An Analysis of the Min-max Algorithm

Jerzy Cisło
Institute of Theoretical Physics
University of Wrocław
pl. M. Borna 9, 50-205 Wrocław, Poland
cislo@ift.uni.wroc.pl

Abstract
We consider the matrix $A_{ij}$, whose elements are independent random variables. We calculate the mean value of the number of the elements that we need to read to find $\min_i \max_j A_{ij}$.

1 The Min-max Algorithm

In this paper we are going to analyze the process of finding $\min_i \max_j A_{ij}$ [1,2,3], for which the reading of any single matrix element is particularly time consuming. However, this process–used in game theory and economy–may be shortened as to find the value of the Min-max, we do not have to read the whole matrix. Let us look at the following example:

$$
\begin{array}{cccc}
5 & 3 & \textbf{14} & 8 \\
2 & 4 & \textbf{9} & 1 \\
7 & \textbf{16} & 13 & 11 \\
12 & 6 & \underline{10} & \textbf{15}
\end{array}
$$

The maximal elements in each row are printed bold, and the elements which we do not have to read are underlined.
The algorithm of finding Min-max of matrix $n \times k$ reads

\[
\begin{align*}
\text{minmax} &= \infty \\
\text{FOR } i = 1 \text{ TO } n \\
& \quad \text{max} = -\infty \\
& \quad \text{FOR } j = 1 \text{ TO } k \\
& \quad \quad t = A[i][j] \\
& \quad \quad \text{IF } t > \text{max} \text{ THEN } \text{max} = t \\
& \quad \quad \text{IF } t > \text{minmax} \text{ THEN BREAK} \\
& \quad \quad \text{IF } \text{max} < \text{minmax} \text{ THEN } \text{minmax} = \text{max}
\end{align*}
\]

The essential instruction in this algorithm is: "IF $t > \text{minmax}$ THEN BREAK".

The presented algorithm is a restricted version of algorithm alpha-beta pruning, that is used in logical computer games such as chess, Othello, etc.

The search function of algorithm alpha-beta pruning reads [4,5]

\[
\begin{align*}
\text{SEARCH}(n, \alpha) \\
& \quad \text{IF } n = 0 \text{ THEN RETURN } - \text{evaluation} \\
& \quad \beta = -\infty \\
& \quad \text{FOR } j = 1 \text{ TO } m \text{ (list of moves)} \\
& \quad \quad \text{MOVE}(j) \\
& \quad \quad t = -\text{SEARCH}(n-1, -\beta) \\
& \quad \quad \text{UNDO MOVE} \\
& \quad \quad \text{IF } t \leq \alpha \text{ THEN RETURN } \beta \\
& \quad \quad \text{IF } t < \beta \text{ THEN } \beta = t \\
& \quad \text{RETURN } \beta
\end{align*}
\]

2 Two equivalent models

Generally, the time of performance of the algorithm depends on the distribution of matrix elements.

In this paper we consider the probability model in which we assume that the matrix elements $(A_{11}, ..., A_{nk})$ are independent uniformly distributed random variables on $\langle 0, 1 \rangle$.

Our aim is to find the mean value of the number of matrix elements that are read while searching for the min-max.
We can also consider the discrete model in which the matrix elements \((A_{11}, ..., A_{nk})\) are permutations of the set \((1, 2, ..., nk)\), each permutation being of the same probability.

For both models, the mean values are the same.

Still another model is the discrete model in which the matrix elements belong to the finite set (i.e. the elements can be repeated). For this model, however, the obtained mean value is different from the ones achieved for the first two models.

### 3 Recursion

When the algorithm begins to operate, the parameter \(a\) takes the maximum value. After each row the parameter \(a\) can decrease. The further search depends on the value of \(a\). Let \(M(a)\) stand for the mean value of the read elements, on condition that the algorithm begins to operate with the parameter \(a \in (0, 1)\).

Now at the beginning of the matrix \(A_{ij}\), let us add a new row \((x_1, x_2, ..., x_k)\) consisting of random independent variables uniformly distributed on \((0, 1)\).

Let us take an arbitrary number \(a \in (0, 1)\).

The probability that \(x_1, x_2, x_{j-1} \leq a < x_j\) is equal to \((1 - a)^{j-1}\).

If \(x_1, x_2, ..., x_{j-1} \leq a < x_j\), the algorithm reads the elements \(x_1, ..., x_j\), and, on average, \(M(a)\) elements from the remaining part of the matrix.

The probability that \(\{x_1, ..., x_k\}\) are in \((t, t + \Delta)\) is equal to \((1 + \Delta)^k - t^k\).

If \(\{x_1, ..., x_k\} \subseteq (t, t + \Delta)\), the algorithm reads all \(k\) elements from the added row, and, on average, \(M(t)\) elements from the remaining part of the matrix.

Therefore the mean number of the read matrix elements is equal to

\[
\tilde{M}(\alpha) = (1 - \alpha)(1 + M(\alpha)) + (1 - \alpha)(2 + M(\alpha)) + ... + (1 - \alpha)(k + M(\alpha)) + \\
+ \int_0^\alpha (k + M(t)) \frac{dt^k}{dt} dt = 1 + \alpha + ... + \alpha^{k-1} + M(\alpha) + \int_0^\alpha t^k \frac{dM(t)}{dt} dt.
\]

### 4 Calculations

The obtained recursion makes it possible to calculate the mean number of the read matrix elements.
Let $M_{nk}(a)$ stand for the mean number of the read matrix elements, on condition that the algorithm begins to operate with parameter $a \in (0, 1)$, $n$ and $k$ representing, respectively, the number of columns and number of rows in the matrix.

Using the relation from the previous section we can write

\[
M_{nk}(a) = M_{n-1,k}(a) - \int_0^a t^k M_{n-1,k}(t) \, dt + 1 + a + a^2 + \ldots + a^{k-1}.
\]  

This relation, together with the initial condition $M_{0k}(a) = 0$, allows us to find $M_{nk}(a)$ for any $n$.

Differentiating the equation (1) we obtain

\[
\frac{dM_{nk}(a)}{da} = (1 - a^k)\frac{dM_{n-1,k}(a)}{da} + (1 + 2a + 3a^2 + \cdots + (k - 1)a^{k-1}).
\]

We remember that

\[
\frac{dM_{0k}(a)}{da} = 0, \quad M_{nk}(0) = 0.
\]

Iterating $n$ times the relation (2) we get

\[
\frac{dM_{nk}(a)}{da} = \sum_{j=0}^{n-1} (1 - a^k)^j (1 + 2a + 3a^2 + \cdots + (k - 1)a^{k-1})
\]

\[
= 1 - \left(1 - a^k\right)^n (1 + 2a + 3a^2 + \cdots + (k - 1)a^{k-1}).
\]

Hence

\[
M_{nk}(a) = \int_0^a \frac{1 - (1 - t^k)^n}{t^k} (1 + 2t + 3t^2 + \cdots + (k - 1)t^{k-1}) \, dt =
\]

\[
M_{nk}(a) = n \sum_{l=0}^{k-1} a^k + \sum_{l=1}^{k-1} \sum_{j=1}^{n-1} \binom{n}{j+1} (-1)^j \frac{l}{l+jk} a^{l+jk}.
\]
5  Results

We obtain our main result, that is the formula for the mean number of the read matrix elements, by substituting $a = 1$ in the formula (5)

$$M_{nk}(1) = nk + \sum_{l=1}^{k-1} \sum_{j=1}^{n-1} \binom{n}{j} (-1)^{j} \frac{l}{l+jk}.$$  

(6)

Using the formula [6]

$$F_{n}(x) = n + \sum_{j=1}^{n-1} \binom{n}{j} (-1)^{j} \frac{x}{j+jx} =$$

$$1 + \frac{1}{(1+x)} + \frac{2!}{(1+x)(2+x)} + \cdots + \frac{(n-1)!}{(1+x)(2+x)...(n-1+x)},$$

we can rewrite our result in the following form:

$$M_{nk}(1) = \sum_{l=0}^{k-1} F_{n}(l/k) = n + k - 1 + \sum_{l=1}^{k-1} \sum_{j=1}^{n-1} \prod_{r=1}^{j} \frac{rk}{rk+l}.$$  

(8)

For small values of $n$ and $m$, we have

$$M_{22}(1) = \frac{2}{3}, \quad M_{23}(1) = \frac{7}{20}, \quad M_{24}(1) = \frac{4}{105},$$

$$M_{32}(1) = \frac{1}{5}, \quad M_{33}(1) = \frac{31}{70}, \quad M_{34}(1) = \frac{2419}{3465},$$

$$M_{42}(1) = \frac{23}{35}, \quad M_{43}(1) = \frac{30}{77}, \quad M_{44}(1) = \frac{6493}{45045}.$$  

Also, we would like to add that the following inequalities may be checked:

$$M_{nk}(1) > \frac{M_{n+1,k}(1)}{n+1}, \quad \frac{M_{nk}(1)}{k} > \frac{M_{n,k+1}(1)}{k+1},$$  

(9)

There is a good approximation for $M_{nk}(1)$

$$M_{nk}(1) \approx \frac{n}{2} + k \left( \frac{n}{\log n} + \frac{n}{\log^2 n} \right).$$

The error of the formula is less then 3 percent for $n \geq 28, k \geq 17$, and less then 1 percent for $n \geq 38, k \geq 20$. 

References

[1] Knuth D E: *The Art of Computer Programming*, Addison Wesley Longman 1997

[2] Cormen T H, Leiserson C E, Rivest R L: *Introduction to Algorithms*, The Massachusetts Institute of Technology Thirteenth printing 1994

[3] Dijkstra E W: *A Discipline of Programming*, Prentice-Hall, New Jersey 1976

[4] Knuth D E, Moore R W: *An Analysis of Alpha-Beta Pruning*, Artificial Intelligence, 6 (1975), 293-326

[5] Reigold E M, Nievergelt J, Deo N: *Combinatorial Algorithms. Theory and Practice*, Prentice-Hall, New Jersey 1977

[6] Graham R L, Knuth D E, Patashnik O: *Concrete Mathematics*, Addison-Wesley Company 1994