ACM-ICPC Algorithm Template

苏鹿鸣

<liugege@live.com>

https://github.com/Suluming1999

既见君子，云胡不喜

August 27, 2024

Contents

Chapter 1

Cấu trúc dữ liệu

1.1 RMQ

Bảng ST có thể được sử dụng cho giá trị tối đa của khoảng truy vấn, yêu cầu xử lý trước thời gian $O(n \log n)$, truy vấn có thể được hoàn thành trong thời gian $O(1)$ và không hỗ trợ sửa đổi.

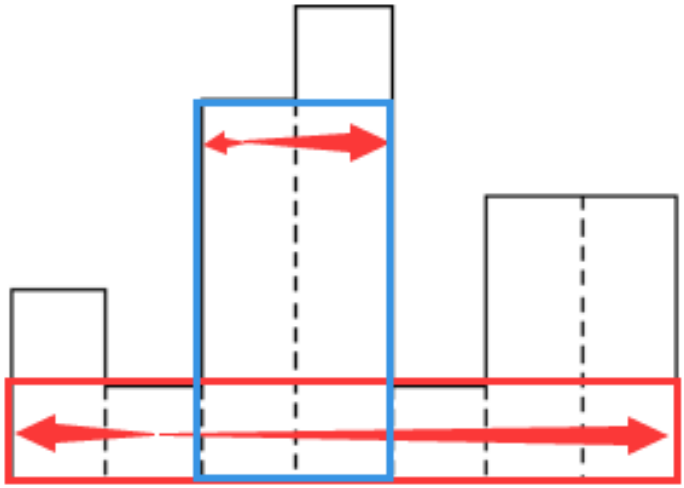
```
const int maxn = 110000;
int st[maxn][21], lg[maxn], a[maxn];
void init(int n) { // Phạm vi subscript [1,n], mà phép gán mảng a phải được hoàn
 ↪  thành trước khi gọi
    lg[1] = 0;
    for (int i = 2; i <= n; ++i) lg[i] = lg[i >> 1] + 1;
    for (int i = 1; i <= n; ++i) st[i][0] = a[i];
    for (int j = 1; j <= lg[n]; ++j)
        for (int i = 1; i + (1 << j) - 1 <= n; ++i)
            st[i][j] = max(st[i][j - 1], st[i + (1 << (j - 1))][j - 1]);
}
inline int rmq(int L, int R) {
    int k = lg[R - L + 1];
    return max(st[L][k], st[R - (1 << k) + 1][k]);
}
```

1.2 stack

Xác định ma trận 01 và tìm số ma trận con có các phần tử bằng 1.

1. Liệt kê các cơ số của các ma trận con theo hàng, sau đó tìm số ma trận con có cơ số của hàng đó.

2. Tiền xử lý int H[N][N]: Mở rộng lên tới độ dài tối đa của cả 1 và sau đó có thể giải quyết bằng ngăn xếp đơn điệu.



3. Để tránh tính toán lặp lại, ngăn xếp đơn điệu có thể được đóng ở bên trái và mở ở bên phải.

Theo ý tưởng trên cũng có thể tìm được diện tích của hình chữ nhật phụ lớn nhất.

Lưu ý rằng phương pháp này có thể được tối ưu hóa thành O (số hình chữ nhật 1), vì vị trí của các số 1 trong cùng một hàng có thể được liệt kê trực tiếp và câu trả lời được tính toán khi xem xét rằng các cột liền kề của hàng đều là 1. [Tính theo phân bố của hình chữ nhật, vì vị trí 0 có thể tách hình chữ nhật phụ]

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

using namespace std;

const int N = 1e3 + 5;
using ll    = int64_t;
int n, m, a[N][N];
int H[N][N];

ll calc(int* h, int k) { // Tính số hình chữ nhật chỉ số [0, k)
    stack<int> s;
    vector<int> l(k), r(k);
    for (int  i = k - 1; i >= 0; i--) {
        while (!s.empty() && h[i] <= h[s.top()]) l[s.top()] = i, s.pop();
        s.push(i);
    }
    while (!s.empty()) l[s.top()] = -1, s.pop();

    for (int  i = 0; i <= k - 1; i++) {
        while (!s.empty() && h[i] < h[s.top()]) r[s.top()] = i, s.pop();
        s.push(i);
    }
```

```
    while (!s.empty()) r[s.top()] = k, s.pop();
    ll res = 0;
    for (int  i = 0; i <= k - 1; i++)  res += (ll)h[i] * (i - l[i]) * (r[i] - i);
    return res;
}

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= m; j++)
            scanf("%d", &a[i][j]);
    ll ans = 0;

    for (int i = 1; i <= n; i++)  {
        for (int j = 1; j <= m; j++) {
            if (a[i][j])
                H[i][j] = H[i-1][j] + 1;
            else
                H[i][j] = 0;
        }
        ans += calc(H[i] + 1, m); // Đếm một lần mỗi hàng
    }

    printf("%lld\n", ans);
    return 0;
}
/**
2 2
0 1
1 1
Số hình chữ nhật con là 5
**/
```
