




CHUYÊN ĐỀ TRẠI HÈ HÙNG VƯƠNG

CẤU TRÚC LEFT-RIGHT VÀ ỨNG DỤNG

MỤC LỤC

A. MỞ ĐẦU.....	3
B. CẤU TRÚC LEFT-RIGHT.....	3
C. BÀI TẬP MINH HỌA	3
Bài 1. SALEMON – Buôn dưa lê.....	3
Ý tưởng:.....	5
Thuật toán:.....	5
Code C++:.....	5
	
Bộ test: SALEMON.zip	5
Bài 2. LLEGENDS – Liên minh huyền thoại	5
	
Bộ test: LLEGENDS.zip	7
Bài 3. MAXAREA – Hình chữ nhật có diện tích lớn nhất.....	7
Dữ liệu:	7
Kết quả:.....	7
Ví dụ:	7
Giới hạn:	8
Ý tưởng:.....	8
Thuật toán:	8
Code C++:.....	8
	
MAXAREA.zip	
Bộ test:	9
Bài 4. GOLFYARD – Sân Golf	9
Dữ liệu:	9
Kết quả:.....	9
Ví dụ:	9
Giới hạn:	10
Ý tưởng:.....	10
Code C++:.....	10



Bộ test: GOLFYARD.zip	11
-----------------------------	----

Bài 5. RECTCNT – Đếm hình chữ nhật	11
--	----

Dữ liệu:	11
----------------	----

Kết quả:.....	11
---------------	----

Ví dụ:	12
--------------	----

Giới hạn:	12
-----------------	----

Ý tưởng:	12
----------------	----

Thuật toán:	12
-------------------	----

Code C++:.....	12
----------------	----



Bộ test: RECTCNT.zip	13
----------------------------	----

D. BÀI TẬP THAM KHẢO	13
----------------------------	----

Bài 6. TWOLETTER – Đếm hình chữ nhật chứa 2 ký tự.....	13
--	----

Dữ liệu:	13
----------------	----

Kết quả:.....	13
---------------	----

Ví dụ:	13
--------------	----

Giới hạn:	14
-----------------	----

Gợi ý:	14
--------------	----



Bộ test: TWOLETTER.zip	14
------------------------------	----

PS - Đoạn dương	14
-----------------------	----



Bộ test: PS.zip	14
-----------------------	----

A. MỞ ĐẦU

Nhiều bài toán trong tin học dẫn tới truy vấn trên dãy số a_1, a_2, \dots, a_n , với mỗi i , cần tìm một đoạn dài nhất chứa phần tử a_i mà a_i là nhỏ nhất (hoặc lớn nhất). Tức là với mỗi i , cần tìm hai chỉ số L_i và R_i sao cho:

$$\begin{cases} 1 \leq L_i \leq i \leq R_i \leq n \\ a_j \geq a_i, \forall L_i \leq j \leq R_i \end{cases}$$

B. CẤU TRÚC LEFT-RIGHT

Với mỗi truy vấn, ta dễ dàng trả lời với ĐPT $O(n)$. Nếu có nhiều truy vấn, ta cũng có thể giải quyết bài toán bằng cách sử dụng hàng đợi hai đầu. Tuy nhiên, trong phạm vi bài viết này tôi đề xuất một kỹ thuật áp dụng cho lớp bài toán dạng trên, xin tạm gọi là “Cấu trúc Left-Right”.

Với mỗi i , ta gọi L_i là chỉ số “bên trái” i sao cho $a_i \leq a_j, \forall L_i \leq j \leq i$, R_i là chỉ số “bên phải” i sao cho $a_i \leq a_j, \forall i \leq j \leq R_i$.

Ta sẽ tính trước mảng L và R :

Tính mảng L : QHĐ từ bên trái sang:

$\forall i = 1..n$:

+ $L_i = i$

+ Trong khi $L_i > 1$ và $a_{L_i-1} \geq a_i$: $L_i = L_{L_i-1}$

Tính mảng R : QHĐ từ bên phải sang:

$\forall i = n..1$:

+ $R_i = i$

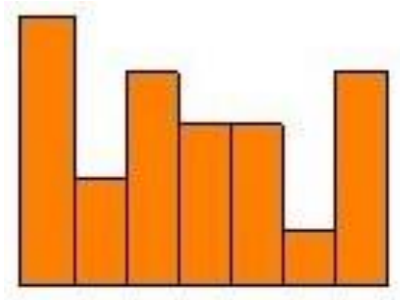
+ Trong khi $R_i < n$ và $a_{R_i+1} \geq a_i$: $R_i = R_{R_i+1}$

ĐPT khi tính L và R là: $O(n)$.

C. BÀI TẬP MINH HỌA

Bài 1. SALEMON – Buôn dưa lê

Sau một thời gian sống với Cuội ở gốc đa trên cung trăng, chị Hằng Nga thấy buồn chán và muốn mở cửa hàng buôn dưa lê. Chị Hằng Nga bắt Cuội làm cho một cái biển quảng cáo cho cửa hàng của mình. Cuội chỉ kiếm được mảnh gỗ trước đây là hàng rào vây gốc đa. Mảnh gỗ gồm n thanh gỗ ghép lại (đánh số từ 1 đến n), mỗi thanh có chiều ngang là 1 đơn vị và chiều dài là h_i đơn vị, các thanh được ghép song song sát nhau và bằng nhau ở một đầu (như hình sau)



Trên cung trăng không có thước đo các loại nên Cuội **chỉ có thể cưa tấm gỗ theo các đường là cạnh của các tấm gỗ**. Hằng Nga lại yêu cầu bôm làm cho mình cái biển hình vuông có diện tích càng lớn càng tốt. Em hãy giúp Cuội tính diện tích tối đa của cái biển có thể làm được từ miếng gỗ trên nhé.

Dữ liệu:

- Dòng đầu chứa số nguyên dương n (là số thanh gỗ của mảnh gỗ);
- Dòng thứ hai chứa n số nguyên dương h_1, h_2, \dots, h_n . Hai số liên tiếp cách nhau một dấu cách.

Kết quả:

- Một số nguyên dương duy nhất là diện tích của tấm biển lớn nhất có thể.

Ví dụ:

Dữ liệu:

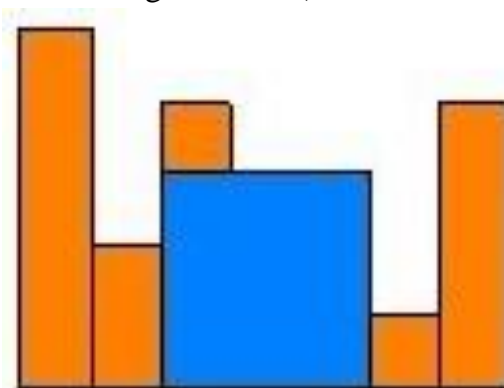
```
7
5 2 4 3 3 1 4
```

Kết quả:

```
9
```

Giải thích:

- Phương án tối ưu được mô tả trong hình sau (ô màu xanh là biển làm được):



Giới hạn:

- $1 \leq n \leq 10^6; 1 \leq h_i \leq 10^9$.

Ý tưởng:

Với mỗi i , ta cần tính toán xem có thể cắt hình vuông cạnh đúng bằng h_i không (chứa thanh thứ i). Để cắt được hình vuông cạnh bằng h_i thì một đoạn liên tiếp các thanh có độ dài lớn hơn hoặc bằng h_i phải không ít hơn h_i thanh.

Thuật toán:

- + Tính mảng L, R như mô tả ở phần đầu.
- + Đặt $s = 0$; //Diện tích hình vuông lớn nhất cắt được
- + $\forall i = 1..n$: Nếu $s < h_i^2$ và $R_i - L_i + 1 \geq h_i$ thì $s = h_i^2$;
- + Ghi ra giá trị s ;

ĐPT: $O(n)$.

Code C++:

```
#include <bits/stdc++.h>
using namespace std;
#define nmax 1000002
typedef long long ll;
int a[nmax], L[nmax], R[nmax], n;
void buildLeftRight() {
    // Build L
    for (int i = 1; i <= n; i++)
        for (L[i] = i; L[i] > 1 && a[L[i] - 1] >= a[i]; L[i] = L[L[i] - 1])
            ;
    // Build R
    for (int i = n; i >= 1; i--)
        for (R[i] = i; R[i] < n && a[R[i] + 1] >= a[i]; R[i] = R[R[i] + 1])
            ;
}
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    buildLeftRight();
    int res = 0;
    for (int i = 1; i <= n; i++)
        if (a[i] <= R[i] - L[i] + 1)
            res = max(res, a[i]);
    cout << ll(res) * res;
}
```



Bộ test: SALEMON.zip

Bài 2. LLEGENDS – Liên minh huyền thoại

Trong cuộc thi chung kết thế giới LEAGUE LEGENDS 3024, đội GAM của Việt Nam đã vào đến trận chung kết và đối đầu với một đội rất mạnh là Fnatic, tất nhiên GAM muốn

đánh một trận quyết chiến với chiến thuật hợp lý nhất. Trong tay GAM có n đại tướng, mỗi đại tướng có những ưu điểm riêng và GAM sắp xếp theo thứ tự từ 1 đến n để tiện cho việc điều binh của mình. Đại tướng thứ i có sức mạnh là s_i . Theo một bí quyết được lưu truyền trong thế giới LEAGUE LEGENDS thì đội quân được chọn ra chiến đấu sẽ phát huy được sức mạnh tối đa nếu là những đại tướng có chỉ số liên tiếp nhau, hơn nữa, sức mạnh của đội quân sẽ bằng sức mạnh của đại tướng yếu nhất (trong các đại tướng được chọn) nhân với số đại tướng. GAM rất muốn thắng trận chung kết để đem vinh quang về cho giới Game thủ Việt Nam, em hãy giúp đội GAM tính toán sức mạnh tối đa của đội quân mà GAM có thể chọn.

Dữ liệu:

- Dòng đầu chứa số nguyên dương n (là số đại tướng trong tay GAM);
- Dòng thứ hai chứa n số nguyên dương s_1, s_2, \dots, s_n . Hai số liên tiếp cách nhau một dấu cách.

Kết quả:

- Một số nguyên dương duy nhất là sức mạnh tối đa của đội quân GAM chọn được.

Ví dụ:

Dữ liệu:

```
4
3 4 3 1
```

Kết quả:

```
9
```

Giải thích:

- GAM chọn các vị đại tướng số 1,2,3 thì sức mạnh đạt được là $3 \times 3 = 9$.

Giới hạn:

- $1 \leq n \leq 10^6; 1 \leq s_i \leq 10^9, \forall i$.

Ý tưởng:

+ Duyệt toàn bộ bằng cách xét mọi đoạn: ĐPT $O(n^2)$.

Ta có cách duyệt khác với ĐPT $O(n)$ như sau:

+ Gọi L, R là hai mảng như mô tả phần cấu trúc Left-Right.

+ Với mỗi i , trong tất cả các đoạn chứa đại tướng i với s_i nhỏ nhất thì đoạn từ L_i đến R_i là đoạn tối ưu.

Thuật toán:

- + Tính toán mảng L, R ;
- + Đặt $\text{BestStrength} = 0$; //Sức mạnh lớn nhất
- + $\forall i = 1..n: s = \max(s, s_i \times (R_i - L_i + 1))$;
- + Ghi ra giá trị s ;

ĐPT: $O(n)$.

Code C++:

```
#include <bits/stdc++.h>
using namespace std;
#define nmax 1000002
int n, s[nmax], l[nmax], r[nmax];
```

```

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> s[i];
    for (int i = 1; i <= n; i++)
        for (l[i] = i; l[i] > 1 && s[i] <= s[l[i] - 1]; l[i] = l[l[i] - 1])
            ;
    for (int i = n; i >= 1; i--)
        for (r[i] = i; r[i] < n && s[i] <= s[r[i] + 1]; r[i] = r[r[i] + 1])
            ;
    long long res = 0;
    for (int i = 1; i <= n; i++) res = max(res, 1ll * s[i] * (r[i] - l[i] + 1));
    cout << res;
}

```



Bộ test: LLEGENDS.zip

Bài 3. MAXAREA – Hình chữ nhật có diện tích lớn nhất

Cho một bảng kích thước $M \times N$, được chia thành lưới ô vuông đơn vị M dòng N cột. Trên các ô của bảng ghi số 0 hoặc 1. Các dòng của bảng được đánh số $1, 2, \dots, M$ theo thứ tự từ trên xuống dưới và các cột của bảng được đánh số $1, 2, \dots, N$ theo thứ tự từ trái qua phải.

Hãy tìm một hình chữ nhật gồm các ô của bảng thoả mãn các điều kiện sau:

- Hình chữ nhật đó chỉ gồm các số 1;
- Cạnh hình chữ nhật song song với cạnh bảng;
- Diện tích hình chữ nhật là lớn nhất.

Dữ liệu:

- Dòng đầu chứa hai số nguyên dương M và N (là số hàng và số cột của HCN);
- M dòng tiếp theo, dòng thứ i chứa một xâu có độ dài N chỉ chứa các số 0 và 1 mô tả dòng thứ i của bảng.

Kết quả:

- Một số nguyên dương duy nhất là diện tích lớn nhất của HCN thoả yêu cầu bài toán.

Ví dụ:

Dữ liệu:

```

4 8
00010000
00111000
11111110
00111000

```

Kết quả:

9

Giải thích:

- HCN có diện tích lớn nhất có tọa độ trái trên là (2,2), phải dưới là (4,5) tổng diện tích là 9 ô đơn vị.

Giới hạn:

- $1 \leq m, n \leq 10^3$.

Ý tưởng:

Cách 1: Duyệt toàn bộ. ĐPT $O(N^3 \times M^3)$.

Cách 2:

+ Với mỗi dòng r , ta xét các hình chữ nhật có cạnh dưới nằm trên dòng r .

+ Gọi h_i là số số 1 liên tiếp tính từ dòng r trở lên của cột i . Gọi L, R là mảng Left, Right của dãy h , khi đó, với mỗi i , diện tích hình chữ nhật lớn nhất có cạnh một cạnh h_i sẽ có diện tích là $(R_i - L_i + 1) \times h_i$. Lấy max của các hình chữ nhật như vậy ta được kết quả.

Thuật toán:

+ Khởi tạo $maxArea = 0$; //Diện tích hình chữ nhật lớn nhất

+ Khởi tạo mảng $h_i = 0, \forall i = 1..N$

+ $\forall r = 1..M$:

- Cập nhật mảng h ;
- Tính mảng L, R của mảng h ;
- $\forall i = 1..N: maxArea = \max(maxArea, h_i \times (R_i - L_i + 1))$;

+ Ghi ra $maxArea$;

ĐPT: $O(M \times N)$.

Code C++:

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    int m, n, sol = 0;
    cin >> m >> n;
    string s;
    int l[n], r[n], h[n];
    memset(h, 0, sizeof(h));
    for (int i = 0; i < m; i++) {
        cin >> s;
        for (int j = 0; j < n; j++)
            if (s[j] == '1')
                h[j]++;
            else
                h[j] = 0;

        for (int j = 0; j < n; j++) {
```



```

        l[j] = j;
        for (int k = j - 1; k >= 0 && h[k] >= h[j]; l[j] = l[k], k = l[k] - 1)
            ;
    }
    for (int j = n - 1; j >= 0; j--) {
        r[j] = j;
        for (int k = j + 1; k < n && h[k] >= h[j]; r[j] = r[k], k = r[k] + 1)
            ;
    }
    for (int j = 0; j < n; j++) sol = max(sol, h[j] * (r[j] - l[j] + 1));
}
cout << sol << endl;
}

```



MAXAREA.zip

Bộ test:

Bài 4. GOLFYARD – Sân Golf

Sau nhiều lần tham gia giải Golf quốc tế PGA Tour và giành được nhiều giải thưởng, Tiger Woods quyết định đầu tư số tiền này để xây dựng một sân golf mang tên mình.

Sân golf mà Tiger Woods định xây nằm trong khuôn viên một khu đất hình chữ nhật kích thước $M \times N$ đã được chia thành lưới M hàng và N cột, các hàng được đánh số từ 1 đến M từ trên xuống dưới, các cột được đánh số từ 1 đến N từ trái sang phải. Ô đất tại hàng i , cột j có độ cao là h_{ij} . Tiger Woods sẽ chọn một hình chữ nhật con gồm các ô đất thuộc lưới để xây sân golf sao cho với hình chữ nhật con này thì các số trên một hàng bất kỳ tính từ trái sang phải, các số trên một cột bất kỳ tính từ trên xuống dưới đều có độ cao không giảm. Sau đó, Tiger Woods sẽ đặt lỗ golf tại góc trái trên và vị trí bắt đầu đánh tại góc phải dưới của hình chữ nhật để khi đánh, quả bóng luôn lăn xuống lỗ (do quá mệt mỏi với việc thi đấu các giải đấu lớn rồi nên Tiger Woods chỉ muốn xây một sân golf mang tính giải trí cao).

Bạn hãy giúp Tiger Woods chọn được sân golf có diện tích lớn nhất thỏa mãn yêu cầu trên.

Dữ liệu:

- Dòng đầu chứa hai số nguyên dương M và N (là số hàng và số cột của khu đất).
- M dòng tiếp theo, dòng thứ i chứa N số nguyên dương $h_{i1}, h_{i2}, \dots, h_{iN}$.

Hai số liên tiếp trên một dòng được ghi cách nhau một dấu cách.

Kết quả:

- Một số nguyên dương duy nhất là diện tích lớn nhất của sân golf xây được.

Ví dụ:

Dữ liệu:

```

3 4
9 2 4 8

```

3 5 7 8
6 8 1 3

Kết quả:

6

Giải thích:

- Khu đất chọn để xây sân golf có tọa độ trái trên là (1,2), phải dưới là (2,4) tổng diện tích là 6 ô đơn vị.

Giới hạn:

- $1 \leq M, N \leq 10^3; 1 \leq h_i \leq 10^9$.

Ý tưởng:

Cách 1: Duyệt toàn bộ. ĐPT $O(N^3 \times M^3)$.

Cách 2:

+ Ta xét riêng các HCN $1 \times d$ và các HCN $d \times 1$ (các HCN nằm trên một dòng, hoặc một cột).

+ Từ mảng $h[M \times N]$ ta lập mảng $a[(M - 1) \times (N - 1)]$ chỉ chứa các số 0, 1 theo quy tắc sau:

$$a_{ij} = \begin{cases} 1 & \text{nếu } h_{i,j} \leq h_{i,j+1} \text{ và } h_{i+1,j} \leq h_{i+1,j+1} \text{ và } h_{i,j} \leq h_{i+1,j} \text{ và } h_{i,j+1} \leq h_{i+1,j+1} \\ 0 & \text{nếu ngược lại} \end{cases}$$

+ Khi đó mỗi HCN gồm toàn số 1 trên mảng a có kích thước là $r \times c$ thì HCN tương ứng trên mảng h có thể làm sân golf có kích thước là $(r + 1) \times (c + 1)$. Áp dụng thuật toán của **bài 3** để giải **bài 4**.

ĐPT: $O(M \times N)$.

Code C++:

```
#include <bits/stdc++.h>
using namespace std;
int m, n, h[1001][1001];
bool a[1001][1001];

// Hình chữ nhật dạng 1 dòng hoặc 1 cột
int maxArea1() {
    int res = 1;
    // Dòng
    for (int y = 1; y <= m; y++)
        for (int x1 = 1, x2 = 2; x2 <= n; x2++) {
            if (h[y][x2 - 1] > h[y][x2])
                x1 = x2;
            res = max(res, x2 - x1 + 1);
        }
    // Cột
    for (int x = 1; x <= n; x++)
        for (int y1 = 1, y2 = 2; y2 <= m; y2++) {
            if (h[y2 - 1][x] > h[y2][x])
                y1 = y2;
            res = max(res, y2 - y1 + 1);
        }
    return res;
}

int maxArea2() {
```

```

int res = 0;
int g[1001], l[1001], r[1001];
for (int y = 1; y < m; y++)
    for (int x = 1; x < n; x++)
        a[y][x] = h[y][x] <= h[y + 1][x] && h[y][x] <= h[y][x + 1] && h[y + 1][x] <= h[y + 1][x + 1] && h[y][x + 1] <= h[y + 1][x + 1];

memset(g, 0, sizeof g);

for (int y = 1; y < m; y++) {
    for (int x = 1; x < n; x++)
        if (a[y][x])
            g[x]++;
        else
            g[x] = 0;
    for (int x = 1; x < n; x++)
        for (l[x] = x; l[x] > 1 && g[x] <= g[l[x] - 1]; l[x] = l[l[x] - 1])
            ;
    for (int x = n - 1; x >= 1; x--)
        for (r[x] = x; r[x] < n - 1 && g[x] <= g[r[x] + 1]; r[x] = r[r[x] + 1])
            ;
    for (int x = 1; x < n; x++) res = max(res, (g[x] + 1) * (r[x] - l[x] + 2));
}
return res;
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> m >> n;
    for (int i = 1; i <= m; i++)
        for (int j = 1; j <= n; j++) cin >> h[i][j];
    cout << max(maxAreal(), maxArea2());
}

```



Bộ test: GOLFYARD.zip

Bài 5. RECTCNT – Đếm hình chữ nhật

Cho một bảng kích thước $M \times N$, được chia thành lưới ô vuông đơn vị M dòng N cột. Trên các ô của bảng ghi số 0 hoặc 1. Các dòng của bảng được đánh số $1, 2, \dots, M$ theo thứ tự từ trên xuống dưới và các cột của bảng được đánh số $1, 2, \dots, N$ theo thứ tự từ trái qua phải.

Hãy đếm số hình chữ nhật con của bảng mà có các cạnh song song với các cạnh của bảng và gồm toàn số 1.

Dữ liệu:

- Dòng đầu chứa hai số nguyên dương M và N (là số hàng và số cột của HCN);
- M dòng tiếp theo, dòng thứ i chứa một xâu có độ dài N chỉ chứa các số 0 và 1 mô tả dòng thứ i của bảng.

Kết quả:

- Một số nguyên dương duy nhất là số HCN đếm được.

Ví dụ:

Dữ liệu:

```
4 3
111
101
111
001
```

Kết quả:

```
24
```

Giới hạn:

- $1 \leq m, n \leq 10^3$.

Ý tưởng:

Cách 1: Duyệt toàn bộ. ĐPT $O(n^3 \times m^3)$.

Cách 2: Với mỗi ô (r, c) , ta đếm số HCN có đỉnh dưới phải là ô (r, c) chỉ chứa toàn số 1.

Để đếm số HCN có đỉnh dưới phải là ô (r, c) ta QHĐ như sau:

Trên dòng r , xét từ trái sang phải từ cột 1 đến cột n , với cột c , gọi L_c là cột bên trái c mà có $h_i \geq h_c, \forall L_c \leq i \leq c$ và $h_{L_c-1} < h_c$ (nếu $L_c > 1$, ở đây h_i là số số 1 liên tiếp trên cột c tính từ dòng r trở lên), khi đó ta có $dp_{r,c} = h_c \times (c - L_c + 1) + dp_{r,L_c}$.

Thuật toán:

+ Khởi tạo biến đếm $cnt = 0$; //Số HCN

+ Khởi tạo $h_i = 0, \forall i = 1..n$;

+ $\forall r = 1..m$:

- Cập nhật mảng h ;
- Tính mảng L ;
- $dp_{r,0} = 0$;
- $\forall c = 1..n$:
 - $dp_{r,c} = dp_{r,L_c-1} + h_c \times (c - L_c + 1)$;
 - $cnt = cnt + dp_{r,c}$;

+ Xuất kết quả là cnt ;

ĐPT $O(m \times n)$.

Code C++:

```
#include <bits/stdc++.h>
using namespace std;
#define nmax 1001
typedef long long ll;
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
```

```

int m, n, l[nmax], h[nmax], c[nmax];
ll res = 0;
cin >> m >> n;
memset(h, 0, sizeof h);
memset(c, 0, sizeof c);

for (int j = 0; j < m; j++) {
    string s;
    cin >> s;
    s = ' ' + s;
    for (int i = 1; i <= n; i++) {
        h[i] = (s[i] == '1' ? h[i] + 1 : 0);
        for (l[i] = i; l[i] > 1 && h[i] <= h[l[i] - 1]; l[i] = l[l[i] - 1])
            ;
        c[i] = (h[i] ? (i - l[i] + 1) * h[i] + c[l[i] - 1] : 0);
        res += c[i];
    }
}
cout << res;
}

```



Bộ test: RECTCNT.zip

D. BÀI TẬP THAM KHẢO

Bài 6. TWOLETTER – Đếm hình chữ nhật chứa 2 ký tự

Cho một bảng hình chữ nhật kích thước $M \times N$, trên các ô của bảng có ghi một trong 3 chữ cái A, B, C . Hãy đếm số hình chữ nhật có cạnh song song với cạnh của bảng và có đúng 2 trong 3 ký tự A, B, C .

Dữ liệu:

- Dòng đầu chứa hai số nguyên dương M và N (là số hàng và số cột của HCN);
- M dòng tiếp theo, dòng thứ i chứa một xâu có độ dài N chỉ chứa các chữ cái A, B, C mô tả dòng thứ i của bảng.

Kết quả:

- Một số nguyên dương duy nhất là số HCN thỏa yêu cầu bài toán đếm được.

Ví dụ:

Dữ liệu:

```

4 3
ABC
AAA
BBC
ABC

```

Kết quả:

28

Giới hạn:

- $1 \leq m, n \leq 1000$.

Gợi ý:

Số HCN có đúng 2 loại ký tự A, B bằng số HCN có một trong hai loại ký tự A, B trừ đi số HCN chỉ chứa ký tự A và số HCN chỉ chứa ký tự B .

Để đếm số HCN chứa tập ký tự X ta thay những ký tự trong tập X của HCN bằng 1, những ký tự khác thay bằng 0 rồi áp dụng thuật toán của bài 5 để giải.

**Bộ test:** TWOLETTER.zip**PS - Đoạn dương**

Cho dãy số nguyên $A = (a_1, a_2, \dots, a_n)$. Hãy tìm một đoạn dài nhất gồm các phần tử liên tiếp trong dãy A : $(a_L, a_{L+1}, \dots, a_R)$ có tổng là số dương.

Dữ liệu:

- Dòng đầu tiên chứa số nguyên dương $n \leq 10^5$;
- Dòng thứ hai chứa n số nguyên a_1, a_2, \dots, a_n ($|a_i| \leq 10^9$), có ít nhất một số dương trong dãy.

Kết quả:

- Ghi ra hai chỉ số L, R trên một dòng;

Ví dụ:**Dữ liệu:**

```
10
-5 -2 -3 4 -6 7 -8 9 -1 -20
```

Kết quả:

3 9

**Bộ test:** PS.zip