Timus 1146

Maximum Sum

Prof. Edson Alves – UnB/FGA

Problema

Given a 2-dimensional array of positive and negative integers, find the sub-rectangle with the largest sum. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the maximal sub-rectangle. A sub-rectangle is any contiguous sub-array of size 1×1 or greater located within the whole array.

As an example, the maximal sub-rectangle of the array:

```
 0  -2  -7   0
 9   2  -6   2
-4   1  -4   1
-1   8   0  -2
```

is in the lower-left-hand corner and has the sum of 15.

Entrada e saída

Input

The input consists of an $N \times N$ array of integers. The input begins with a single positive integer N on a line by itself indicating the size of the square two dimensional array. This is followed by N^2 integers separated by white-space (newlines and spaces). These N^2 integers make up the array in row-major order (i.e., all numbers on the first row, left-to-right, then all numbers on the second row, left-to-right, etc.). N may be as large as 100. The numbers in the array will be in the range $[-127, 127]$.

Output

The output is the sum of the maximal sub-rectangle.

Exemplo de entradas e saídas

Sample Input

```
4
0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
```

Sample Output

```
15
```

Solução $O(N^3)$

- Uma solução de força bruta computaria a soma todas as N^4 submatrizes, sendo que cada soma é feita em $O(N^2)$, de modo que a solução teria complexidade $O(N^6)$
- Contudo, o uso de combinado de somas prefixadas e o algoritmo de Kadane permite identificar a submatriz de soma máxima com complexidade $O(N^3)$
- Para cada par de colunas (i, j), deve ser computado, por meio do algoritmo de Kadane nas somas $p_k(i, j)$, para $1 \leq k \leq N$, o intervalo de maior soma, onde

$$p_k(i, j) = \sum_{t=i}^{j} a_{kt}$$

- Veja que, dados os limites do problema, mesmo nos casos extremos a soma máxima ainda pode ser armazenada em variáveis inteiras

Solução $O(N^3)$

```cpp
#include <bits/stdc++.h>

using namespace std;
const int oo { 1'000'000'010 };

int kadane(int N, const vector<int>& as)
{
    vector<int> s(N + 1);
    s[1] = as[1];

    for (size_t i = 2; i < as.size(); ++i)
        s[i] = max(as[i], s[i - 1] + as[i]);

    return *max_element(s.begin() + 1, s.end());
}

int solve(int N, const vector<vector<int>>& A)
{
    vector<vector<int>> p(N + 1, vector<int>(N + 1, 0));
    int ans = -oo;
```

Solução $O(N^3)$

```cpp
    for (int i = 1; i <= N; ++i)
    {
        vector<int> r(N + 1, 0);

        for (int j = i; j <= N; ++j)
        {
            for (int k = 1; k <= N; ++k)
                r[k] += A[k][j];

            ans = max(ans, kadane(N, r));
        }
    }

    return ans;
}

int main()
{
    ios::sync_with_stdio(false);
```

Solução $O(N^3)$

```cpp
    int N;
    cin >> N;

    vector<vector<int>> A(N + 1, vector<int>(N + 1));

    for (int i = 1; i <= N; ++i)
        for (int j = 1; j <= N; ++j)
            cin >> A[i][j];

    auto ans = solve(N, A);

    cout << ans << endl;

    return 0;
}
```