# Bài 1: Di chuyển robot Tên file: MOVE.CPP hoặc MOVE.PAS

Một sân chơi có kích thước *n* x *n* (*n* lẻ) được chia thành lưới *n* x *n* ô vuông. Ô vuống chính giữa là vị trí đích. Ở một số ô khác có các robot khác nhau. Mỗi lần, một robot chỉ có thể thực hiện hoặc chuyển động đến ô bên cạnh chung cạnh mất 10 đơn vị năng lượng hoặc chuyển động đến ô bên cạnh chung đỉnh mất 15 đơn vị. Không được phép có 2 robot cùng một ô (trừ ô đích). Hãy tính xem chi phí tối thiểu để chuyển các robot trên về đích là bao nhiêu?

**INPUT: MOVE.INP**

+ Dòng đầu tiên ghi *n* (*n*≤100)

+ Dòng thứ hai ghi K là số robot (K≤100)

+ K dòng tiếp theo, mỗi dòng ghi hàng và cột của một robot. Không có 2 robot cùng một ô

**OUTPUT: MOVE.OUT**

+ Một số nguyên duy nhất là tổng năng lượng ít nhất để chuyển các robot đến ô đích.

|  |  |
| --- | --- |
| **ROBOT.INP** | **ROBOT.OUT** |
| 7  3  2 4  3 3  6 5 | 60 |

# Bài 2. PHẦN THƯỞNG Tên file: BONUS.CPP hoặc BONUS.PAS

Tuấn là người chiến thắng trong một cuộc thi “tìm hiểu kiến thức vũ trụ” và được nhận các phần thưởng do công ty XYZ tài trợ. Các phần thưởng được bố trí trên một bảng hình vuông nxn có dạng một lưới ô vuông kích thước đơn vị. Các dòng của bảng được đánh số từ 1 đến n, từ trên xuống dưới và các cột của bảng được đánh số từ 1 đến n, từ trái qua phải. Ô nằm trên giao của dòng i và cột j được gọi là ô (i,j) và trên ô đó chứa một món quà có giá trị là a[i,j] (1 <= i, j <= n)

Đề nhận phần thưởng, Tuấn được phép chọn một hình vuông kích thước k x k chiếm trọn trong một số ô của bảng và nhận tất cả các phần quà có trong các ô nằm trong hình vuông đó.

Yêu cầu: Hãy xác định tổng giá trị lớn nhất của món quà mà Tuấn có thể nhận được.

**Dữ liệu: BONUS.INP**

• Dòng thứ nhất chứa hai sô nguyên dương n, k (n <= 1000, n/3 <= k <= n).

• Dòng thứ i trong số n dòng tiếp theo chứa n số nguyên dương, số thứ j là a[i,j] (a[i,j] <= 1000)

**Kết quả: BONUS.OUT**

* Ghi ra một số nguyên duy nhất là tổng giá trị lớn nhất của các món quà mà Tuấn có thể nhận được.

Ví dụ:

|  |  |
| --- | --- |
| **BONUS.INP** | **BONUS.OUT** |
| 4 3  1 9 1 1  9 9 9 9  1 9 9 9  1 9 9 14 | |  |  |  |  | | --- | --- | --- | --- | | 1 | 9 | 1 | 1 | | 9 | **9** | **9** | **9** | | 1 | **9** | **9** | **9** | | 1 | **9** | **9** | **14** |   86 |

# Bài 3. Bàn cờ vua Tên file: VBOARD.CPP hoặc VBOARD.PAS

Cho một bàn cờ vua kích thước nxn, trên mỗi ô của bàn cờ có ghi một con số. Biết ô trên trái của bàn cờ vua là ô trắng (các ô của bàn cờ vua có dạng xen kẽ trắng đen). Các cột được đánh số từ 1 đến n từ trái sang phải, các hàng được đánh số từ 1 đến n từ trên xuống dưới. Ô ở hàng i, cột j của bàn cờ được ký hiệu là ô (i, j).

Cuội đưa ra những câu đố cho Bờm như sau: Cuội sẽ cho Bờm biết các vùng hình chữ nhật trên bàn cờ, nhiệm vụ của Bờm là phải tính giá trị tuyệt đối của độ chênh lệch giữa tổng giá trị các ô trắng và tổng giá trị các ô đen trên vùng hình chữ nhật đó. Bạn hãy lập trình giúp Bờm trả lời các câu đố của Cuội.

**INPUT: VBOARD.INP**

* Dòng đầu tiên: chứa số nguyên n (1 ≤ n ≤ 500).
* Dòng thứ i trong số n dòng tiếp theo chứa n số nguyên ai1, ai2, ..., aij, cho biết các số trên hàng i của bàn cờ (0 ≤ aij < 100).
* Dòng thứ n+2: chứa số q, cho biết số câu đố của Cuội (1 ≤ q ≤ 10000).
* q dòng tiếp theo, mỗi dòng chứa 4 số nguyên i1, j1, i2, j2 cho biết các tọa độ của vùng hình nhật trong một câu đố của Cuội: tọa độ đỉnh trái trên là (i1, j1) và đỉnh dưới phải là (i2, j2).

**OUTPUT: VBOARD.OUT**

* In ra q dòng, mỗi dòng cho biết đáp án của Bờm đối với câu đố tương ứng của Cuội.

**Giới hạn:**

Có 50% số test, tương ứng với 50% số điểm, trong đó 1 ≤ n ≤ 100 và 1 ≤ q ≤ 2000.

Ví dụ:

|  |  |
| --- | --- |
| **VBOARD.INP** | **VBOARD.OUT** |
| 3  1 3 5  2 4 6  0 10 5  2  1 1 2 2  1 2 3 3 | 0  5 |

# Bài 4. Bánh sinh nhật Tên file: CAKES.CPP hoặc CAKES.PAS

Bánh sinh nhật mà bố mẹ mua cho An là một hình chữ nhật gồm m dòng và n cột. Tại mỗi ô của chiếc bánh, người ta đổ các cột Socola ngon lành với độ cao là h[i][j] (h[i][j] < 0 nghĩa là ô đó bị khoét sâu xuống dưới và quết Socola vào ô đó)

Đến giờ sinh nhật An muốn cắt chiếc bánh thành 4 phần bằng 2 nhát dao dọc và ngang chiếc bánh. Gọi Smax là tổng độ cao các cột Socola của phần lớn nhất trong 4 phần và Smin là tổng độ cao các cột Socola của phần nhỏ nhất trong 4 phần.

An muốn biết phải cắt thế nào để chênh lệch giữa Smin và Smax là nhỏ nhất.

**INPUT: CAKES.INP**

* Dòng 1: chứa hai số nguyên dương m và n (m, n ≤ 1000)
* m dòng tiếp theo, mỗi dòng ghi n số nguyên h[i][j] thể hiện độ cao của các cột Socola được đổ lên mỗi ô của chiếc bánh.

**OUTPUT: CAKES.OUT**

* Một số duy nhất là độ chênh lệnh giữa Smin và Smax nhỏ nhất

Ví dụ:

|  |  |  |
| --- | --- | --- |
| CAKES.INP | CAKES.OUT | Giải thích: |
| 3 4  1 -3 2 2  4 2 -1 2  -1 2 2 1 | 4 | Sẽ cắt theo như hình dưới |

# Bài 5. Khai thác gỗ Tên file: WOOD.CPP hoặc WOOD.PAS

Năm 2050, lúc này JACKY đã trở thành Thủ tướng đất nước XYZ. Ông nhận được một đề nghị cho phép khai thác K m3 gỗ ở một khu rừng để xuất khẩu. Khu rừng này có dạng hình chữ nhật MxN km2. Để tiện quản lý thì người ta chia khu vực thành MxN vùng (M hàng, N cột) và lượng gỗ tại mỗi khu vực (tính theo m3) đã được biết. Các hàng được đánh số 1 đến M từ trên xuống dưới. Các cột được đánh số từ 1 đến N từ trái sang phải. Tọa độ của vùng nằm tại hàng i, cột j là (i, j).

Ngài Thủ tướng quyết định cho phép khai thác và vùng khai thác để dễ quản lý nên là một vùng hình chữ nhật. Ngài Thủ tướng muốn tìm một phương án khai thác gỗ sao cho diện tích khai thác là nhỏ nhất và vẫn đủ lượng gỗ cần thiết để xuất khẩu.

Do lâu ngày không lập trình nên ngài Thủ tướng cần đến sự giúp đỡ của các bạn. Các bạn hãy giúp ngài Thủ tướng nào.

**Dữ liệu: WOOD.INP*:***

* Dòng thứ nhất ghi ba số M, N, K (1 ≤ M, N ≤ 500, 1 ≤ K ≤ 109).
* Dòng thứ i trong M dòng tiếp theo ghi N số nguyên không âm, trong đó số thứ j cho biết lượng gỗ tại khu vực (i, j). Biết lượng gỗ tại mỗi khu vực không vượt quá 104 m3.

**Kết quả: WOOD.OUT:**

* Nếu không tồn tại vùng khai thác gỗ nào cho đủ gỗ xuất khẩu, in ra -1. Ngược lại in ra:
* Dòng thứ nhất ghi diện tích nhỏ nhất có thể của vùng khai thác gỗ.
* Dòng tiếp theo ghi bốn số là chỉ số của góc trái trên và góc phải dưới của vùng khai thác gỗ. Nếu có nhiều vùng cùng thỏa mãn thì in ra tọa độ của một vùng bất kỳ.

Ví dụ:

|  |  |
| --- | --- |
| **WOOD.INP** | **WOOD.OUT** |
| 3 3 19  5 4 0  4 7 0  0 0 2 | 4  1 1 2 2 |

# Bài 6. Ma trận Tên file: MATRIX.CPP hoặc MATRIX.CPP

Cho lưới ô vuông A kích thước M x N, trong đó các dòng được đánh thứ tự từ 1 đến M từ trên xuống dưới, các cột được đánh thứ tự từ 1 đến N từ trái sang phải, ô nằm trên dòng i, cột j có chứa giá trị nguyên A[i,j].

Nhiệm vụ của bạn là tìm lưới ô vuông con ( là hình chữ nhật nằm trong lưới đã cho ) có tổng các phần tử trong đó là lớn nhất.

**INPUT: MATRIX.INP**

* Dòng đầu tiên là hai số nguyên M và N (1 ≤ M, N ≤ 500)
* M dòng tiếp theo, dòng thứ i chứa N số Ai1, Ai2, …, AiN (|Aij| ≤ 5\*104)

Các số cách nhau ít nhất 1 dấu cách

**OUTPUT: MATRIX.OUT**

* Một dòng duy nhất là tổng lớn nhất của các phần tử thuộc lưới ô vuông con tìm được.

**Ví dụ:**

|  |  |  |
| --- | --- | --- |
| **MATRIX.INP** | **MATRIX.OUT** |  |
| 3 5  -4 5 -18 9 5  -16 4 0 -4 9  5 -1 4 -1 2 | 20 | \* Giải thích: lưới con có tổng lớn nhất từ ô (1,4) đến ô (3,5) |

\* Chú ý: có 60% test ứng với M, N ≤ 100

# Câu 7. Ma trận nguyên tố Tên file: PRIMATR.CPP hoặc PRIMATR.PAS

Cho một ma trận kích thước M x N, mỗi phần tử trong ma trận là một số nguyên dương. Chúng ta có thể thực hiện một thao tác biến đổi trên ma trận như sau: chọn 1 phần tử bất kỳ và tăng nó lên 1 đơn vị, mỗi phần tử trong ma trận có thể tăng với một số lần không hạn chế.

Một ma trận được gọi là ma trận nguyên tố nếu thỏa mãn một trong 2 điều kiện sau:

* Hoặc ma trận có ít nhất 1 hàng chứa toàn bộ là số nguyên tố
* Hoặc ma trận có ít nhất 1 cột chứa toàn bộ là số nguyên tố.

Nhiệm vụ của bạn là đếm số thao tác biến đổi ít nhất để chuyển ma trận ban đầu sang ma trận nguyên tố.

**INPUT: PRIMATR.INP**

* Dòng đầu chứa hai số nguyên dương M và N (1 ≤ M, N ≤ 1000)
* M dòng tiếp theo, mỗi dòng chứa N số nguyên dương, các số nguyên dương có giá trị ≤ 106.

**OUTPUT: PRIMATR.OUT**

* Một dòng duy nhất là số thao tác biến đổi ít nhất cần thực hiện.

Ví dụ:

|  |  |
| --- | --- |
| **PRIMATR.INP** | **PRIMATR.OUT** |
| 2 3  4 8 8  9 2 9 | 3 |

**\* Giải thích:**

Chọn phần tử tại vị trí (1,2) và tăng nó thêm 3 lần

**\* Ghi chú:**

- 30% test có n, m ≤ 100 và các phần tử trong bảng ban đầu ≤ 103

- 30% test tiếp theo có n, m ≤ 500 và các phần tử trong bảng ≤ 106

- 40% test còn lại n, m ≤ 1000 và các phần tử trong bảng ≤ 106