On Linear Solution of “Cherry Pickup II”.
Max Weight of Two Disjoint Paths in Node-Weighted Gridlike DAG

Igor N. Tunev
itnvi@mail.ru

Abstract

“Minimum Falling Path Sum” (MFPS) is classic question in programming – “Given a grid of size $N \times N$ with integers in cells, return the minimum sum of a falling path through grid. A falling path starts at any cell in the first row and ends in last row, with the rule of motion – the next element after the cell $(i, j)$ is one of the cells $(i+1, j-1), (i+1, j)$ and $(i+1, j+1)$”. This problem has linear solution (LS) (i.e. $O(N^2)$) using dynamic programming method (DPM).

There is an Multi-Agent version of MFPS called “Cherry Pickup II” (CP2) [1]. CP2 is a search for the maximum sum of 2 falling paths started from top corners, where each covered cell summed up one time. All known fast solutions of CP2 uses DPM, but have $O(N^3)$ time complexity on grid $N \times N$. Here we offer a LS of CP2 (also using DPM) as finding maximum total weight of 2 vertex-disjoint paths. Also, we extend this LS for some extended version of CP2 with wider motion rules.

Key words: dynamic programming, directed acyclic graph, grid, time complexity, combinatorial optimization, linear algorithm, disjoint paths, set

1 Introduction

CP2 is Multi-Agent extension of well known problem, sometimes called as “Minimum Falling Path Sum” in [2], and its variations like “Gold Mine” in [3] and “Minimum Path Sum” in [4].

There is variation of CP2 called “Cherry Pickup” in [5] sometimes called as “Diamond Mine” (DM) in [6]. DM extended with ability to lock cells, but still has linear reducing to CP2, even as finding maximum sum of 2 node-disjoint paths, as will be described below.

For solution of CP2 we offer algorithm for search of 2 paths without crossing with maximum common sum. Thus, this LS can be represented as LS for a simple case of Multi-Agent Path Finding problem (MAPF) with maximizing/minimizing deliveries/cost. The MAPF is the problem of finding collision-free paths for a team of robots from their locations to given destinations in a known environment.

Disjoint paths (DP) problem is one of the well known problems in algorithmic graph theory and combinatorial optimization. There are many LSs of finding fixed number of DP on special cases of graphs. For example, Scheffler found LS on graphs with bounded tree-width [7]. In the paper of Golovach, Kolliopoulos, Stamoulis and Thilikos [8] offered LS on a planar graphs. Most closely for our purpose is LS proposed by Tholey for 2 DP on directed acyclic graphs (DAGs) [9]. But we need in LS on node- or edge-weighted DAGs.

Suitable for our purpose the Suurballe’s algorithm (SA) on edge-weighted digraphs [10], but with not linear complexity, as we will show further. We offer LS for finding 2 node-DP with maximum total weight on some special case of node-weighted DAGs.

1.1 Problem description

Given a grid $g$ of size $H \times W$ with addressable cells from $(0, 0)$ to $(H-1, W-1)$. Each cell in grid represents the number of cherries that we can collect. There are 2 robots in corners $(0, 0)$ and $(0, W-1)$, that can collect cherries. When a robot is located in a cell, it picks up all cherries of this cell, and this cell becomes an empty. We need to collect maximum number of cherries, using these robots. Robots can move according to following rules:

(r1) From cell $(i, j)$, robots can move to cell $(i + 1, j - 1), (i + 1, j)$ or $(i + 1, j + 1)$;
When both robots stay on the same cell, only one of them takes the cherries;

Both robots cannot move outside of the grid at any moment;

Both robots should reach the bottom row in the grid.

The fastest solutions, found by us on the network, have \( O(H \cdot W \cdot \min\{H, W\}) \) complexity. Same complexity can be reached using next naive DPM with 3D structure \( dp \): for each \( i = 0, \ldots, H-2 \) and \( 0 \leq j_1 < j_2 \leq W-1 \)

\[
dp[i][j_1][j_2] = \max_{j_1-1 \leq k_1 \leq j_1, j_2-1 \leq k_2 \leq j_2, 1.0 \leq k_1 < k_2 < W} \{\dp[i+1][k_1][k_2] + \g{k_1,j_1} + \g{k_2,j_2}\}
\]

where \( \dp[H-1][j_1][j_2] = g_{H-1,j_1} + g_{H-1,j_2} \).

Thus, if \( 2H > W \), then we can to find this \( dp \) table and return \( \dp[0][0][W-1] \). If \( 2H \leq W \), then any paths that started from \( (0,0) \) and \( (0, W-1) \) don’t intersect with each other, then this case can be reduced to the original problem with one path.

Here we answer the question – is there a solution of CP2 with \( O(H \cdot W) \) complexity? Also, we show LS for some extension of CP2 (without strong proof of correctness).

### 1.2 Near linear solution using Suurballe’s algorithm

Here we show the simple reduction of CP2 to the well known method for finding 2 node-DP in a nonnegatively-weighted (edge-weighted) digraph, such that both paths connect the same pair of nodes and have minimum total weight.

Let \( m \) be maximum value of \( \g \). Denote by \( \g' \) the edge-weighted DAG with \( WH+2 \) nodes and \( 3(W-2)(H-1) + 4(H-1) + W+2 \) links (directed edges) such that:

1. Each cell of \( \g \) contains one node of \( \g' \). And 2 more nodes \( s \) and \( t \).
2. Weight of link from node in cell \((i,j)\) to node in cell \((i+1,j')\) is \( m-\g{i,j} \), for each \( 0 \leq i < H-1, 0 \leq j < W \) and \( m\{0,j-1\} \leq j' \leq \min\{j+1, W-1\} \). Weights of 2 links from node \( s \) to nodes in cells \((0,0)\) and \((0, W-1)\) are 0. And weight of link from node in cell \((H-1,j)\) to \( t \) is \( m-\g{H-1,j} \) for each \( 0 \leq j < W \).

Now we can find 2 node-DP from \( s \) to \( t \) in \( \g' \) using SA. The total weight of found 2 paths is minimum sum \( M' \). Then required answer is \( m \cdot H - M' \).

#### 1.2.1 Complexity analysis

Let denote the set of edges and nodes of graph \( \g' \) as \( E(\g') \) and \( V(\g') \) respectively. The case when \( W > 2 \cdot H \) is trivial (because of this case can be reduced to problem with one robot in linear time), therefore we can assume that \( W \leq 2 \cdot H \).

Complexity of SA equal to complexity of Dijkstra’s algorithm (DA) \([11]\). As published in \([12]\) by Fredman and Tarjan the DA can be improved using Fibonacci heap and performed in \( O(|E(\g')| + |V(\g')| \log(|V(\g')|)) \). Then we get complexity \( O(H \cdot W \cdot \log(H \cdot W)) = O(H \cdot W \cdot \log(H)) \).

There are other optimisations of DA for our purposes. One of them is algorithm of shortest path (SP) on DAGs. Using topological sorting we can find SP on DAG in linear time as in \([13]\). But SA uses search of SP twice. And before second search of SP, the graph is not a DAG in common case.

Other optimisation for bounded integers weights by some value \( C \). But all such optimisations is not linear. Most fast of them, in our case, published in \([13]\) by Aluja, Mehlhorn, Orlin and Tarjan (AMOT). This algorithm works in \( O(|E(\g')| \cdot \log(C)) \) time. Thus, if \( C \) has polynomial dependence on \( H \), then SA with AMOT optimisation has complexity \( O(H \cdot W \cdot \log(H)) \).

Here we offer linear solution when almost all absolute values of the grid \( \g \) are close to \( C \).

### 2 Defaults

We can assume that absolute values in cells of grids are bounded by value \( C = f(H) \), for some positive real function \( f \) (i.e. in common case some values of \( \g \) can be negative). Exception for values equals to \( -\infty \) – this value is used for bounding of paths.

Also we assume that \( \Theta(H \cdot W) \) of cells have values \( \Theta(f(H)) \). I.e. the length of input data is \( O(H \cdot W \cdot \log(f(H))) \). And assume that \( H, W \geq 2 \).
Definition 1. The $F_{i,j}(g')$ is table, defined by grid $g'$ of size $H \times W$, such that

$$F_{i,j}(g') = \begin{cases} 0 & i = H, \\ g'_{i,j} + \max\{F_{i+1,\max(j-1,0)}(g'), F_{i+1,j}(g'), F_{i+1,\min(j+1,W-1)}(g')\} & 0 \leq j \leq W - 1. \end{cases}$$

for each $0 \leq j \leq W - 1$. By default $F_{i,j}$ means $F_{i,j}(g)$

Definition 2. By path we call an ordered finite sequence (vector) of cells in grid (by default in $g$) using rules $(r1)$ and $(r3)$. I.e. after not the last cell $(i,j)$ the next cell either $(i+1, \max(j-1,0))$ or $(i+1, j)$ or $(i+1, j + 1, W - 1)$.

Location of path in grid can be obtained by addressing to row number. For example, at $i$-th row the path $t$ located at $t(i)$-th column.

Definition 3. Let $t$ is path from row $i_1$ to row $i_2$ ($i_1 \leq i_2$), then denote sum of $t$ as $PS(t)$. I.e.

$$PS(t) = \sum_{k=i_1}^{i_2} g_{k,t(k)}.$$  

Table $F$ is known dynamic programming method of search for maximum (or minimum, if we change the max to min in the definition of $F$) sum of falling path. Also, using $F$ we can choose one of these paths with maximum sum.

Definition 4. Call path $t$ as path defined by $F_{i',j'}$ if $t(i') = j'$ and for each $i = i' + 1, ..., H - 1$

$$t(i) \in \text{arg max}_{j = \max\{t(i-1) - 1,0\}, \ldots, \min(t(i-1) + 1, W-1\}} \{F_{i,j}\}.$$  

Since the $F$ is well known DPM for solution of MFPS, then next simple notes we will not prove

**Note 1.** $t$ defined by $F_{i,j}(g)$ iff $PS(t) = F_{i,j}(g)$.

**Note 2.** If $t$ starts from cell $(i,j)$ then $PS(t) \leq F_{i,j}(g)$.

Definition 5. $l_p$ is leftmost path defined by $F_{0,0}$. I.e. $l_p(0) = 0$ and for each $i = 1, ..., H - 1$

$$l_p(i) = \min_{j = \max\{l_p(i-1) - 1,0\}, \ldots, \min(l_p(i-1) + 1, W-1\}} \{F_{i,j}\}.$$  

And $r_p$ is rightmost path defined by $F_{0,W-1}$. I.e. $r_p(0) = W - 1$ and for each $i = 1, ..., H - 1$

$$r_p(i) = \max_{j = \max\{r_p(i-1) - 1,0\}, \ldots, \min(r_p(i-1) + 1, W-1\}} \{F_{i,j}\}.$$  

By Note 1 we get $PS(l_p) = F_{0,0}(g)$ and $PS(r_p) = F_{0,W-1}(g)$. Then, if $l_p$ don’t intersect with $r_p$, then required answer is $F_{0,0} + F_{0,W-1}$. This case can be checked in $O(H \times W)$ of linear operations with numbers of length $\log(H)$. Further we suppose that $l_p$ intersects with $r_p$.

Due to simmetry of rules by left and right for input data and moving, all properties we will formulate for one side only. For other side all these properties can be formulated and proved similarly.

By default, if name of pair of paths starts from letters ’”l”’ and ’”r”’, then it means that path with first letter ””l”” located on the left side of path with first letter ””r””.

When we talk ”for each $i$” for rows, we mean ”for each $i = 0, ..., H - 1$”. When we talk ”for each $j$” for columns, we mean ”for each $j = 0, ..., W - 1$”.

3 Definitions and properties

Definition 6. Let $0 \leq i_1 < i_2 \leq H - 1$ and path $t$ with begining not after $i_1$-th row and with ending not before $i_2$-th row. By subpath between rows $i_1$ and $i_2$ of $t$ we call path $((i_1, t(i_1)), (i_1 + 1, t(i_1 + 1)), ..., (i_2, t(i_2)))$ and denote it as $t[i_1, ..., i_2]$.

By default $i_1 = 0, i_2 = H - 1$.

Definition 7. Let $t$ is path from row $i_1$ to row $i_2$. By tail of path $t$ from $(i, t(i))$ (or from $i$-th row) we call subpath $t[i, ..., i_2]$ and denote as $t[i, ..., i]$.

By prefix (or head) of path $t$ with end on $(i, t(i))$ we call subpath $t[i_1, ..., i]$ and denote $t[i, ..., i]$.  

3
Definition 8. Let $t_1$ and $t_2$ are paths. Suppose that $t_1$ ends after $(i-1)$-th row and $t_2$ starts before $(i+2)$. By **concatenation** $t$ of $t_1[...i]$ and $t_2[i+1,...]$ we call the sequence of cells ordered by rows where $t[...i] = t_1[...i]$ and $t[i+1,...] = t_2[i+1,...]$.

Note 3. Let $t_1$ and $t_2$ are paths and $t_1(i) = t_2(i)$ then concatenation $t$ of $t_1[...i]$ and $t_2[i+1,...]$ is path. I.e. $t$ satisfies the rules (r1) and (r3).

Definition 9. The path $t$ **intersect cell** $(k,m)$ when $t(k) = m$.
The path $t_1$ **intersects path** $t_2$ at $i$-th row when either $(t_1(i-1) \leq t_2(i-1)$ and $t_1(i) \geq t_2(i))$ or $(t_1(i-1) \geq t_2(i-1)$ and $t_1(i) \leq t_2(i))$.

Note that “paths without intersection” (PW0I) is more stronger than “node-disjoint paths” (or “cell-disjoint paths” in our case).

Property 1. Let path $p_1$ intersects the path $p_2$ at row $i+1$ where $p_1(i) \leq p_2(i)$ and $p_2(i+1) \geq p_2(i+1)$, then tails of $p_1$ and $p_2$ from row $i+1$ are swapable. It mean that concatenation of $p_1[0,...,i]$ and $p_2[i+1,...]$ is path, and concatenation of $p_2[...i]$ and $p_1[i+1,...]$ is path too.

Proof. There are 2 case of intersections:

- When $p_1(i) = p_2(i)$.
  Then using rule (r1) we get $p_1(i) - 1 = p_2(i) - 1 \leq p_2(i+1) \leq p_2(i) + 1 = p_1(i) + 1$.
  I.e. $p_1(i) - 1 \leq p_2(i+1) \leq p_1(i) + 1$. Thus $p_1[...i]$ can be continued by $p_2[i+1,...]$ without breaking of rule (r1). A similar proof for concatenation of $p_2[...i]$ and $p_1[i+1,...]$.

- When $p_1(i) < p_2(i)$.
  Then using rule (r1) we get $p_1(i) - 1 < p_2(i) - 1 \leq p_2(i+1) \leq p_1(i) + 1$. And again, $p_1[...i]$ can be continued by $p_2[i+1,...]$ without breaking of rule (r1).
  Also using (r1) we get $p_2(i) - 1 \leq p_2(i+1) \leq p_1(i+1) \leq p_1(i) + 1 < p_2(i) + 1$. Thus $p_2[...i]$ can be continued by $p_1[i+1,...]$ without breaking of rule (r1).

Since $p_1$ and $p_2$ satisfy the rule (r3), then any subpaths of them are satisfy the rule (r3).

Thus all these concatenations satisfy the rules (r1) and (r3). I.e. concatenation of $p_1[0,...,i]$ and $p_2[i+1,...]$ is path, and concatenation of $p_2[...i]$ and $p_1[i+1,...]$ is path too. 

\[\blacksquare\]

Note 4. If path $t$ defined by $F_{i,j}$, then for any row $i' \geq i$ we get $\text{PS}(t[i',...]) = F_{i',t(i')}$. 

Property 2. Consider path $t_1$ started from cell $(i_1,j_1)$ and has maximum sum (i.e. $t_1$ is path defined by $F_{i_1,j_1}$). Suppose that $t_1$ intersect $(k_1,m_1)$-th and $(k_2,m_2)$-th cells, where $k_2 > k_1 \geq i_1$. Then:

1. $\text{PS}(t_1[i_1,...,k_2-1]) = F_{k_1,m_1} - F_{k_2,m_2}$;
2. Let path $t$ intersect cells $(k_1,m_1)$ and $(k_2,m_2)$ then $\text{PS}(t[i_1,...,k_2]) \leq \text{PS}(t_1[i_1,...,k_2])$;
3. Let path $t$ intersect cell $(k_1,m_1)$ and $t$ intersect $t_1$ at row $k_2$ then $\text{PS}(t[i_1,...,k_2-1]) \leq \text{PS}(t_1[i_1,...,k_2-1])$;
4. Let path $t$ intersect cells $(k_1,m_1)$ and $(k_2,m_2)$, and $\text{PS}(t[i_1,...,k_2-1]) = F_{k_1,m_1} - F_{k_2,m_2}$. Then for any $k_1 \leq k_1' \leq k_2 \leq k_2$ we get $\text{PS}(t[k_1',...,k_2-1]) = F_{k_1',t(k_1')} - F_{k_2',t(k_2')}$;
5. Let path $t$ intersect cells $(k_1,m_1)$ and $(k,m)$ for some $k > k_1$ and $0 \leq m \leq W - 1$, then $\text{PS}(t[i_1,...,k-1]) \leq F_{k_1,m_1} - F_{k,m}$.

Proof. 1. Since $t_1$ defined by $F$, then for any row $i \geq i_1$ by Note 2 we get $\text{PS}(t_1[i,...]) = F_{i,t_1(i)}$. Thus $\text{PS}(t_1[i_1,...,k_2-1]) = \text{PS}(t_1[i_1,...] - \text{PS}(t_1[k_2,...]) = F_{k_1,m_1} - F_{k_2,m_2}$.

2. Suppose that $\text{PS}(t[k_1,...,k_2]) > \text{PS}(t_1[k_1,...,k_2])$.
   Let $t'$ is concatenation with begining on cell $(k_1,m_1)$ such that $t'[k_1,...,k_2] = t[k_1,...,k_2]$ and $t'[k_2+1,...] = t_1[k_2+1,...]$. By Note 4 the $t'$ is path.
   Then $F_{k_1,m_1} \geq \text{PS}(t')$ and the other side:
   $\text{PS}(t') = \text{PS}(t[k_1,...,k_2]) + \text{PS}(t_1[k_2+1,...]) > \text{PS}(t_1[k_1,...] + \text{PS}(t_1[k_2+1,...]) = \text{PS}(t_1[k_1,...]) = F_{k_1,m_1}$.
   This contradiction proves statement 2.
3. Let \( t' \) is concatenation of \([t[k_1, ..., k_2 - 1]] \) and \([t_1[k_2, ...]] \). By Property 4, \( t' \) is path. Also \( t' \) intersects with cells \((k_1, m_1) \) and \((k_2, m_2) \). Then using Property 2, we get \( PS([t[k_1, ..., k_2 - 1]]) = PS(t[k_1, ..., k_2]) - g_{k_2, m_2} \geq PS(t'[k_1, ..., k_2]) - g_{k_2, m_2} = PS([t[k_1, ..., k_2 - 1]]) \).

4. Let \( t_2 \) is path defined by \( F_{k_2, m_2} \). And \( t' \) is concatenation of \([t[k_1, ..., k_2 - 1]] \) and \([t_2[k_2, ...]] \). Then by Note 3, \( t' \) is path, with sum \( PS(t') = PS([t[k_1, ..., k_2 - 1]] + PS(t_2[k_2, ...]) = F_{k_1, m_1} - F_{k_2, m_2} = F_{k_1, m_1}. \) i.e. \( t' \) defined by \( F_{k_1, m_1} \).

Then using Property 2, we get \( PS(t'[k_1', ..., k_2'-1]) = F_{k_1', t'} - F_{k_2', t'} \). Since \( t(k_2) = t'(k_2) \) then \( t[k_1', ..., k_2'] = t'[k_1', ..., k_2'] \) then \( PS(t'[k_1', ..., k_2'-1]) = F_{k_1', t'} - F_{k_2', t'} \).

5. Let \( b_1 = \max(0, t(k - 1) - 1) \) and \( b_2 = \min(t(k - 1) + 1, W - 1) \). Then \( m \in \{b_1, ..., b_2\} \).

Let prove by induction on difference \( k - k_1 \)

**Base case:**
If \( k - k_1 = 1 \) then \( PS([t[k_1, ..., k_2 - 1]]) = g_{k_1, m_1} = g_{k_1, t(k_1)} \leq g_{k_1, t(k_1) - 1} + \max_{j=b_1, ..., b_2} \{F_{k_1, j}\} \leq F_{k_1, m_1} = F_{k_1, m_1} - F_{k_1, m_1} = F_{k_1, m_1} - F_{k_1, m_1}.

**Induction step:**
Let \( k - k_1 > 1 \), and \( PS([t[k_1, ..., k_2 - 1]]) \leq F_{k_1, m_1} - F_{k_1, t(k_1)}.

Then \( PS([t[k_1, ..., k_2 - 1]]) = PS([t[k_1, ..., k_2 - 2]]) + g_{k_1, t(k_1)} \leq \max_{j=b_1, ..., b_2} \{F_{k_1, j}\} \leq F_{k_1, m_1} = F_{k_1, m_1} - F_{k_1, t(k_1)} - F_{k_1, t(k_1)} = F_{k_1, m_1} = F_{k_1, m_1} - F_{k_1, m_1}.

**Note 5.** \( l_p(i) \leq r_p(i) \) for each \( i = 0, ..., H - 1 \).

**Note 6.** \( PS(l_p) = F_{0, 0} \) and \( PS(r_p) = F_{0, W-1} \).

**Definition 10.** \( g_i \) is grid defined for each \( i = 0, ..., H - 1 \) as:

\[
g_{i,j} = \begin{cases} \infty & j = l_p(i) + 1, ..., W - 1, \\ g_{i,j} & j = 0, ..., l_p(i). \end{cases}
\]

And \( g_r \) is grid defined for each \( i = 0, ..., H - 1 \) as:

\[
g_{i,j} = \begin{cases} g_{i,j} & j = r_p(i), ..., W - 1, \\ -\infty & j = 0, ..., r_p(i) - 1. \end{cases}
\]

**Property 3.** For each \( i = 0, ..., H - 1 \) and \( j \leq l_p(i) \) we get \( F_{i,j}(g) \leq F_{i,j}(g_i) \), and for \( j \geq r_p(i) \) we get \( F_{i,j}(g) \leq F_{i,j}(g_r) \).

**Proof.** Due to \( g_{i,j} \geq g_{i,j} \) for each \( i \) and \( j \), we get \( F_{i,j}(g) \geq F_{i,j}(g_i) \) for each \( i \) and \( j \).

Let \( t \) is path defined by \( F_{i_1,j_1}(g) \) for some \( i_1 \) and \( j_1 \leq l_p(i_1) \), then \( PS(t) = F_{i_1,j_1}(g) \).

Consider 2 cases:

- If \( l_p(i) \leq l_p(i) \) for each \( i \), then \( F_{i_1,j_1}(g) \geq PS(t) = F_{i_1,j_1}(g) \).
- Let \( i_2 \) is lowest row such that \( l_p(i_2) > l_p(i) \) (i.e. \( i_2 > i_1 \)). Then due to Property 1 a concatenation \( t' \) of \( l_p[i_2, ..., i_2 - 1] \) and \( t[i_2, ...] \) is path.

Since \( t \) defined by \( F(g) \) then by Note 4 we get \( PS([t[i_2, ...]] = F_{i_2,t(i_2)}(g) \). Since \( l_p \) defined by \( F(g) \) then by Property 2 we get \( PS([l_p[i_2, ...]] = F_{i_2,l_p(i_2)}(g) \).

Then \( F_{i_2,l_p(i_2)}(g) \geq PS(t') = PS([l_p[i_2, ...]] + PS([t[i_2, ...]] = F_{i_2,l_p(i_2)}(g) + F_{i_2,t(i_2)}(g) \). Thus \( F_{i_2,l_p(i_2)}(g) \geq F_{i_2,t(i_2)}(g) \).

Consider concatenation \( t'' \) of \( l_p[i_2, ..., i_2 - 1] \) and \( l_p[i_2, ...] \). Then due to Property 1 the \( t'' \) is path.

Since \( l_p \) defined by \( F(g) \), due to Note 4 we get \( PS([l_p[i_2, ...]] = F_{i_2,l_p(i_2)}(g) \). By Property 2 we get \( PS([l_p[i_2, ...]] = F_{i_2,l_p(i_2)}(g) \). Then \( PS(t'') = PS([l_p[i_2, ..., i_2 - 1]] + PS([l_p[i_2, ...]] = F_{i_1,j_1}(g) - F_{i_2,l_p(i_2)}(g) + F_{i_2,t(i_2)}(g) \).

By our choice of \( t' \) we get \( t''(i) \leq l_p(i) \) for each \( i \). Then \( F_{i_1,j_1}(g) \geq PS(t'') \geq F_{i_1,j_1}(g) \).
Similarly we can proof that \( F_{i,j}(g) = F_{i,j}(g_i) \).

**Property 4.** Let \( 0 \leq i_1 < i_2 \leq H-1 \), and consider path \( t \) from cell \( (i_1, l_p(i_1)) \) to cell \( (i_2, l_p(i_2)) \), and path \( t' \) from cell \( (i_1, r_p(i_1)) \) to cell \( (i_2, r_p(i_2)) \). Then:

1. Due to Property 2 and Note 2 we get \( PS(t) \leq PS(l_p[i_1, ..., i_2]) \). Similarly we get \( PS(t') \leq PS(r_p[i_1, ..., i_2]) \).

2. Due to Property 1, leftmost of \( l_p \) and rightmost of \( r_p \) we get implication:
   - if \( PS(t) = PS(l_p[i_1, ..., i_2]) \) then \( t(i) \geq l_p(i) \) for each \( i = i_1, ..., i_2 \); if \( PS(t') = PS(r_p[i_1, ..., i_2]) \) then \( t'(i) \leq r_p(i) \) for each \( i = i_1, ..., i_2 \).

3. If \( t \) is LP path and \( PS(t) = PS(l_p[i_1, ..., i_2]) \), then by Property 2 we get \( t = l_p[i_1, ..., i_2] \). Similarly, if \( t' \) is RP path and \( PS(t') = PS(r_p[i_1, ..., i_2]) \), then \( t' = r_p[i_1, ..., i_2] \).

4. If \( p \) is \( LP_{i_1, t_1(i_1)} \) path and \( PS(p) = PS(l_p[i_1, ...,]) \), then due to leftmost and maximum sum of \( l_p \) we get \( p = l_p[i_1, ...] \).
   - Similarly, if \( p' \) is \( RP_{i_1, t_2(i_1)} \) path and \( PS(p') = PS(r_p[i_1, ...,]) \), then \( p' = r_p[i_1, ...] \).

**Definition 11.** Let path \( t \) begin at cell \((i, j)\) and ends at \((i', j')\) then \( t \) is leftmost of \( PS \).

**Note 7.** If \( t \) is LP path, and \( t(i) = r_p(i) \), then \( l_p(i) = r_p(i) \). If \( t \) is RP path, and \( t(i) = l_p(i) \), then \( l_p(i) = r_p(i) \).

**Note 8.** Let paths \( t_1, ..., t_n \) don't intersect the path \( t_0 \), and all \( t_1, ..., t_n \) placed on the same side of \( t_0 \). And \( t \) is concatenation of \( t_1, ..., t_n \) subpaths, such that \( t \) is path. Then \( t \) is path without intersection with any subpath of \( t_0 \).

**Note 9.** Let \( t_1, ..., t_n \) are \( RP_{i_1, t_1(i_1)} \), ..., \( RP_{i_n, t_1(i_n)} \) paths respectively, and \( t \) is concatenation of \( t_1, ..., t_n \) subpaths, such that \( t \) is path. Then \( t \) is \( RP_{i_n, t_n(i_n)} \) path for some \( i_1, ..., i_n \).

**Definition 12.** Let \( t_1 \) and \( t_2 \) are \( LP_{i_2, j_1} \) and \( RP_{i_1, j_2} \) PWOI, such that \( PS(t_1) + PS(t_2) \) is maximum among all \( LP_{i_1, j_1} \) and \( RP_{i_1, j_2} \) pairs of PWOI and ending at bottom (BE), then we call this pair as \( \text{pair} \) (l)eft and (r)ight (d)isjoint (t) racks with (m) aximum (s) um.

**Definition 13.** \( M_i \) is table, where \( M_i(j, j) = PS(l) + PS(r) \) for any \( \text{lrdts}(i, j, j) \) \( \text{pair} \) \( (l) \)eft and \( (r) \)ight \( (d) \) isjoint \( (t) \) racks with \( (m) \) aximum \( (s) \) um.

**Note 10.** For each row \( i \) the \( M_i \) defined in columns \( j \leq \min\{l_p(i), r_p(i) - 1\} \) only. For each row \( i \) the \( M_i \) defined in columns \( j \geq \max\{l_p(i) + 1, r_p(i)\} \) only.

### 3.1 Linear search of \( M_i \) and \( M_r \)

**Property 5.** Let \( l \) and \( r \) are \( \text{lrdts}(i, j_1, j_2) \) \( \text{pair} \) for some \( j_1 \leq l_p(i) \) and \( j_2 \geq r_p(i) \).

1. If \( l \) intersect \( l_p \) at 2 rows \( i_2 > i_1 > i \), and \( r \) don’t intersect \( l_p \) between these rows, then \( l[t[i_1, ..., i_2]] = l_p[i_1, ..., i_2] \).

2. If \( l \) intersect \( l_p \) at row \( i' \), and \( r \) don’t intersect \( l_p \) after this row, then \( l[t[i', ...]] = l_p[i', ...] \).

**Proof.** 1. Suppose that \( l[t[i_1, ..., i_2]] \neq l_p[i_1, ..., i_2] \).

   If suppose that \( PS(l[t[i_1, ..., i_2]]) = PS(l_p[i_1, ..., i_2]) \) then by Property 3 we get \( l[t[i_1, ..., i_2]] = l_p[i_1, ..., i_2] \) that contradicts to our assumption. Thus, using Property 1, we get inequality \( PS(l[t[i_1, ..., i_2]]) < PS(l_p[i_1, ..., i_2]) \).

   Since \( l \) is LP path then because of the intersection with \( l_p \) on \( i_1 \) and \( i_2 \) we get \( l(t[i_1]) = l_p(i_1) \) and \( l(t[i_2]) = l_p(i_2) \). Then consider concatenation \( l' \):
   
   \[
   l'[i, ..., i_1 - 1] = l[t[i, ..., i_1 - 1]], \\
   l'[i_1, ..., i_2] = l_p[i_1, ..., i_2], \\
   l'[i_2 + 1, ...] = l[t[i_2 + 1, ...]].
   \]
By Note \text{3} the $lt'[i, \ldots]$ is path. Then by Note \text{8} the $lt'$ is path. By Note \text{9} the $lt'$ is $LP_{0,0}$ path. By Note \text{8} $lt'$ don’t intersects with $rt$.

Consider relation between $PS(lt)$ and $PS(lt')$:
\[
PS(lt) = PS(lt[i, \ldots, i_1=1]) + PS(lt[i_1, \ldots, i_2]) + PS(lt[i_2+1, \ldots]) < PS(lt[i, \ldots, i_1=1]) + PS(lp[r_p, \ldots, r_p]) + PS(lt[i_2+1, \ldots]) = PS(lt').
\]

Thus we get $lt'$ and $rt$ are $LP_{i,j_1}$ and $RP_{i,j_2}$ paths without intersection with sum $PS(lt') + PS(rt) > PS(lt) + PS(rt)$. That contradict to maximum sum of $ldtms(i, j_1, j_2)$. Let $lt$ and $rt$.

2. Suppose that $lt'[i', \ldots] \neq lp[i', \ldots]$. Since $lt$ is $LP$ path then because of the intersection with $lp$ on $i'$ we get $lt(i') = lp[i']$ and $lt[i' + 1, \ldots] \neq lp[i', \ldots]$. Then consider concatenations $lt'$ and $lt''$:
\[
lt'[i', \ldots] = lt[i', \ldots], \quad lt''[i' + 1, \ldots] = lp[i'] + 1, \ldots.
\]

Then by Note \text{8} the $lt'$ and $lt''$ are paths. By Note \text{9} the $lt'$ and $lt''$ are $LP$ paths. By Note \text{8} $lt'$ don’t intersects with $rt$.

Since $lt''[i' + 1, \ldots] = lt[i' + 1, \ldots] \neq lp[i', \ldots]$, then $lt'' \neq lp$. Then due to leftmost of $lp$ among all LP paths with maximum sum we get $PS(lp) > PS(lt'')$. Then $PS(lt[i'+1, \ldots]) = PS(lt'') - PS(lp[i', \ldots]) < PS(lp) - PS(lp[i', \ldots]) = PS(lp[i'+1, \ldots])$.

Then $PS(lt) = PS(lt[i, \ldots, i']) + PS(lt[i' + 1, \ldots]) < PS(lt[i, \ldots, i']) + PS(lt[i' + 1, \ldots]) = PS(lt')$.

Thus we get $LP_{i,j_1}$ and $RP_{i,j_2}$ paths $lt'$ and $rt$ without intersections with sum $PS(lt') + PS(rt) > PS(lt) + PS(rt)$. That contradict to maximum sum of $ldtms(i, j_1, j_2)$. Let $lt$ and $rt$.

\begin{property}
Let $lt$ and $rt$ are $ldtms(i, j_1, j_2)$ pair. Then for any $i' \geq i$ the path $lt[i', \ldots]$ and $rt[i', \ldots]$ are $ldtms(i', lt(i'), rt(i'))$ pair.
\end{property}

\begin{proof}
By Note \text{8} the $lt[i', \ldots]$ don’t intersects with $rt[i', \ldots]$. By Note \text{9} the $lt[i', \ldots]$ and $rt[i', \ldots]$ are $LP_{i', rt(i')}$ and $RP_{lt(i'), rt(i')}$ paths respectively.

Let $lmt$ and $rmt$ are $ldtms(i', lt(i'), rt(i'))$ pair. Suppose that $PS(lmt) + PS(rmt) > PS(lt[i', \ldots]) + PS(rt[i', \ldots])$. Consider concatenations $lp$ and $rp$ such that:
\[
lp[i, \ldots, i-1] = lt[i, \ldots, i-1], \quad rp[i', \ldots, i'-1] = rt[i', \ldots, i'-1], \quad lmt[i', \ldots], \quad rmt[i', \ldots].
\]

By Note \text{3} the $lp$ and $rp$ are paths. By Note \text{8} $lp$ is $LP_{i,j_1}$ path and $rp$ is $RP_{i,j_1}$ path. Since $lt[i, \ldots, i-1]$ don’t intersects with $rt[i, \ldots, i-1]$, and $lmt[i', \ldots]$ don’t intersects with $rmt[i', \ldots]$, then $lp$ don’t intersects with $rp$. Then due to maximum sum of $lt$ and $rt$ we get $PS(lp) + PS(rp) \leq PS(lt) + PS(rt)$. But the other side
\[
PS(lp) + PS(rp) = PS(lt[i, \ldots, i-1]) + PS(lmt[i', \ldots]) + PS(rt[i, \ldots, i-1]) + PS(rmt[i', \ldots]) > PS(lt[i, \ldots, i-1]) + PS(lt[i', \ldots]) + PS(rt[i, \ldots, i-1]) + PS(rt[i', \ldots]) = PS(lt) + PS(rt).
\]

This contradiction proves that $PS(lmt) + PS(rmt) = PS(lt[i', \ldots]) + PS(rt[i', \ldots])$.

Thus we get $LP_{i,lt(i')}$ and $RP_{i',rt(i')}$ paths $lt[i', \ldots]$ and $rt[i', \ldots]$ respectively without intersection with maximum sum. I.e. $lt[i', \ldots]$ and $rt[i', \ldots]$ are $ldtms(i', lt(i'), rt(i'))$ pair.
\end{proof}

\begin{property}
Let $lt$ and $rt$ are $ldtms(i, lt(i), rt(i))$ pair, $lt[i, \ldots, ri]$ don’t intersects with $lp[i, \ldots, ri]$ and $rt(r(i)) = lp(r(i))$ for some $i < ri$. Let $i < i' < ri$ and $lp(i') \leq i' \leq rt(i')$. Consider $RP_{i', j'}$ path $rt'$ where $rt'[ri, \ldots] = rt[ri, \ldots]$ and $PS(rt'[i', \ldots]) = rt'[i', \ldots]$. Then $lt[i', \ldots]$ and $rt'[i', \ldots]$ are $ldtms(i', lt(i'), j')$ pair.
\end{property}

\begin{proof}
Since $lt$ is $LP_{i,lt(i)}$ path and don’t intersects with $lp[i, \ldots, ri]$, then $lt(k) < lp(k) \leq rp(k) \leq rt'(k)$ for each $k = i', \ldots, ri$. Since $lt$ don’t intersects with $rt$, then by Note \text{3} the $lt'[i', \ldots]$ don’t intersects with $rt'$.

Let denote $lt'[i', \ldots]$ and $rt'[i', \ldots]$ as $IT$ and $RT$ respectively. Consider $ldtms(i', lt(i'), j')$ pair $LP$ and $RP$. Since $RP$ is $RP_{i', j'}$ path and $j' \leq rt(i') = rT(i')$, then $RP$ intersects with $RT$ on some row $ri \leq ri$. Then $RT$ don’t intersects with $LP$ before $RI$. Since $IT$ don’t intersects with any of $RP$ path before $ri$, then $LT$ don’t intersects with $RP$ before $ri$.

Let $rP_1$ and $rP'$ are concatenations:
\[
rt'[i', \ldots, ri-1] = rP[lp[i', \ldots, ri-1], \quad rT'[ri, \ldots] = rT[ri, \ldots],
\]
\[
rP'[i', \ldots, ri-1] = rT'[i', \ldots, ri-1], \quad rP'[ri, \ldots] = rP[ri, \ldots].
\]
If $rP(rI) = rT(rI)$ then by Note $\text{X}$ the $rT'$ and $rP'$ are paths. If $rP(rI) \neq rT(rI)$ then $rP(rI) > rT(rI)$ then by Property $\text{I}$ the $rT'$ and $rP'$ are paths. Then by Note $\text{X}$ $rT'$ and $rP'$ are RP paths. Using Note $\text{X}$ the $LP$ don't intersects with $rP'$, and $IT$ don't intersects with $rT'$.

Consider relations of differences $d_1 = PS(lP) - PS(lT)$ and $d_2 = PS(rT[rI, ...]) - PS(rP[rI, ...])$:

- $d_1 > d_2$. We get $LP_{r*I', lT(i')}$ and $RP_{r*I', rT(i')}$ paths $LP$ and $rP'$ without intersections with sum $PS(lP) + PS(rP') = d_1 + PS(lT) + PS(rT[i', ..., I - 1]) + PS(rP[rI, ...]) = d_1 + PS(lT) + PS(rT[i', ..., I - 1]) + PS(rP[rI, ...]) - d_2 > > PS(lT) + PS(rT)$, which contradicts to maximum of $PS(lT) + PS(rT)$ due to Property $\text{X}$.

- $d_1 \leq d_2$. We get $LP_{r*I', lT(i')}$ and $RP_{r*I', rT(i')}$ paths $IT$ and $rT'$ without intersections with sum $PS(lT) + PS(rT') = PS(lP) + d_1 + PS(rP[i', ..., I - 1]) + PS(rP[rI, ...]) = PS(lP) + d_1 + PS(rP[i', ..., I - 1]) + PS(rP[rI, ...]) + d_2 \geq \geq PS(lP) + PS(rP)$.

Inequality $PS(lT) + PS(rP)$ > $PS(lP) + PS(rP)$ contradicts the maximum of $PS(lP) + PS(rP)$ among all pairs of $LP_{r*I', lT(i')}$ and $RP_{r*I', rT(i')}$ paths without intersections.

Thus we get one valid case $d_1 = d_2$ with equation $PS(lT) + PS(rT') = PS(lP) + PS(rP)$. I.e. $IT = lT[i', ...]$ and $rT'$ are $lrdms(i', lT(i'), j')$ pair. Since $rI \leq rI$ then $rT'(rI) = rT(rI)$.

Thus we get $RP_{r*I', rT(i')}$ path $rT'$ where $rT'[rI, ...] = rT[rI, ...]$. Using Properties $\text{X}$ and $\text{X}$ we get $PS(rT'[i', ..., rI]) \leq \leq \leq F_{rT', F_{rT, rT'(i')}} + g_{rT, rT'(rI)} = F_{rT', rT'(i')} - F_{rT, rT'(rI)} + g_{rT, rT'(rI)} = PS(rT'[i', ..., rI]).$

Then, using condition $rI \leq rI$, we get $PS(lT[i', ...]) + PS(rT') = PS(lT) + PS(rT'[i', ..., rI]) \geq \geq PS(lT) + PS(rT'[i', ..., rI]) + PS(rT[rI, ...]) = PS(lT) + PS(rT'[i', ..., rI]) + PS(rT[rI, ...]) = PS(lT) + PS(rT')$. Then we get that $lT[i', ...]$ and $rT'$ are $LP_{r*I', lT(i')}$ and $RP_{r*I', rT(i')}$ paths respectively without intersections and with maximum sum. I.e. $lT[i', ...]$ and $rT'$ are $lrdms(i', lT(i'), j')$ pair.

**Property 8.** Let $lI$ and $rI$ are $lrdms(i - 1, lP(i - 1), j)$ pair, where $j > rP(i - 1)$. And $lT(i) < lP(i)$, $rT(i) > rP(i)$. Then:

1. Exist $rI_i > i$ such that $rT(rI_i) = lP(rI_i)$, and $lT(k) < lP(k)$ for each $k = i, ..., rI_i$;

2. Consider concatenation $rT'$ of $lP[i, ..., rI_i - 1]$ and $rT[i, ...]$ (i.e. $rT'[i, ..., rI_i] = rT[i, ..., rI_i]$). Then $lT[i, ...]$ and $rT'$ are $lrdms(i, lT(i), rT(i))$ pair;

3. $PS(lT[i - 1, ..., rI_i]) = F_{lT, rT(i - 1)} - F_{lT, rT(i)} + g_{lT, rT(i)}$. And $PS(rT[i, ..., rI_i - 1]) = F_{lT, rT(i)} - F_{lT, rT(i)}$ by Property $\text{X}$ $I$;

4. Let $b_1 = \max\{0, lP(i - 1) - 1\}, b_2 = \min\{lP(i - 1) + 1, lP(i) - 1\}$ and $b_3 = \max\{rP(i) + 1, j - 1\}, b_4 = \min\{j + 1, W - 1\}$ then $PS(lT[i, ...]) + PS(rT[i, ...]) = \max_{k = \max\{b_1, b_2\}} \{M_{lT}(i, k)\} + \max_{k = \max\{b_3, b_4\}} \{F_{lT, i, k}\} - F_{lT, rP(i)}$.

**Proof.** 1. Suppose that $rT$ don't intersect $lP$ after $(i - 1)$-th row. Then due to Property $\text{X}$ we get $lT[i, ...] = lP[i, ...]$ that contradicts with condition $lT(i) < lP(i)$. I.e. $rT$ intersect $lP$ after $(i - 1)$-th row.

Let $rI_i \geq i$ such that $rT(rI_i) = lP(rI_i)$, and $rT(k) \neq lP(k)$ for each $k = i, ..., rI_i - 1$.

Suppose that $lT$ intersects with $lP$ on row $lI_i$ between $i$ and $rI_i$. Since $lT(i - 1) = lP(i - 1)$, then due to Property $\text{X}$ we get $lT[i - 1, ..., lI_i] = lP[i - 1, ..., lI_i]$ that contradicts with $lT(i) < lP(i)$.

Thus $lT(k) \neq lP(k)$ for each $k = i, ..., rI_i$. Then because of $lT$ is LP path then $lT(k) < lP(k)$ for each $k = i, ..., rI_i$. Since $rP(i) < rT(i)$ then $rI_i > i$.

2. Since $rT(rI_i) = lP(rI_i)$ then using Note $\text{X}$ we get $rT'[i, ..., rI_i] = rP[i, ..., rI_i]$. Due to Note $\text{X}$ the $rT'$ is path. By Note $\text{X}$ the $rT'$ is $RP_{rP, rI}$ path.

Since $rP$ defined by $F$ then $PS(rT'[i, ..., rI_i]) = PS(rP[i, ..., rI_i]) = F_{lT, rT'(i)} - F_{lT, rT'(rI)} + g_{lT, rT'(rI)}$. Then due to Property $\text{X}$ the $lT[i, ...]$ and $rT'$ are $lrdms(i, lT(i), rT'(i))$ pair. Since $rT'(i) = rP(i)$ we get proof of statement 2.
3. Since $l_p(i-1) = l_l(i-1)$ and $l_l$ is LP path then by Property 4.1 $PS(l_p[i-1, ...]) \geq PS(l_l)$. Since $l_p(i) < l_l(i)$ then $l_p[i-1, ...] \neq l_l[i-1, ...]$. Then since $l_p(i-1) = l_l(i-1)$ and $l_l$ is LP path by Property 4.1 we get $PS(l_p[i-1, ...]) > PS(l_l[i-1, ...])$

Consider $RP_{i-1,j}$ path $rt''$ defined by $F_{i-1,j}(r)$. Then $PS(rt') \leq F_{i-1,j}(r) = PS(rt'')$.

In case when $rt'(k) < rt''(k)$ for each $k \geq i-1$ we get $l_p[i-1, ...]$ and $rt''$ are $LP'_{i-1,j}(i-1)$ and $RP_{i-1,j}$ paths without intersections and with sum $PS(l_p[i-1, ...]) + PS(rt'') > PS(l_l) + PS(rt)$ that contradict to maximum sum of $l_l$ and $rt$. I.e. this case impossible.

Then $rt'(i') \geq rt''(i')$ for some $i' > i-1$. WLOG we can assume that $rt'(k) < rt''(k)$ for each $k = i-1, ..., i' - 1$.

Let $ri' > ri$ such that $r_p(ri') < rt(ri')$ and $r_p(k) = rt(k)$ for each $k = ri, ..., ri' - 1$. I.e. using Property 8.1 we get $lt(k) < rp(k)$ for each $k = i, ..., i' - 1$. And since $rt''(i-1) = j > r_p(i-1) \geq l_p(i-1)$ then $lt$ don’t intersects with $rt''[i-1, ... , i' - 1]$. If $rt[ri, ...] = r_p[ri, ...]$ then we can assume that $ri' = H$ and $F_{i,k} = 0$ for each $k = 0, ..., W - 1$.

If $i' < ri$ then $rt''(i') = rt'(i') = r_p(i')$ then due to Property 4.1 we get $rt''[i', ...] = r_p[i', ...]$. Then $rt''(ri' - 1) = r_p(ri' - 1) = rt(ri' - 1)$.

Then due to Properties 2,3 we get $PS(rt'[i-1, ..., ri' - 2]) \geq PS(rt[i-1, ..., ri' - 2])$.

Suppose that $PS(rt''[i-1, ..., ri' - 2]) > PS(rt[i-1, ..., ri' - 2])$. Then consider concatenation $rp$ of $rt''[i-1, ..., ri' - 2]$ and $rt[ri' - 1, ...]$. Since $rt''(ri' - 1) = rt(ri' - 1)$ then by Property 1 the $rp$ is path. By Note 1 the $lt$ don’t intersects with $rp$. Thus $lt$ and $rp$ are $LP'_{i-1,lp(i-1)}$ and $RP_{i-1,j}$ paths without intersection with sum $PS(l_l) + PS(rp) = PS(l_l) + PS(rt''[i-1, ..., ri' - 2]) + PS(rt[ri' - 1, ...]) > PS(l_l) + PS(rt)$ that contradict to maximum sum of $l_l$ and $rt$.

Thus $PS(rt[i-1, ..., ri' - 2]) = PS(rt''[i-1, ..., ri' - 2])$. Then using Property 2, 4 and $ri \leq ri' - 1$ we get $PS(rt[i, ..., ri - 1]) = F_{i,rt(i)} - F_{i,r,rt(i)}$ that proves this case.

It remains to consider case $ri' \leq i'$. Then $i < ri < ri' \leq i'$. Let $rt_1$ is concatenation of $rt'[i-1, ..., i' - 1]$ and $rt''[i', ...]$. By Property 1 the $rt_1$ is path. By Note 1 the $rt_1$ is RP path. Since $lt(k) < rt(k) \leq rt''(k)$ for each $k = i, ..., i' - 1$ and $lt(i-1) < j = rt''(i-1)$ then using Note 3 the $lt$ don’t intersects with $rt_1$.

Since $ri < ri' \leq i'$ then $rt''[i', ...] = rt[ri', ...]$ then $rt_1$ intersects $rt$ at row $i'$. Using Property 2, 3 we get $PS(rt''[i-1, ..., i' - 1]) \geq PS(rt[i-1, ..., i' - 1])$.

If $PS(rt''[i-1, ..., i' - 1]) > PS(rt[i-1, ..., i' - 1])$ then $PS(rt_1) = PS(rt''[i-1, ..., i' - 1]) + PS(rt''[i', ...]) \geq PS(rt[ri', ...]) = PS(rt)$. Then $lt$ and $rt_1$ are $LP'_{i-1,lt(i-1)}$ and $RP_{i-1,j}$ paths without intersection and with sum $PS(l_l) + PS(rt_1) > PS(l_l) + PS(rt)$ that contradict to maximum sum of $l_l$ and $rt$.

Then $PS(rt''[i-1, ..., i' - 1]) = PS(rt[i-1, ..., i' - 1])$.

Let $rt_2$ is concatenation of $rt[i-1, ..., i' - 1]$ and $rt''[i', ...]$. Since $rt''$ intersects $rt$ at row $i'$ then by Property 1 we get that $rt_2$ is path. Then $PS(rt_2) = PS(rt[i-1, ..., i' - 1]) + PS(rt''[i', ...]) = PS(rt'') = F_{i-1,j}$. Thus using Note 1 we get that $rt_2$ defined by $F_{i-1,j}$.

Since $ri < i'$ then $rt_2(ri) = rt(ri)$ and $rt_2(ri) = rt(i)$. Then using Property 2, 1 and $ri < i'$ we get $PS(rt[i-1, ..., ri]) = PS(rt_2[i-1, ..., ri]) = F_{i-1,rt_2(i)} - F_{i,rt_2(i)} + g_{ri,rt_2(i)} = F_{i-1,rt(i-1)} - F_{i,rt(i)} + g_{ri,rt(i)}$.

4. The set $\{b_1, ..., b_k\}$ are all possible columns which can be intersected at row $i$ by $LP'_{i-1,l'(i-1)}$ path $t_2$ with restriction $t_2(i) < l_p(i)$. The set $\{b_1, ..., b_k\}$ are all possible columns which can be reached at row $i$ by $RP_{i-1,j}$ path $t_2$ with restriction $t_2(i) > r_p(i)$. Since $r_p(i) + 1 \leq W - 1$, $l_p(i-1) \leq l_p(i) + 1 - 1 \geq 0$ then $b_2 \leq b_2$ and $b_3 \leq b_4$ i.e. these sets are not empty.

By Property 8.3 we get $PS(rt[i, ..., ri - 1]) = F_{i,rt(i)} - F_{i,rt(i)}$. Since $lt[i, ..., ri - 1]$ are $LP'_{i,rt(i)}$ pair and $lt(i-1) = l_p(i-1)$ then $PS(l_l[i, ..., i]) + PS(rt''[i, ..., ri - 1]) = max_k b_{k1}, ..., b_{k2} \{M[i, k]\}$. Recall that $PS(rt'[i, ..., ri - 1]) = PS(t_p[i, ..., ri - 1]) = F_{i,rt(i)}$ and $rt'(ri) = rt(ri) = r_p(ri)$. Then $PS(l_l[i, ..., i]) + PS(rt''[i, ..., ri - 1]) = F_{i,rt(i)} - F_{i,rt(i)} + PS(rt''[i, ..., ri - 1]) + PS(rt[i, ..., ri - 1]) = PS(l_l[i, ..., i]) + PS(rt'[i, ..., ri - 1]) + PS(rt[i, ..., ri - 1]) = PS(l_l[i, ..., i]) + PS(rt'[i, ..., ri - 1]) + PS(rt[i, ..., ri - 1]) = PS(l_l[i, ..., i]) + PS(rt'[i, ..., ri - 1]) + PS(rt[i, ..., ri - 1])$. 

9
Let prove that $F_{i,rt(i)} = \max_{k=b_3, \ldots, b_4} \{F_{i,k}\}$. Since $\tau_p(i) < rt(i)$ then $rt(i) \in \{b_3, \ldots, b_4\}$ then $F_{i,rt(i)} \leq \max_{k=b_3, \ldots, b_4} \{F_{i,k}\}$.

Suppose that exists $j' \in \{b_3, \ldots, b_4\}$ such that $F_{i,j'} > F_{i,rt(i)}$. Consider $R_{p,j'}$ path $rt_2$ defined by $F_{i,j'}$. Let $rt'_2$ is concatenation of $rt'[i-1]$ and $rt_2[i, ...]$. Due to $\max(j-1, \tau_p(i)) \leq b_3 \leq b_4 \leq \min(j+1, W-1)$ then $rt'_2$ is $R_{p,1-i,j}$ path.

If $rt'(i) \leq rt_2'(i)$ for each $k \geq i$ then path and $rt'_2$ are $\text{ldtms}(i-1, l_p(i-1), j)$ pair with sum $PS(lt') + PS(rt'_2) = PS(lt) + F_{i,j'} + g_{i-1,j} \leq PS(lt) + F_{i,rt(i)} + g_{i-1,j} \geq PS(lt) + PS(rt)$. That contradict to maximum sum of $lt$ and $rt$.

Then let $i_2 \geq i$ such that $rt'(i_2) > rt'_2(i_2)$ and $rt'(k) \leq rt'_2(k)$ for each $k = i-1, \ldots, i_2-1$. Consider concatenation $rt'_2'[i-1, \ldots, i_2-1]$ and $rt'[i_2, ...]$. By Property 9 the $rt'_2'$ is path. By Note 9 the $rt'_2$ don't intersects with $lt$.

If $i_2 \geq i$ then $rt'_2(i_2) = rt'(i_2)$ then using Property 3

$$PS(lt) + PS(rt'_2) = PS(lt) + F_{i,j'} - F_{i,rt(i_2)} + PS(rt[i_2, ...]) + g_{i-1,j} =$$

$$= PS(lt) + F_{i,rt(i)} - F_{i,rt(i_2)} + PS(rt[i_2, ...]) + g_{i-1,j} \geq =$$

$$= PS(lt) + PS(rt)$$.

That contradict to maximum sum of $lt$ and $rt$.

Then $i_2 < i$. Due to $PS(rt'[i_2, ..., ri-1]) = PS(r_p[i_2, ..., ri-1]) = F_{i_2,rt(i_2)} - F_{i,rt(i)} = F_{i_2,rt(i_2)} - F_{i,rt(i)}$ we get

$$PS(lt) + PS(rt'_2) = PS(lt) + PS(rt'[i_2, ..., ri-1]) + PS(rt'[i_2, ..., ri-1]) + PS(rt'[ri, ..., ni]) =$$

=$$PS(lt) + g_{i-1,j} + F_{i,j'} - F_{i,rt(i_2)} + PS(r_p[i_2, ..., ri-1]) + PS(rt'[ri, ..., ni]) >$$

$$PS(lt) + g_{i-1,j} + F_{i,rt(i)} - F_{i,rt(i_2)} + PS(rt'[ri, ..., ni]) =$$

$$PS(lt) + g_{i-1,j} + PS(rt'[i, ..., ri-1]) + PS(rt'[ri, ..., ni]) = PS(lt) + g_{i-1,j} + PS(rt[i, ..., ri-1]) = PS(lt) + PS(rt)$$.

Thus $F_{i,rt(i)} = \max_{k=b_3, \ldots, b_4} \{F_{i,k}\}$. Then $PS(lt[i, ...]) < PS(rt'[i, ...]) + F_{i,rt(i)} - F_{i,rt(i_2)} = \max_{k=b_3, \ldots, b_4} \{M_{i}(k, j)\} + \max_{k=b_3, \ldots, b_4} \{F_{i,k}\} = F_{i,rt(i)}$.

Property 8 divided into 4 keys notes, where each next note depends on previous. Property 8 tells that $rt'[i-1, ..., ri]$ is subpath of some path defined by $F_{i,rt(i-1)}$.

Property 9 tells that, if we will swap the tails of $rt[i, ...]$ and $r_p[i, ...]$ at row $ri$, then we get $rt'$ (with head of $\tau_p$ and tail of $rt$) for which the difference between $PS(rt'[i, ...])$ and $PS(rt[i, ...])$ is the difference between $F_{i,rt(i)}$ and $F_{i,rt(i)}$.

Property 9 is main property that allows to find $M_l$ and $M_r$ in linear time, using DPM.

**Property 9.** Let $lt$ and $rt$ are $\text{ldtms}(i-1, l_p(i-1), j)$ pair. And let $\max\{0, l_p(i-1) - 1\} \leq b_1 \leq b_2 < \min\{l_p(i-1) + 1, l_p(i) - 1\} + \max\{l_p(i) - 1, l_p(i), j-1\} \leq b_3 \leq b_4 \leq \min\{j+1, W-1\}$. Then

$$PS(lt[i, ...]) + PS(rt[i, ...]) \geq \max_{k=b_1, ..., b_2} \{M_{i}(k, j)\} + \max_{k=b_3, ..., b_4} \{F_{i,k}\} - F_{i,rt(i)}.$$

**Proof.** Denote $lt[i, ...]$ as $lt^{-}$ and $rt[i, ...]$ as $rt^{-}$. And suppose that $PS(lt^{-}) + PS(rt^{-}) < M_{i}(i, k_1) + F_{i,k_2} - F_{i,rt(i)}$ for some $k_1 \in \{b_1, ..., b_2\}$ and $k_2 \in \{b_3, ..., b_4\}$.

Let $lt^{-}$ and $rt^{-}$ are $\text{ldtms}(i, k_1, r_p(i))$ pair. Then $PS(lt') + PS(rt') = M_{i}(i, k_1)$.

The set $\{b_1, ..., b_2\}$ is set of all columns which can be reached by $LP_{i-1,l_p(i)}$ path at row $i$ except column $r_p(i)$. Then concatenation $lt'^+ \cup$ of $lt[i-1]$ and $lt'$ is $LP_{i-1,l_p(i)}^{-}(i-1)$ path. Then by definition of $F_{i,rt(i-1)}$ and $l_p$ we get $F_{i,k_1} \leq F_{i,rt(i-1)}$. Since $rt'(i) = r_p(i)$ then concatenation $rt'^+$ of $r_p[i-1]$ and $rt'$ in $R_{p,1-i,j}$ path.

Consider $R_{p,k_2}$ path $rt'^+$ defined by $F_{i,k_2}(g_i) = F_{i,k_2}$. The set $\{b_3, ..., b_4\}$ is set of all columns which can be reached by $RP_{i-1,j}$ path at row $i$ except column $l_p(i)$. Then concatenation $rt'^+$ of $rt'[i-1]$ and $rt'$ in $R_{p,1-i,j}$ path. By definition $PS(lt) + PS(rt) = M_l(i-1, j) \geq PS(lt'^+) + PS(rt'^+)$. Then $PS(lt'^+) + PS(rt'^+) \geq F_{i,k_1} + F_{i,k_2} \geq PS(lt') + F_{i,k_2}$. Since $rt'^+(i-1) = r_p(i-1)$ then $PS(rt'^+) \leq F_{i,rt(i-1)}$ then $PS(lt'^+) \leq F_{i,rt(i-1)}$.

Thus $PS(lt) + PS(rt) = M_l(i-1, j) \geq PS(lt'^+) + PS(rt'^+) \geq F_{i,k_1} + F_{i,k_2} \geq PS(lt') + F_{i,k_2}$. Then $PS(lt') + PS(rt') \geq F_{i,k_1} + F_{i,k_2} \geq PS(lt') + F_{i,k_2}$. Since $rt'^+(i-1) = r_p(i-1)$ then $PS(rt'^+) \leq F_{i,rt(i-1)}$.

Thus $PS(lt) + PS(rt) = M_l(i-1, j) \geq PS(lt'^+) + PS(rt'^+) \geq F_{i,k_1} + F_{i,k_2} \geq PS(lt') + F_{i,k_2}$. Then $PS(rt') \leq F_{i,rt(i-1)}$.

Thus $PS(lt) + PS(rt) = M_l(i-1, j) \geq PS(lt'^+) + PS(rt'^+) \geq F_{i,k_1} + F_{i,k_2} \geq PS(lt') + F_{i,k_2}$. Then $PS(rt') \leq F_{i,rt(i-1)}$.

Thus $PS(lt) + PS(rt) \geq M_l(i-1, j) + F_{i,k_2} - F_{i,rt(i-1)}$. That contradicts to our assumption.
Then exists $r_i \geq i - 1$ such that $r_i''(r_i') = r_{p_1}(r_i')$. Then, due to $r_{p_1}(i) = r_i'(i)$, exists $i' \in \{i, ..., r_i\}$ such that $r_i''(i') \geq r_i''(i')$. WLOG we can assume that $r_i''(k) < r_i''(k)$ for each $k = i, ..., i' - 1$ when $i' > i$. Then $l't'$ don't intersect $rt'[i', i' - 1]$. If $i' = i$ then assume that $r_i''(i', i' - 1]$ and $rt'[i', i' - 1]$ is empty paths.

Consider concatenation of $rt'[i', i' - 1]$ and $r_i''(i', i' - 1]$). By Property 1 of the $rt_1$ is path. By Note 3 the $rt_1[i, ...]$ is $RP_{i, k_1}$ path. By Note 3 $rt'[i, ...]$ don’t intersect $rt_1[i, ...]$

Let $rt_2$ is concatenation of $rt'[i', i' - 1]$ and $r_i''(i', i' - 1]$. By Property 1 the $rt_2$ is path. Using Property 3 we get $PS(rt_2[i, ...]) \leq PS(r_{p_1}(i), ...)] = F_{i, r_{p_1}(i)}$. Then $PS(rt_1[i, ...]) = PS(rt') + PS(r_i''(i', i' - 1]) - PS(rt_2[i, ...]) \geq PS(rt') + F_{i, k_2} - F_{i, r_{p_1}(i)}$

Thus $lt''[i, ...]$ and $rt_1[i, ...]$ are $LPP_{i, k_1}$ and $RP_{i, k_2}$ paths without intersections and with sum $PS(lt''[i, ...]) + PS(rt_1[i, ...]) \geq PS(lt') + PS(r_i''(i', i' - 1]) + F_{i, k_2} - F_{i, r_{p_1}(i)} = M_i(i, i_1) + F_{i, k_2} - F_{i, r_{p_1}(i)}$.

This contradiction proves our Property.

**Lemma 1.** Tables $M_i$ and $M_r$ can be found in $O(H \cdot W)$.

**Proof.** Before calculation of $M_i$ and $M_r$ we need to find table $F_{i, j}(g)$ for each $i, j$. This table can be found in $O(H \cdot W)$. Also, we need in $l_{p_1}$ and $r_{p_1}$, which can be found in $O(H)$.

It is enough to prove that every row of tables $M_i$ and $M_r$ can be found in $O(W)$. Let prove it by induction on $H$.

**Base case:** Let find values for last row. For last row these tables contains the sum of pair paths with length 1. Thus, any pair (with different beginning) don’t intersects between themselves.

$$M_i(H-1, j) = g_{H-1, l_{p_1}(H-1)} + g_{H-1, j} \quad \text{for each } j = \max\{l_{p_1}(H-1), l_{p_1}(H-1) + 1, ..., W-1\}.$$  
$$M_r(H-1, j) = g_{H-1, r_{p_1}(H-1)} + g_{H-1, j} \quad \text{for each } j = 0, ..., \min\{l_{p_1}(H-1), r_{p_1}(H-1) - 1\}.$$  

This calculation requires $O(W)$ time.

**Induction step:** Suppose that known $M_i$ and $M_r$ for rows $i, ..., H-1$, where $i > 0$.

Then let find $M_i$ for $(i - 1)$-th row. By Note 10 it is enough to find the $M_i(i - 1, j)$, for any $j \geq \max\{l_{p_1}(i-1) + 1, r_{p_1}(i-1)\}$.

Let $l_{p_1} and $r_{p_1}$ are $l_{r_{p_1}}(i-1, l_{p_1}(i-1), j)$ pair. Consider all possible cases and find the sum $PS(l_{r_{p_1}}[i, ...]) + PS(r_{r_{p_1}}[i, ...]):$

1. For case $l_{r_{p_1}}(i) < l_{p_1}(i)$ and $r_{r_{p_1}}(i) = r_{p_1}(i)$. Denote $PS(l_{r_{p_1}}[i, ...]) + PS(r_{r_{p_1}}[i, ...])$ for this case as $max_1(j)$. Then we get $max_1(j) = M_i(i, l_{r_{p_1}}(i))$ i.e.

$$max_1(j) = \max_{k=b_1, ..., b_2} \{M_r(i, k)\}$$

where $b_1 = \max\{l_{p_1}(i-1), 1\}$ and $b_2 = \min\{l_{p_1}(i-1) + 1, l_{p_1}(i-1), r_{p_1}(i-1)\}$. Due to rule (r1) we get $l_{p_1}(i-1) + 1 > l_{p_1}(i)$ and $l_{p_1}(i) - 1 \leq r_{p_1}(i) - 1$, then $b_2 = l_{p_1}(i)$. Note that $b_1 \leq b_2$ if $l_{p_1}(i-1), 1 \leq l_{p_1}(i)$.

Let find when restrictions of this case don’t contradict to (r1), (r3). It is enough to check for possible positions of $l_{r_{p_1}}(i)$ and $r_{r_{p_1}}(i)$.

For $r_{r_{p_1}}(i)$ we get $j - 1 \leq r_{r_{p_1}}(i) \leq j + 1$ and $r_{r_{p_1}}(i) = r_{p_1}(i)$, then sufficient conditions for $r_{p_1}(i)$ are $j - 1 \leq r_{p_1}(i) \leq j + 1$. But by proposition $j \geq r_{p_1}(i) - 1$ then by (r1) the condition $j \geq r_{p_1}(i) - 1$ is true always.

Restrictions for $l_{r_{p_1}}(i)$ are $l_{p_1}(i) - 1 = l_{r_{p_1}}(i) - 1 \leq l_{p_1}(i) < l_{p_1}(i)$ and $0 \leq l_{p_1}(i)$.

Thus we get conditions when this case need to check

$$j - 1 \leq r_{p_1}(i), \max\{l_{p_1}(i-1), 1\} \leq l_{p_1}(i).$$

Thus, in common case, we can assume

$$max_1(j) = \begin{cases} \max_{k=b_1, ..., b_2} \{M_r(i, k)\}, & \text{if } j - 1 \leq r_{p_1}(i), \max\{l_{p_1}(i-1), 1\} \leq l_{p_1}(i) \\ 0, & \text{otherwise.} \end{cases}$$

11
2. For case $IP(i) = l_p(i)$. Denote $PS(IP[i,...]) + PS(rP[i,...])$ for this case as $max_2(j)$. Then we get $max_2(j) = M_i(i, rP(i))$ i.e.

$$max_2(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\}$$

where $b_1 = \max\{j-1, r_p(i), l_p(i) + 1\}$ and $b_2 = \min\{j + 1, W - 1\}$

Note that $b_1 \leq b_2$ iif $lpath(i) + 2 \leq W$.

For $rP(i)$ we get restrictions $\max\{l_p(i) + 1, r_p(i)\} \leq rP(i) \leq W - 1$ and $j - 1 \leq rP(i) \leq j + 1$. Since always $r_p(i) \leq \min\{j + 1, W - 1\}$ and $j - 1 \leq W - 1$ then required conditions for $rP(i)$ are $lpath(i) + 1 \leq j + 1$ and $l_p(i) + 2 \leq W$. But by proposition and (r1) we get $j \geq l_p(i) + 1 \geq l_p(i)$ then we get that $l_p(i) \leq j$ is true always.

Restrictions for $IP(i)$ are $lP(i) = l_p(i)$ and $lP(i) - 1 = l_p(i) - 1$. Since $l_p[i-1,i]$ satisfy to (r1) and (r3) then this restriction always true for $lP[i-1,i]$.

Thus we get conditions for this case checking

$$l_p(i) + 2 \leq W.$$  \hspace{1cm} (2)

Thus, in common case, we can assume

$$max_2(j) = \begin{cases} \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} & \text{if } l_p(i) + 2 \leq W, \\ 0 & \text{otherwise.} \end{cases}$$

3. Consider case when $IP(i) < l_p(i)$, $rP(i) > r_p(i)$ and $j = r_p(i) - 1$.

Due to contradiction with Properties [5]1 and [5]2 this case impossible for $lrdtms(i-1,l_p(i-1),j)$ pair $IP$ and $rP$.

4. Consider case when $IP(i) < l_p(i)$, $rP(i) > r_p(i)$ and $j > r_p(i) - 1$. Denote $PS(IP[i,...]) + PS(rP[i,...])$ for this case as $max_3(j)$. Then by Property [8]4 we get

$$max_3(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_{1,k} - F_{i,r_p(i)}\}$$

where $b_1 = \max\{0, l_p(i-1) - 1\}$, $b_2 = \min\{l_p(i-1) + 1, l_p(i) - 1\} = l_p(i) - 1$ and $b_3 = \max\{r_p(i) + 1, j - 1\}$, $b_4 = \min\{j + 1, W - 1\}$.

Note that $b_1 \leq b_2$ and $b_3 \leq b_4$ iif $\max\{1, l_p(i-1)\} \leq l_p(i)$, $r_p(i) + 2 \leq W$.

This case possible only when $r_p(i) < rP(i) \leq W - 1$, $j - 1 \leq rP(i) \leq j + 1$, $r_p(i-1) < j$, $l_p(i-1) - 1 \leq IP(i) < l_p(i)$ and $0 \leq IP(i)$.

Then we get condition of $max_3(j)$ existing

$$\max\{1, l_p(i-1)\} \leq l_p(i)$, $r_p(i) + 2 \leq W$, $r_p(i-1) < j$.  \hspace{1cm} (3)

Thus, in common case, we can assume

$$max_3(j) = \begin{cases} \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_{1,k} - F_{i,r_p(i)}\} & \text{if } \max\{1, l_p(i-1)\} \leq l_p(i), \\
0 & \text{otherwise.} \end{cases}$$

Note that condition $b_1 \leq b_2$ and $b_3 \leq b_4$ follows from (3).

Thus exists $m \in \{1, 2, 3\}$ such that $PS(IP[i,...]) + PS(rP[i,...]) = max_m(j)$. Then

$$PS(IP[i,...]) + PS(rP[i,...]) \leq \max\{max_1(j), max_2(j), max_3(j)\}.$$

Since $PS(IP[i,...]) + PS(rP[i,...]) \geq 0$ then $PS(IP[i,...]) + PS(rP[i,...]) \geq max_m(j)$ when condition (m) is false for each $m \in \{1, 2, 3\}$. Since $max_1(j)$ and $max_2(j)$ is result of reducing to an existing pairs of paths with maximum sum then $PS(IP[i,...]) + PS(rP[i,...]) \geq max_m(j)$ for each $m \in \{1, 2\}$.

Since $b_1 \leq b_2$ and $b_3 \leq b_4$ in case 4 follows from condition (3) then by Property [8] we get that $PS(IP[i,...]) + PS(rP[i,...]) \geq max_3(j)$. Thus using $M_i(i-1, j) = PS(IP[i,...]) + PS(rP[i,...]) + g_{i-1,r_p(i-1)} + g_{i-1,j}$ we get

$$M_i(i-1, j) = g_{i-1,l_p(i-1)} + g_{i-1,j} + \max\{max_1(j), max_2(j), max_3(j)\}.$$

Thus in $O(1)$ we can find $M_i(i-1, j)$ for any $j \in \{\max\{l_p(i-1) + 1, r_p(i-1)\}, \ldots, W - 1\}$. Then in $O(W)$ we can find $M_i$ for row $i - 1$. Similarly in $O(W)$ we can find $M_i$ for row $i - 1$.  \hspace{1cm} \square
More exactly, this algorithm spent $O(H \cdot W)$ of comparisons and sums of numbers like $F_{i,j}$, $M_i(j, j)$, $l_p(j)$. Since values of $g$ bounded by value $C$, then these numbers have length $O(\log(H \cdot C))$.

If $C = f(H)$ is less than any polynomial function of $H$, then these values have length $o(\log(H))$ i.e. less than length of addresses to elements of input data, therefore we ignore linear operations with these values.

If $f$ is not less than some polynomial function of $H$, then complexity is $O(H \cdot W)$ of linear operations with integers of length $O(\log(f(H)))$. Thus, we have full complexity $O(H \cdot W \cdot \log(f(H)))$. But by our assumption, the length of input data is $\Theta(H \cdot W \cdot \log(f(H)))$. Thus, we got pure linear algorithm.

### 3.1.1 Simplification of $M_i$ and $M_r$ search

Here we use designations from induction step of Lemma[1].

Assume that $lP(i) < l_p(i)$. Note that pair $b_1, b_2$ of case 1 are same as pair $b_1, b_2$ of case 4. Also using restriction $rP(i) = r_p(i)$ in case 1 we get

$$\max_{k=r_p(i), \ldots, b_4} \{F_{i,k}\} = F_{i, r_p(i)}$$

for any $r_p(i) \leq b_4 \leq \min\{j + 1, W - 1\}$. Thus we can assume that $b_4$ from case 4 and

$$max_1(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=r_p(i), \ldots, b_4} \{F_{i,k}\} - F_{i, r_p(i)}.$$

Also we can extend restriction for case 4 by addition of restriction of cases 1 and 3. Let $b_3 = \max\{r_p(i), j - 1\}$. Then in case $rP(i) = r_p(i)$ we get $r_p(i) = rP(i) \geq j - 1$ then we get $b_3 = r_p(i)$ then

$$max_1(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=r_p(i), \ldots, b_4} \{F_{i,k}\} - F_{i, r_p(i)}.$$

If $rP(i) > r_p(i)$ then by case 3 we get that case $j = r_p(i - 1)$ impossible. Then $j > r_p(i - 1)$ and we get restrictions of case 4 and conditions of Property [8].

In case when $j - 1 > r_p(i)$ we get $b_3 = b_3$.

Consider case when $j - 1 \leq r_p(i)$ i.e. $b_3 = r_p(i) = b_3 - 1$. Then by Property [8] exists $r \in i$ such that $rP[i - 1, \ldots, r]$ is subpath of some $RP$ path defined by $F$ then

$$rP(i) \in \arg \max_{k=b_1, \ldots, b_4} \{F_{i,k}\}.$$ 

Since $b_3 = r_p(i) < rP(i)$ and $b_3 + 1 = b_3$ then $rP(i) \in \{b_3, \ldots, b_4\}$ then

$$\max_{k=b_3, \ldots, b_4} \{F_{i,k}\} = \max_{k=b_3, \ldots, b_4} \{F_{i,k}\}.$$ 

Thus if $rP(i) > r_p(i)$ we get

$$max_2(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_{i,k}\} - F_{i, r_p(i)}.$$

Thus, in common case, we can combine cases 1, 3 and 4 with one restriction $lP(i) < l_p(i)$ and common maximum formula

$$max_1(j) = \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_{i,k}\} - F_{i, r_p(i)}.$$

Let find conditions of $max_1'$ existing.

For $rP(i)$ we get $r_p(i) \leq rP(i) \leq W - 1$ and $j - 1 \leq rP(i) \leq j + 1$ then we get $r_p(i) \leq j + 1$. But $j = rP(i - 1) \geq rP(i - 1) \geq r_p(i) + 1$ always.

For $lP(i)$ we get $l_p(i - 1) - 1 \leq lP(i) < l_p(i)$ and $0 \leq lP(i)$. Then we get conditions of $max_1'(j)$ existing

$$\max\{1, l_p(i - 1)\} \leq l_p(i). \quad (4)$$

Thus in common case we can assume

$$max_1'(j) = \begin{cases} \max_{k=b_1, \ldots, b_2} \{M_i(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_{i,k}\} - F_{i, r_p(i)} & \text{if } 1 \leq l_p(i - 1) \\ 0 & \text{otherwise.} \end{cases}$$

Then

$$M_i(i - 1, j) = g_{i - 1, l_p(i - 1)} + g_{i - 1, j} + \max\{max_1'(j), max_2(j)\}$$

Implementation of this version search of $M_i$ and $M_r$ represented at listing 3 in function get_M using programing language Python.
3.2 Reducing problem to lrdtms(0,0,W-1) pair

**Definition 14.** Denote subset of common cells between paths \(t_1\) and \(t_2\) as \(t_1 \cap t_2\). 
Set of all cells of paths \(t_1\) and \(t_2\) as \(t_1 \cup t_2\). 
Set of all cells of path \(t_1\) without cells of path \(t_2\) as \(t_1 \setminus t_2\).

**Definition 15.** Consider paths \(lt\) and \(rt\). Let rows \(i_1\) and \(i_2\) such that \(lt(i) = rt(i)\) for each \(i = i_1 + 1, \ldots, i_2 - 1\) and either \(lt(i) < rt(i_1)\), \(lt(i_2) > rt(i_2)\) or \(lt(i_1) > rt(i_1)\), \(lt(i_2) < rt(i_2)\). Then call pair \(i_1, i_2\) as cross over pair.

**Property 10.** For any paths \(lt\) and \(rt\), with beginning from cells \((0,0)\) and \((0,W-1)\), exists paths \(lt'\) and \(rt'\) with beginning from \((0,0)\) and \((0,W-1)\) respectively, with \(lt \cup rt = lt' \cup rt'\) (as corollary with same common sum i.e. \(PS(lt \cup rt) = PS(lt' \cup rt')\)), and inequality \(lt'(i) \leq rt'(i)\) for each \(i\).

*Proof.* WLOG suppose that \(lt\) and \(rt\) have minimum cross over pairs from all paths \(lt'\) and \(rt'\) starts from \((0,0)\) and \((0,W-1)\) respectively with same common sum (equal to \(N\)), and \(lt \cup rt = lt' \cup rt'\). And suppose that between \(lt\) and \(rt\) exists cross over pair.
Then, using Property 10, we can reduce number of cross over pairs by swapping tails of \(lt\) and \(rt\). Since swapping don’t changes the set of cells of paths then we get \(lt \cup rt = lt' \cup rt'\). Thus we get contradiction with minimum cross over pairs between \(lt\) and \(rt\).

Thus we get \(lt(i) \leq rt(i)\) for each \(i\).

**Property 11.** Suppose that our grid \(g\) with no negative values. Consider paths \(lt\) and \(rt\) with beginning from \((0,0)\) and \((0,W-1)\), and \(lt(i) \leq rt(i)\) for each \(i\).

Then exists paths \(lt'\) and \(rt'\) with beginning from \((0,0)\) and \((0,W-1)\) respectively, such that \(lt'(i) < rt'(i)\) for each \(i\) (i.e. \(lt'\) don’t intersects with \(rt'\), and \(PS(lt') + PS(rt') \geq PS(lt) + PS(rt) - PS(lt \cap rt)\)).

*Proof.* Denote \(PS(lt) + PS(rt) - PS(lt \cap rt)\) as \(N\). WLOG assume that \(lt\) and \(rt\) have minimum common cells among all paths starts from \((0,0)\) and \((0,W-1)\) cells, and with common sum equal to \(N\) or greater (i.e. \(PS(lt) + PS(rt) - PS(lt \cap rt) \geq N\)). And suppose that row \(i_1\) such that \(lt(i_1) = rt(i_1)\) and \(lt(i) < rt(i)\) for each \(i < i_1\). Denote \(lt(i_1)\) as \(j_1\).
Consider case when \(lt(i_1 - 1) < j_1\).
Due to rule of moving \((r1)\), after \(k\) steps from cell \((i_1,j_1)\) left and right robots will be located on cells \((i_1 + k, j')\) and \((i_1 + k, j'')\) respectively, for some \(j', j'' \leq j_1 + k\). I.e. \(lt(i_1 + k) \leq rt(i_1 + k) \leq j_1 + k\).

Consider cases:
- Suppose that not all moves of left robot are rightmost after row \(i_1\). I.e. exists \(i' > i_1\) such that \(lt(i_1 + i - 1) \geq i(i_1 - 1)\) for each \(i = i_1, \ldots, i' - 1\) and \(lt(i') - j_1 < (i' - i_1)\),

Then \(j_1 + 1 + i - i_1 \leq lt(i_1 + i - i_1) \leq rt(i_1 + i - i_1) \leq j_1 + i - i_1\) for each \(i = i_1, \ldots, i' - 1\). I.e. \(lt[i_1, \ldots, i' - 1] = lt[i_1, \ldots, i' - 1]\).

Consider concatenation \(lmp'\) such that:
- \(lmp'[1, \ldots, i_1 - 1] = lt[1, \ldots, i_1 - 1]\),
- \(lmp'[i_1 + k] = j_1 + 1 + k, k = 0, \ldots, i' - 1 - i_1\),
- \(lmp'[i', \ldots] = lt[i', \ldots]\).

Then \(lmp'[\ldots, i_1 - 1]\) and \(lmp'[i', \ldots]\) are subpaths. Also, \(lmp'[\ldots, i' - 1]\) is subpath with rightmost moves.

Let prove that moves from \(lmp'(i_1 - 1)\) to \(lmp'(i_1)\) and from \(lmp'(i' - 1)\) to \(lmp'(i')\) are corresponds to move rules.

Using rules of move for \(lt\) we get \(lmp'(i_1 - 1) = lt(i_1 - 1) \geq lt(i_1) - 1 = lmp'(i_1)\). The other side \(lmp'(i_1) = lt(i_1) - 1 > lt(i_1) - 1 = lmp'(i_1 - 1)\). I.e. \(lmp'(i_1) = lmp'(i_1 - 1)\).
Thus move from \(lmp'(i_1 - 1)\) to \(lmp'(i_1)\) is correct (i.e. corresponds to moving rules).

By assumption \(lmp'(i') - j_1 < (i' - i_1)\) we get \(j_1 > lt(i') - (i' - i_1)\). Then for \(k = i' - 1 - i_1\) we get \(lmp'(i' - 1) = lmp'(i_1 + k) = j_1 + 1 + k = lt(i') - 2 = lmp'(i') - 2\). I.e. \(lmp'(i' - 1) = lmp'(i')\).

By assumption \(lt(i' - 1) - j_1 \geq (i' - 1 - i_1)\) we get \(j_1 \leq lt(i' - 1) - (i' - 1 - i_1)\). Then for \(k = i' - 1 - i_1\) we get \(lmp'(i' - 1) = j_1 + k - 1 \leq lt(i' - 1) - 1 \leq lt(i').\)
Proof. Let \( i \geq i_1, \ldots, i' \). Then, using assumption \( \ell(i) - j_1 \geq (i - i_1) \) for each \( i = i_1, \ldots, i' \) and \( \ell(i) \geq (i - i_1) + \ell(i') \) for each \( i = i_1, \ldots, i' - 1 \). Since \( g \) consists of nonnegative values, then \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \). Then \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) is correct too. Thus \( \mathit{PS} \) is path.

By definition \( \ell(i) = i - 1 + (i - i_1) \) for each \( i = i_1, \ldots, i' - 1 \). Then, using assumption \( \ell(i) \geq j_1 \) for each \( i = i_1, \ldots, i' - 1 \) and \( \ell(i) \geq (i - i_1) + \ell(i') \) for each \( i = i_1, \ldots, i' - 1 \). Denote \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) and \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) as \( d \) and \( d' \) respectively.

Since \( \ell(i) \neq \ell(i') \) for each \( i = i_1, \ldots, i' - 1 \), then \( d' = d - \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \). Since \( g \) consists of nonnegative values, then \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) is correct too. Thus \( \mathit{PS} \) is path.

Then \( \ell(i) = \ell(i_1 - 1) < j_1 \leq j_1 + i - i_1 = \ell(i) \) for each \( i \geq i_1 \).

Denote \( \mathit{PS}(\ell[i] - 1) \) and \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) as \( d \) and \( d' \) respectively.

Since \( \ell(i) < \ell(i') \) for each \( i \geq i_1 \), then \( d' = d - \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \). Since \( g \) consists of nonnegative values, then \( \mathit{PS}(\ell[i_1, \ldots, i'] - 1) \) is correct too. Thus \( \mathit{PS} \) is path.

Then \( \ell(i_1 - 1) = \ell(i_1 - 1) \) for each \( i \geq i_1 \).

Thus, like in previous case, we get contradiction with minimum of common cells between \( \ell \) and \( rt \).

It remains to consider case when \( \ell(i_1 - 1) \geq j_1 \). Then \( \ell(i_1 - 1) > \ell(i_1 - 1) \geq j_1 = \ell(i_1) \) and, due to symmetry, this case lead us to contradiction like in previous case.

**Property 12.** Consider paths \( \ell \) and \( rt \) with beginning from \( (0, 0) \) and \( (0, W - 1) \) respectively, and \( \ell(i) < \ell(i') \) for each \( i \). Then exists \( \ell(P, 0) \) and \( \ell(P, 0, W - 1) \) paths \( \ell' \) and \( rt' \) respectively such that \( \ell'(i) < \ell'(i') \) for each \( i \), and \( \mathit{PS}(\ell') + \mathit{PS}(rt') \geq \mathit{PS}(\ell) + \mathit{PS}(rt) \).

**Proof.** Denote \( \mathit{PS}(\ell) + \mathit{PS}(rt) \) as \( N \). WLOG we can assume that \( \ell(i) \) has minimum amount of rows \( i \) such that \( \ell(i) > l_0(i) \) among all paths \( \ell' \) with beginning on \( (0, 0) \) without intersections with \( rt \), and with sum \( \mathit{PS}(\ell') + \mathit{PS}(rt') \geq N \).

Suppose that \( \ell(i) \) isn't \( \ell(P, 0) \) path. Then exists row \( i_1 \) such that \( \ell(i) \leq l_0(i) \) for each \( i < i_1 \) and \( \ell(i_1) > l_0(i_1) \) (i.e. \( i_1 > 0 \)). Then consider cases:

- If exists \( i_2 > i_1 \) such that \( \ell(i) > l_0(i) \) for each \( i = i_1, \ldots, i_2 - 1 \) and \( \ell(i_2) \leq l_0(i_2) \).

  Then consider concatenation \( t_1: t_1[i_1, i_1 - 1] = \ell[i_1, i_1 - 1], t_1[i_1, \ldots, i_1 - 1] = l_0[i_1, \ldots, i_1 - 1], t_1[i_1, \ldots, i_2 - 1] = l_0[i_1, \ldots, i_2 - 1] \).

  And concatenation \( \ell(i) = \ell(i_2, \ldots, i_2 - 1) = \ell(i_2, \ldots, i_2 - 1), l_0(i_2, \ldots, i_2) = l_0[i_2, \ldots, i_2] \).

  Due to Property the \( t_1 \) is path. Then due to Property the \( \ell(i) \) is path too.
Thus we get path $lmp'$:
\[
\begin{align*}
lmp'[\ldots, i_1 - 1] &= lt[\ldots, i_1 - 1], \\
lmp'[i_1, \ldots, 2_1 - 1] &= l_p[i_1, \ldots, i_2 - 1], \\
lmp'[i_2, \ldots] &= lt[i_2, \ldots].
\end{align*}
\]
Similarly we can prove that concatenation $t_2$:
\[
\begin{align*}
t_2[\ldots, i_1 - 1] &= l_p[\ldots, i_1 - 1], \\
t_2[i_1, \ldots, 2_1 - 1] &= lt[i_1, \ldots, i_2 - 1], \\
t_2[i_2, \ldots] &= l_p[i_2, \ldots].
\end{align*}
\]
is path too.

Due to $l_p$ defined by $F_{0,0}$ and $t_2$ is path with begining on $(0,0)$
\[
\operatorname{PS}(lt[i_1, \ldots, i_2 - 1]) = \operatorname{PS}(t_2) - \operatorname{PS}(l_p[\ldots, i_1 - 1]) - \operatorname{PS}(l_p[i_2, \ldots])
\leq \operatorname{PS}(l_p) - \operatorname{PS}(l_p[\ldots, i_1 - 1]) - \operatorname{PS}(l_p[i_2, \ldots]) = \operatorname{PS}(l_p[i_1, \ldots, i_2 - 1]).
\]
Then
\[
\operatorname{PS}(lmp') = \operatorname{PS}(lt[\ldots, i_1 - 1]) + \operatorname{PS}(l_p[i_1, \ldots, i_2 - 1]) + \operatorname{PS}(lt[i_2, \ldots]) \geq
\geq \operatorname{PS}(lt[\ldots, i_1 - 1]) + \operatorname{PS}(lt[i_1, \ldots, i_2 - 1]) + \operatorname{PS}(lt[i_2, \ldots]) = \operatorname{PS}(lt).
\]
Since $l_p(i) \leq lt(i)$ for each $i = i_1, \ldots, i_2 - 1$, then $lmp'(i) \leq lt(i) < rt(i)$ for each $i$.

Thus we get path $lmp'$ without intersections with $rt$ and $\operatorname{PS}(lmp') + \operatorname{PS}(rt) \geq \operatorname{PS}(lt) + \operatorname{PS}(rt) = N$.

But $lmp'$ has less rows $i$ such that $lmp'(i) > l_p(i)$ which contradicts to minimum of these rows in $lt$. Thus $lt$ is $LP_{0,0}$ path.

- $lt(i) > lpath(i)$ for each $i \geq i_1$.

Then consider concatenations $lmp'$ and $t_2$:
\[
\begin{align*}
lmp'[\ldots, i_1 - 1] &= lt[\ldots, i_1 - 1], \\
lmp'[i_1, \ldots] &= l_p[i_1, \ldots], \\
t_2[\ldots, i_1 - 1] &= lpath[\ldots, i_1 - 1], \\
t_2[i_1, \ldots] &= lt[i_1, \ldots].
\end{align*}
\]

Due to Property $\Box$ the $lmp'$ and $t_2$ are paths.

Due to $l_p$ defined by $F_{0,0}$ and $t_2$ is path with begining on $(0,0)$
\[
\operatorname{PS}(lt[i_1, \ldots]) = \operatorname{PS}(t_2) - \operatorname{PS}(l_p[i_1, \ldots, i_2 - 1]) \leq
\leq \operatorname{PS}(l_p) - \operatorname{PS}(l_p[\ldots, i_1 - 1]) = \operatorname{PS}(l_p[i_1, \ldots])
\]
Then
\[
\operatorname{PS}(lmp') = \operatorname{PS}(lt[\ldots, i_1 - 1]) + \operatorname{PS}(l_p[i_1, \ldots]) \geq
\geq \operatorname{PS}(lt[\ldots, i_1 - 1]) + \operatorname{PS}(lt[i_1, \ldots]) = \operatorname{PS}(lt).
\]
Since $l_p(i) \leq lt(i)$ for each $i \geq i_1$, then $lmp'(i) \leq lt(i) < rt(i)$ for each $i$.

Thus we get path $lmp'$ without intersections with $rt$ and $\operatorname{PS}(lmp') + \operatorname{PS}(rt) \geq \operatorname{PS}(lt) + \operatorname{PS}(rt) = N$.

But $lmp'$ has less rows $i$ such that $lmp'(i) > l_p(i)$ which contradicts to minimum of these rows in $lt$. Thus $lt$ is $LP_{0,0}$ path.

Similarly we can prove that $rt$ is $RP_{0,W-1}$ path.

**Definition 16.** Consider pair of paths $l$ and $r$, with intersection in $i$-th row. Assume that there are no paths $l'$ and $r'$ such that they contains all cells of $l$ and $r$, but without intersections at $i$-th row (i.e. $(l \cup r) \subseteq (l' \cup r')$, and all cells $(l' \cup r') \setminus (l \cup r)$ with nonnegative values. Then call paths $l$ and $r$ as $l(i,i)$-linked pair. And call cell $(i,j)$ as bottleneck if there are $(i,j)$-linked pair.

**Lemma 2.** Let $N$ is maximum number of cherries which can be collected by 2 robots with begining on $(0,0)$ and $(0, W-1)$ cells. If is true at least one of next conditions:

1. All values of grid $g$ is nonnegative.
2. The $g$ don't has bottlenecks.

then any LRTMS$(0,0, W-1)$ pair $lt$ and $rt$ have $\operatorname{PS}(lt) + \operatorname{PS}(rt) = N$.

**Proof.** Let paths $lmp$ and $rmp$ starts from $(0,0)$ and $(0, W-1)$ cells respectively and pickups maximum cherries. I.e. $\operatorname{PS}(lmp) + \operatorname{PS}(rmp) - \operatorname{PS}(lmp \cap rmp) = N$.

By Property $\Box$ we can assume that $lmp(i) \leq rmp(i)$ for each $i$.

1. Suppose that all values of $g$ is nonnegative.

   Then by Property $\Box$ we can assume that $lmp(i) < rmp(i)$ for each $i$.
2. Suppose that $g$ don’t has bottlenecks. WLOG assume that $lt$ and $rt$ have minimum of intersections among all pairs of paths with begining from $(0,0)$ and $(0, W-1)$, and common sum equal to $N$ or grater.
Suppose that $lt$ intersects with $rt$.
Since grid $g$ don’t has bottlenecks, then exists paths $lt'$ and $rt'$ such that $(lt' \cup rt') \subseteq (lt' \cup rt')$, and cells $(lt' \cup rt') \setminus (lt \cup rt)$ without negative values.
By Property $[13]$ exists $lt''$ and $rt''$ started from $(0,0)$ and $(0, W-1)$ without cross over pairs, and $(lt'' \cup rt'') = (lt'' \cup rt')$.
Thus we get paths $lt''$ and $rt''$ started from $(0,0)$ and $(0, W-1)$ such that $lt''(i) \leq rt''(i)$ for each $i$, and with common sum $PS((lt'' \cup rt'') = PS((lt' \cup rt') + PS((lt' \cup rt') \setminus (lt \cup rt)) \geq PS(lt \cup rt) = N$.
But $lt''$ and $rt''$ have less intersections than $lt$ and $rt$, that contradicts with our assumption.
Thus $lt$ don’t intersects with $rt$. Then $lt(i) < rt(i)$ for each $i$. Then, as in previous case, we can assume that $lmp(i) < rmp(i)$ for each $i$.
Then by Property $[12]$ exists $LP_{0,0}$ and $RP_{0,W-1}$ paths $lmp'$ and $rmp'$ respectively without intersections, and $PS(lmp') + PS(rmp') = N$. Since $N$ is upper bound for collected cherries by any pair of $LP_{0,0}$ and $RP_{0,W-1}$ paths then $lmp'$ and $rmp'$ are $lrdtms(0,0, W-1)$ pair.
Due to uniqueness of maximum, all $lrdtms(0,0, W-1)$ pairs have same sum i.e. $N$.

4 Linear solution

**Theorem 1.** The “Cherry Pickup II” problem has a linear solution.

**Proof.** Since count of cherries in cells are nonnegative values, then all values of $g$ are nonnegative. According to Lemma $[2]$ and start positions of robots it is enough to find the sum of any $lrdtms(0,0, W-1)$ pair in grid with nonnegative values. According to definitions of $M_l$ and $M_r$ this sum is equal to $M_l(0, W-1)$ and $M_r(0,0)$. According to Lemma $[1]$ we can find the tables $M_l$ and $M_r$ in $O(H \cdot W)$.

Algorithm implementation in Python showed in listings below. Finding $F$ showed in listing 1, for $l_p$ and $r_p$ in listing 2, for $M_l$ and $M_r$ in listing 3. Main function with solution in listing 4.

**Theorem 2.** If there are negative values in $g$, but there are no bottlenecks, then problem can be solved by finding maximum sum of two node-disjoint paths on $g$.

**Proof.** Since $g$ don’t has bottlenecks, then according to Lemma $[2]$ and start positions of robots it is enough to find the sum of any $lrdtms(0,0, W-1)$ pair. According to definitions of $M_l$ and $M_r$ this sum is equal to $M_l(0, W-1)$ and $M_r(0,0)$. According to Lemma $[1]$ we can find the tables $M_l$ and $M_r$ in $O(H \cdot W)$.

4.1 Reducing of DM to finding maximum sum of two node-DP

**Problem description:**
Given a grid $g_{DM}$ of size $N \times N$ with values in cells 0, 1 and $−1$:  
0 means there is no diamond, but you can go through this cell; 
1 means the diamond (i.e. you can go through this cell and pick up the diamond); 
$−1$ means that you can’t go through this cell.
We start at cell $(0,0)$ and reach the last cell $(N−1,N−1)$, and then return back to $(0,0)$ collecting maximum number of diamonds: 
Going to last cell we can move only right and down; 
Going back we can move only left and up.

**Solution:**
Let $g_1$, $g_2$ and $g_3$ are grids of size $(2N−1)\times (2N−1)$. And $g_4$ is grid of size $(3N−2)\times (2N−1)$. Denote $N−1$ as $n$. Then $g_{DM}$ can be reduced to our LS of CP2 (without proof of correctness):
1. Check matrix for reachability by 1 robot. If not richable then return 0.
2. Turn matrix clockwise by $45^\circ$. I.e. for each $i = 0, ..., n, j = 0, ..., n$ 
   \[ g_1[i+j][n+i−j] = g_{DM}[i][j]. \]
3. Add cells between horizontally neighboring cells. Also add under upper cells, except \((0, n)\), by one cell. Fill cell by \(-10N\) if bottom neighbor is \(-1\), or both horizontally neighboring cells are \(-1\). Otherwise, fill by 0. I.e. for each \(i = 0, \ldots, n, j = 0, \ldots, n\) where \(i + j \geq 1\)

\[
g_1[i+j-1][n+i-j] = \begin{cases} -10N & i = 0 \text{ and } j < n \\
0 & \text{otherwise}. 
\end{cases}
\]

4. Add corners, and fill them by \(-10N\), except top and bottom rows. Fill unvalued cells by 0. I.e. for each \(i = 0, \ldots, 2n, j = 0, \ldots, 2n\)

\[
g_2[i][j] = \begin{cases} -10N & 0 < i < 2n \text{ and } (i+j < n \text{ or } i+j > 3n \text{ or } i+n < j \text{ or } i > j+n), \\
0 & i = 0 \text{ and } (j < n-1 \text{ or } j > n+1), \\
0 & i = 2n \text{ and } j \neq n, \\
g_1[i][j] & \text{otherwise}. 
\end{cases}
\]

5. Change values \(-1\) by \(-10N\). I.e. for each \(i = 0, \ldots, 2n, j = 0, \ldots, 2n\)

\[
g_3[i][j] = \begin{cases} -10N & g_2[i][j] = -1, \\
g_2[i][j] & \text{otherwise}. 
\end{cases}
\]

6. Add on top the matrix of size \(n \times (2n + 1)\) filled by 0. I.e. for each \(i = 0, \ldots, 3n, j = 0, \ldots, 2n\)

\[
g_4[i][j] = \begin{cases} 0 & i < n, \\
g_3[i-n][j] & i \geq n. 
\end{cases}
\]

7. Apply our LS of CP2 for grid \(g_4\) and return answer.

Since our algorithm looking for paths without intersections, therefore by instruction3 we make double "road" with zero-sum for every reachable path to avoid bottlenecks. Therefore, after instruction3 due to Theorem2 we can get answer by applying our LS to \(g_4\).

First instruction can be checked by linear time using BFS. Instructions2 - 6 are linear transformations. And last instruction has linear complexity.

More exactly this reducing used linear operations with values at most \(O(N^2)\). I.e. these values have lengths \(O(\log(N))\) same as lengths of addresses to rows. Therefore, we ignore these operations for complexity estimation.

### 4.2 Some optimisation

**Definition 17.** Let \((fi, fj)\) is cell of first (least by rows) intersection of \(lp\) with \(rp\).

**Definition 18.** Let \(lP\max\) and \(rP\max\) are \(lrdtms(0, 0, W-1)\) pair.

**Property 13.** Either \(lP\max[0, \ldots, fi] = lp[0, \ldots, fi]\) or \(rP\max[0, \ldots, fi] = rp[0, \ldots, fi]\).

**Proof.** Suppose that one of these paths don’t passes through intersection of \(lp\) and \(rp\). WLOG let it be \(rP\max\). Then \(rP\max\) don’t intersect \(lp\). Then, due to Property5, we get \(lP\max = lp\). I.e. \(lP\max[0, \ldots, fi] = lp[0, \ldots, fi]\).

It remains to consider when \(lP\max\) intersect \(rp\) in some \(i_1\)-th row and \(rP\max\) intersect \(lp\) in some \(i_2\)-th row. By Note2 \(fi \leq \min\{i_1, i_2\}\). WLOG let \(i_1 < i_2\), then due to Property2 we get \(lP\max[0, \ldots, i_1] = lp[0, \ldots, i_1]\). Since \(fi \leq i_1\), then \(lP\max[0, \ldots, fi] = lp[0, \ldots, fi]\).

Using Lemma2 it is enough to find \(lrdtms(0, 0, W-1)\) pair \(lP\max\) and \(rP\max\). Also, due to Property13 either \(lP\max(fi) = fj\) or \(rP\max(fi) = fj\).

WLOG let \(lP\max(fi) = fj\). Let \(maxPath(i, j)\) is path \(p\) from \((0, W-1)\) to \((i, j)\) with maximum sum. Then, using Property13 it is enough to find maximum of

\[
PS(lp[\ldots, fi-1]) + PS(lp_j) + PS(rp_j) + maxPath(fi, j) - g_{fi,j}
\]

for each \(j > fj\), where \(lp_j\) and \(rp_j\) are \(lrdtms(fi, fj, j)\) pair.

Sum of \(lrdtms(fi, fj, j)\) pair equal to \(M_b(fi, j)\). For calculation of \(maxPath(fi, j)\) for each \(j > fj\) let consider next tables
Theorem 1. Let $tg$ is grid:

$$tg_{i,j} = \begin{cases} \infty & i \geq fi \text{ or } j < W - 1 - i, \\ g_{i,j} & i < fi \text{ and } (j \leq i \text{ or } j \geq W - 1 - i). \end{cases}$$

Definition 20. For $j = 0, \ldots, W - 1$ the $udF_{i,j}(g')$ is table defined under grid $g'$ as:

$$udF_{i,j}(g') = \begin{cases} g_{i,j}' & i = 0, \\ g_{i,j}' + \max \{udF_{i-1,j-1}(g'), udF_{i-1,j}(g'), udF_{i-1,j+1}(g')\} & i = 1, \ldots, H - 1. \end{cases}$$

Similarly to $F$ the $udF$ allows to find the path with maximum sum. For $j < f_j$ the $udF_{f_i,j}(tg)$ gives sum of path with maximum sum between cells $(f_i, j)$ and $(0, 0)$. And for $j > f_j$ the $udF_{f_i,j}(tg)$ gives maximum sum of path between $(f_i, j)$ and $(0, W - 1)$. Thus $maxPath(f_i,j) = udF_{f_i,j}(tg)$ for any $j > f_j$.

Then, for our task we can find

$$lMax = \max_{j = f_j+1, \ldots, W-1} \{M_0(f_i,j) + udF_{f_i,j}(tg) - g_{f_i,j}\} + F_{0,0} - F_{f_i - 1, f_j},$$

$$rMax = \max_{j = 0, \ldots, f_j - 1} \{M_r(f_i,j) + udF_{f_i,j}(tg) - g_{f_i,j}\} + F_{0,W-1} - F_{f_i - 1, f_j}.$$

Then $max\{lMax, rMax\}$ is required answer.

### 4.3 Linear solutions for some extensions

Let $0 \leq d_i < W$ for each $i > 0$. Then rule (r1) can be extended as (r'1). From cell $(i-1,j)$ robots can move to cell $(i,j-d_i)$, $(i,j+d_i)$, $(i-1,j+1)$, ... or $(i,j+d_i)$.

Note that all Properties, Lemmas and Theorems can be generalized for extended rule (r'1). Therefore further we assume that it is true.

The length of input data is the length of grid plus the length of vector $d$. Thus, the length of input data is $O(H \cdot W)$. Let prove that there are LS i.e. with complexity $O(H \cdot W)$.

Let $SWM_{v,w}(j) = \max\{v(j-w), \ldots, v(j+w)\}$ where $v$ is vector. $SWM_{v,w}$ is sliding window maximum (SWM) with window size $2w + 1$. The SWM is well known structure in programming, and can be defined as array of maximums of each subarray of size $2w + 1$ in $v$. SWM has $O(|v|)$ complexity. I.e. array $SWM_{v,w}$ can be prepared in $O(|v|)$, and (after preparing) the value $SWM_{v,w}(j)$ can be obtained in $O(1)$ for each $j$ (as in [LS]). Then $F$ can be extended as

$$F_{i,j} = \begin{cases} 0 & i = H, \\ g_{i,j} & i = H - 1, \\ g_{i,j} + SWM_{R_{i+1,F_i+d_i+1}}(j) & i = 0, \ldots, H - 2 \end{cases}$$

where $R_{i,F}$ is vector of length $W + 2d_i$ such that

$$R_{i,F}(j) = \begin{cases} 0 & -d_i \leq j < 0 \text{ or } W \leq j < W + d_i, \\ F_{i,j} & 0 \leq j < W. \end{cases}$$

Then each row for $F, R$ and $SWM$ can be found sequentially: the first $F_{H,*}$, then $F_{H-1,*} \rightarrow R_{H-1,*} \rightarrow SWM_{R_{H-1,F,H-1}} \rightarrow F_{H-2,*} \rightarrow \cdots \rightarrow R_{1,F} \rightarrow SWM_{R_{1,F,d_1}} \rightarrow F_{0,*}.$

Since $SWM_{R_{i,F},d_i}$ can be found in $O(W)$ for each $i$, then table $F$ can be found in $O(H \cdot W)$.

Let prove that $M_0$ and $M_r$ can be found in $O(H \cdot W)$.

Assume that $i,j, max_1, max_2$ and $max_3$ are designations from induction step of Lemma[1]

Let $b'_1 = \max\{b_1(i-1)-d_i, 0\}$ and $b_2 = \lfloor p(i) \rfloor - 1$.

Let $b_i' = \max\{j - d_i, b_2(i) + 1\}$ and $b_i'' = \min\{j + d_i, W - 1\}$.

And let $b_1 = \max\{0, b_1(i-1)-d_i\}$, $b_2 = \lfloor p(i) \rfloor - 1 = b_2$ and $b_3 = \max\{\lfloor p(i) \rfloor + 1, j - d_i\}$, $b_4 = \min\{j + d_i, W - 1\}$.

I.e. $b'_1, b_2$ are extended $b_1, b_2$ from case 1 of Lemma[1] $b'_1, b''_1$ are extended $b_1, b_2$ from case 2, and $b_1, b_2, b_3, b_4$ are extended $b_1, b_2, b_3, b_4$ from case 4.

$max_1(j), max_2(j)$ and $max_3(j)$ can be found in $O(1)$ using precalculated the SWM with window size $2d_i + 1$ for $i$-th row of $M_0, M_0$ and $F.$

19
Let $M_t$ is vector defined between positions $b_i' - d_i$ and $W + d_i$ such that $M_t(k) = M_t(i, k)$ for each $b_i' \leq k < W$, and $M_t(k) = 0$ for each $b_i' - d_i \leq k < b_i'$ and $W \leq k \leq W + d_i$. Then

$$\text{max}_2(j) = \max_{k=b_1', \ldots, b_2'} \{M_t(i, k)\} = SWM_{M_t, d_t}(j).$$

Let $M_{ri} = \max_{k=b_1', \ldots, b_2'} \{M_t(i, k)\}$ i.e. $\text{max}_1(j) = M_{ri}$ independ on $j$.

Let $F_t$ is vector defined between positions $b_3 - d_i$ and $W + d_i$ such that $F_t(k) = F(k)$ for each $b_3 \leq k < W$, and $F_t(k) = 0$ for each $b_3 - d_i \leq k < b_3$ and $W \leq k \leq W + d_i$. Then

$$\text{max}_3(j) = \max_{k=b_3, \ldots, b_4} \{M_t(i, k)\} + \max_{k=b_3, \ldots, b_4} \{F_t(k)\} = M_{ri} + SWM_{F_t, d_t}(j) - F_t, r_t(i).$$

I.e. $\text{max}_1(j), \text{max}_2(j)$ and $\text{max}_3(j)$ can be found in $O(1)$ with prepared $SWM_{M_t, d_t}, M_{ri}$ and $SWM_{F_t, d_t}$ for each $j$.

The $M_{ri}$ can be found in $O(W)$ and doesn’t depend on $j$. I.e. $M_{ri}$ can be represented as structure with $O(W), O(1)$ complexity. The $SWM$ can be found for $M_{ri}$ and $F_t$ with window $2d_i + 1$ in $O(W + 2d_i) = O(W)$ for any row. I.e. $SWM_{F_t, d_t}$ and $SWM_{M_t, d_t}$ are structures with $O(W), O(1)$ complexity.

Thus every row of $M_t$ and $M_t$ can be found in $O(W)$. I.e. this extension can be solved in $O(H \cdot W)$ i.e. has linear solution.

And another natural extension of CP2 we formulate as:

**Conjecture 1.** Let $n > 0$ and $W \geq n$. And let there are $n$ robots located on different cells in the top row of $g$, which moves by rules (r1), (r2) and (r3) to bottom row. Then exists an algorithm for finding the maximum number of cherries, that can be collected by these robots, in $O(H \cdot W \cdot 2^n)$.

For $n = 1$ using $F_{0,j}(g)$ we get a proof of this Conjecture immediately for robot at $j$-th column.

For $n = 2$ let robots starts from $j_1$ and $j_2$ columns where $j_1 < j_2$. Consider 2 cases:

1. When $j_2 - j_1 > 2H$ then any paths of robots don’t intersect with each other. Then this case can be reduced to sum of 2 independent solutions for $n = 1$.

2. $j_2 - j_1 \leq 2H$ then all reachable columns by these robots in interval from $j_1 - H$ to $j_2 + H$. Then we can get subgrid of size $H \times (4H)$ contains this interval of all reachable columns. Let denote this subgrid as $g_d$. Let $g_n$ is grid of size $(2H) \times (4H)$ with zeros. Then let $g'$ obtained by attaching the $g_n$ under the $g_d$. Thus, we get $g'$ of size $(3H) \times (4H)$.

Now let $m$ is maximum value of $g'$. Then let $g''$ is $g'$ but with increased values by $m \cdot H$ in cells $(2H, j_1)$ and $(2H, j_2)$. Then after applying our LS for $g''$ we get the sum of 2 DP, passes through the cells $(2H, j_1)$ and $(2H, j_2)$ with maximum sum M. Then required value is $M - 2m \cdot H$.

Thus, we reduce the case $n = 2$ to CP2 by linear time. Then using Theorem [1] we confirm our Conjecture for $n = 2$.

**References**

[1] “1463. Cherry Pickup II”. (n.d.). In leetcode.com. Retrieved 14 Mar, 2021, from: https://leetcode.com/problems/cherry-pickup-ii/.

[2] “931. Minimum Falling Path Sum”. (n.d.). In leetcode.com. Retrieved 14 Mar, 2021, from: https://leetcode.com/problems/minimum-falling-path-sum/.

[3] “Gold Mine Problem”. (n.d.). In tutorialspoint.dev. Retrieved 14 Mar, 2021, from: https://tutorialspoint.dev/algorithms/dynamic-programming-algorithms/gold-mine-problem

[4] “64. Minimum Path Sum”. (n.d.). In leetcode.com. Retrieved 14 Mar, 2021, from: https://leetcode.com/problems/minimum-path-sum/.

[5] “741. Cherry Pickup”. (n.d.). In leetcode.com. Retrieved 14 Mar, 2021, from: https://leetcode.com/problems/cherry-pickup/.

[6] “SumoLogic Interview Question for SDE-3s”. (2015, 9 Aug). In careercup.com. Retrieved 14 Mar, 2021, from: https://www.careercup.com/question?id=5653217876639744.
[7] Scheffler P., “A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree-width”. Report 396, Fachbereich 3 Mathematik, TU Berlin, 1994.

[8] Golovach P. A., Kolliopoulos S. G., Stamoulis G., Thilikos D. M., “Planar Disjoint Paths in Linear Time”. arXiv:1907.05940 [cs.DS], 2019.

[9] Tholey T., “Linear time algorithms for two disjoint paths problems on directed acyclic graphs”. TCS, vol. 465, pp. 35-48, 2012. doi:10.1016/j.tcs.2012.09.025.

[10] Suurballe J. W., “Disjoint paths in a network”. Netw., vol. 4, pp. 125-145, 1974. doi:10.1002/net.3230040204.

[11] Dijkstra, E. W., “A note on two problems in connexion with graphs”. Numerische Mathematik, 1(1):269-271, 1959. doi:10.1007/BF01386390.

[12] Fredman M.L., Tarjan R.E., “Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms”. J. ACM, 34(3):596–615, 1987. doi:10.1109/SFCS.1984.715934.

[13] Ahuja R.K., Mehlhorn K., Orlin J.B., Tarjan R.E., “Faster Algorithms for the Shortest Path Problem”. J. ACM, 37(2):213-223, 1990. doi:10.1145/77600.77615.

[14] “Shortest Path in Directed Acyclic Graph”. (n.d.). In tutorialspoint.dev. Retrieved 30 Mar, 2021, from: https://tutorialspoint.dev/data-structure/graph-data-structure/shortest-path-for-directed-acyclic-graphs.

[15] “Sliding Window Maximum (Maximum of all subarrays of size k)”. (n.d.). In tutorialspoint.dev. Retrieved 14 Mar, 2021, from: https://tutorialspoint.dev/data-structure/queue-data-structure/sliding-window-maximum-maximum-of-all-subarrays-of-size-k.
import numpy as np

def get_F(g):
    H = len(g)
    W = len(g[0])
    F = np.empty((H, W))  # create table HxW
    F[H−1] = g[H−1].copy()  # copy last row
    for i in reversed(range(0, H−1)):  # i = H−2, ..., 0
        F[i] = g[i] + max(F[i+1][0], F[i+1][1])
        F[i][W−1] = g[i][W−1] + max(F[i+1][W−2], F[i+1][W−1])
    return F

def get_bounds(F):
    H = len(F)
    W = len(F[0])
    lp = np.arange(0, H)  # lp = [0, ..., H−1]
    rp = np.arange(0, H)  # rp = [0, ..., H−1]
    lp[0] = 0
    rp[0] = W − 1
    for i in np.arange(1, H):  # i = 1, ..., H−1
        lj = max(rp[H−1], lp[H−1]+1)
        if lj > 0 and F[i][lj−1] >= F[i][lj]:
            lp[i] = lj − 1
        if lj < W−1 and F[i][lj] < F[i][lj+1]:
            lp[i] = lj + 1
        rj = min(rp[i], rp[i−1])
        if rj < W−1 and F[i][rj+1] >= F[i][rj]:
            rp[i] = rj + 1
        if rj > 0 and F[i][rj] < F[i][rj−1]:
            rp[i] = rj − 1
    return lp, rp

def get_max(fromk, tok, Table, i):
    _max = float('−inf')
    for k in np.arange(fromk, tok+1):  # k = fromk, ..., tok
        _max = max(_max, Table[i][k])
    return _max

def get_M(g, F, lp, rp):
    H, W = len(F), len(F[0])
    Ml, Mr = np.empty((H, W)), np.empty((H, W))
    # base case Mr[H−1]
    lj = max(rp[H−1], lp[H−1]+1)
    for j in np.arange(lj, W):  # j = max(rp[H−1], lp[H−1]+1), ..., W−1
        Ml[H−1][j] = g[H−1][lp[H−1]] + g[H−1][j]
    rj = min(lp[H−1], rp[H−1]−1)
    for j in np.arange(0, rj+1):  # j = 0, ..., min(lp[H−1], rp[H−1]−1)
        Mr[H−1][j] = g[H−1][rp[H−1]] + g[H−1][j]
# induction step M*[0,...,H−2]

```python
for i in reversed(np.arange(0, H-1)):  # i = H-2,...,0
    Mri = get_max(max(0, lp[i]-1), lp[i+1]-1, Mr, i+1)
    Mli = get_max(rp[i+1]+1, min(W-1, rp[i]+1), Ml, i+1)

    # MI[i] search
    for j in np.arange(max(lp[i]+1, rp[i]), W):
        max1, max2 = 0, 0
        # case lPmax(i+1)<lP(i+1)
        if max(lp[i], 1) <= lp[i+1]:
            max1 = get_max(max(rp[i+1], j-1), min(j+1, W-1),
                            F, i+1) + Mri - F[i+1][rp[i+1]]
        # case lPmax(i+1)=lP(i+1)
        if lp[i+1]+2 <= W:
            max2 = get_max(max(j-1, rp[i]+1, lp[i]+1),
                            min(j+1, W-1),
                            Ml, i+1)
        MI[i][j] = g[i][lp[i]] + g[i][j] + max(max1, max2)

    # Mr[i] search
    for j in np.arange(0, min(lp[i], rp[i]-1)+1):
        max1, max2 = 0, 0
        # case rPmax(i+1)>rP(i+1)
        if rp[i+1] <= min(W-2, rp[i]):
            max1 = get_max(max(0, j-1), min(j+1, lp[i]+1),
                            F, i+1) + Mli - F[i+1][lp[i]+1]
        # case rPmax(i+1)=rP(i+1)
        if 1 <= rp[i+1]:
            max2 = get_max(max(j-1, 0),
                            min(j+1, lp[i]+1, rp[i+1]-1),
                            Mr, i+1)
        Mr[i][j] = g[i][rp[i]] + g[i][j] + max(max1, max2)
```

return MI, Mr

---

```python
def Pickup_Cherries_II(grid):
    W = len(grid[0])
    F = get_F(grid)
    lp, rp = get_bounds(F)
    MI, Mr = get_M(grid, F, lp, rp)
    return MI[0][W-1]
```