HƯỚNG DẪN GIẢI ĐỀ THI CODE TOUR ONLINE CODE CHALLENGE #1

A. TRÒ CHƠI HALLOWEEN

Solution:

C++	https://ideone.com/1PQaI4

Tóm tắt đề:

Cho bảng vuông $n * n$ $(1 \leq n \leq 400)$, trên 1 ô có thể chứa 1 chướng ngại vật, hoặc 1 đồng tiền vàng, hoặc không chứa gì cả. Các ô thuộc chung miền nếu như có thể đi qua lại giữa chúng bằng các ô kề cạnh (không được đi vào ô có chướng ngại).

Có thể chọn 1 hình vuông con trên bảng có kích thước k $(1 \leq k \leq n)$ và xóa toàn bộ chướng ngại nằm trong đó. Chỉ được thực hiện thao tác này 1 lần.

Yêu cầu tìm số đồng tiền vàng lớn nhất nằm chung miền nếu thực hiện thao tác trên 1 cách tối ưu.

Input

- Dòng đầu tiên là 2 số nguyên dương n $(1 \leq n \leq 400)$ và k $(1 \leq k \leq n)$
- n dòng tiếp theo là ma trận mô tả bảng vuông. Mỗi dòng gồm n số nguyên có giá trị thuộc $\{-1,0,1\}$ với:
  - -1: Ô có chướng ngại vật.
  - 0: Ô trống.
  - 1: Ô chứa 1 đồng tiền vàng.

Output

In ra một số nguyên duy nhất là số đồng tiền vàng lớn nhất nằm chung miền nếu thực hiện thao tác 1 cách tối ưu.

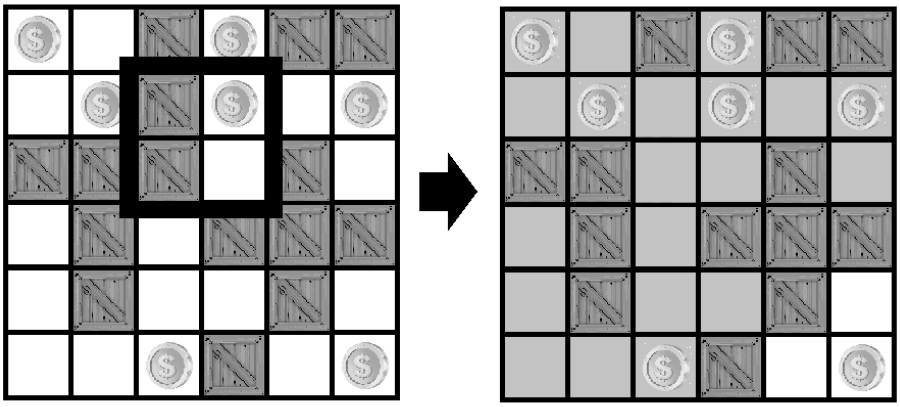
Ví dụ

6 2	6

1 0 -1 1 -1 -1 0 1 -1 1 0 1 -1 -1 -1 0 -1 0 0 -1 0 -1 -1 -1 0 -1 0 0 -1 0 0 0 1 -1 0 1	

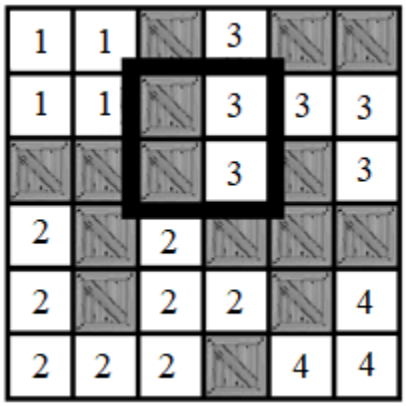
Giải thích:

- Ở ví dụ trên, chọn hình vuông con $2 * 2$ có tọa độ góc trái trên là $(2, 3)$. Sau khi xóa các chướng ngại vật trong đó, sẽ có 3 miền được liên thông với nhau tạo nên miền mới, lấy 6 đồng tiền vàng trên miền này là phương án tối ưu.



Hướng dẫn giải:

Ta thấy rằng, khi chọn 1 hình vuông con kích thước k và xóa các chướng ngại trong đó, thì các miền có thành phần thuộc hình vuông con hoặc có thành phần kề cạnh với hình vuông con sẽ trở nên liên thông với nhau (cùng kết hợp lại tạo thành miền mới).



Như ví dụ hình trên, với $k = 2$ và chọn hình vuông con như trên. Các miền 1 (có chứa ô kề cạnh với hình vuông con), miền 2 (có chứa ô kề cạnh hình vuông con) và miền 3 (có chứa ô kề cạnh lẫn nằm trong hình vuông con) sẽ kết nối với nhau tạo ra miền mới.

Vậy, mỗi khi thử và muốn biết xem tổng số đồng tiền vàng trên miền mới là bao nhiêu, ta xét những ô vuông nằm trong hình vuông con hoặc kề cạnh với hình vuông con. Tìm xem trong đó xuất hiện những miền nào, tổng số tiền của các miền tìm được sẽ là tổng cần tìm.

Với mỗi lần thử chọn 1 hình vuông con, nếu ta duyệt toàn bộ các ô vuông thỏa điều kiện trên (nằm trong hoặc kề cạnh với hình vuông con đang xét), thì độ phức tạp thuật toán sẽ lên đến $O((n-k+1)^2 * k^2) \approx O(n^4)$, không khả thi khi $1 \leq n \leq 400$. Để việc tính toán hiệu quả hơn, ta sử dụng kĩ thuật đếm phân phối và "sliding window".

Do hình vuông con trượt khắp bảng, ta chỉ cần quan tâm đến số tiền ở miền mới được tạo tại đó.

Thuật toán giải cụ thể như sau:

- **Bước 1:** Khởi tạo

  Dùng thuật toán DFS loang qua tất cả các ô không có vật cản, ta sẽ tính các đại lượng sau:

  - $\text{group}[\text{u}][\text{v}]$: Cho biết ô (u, v) thuộc miền nào.
  - $\text{groupCnt}[\text{x}]$: Số đồng tiền vàng thuộc miền x.

- **Bước 2:** Tìm kết quả

  Ta sẽ duyệt lần lượt tọa độ (i, j) là góc trái trên của hình vuông con.

  Gọi S là tập hợp các ô vuông kề cạnh hoặc nằm trong hình vuông con đang xét, $cnt[x]$ là số ô vuông thuộc S của miền x, sum là tổng số đồng tiền vàng trong miền mới tạo thành.

  Khi thêm ô (x, y) vào S, ta tăng $cnt[group[x][y]]$ lên 1 đơn vị. Nếu $cnt[group[x][y]] = 1$, tức miền group[x][y] mới xuất hiện lần đầu, thì cập nhật $sum \mathrel{+}= groupCnt[group[x][y]]$.

Khi xóa ô (x, y) khỏi S, ta giảm $cnt[group[x][y]]$ xuống 1 đơn vị. Nếu $cnt[group[x][y]] = 0$, tức miền group[x][y] không còn xuất hiện nữa, thì cập nhật $sum -= groupCnt[group[x][y]]$.

Trước tiên ta duyệt hàng i ($1 \le i \le n - k + 1$):

- Xét $j = 1$:
  - Khởi tạo $sum = 0, cnt[x] = 0$ (với tất cả miền x)
  - Thêm các ô nằm trong hình vuông con vào S, có tọa độ là (u, v) với $(i \le u \le i + k - 1,\ 1 \le v \le k)$
  - Thêm các ô kề cạnh với hình vuông con vào S, các tọa độ là:
    - $(i - 1, v)$ với $1 \le v \le k$ (đường viền trên).
    - $(i + k, v)$ với $1 \le v \le k$ (đường viền dưới).
    - $(u, k + 1)$ với $i \le u \le i + k - 1$ (đường viền phải).

    Đường viền trái ban đầu nằm ngoài bảng nên ta không cần thêm vào. Lưu ý, cần kiểm tra xem tọa độ ô có hợp lệ hay không (nằm trong bảng và không có vật cản), nếu không thì bỏ qua.
  - Cập nhật lại kết quả: $res = max(res, sum)$.
- Duyệt j: $1 < j \le n - k + 1$:

  Ta thấy, mỗi khi j dịch sang phải 1 đơn vị, thì tập các ô thuộc S cũng tịnh tiến sang phải 1 đơn vị. Vậy, ta chỉ cần xóa những ô rìa bên trái vừa bị trượt qua khỏi S, thêm vào S những ô rìa bên phải vừa được dịch tới. Không cần phải duyệt lại toàn bộ các ô của S. Lưu ý trước khi thêm hoặc xóa ô, phải kiểm tra tọa độ ô đó có hợp lệ hay không.
  - Xóa các ô ở rìa bên trái, có tọa độ:
    - $(u, j - 2)$ với $i \le u \le i + k - 1$
    - $(i - 1, j - 1)$
    - $(i + k, j - 1)$
  - Thêm các ô ở rìa bên phải, có tọa độ:
    - $(u, j + k)$ với $i \le u \le i + k - 1$
    - $(i - 1, j + k - 1)$
    - $(i + k, j + k - 1)$

  Sau đó, cập nhật lại kết quả: $res = max(res, sum)$.

Độ phức tạp thời gian: $\boldsymbol{O(n^2 + (n - k + 1)[k^2 + (n - k + 1) * k])}$. Trong trường hợp xấu nhất, khi $k \approx \frac{n}{2}$, độ phức tạp trở thành xấp xỉ $\boldsymbol{O(n^3)}$.

B. CHECK IN

Solution:

C++	https://ideone.com/KWellN
Java	https://ideone.com/VL5ymu
Python	https://ideone.com/DymLkz

Tóm tắt đề:

Viết chương trình nhập vào thời khóa biểu các môn học và thông tin điểm danh của sinh viên. Mỗi môn học sẽ chỉ diễn ra tại một ngày trong tuần. Thông tin của một môn học bao gồm ngày bắt đầu, ngày kết thúc và ngày trong tuần mà môn học đó diễn ra. Cho biết điểm danh đó là hợp lệ (VALID) hay không hợp lệ (INVALID). Một điểm danh trong môn học, chỉ hợp lệ nếu sinh viên thực hiện đúng vào ngày học của môn đó.

Input

Dòng đầu tiên gồm 6 con số nguyên Y1, M1, D1, Y2, M2, D2, X - với (Y1, M1, D1) lần lượt là năm, tháng, ngày bắt đầu của môn học, (Y2, M2, D2) lần lượt là năm, tháng, ngày kết thúc môn học và X là ngày trong tuần mà buổi học đó diễn ra. Giá trị của X đi từ 2 cho đến 8, tượng trưng cho các ngày trong tuần từ Thứ Hai cho đến Chủ Nhật.

Dòng thứ hai chứa số nguyên dương N - Số lượng lượt điểm danh của học sinh.

N dòng tiếp theo, mỗi dòng gồm 3 số nguyên Y, M, D - lần lượt là năm, tháng, ngày của lượt điểm danh này.

Giới hạn:

- $1900 \le Y1, Y2, Y < 10000$
- $1 \le M1, M2, M \le 12$
- $1 \le D1, D2, D \le 31$
- $1 \le N \le 1000$
- Các giá trị ngày tháng năm là các số nguyên không chứa bất kỳ chữ số 0 không có nghĩa nào. (VD: ngày 01/07/2020 sẽ được biểu diễn bằng 3 con số nguyên 2020 7 1).
- Đảm bảo rằng các ngày tháng là hợp lệ.
- Biết rằng ngày 01/01/1900 là một ngày thứ 2.

Output

Với mỗi lượt điểm danh, in VALID trên một dòng nếu lượt điểm danh này hợp lệ với thời gian của môn học, ngược lại in ra INVALID trên một dòng nếu lượt điểm danh này không hợp lệ.

Ví dụ

1900 1 1 9999 12 26 2 3 2020 6 8 2020 6 9 2020 6 15	VALID INVALID VALID
2020 6 1 2020 6 13 2 3 2020 6 8 2020 6 9 2020 6 15	VALID INVALID INVALID

Hướng dẫn giải:

Để kiểm tra xem một lượt điểm danh có hợp lệ hay ko, nó cần thỏa mãn hai điều sau đây:

- Ngày của lượt điểm danh nằm trong khoảng thời gian của môn học. Giả sử ngày, tháng, năm của ngày bắt đầu môn học là D1, M1, Y1 và ngày, tháng, năm của ngày kết thúc môn học là D2, M2, Y2 và ngày, tháng, năm của lượt điểm danh ta quan tâm là D, M, Y. Lượt điểm danh sẽ nằm trong khoảng thời gian của môn học nếu các điều kiện sau là đúng:
  - Y1 < Y or (Y1 = Y and M1 < M) or (Y1 == Y and M1 == M and D1 <= D)
  - Y < Y2 or (Y = Y2 and M < M2) or (Y == Y2 and M == M2 and D <= D2)
- Ngày điểm danh xảy ra vào ngày trong tuần trùng với ngày của môn học. Để tính được ngày trong tuần (thứ hai, thứ ba, thứ tư, thứ năm, thứ sáu, thứ bảy hay chủ nhật) khi biết được giá trị ngày, tháng, năm của một ngày, có nhiều thuật toán để xác định, tuy nhiên, một thuật toán có thể tham khảo được tạo ra bởi Tomohiko Sakamoto.

Độ phức tạp: O(N) với N là số lượng lượt điểm danh cần xác định.

C. CONAN MOVIE 24

Solution:

C++	https://ideone.com/IpBSqP
Python	https://ideone.com/x3NwbQ

Tóm tắt đề:

Cho 1 bảng M*N đánh số từ 1 ở dòng và cột. Với Q truy vấn, truy vấn i, tăng bảng chữ nhật con với ô trái trên (X_{i1}, Y_{i1}) và phải dưới (X_{i2}, Y_{i2}) lên P_i đơn vị. Cuối cùng đưa ra ô có giá trị lớn nhất.

Input

Dòng đầu chứ 2 số M,N ($\mathbf{1} \leq \boldsymbol{M}, \boldsymbol{N} \leq \mathbf{2000}$) là số hàng, số cột của ghế trong rạp chiếu phim. Dòng thứ 2 chứa Q ($\mathbf{1} \leq \boldsymbol{Q} \leq \mathbf{100000}$) là số ngày công chiếu. Q dòng tiếp theo, dòng thứ i chứa số nguyên $X_{i1}, Y_{i1}, X_{i2}, Y_{i2}, P_i$ ($\mathbf{1} \leq \boldsymbol{X_{i1}} \leq \boldsymbol{X_{i2}} \leq \boldsymbol{M}; \mathbf{1} \leq \boldsymbol{Y_{i1}} \leq \boldsymbol{Y_{i2}} \leq \boldsymbol{N}; \mathbf{1} \leq \boldsymbol{P_i} \leq \mathbf{5000}$), trong đó P_i là giá của mỗi vé khi xem phim vào ngày i, và trong ngày đó đoàn khách sẽ ngồi từ hàng từ hàng X_{i1} đến X_{i2} và cột từ Y_{i1} đến Y_{i2}. Các con số trên cùng 1 dòng cách nhau bởi 1 khoảng trắng.

Output

Xuất ra 3 số trên cùng 1 dòng cách nhau bởi 1 khoảng trắng, lần lượt là chỉ số hàng, chỉ số cột, tổng số tiền thu được của chiếc ghế thu được lợi nhuận lớn nhất. Nếu có nhiều ghế có tổng lớn nhất thì ưu tiên xuất ra ghế có chỉ số hàng nhỏ hơn, sau đó sẽ ưu tiên ghế có cột nhỏ hơn.

Ví dụ

5 6 3 2 2 4 5 3 3 3 5 6 2	2 2 7

Hướng dẫn giải:

- Chúng ta sẽ tính xem, sau q ngày, mỗi chiếc ghế thu được tổng số tiền là bao nhiêu.
- Sử dụng mảng A[M][N] với ý nghĩa A[i][j] là số tiền của chiếc ghế (i,j) thu được.
- Không cập nhật mảng A bằng cách thủ công, như vậy độ phức tạp là O(Q*M*N), rất lớn
- Xét 1 bài toán nhỏ hơn, giả sử ghế chỉ là 1 đoạn thẳng và đoàn khách ngồi các ghế liên tiếp. Tức là giả sử có 1 mảng A[N], và mỗi lần thì tăng đoạn từ $l \to r$
  với $(1 \le l \le r \le N)$ lên p đơn vị. Ta sẽ chỉ tăng A[l] thêm p, và giảm A[r+1] đi p đơn vị. Sau đó ta duyệt lại toàn bộ mảng A với A[i]=A[i-1]+A[i].
- Khi thực hiện trên ma trận HCN, ta tăng trong khu vực $(X1, Y1) \to (X2, Y2)$. Ma trận cũng là nhiều mảng 1 chiều hợp lại. Do đó ta sẽ tăng ở cột Y1, các hàng từ $X1 \to X2$ lên p đơn vị. Sau đò sẽ giảm ở cột Y2 + 1, các hàng từ $X1 \to X2$ đi p đơn vị. Cuối cùng duyệt lại ma trận với A[i][j]=A[i][j-1]+A[i][j]
- Tuy nhiên việc tăng giảm ở các cột như trên cũng là 1 đoạn liên tiếp, do đó ta cũng dùng cách xử lí như trên theo 1 tầng. Tức là trước khi thực hiện bước phía trên ta thực hiện tăng A[X1][Y1] lên P, giảm A[X2+1][Y1] đi P, giảm A[X1][Y2+1] đi P, và tăng A[X2+1][Y2+1] lên P. Cuối cùng duyệt lại ma trận với A[i][j]=A[i-1][j]+A[i][j].
- Tóm lại các bước thực hiện như sau:
  + B1: Thực hiện Q ngày tăng A[X1][Y1] lên P, giảm A[X2+1][Y1] đi P, giảm A[X1][Y2+1] đi P, và tăng A[X2+1][Y2+1] lên P.
  + B2: Duyệt ma trận với tính A[i][j]=A[i-1][j]+A[i][j] => Tương ứng với tăng ở cột Y1 và Y2
  + B3: Duyệt ma trận với tính A[i][j]=A[i][j-1]+A[i][j] => Ra được ma trận cuối cùng là tổng số tiền của mỗi ghế.
  + B4: Duyệt ma trận và tìm ra ô lớn nhất.
- Độ phức tạp: Việc tăng P trong Q ngày sẽ mất O(Q). Việc duyệt ma trận sẽ mất O(M*N). Vậy độ phức tạp là O(Q+M*N).

Độ phức tạp: Việc tăng P trong Q ngày sẽ mất O(Q). Việc duyệt ma trận sẽ mất O(M*N). Vậy độ phức tạp là **O(Q+M*N)**

D. PROGENITOR VIRUS

Solution:

C++	https://ideone.com/1zSa0Z
Java	https://ideone.com/8EGl0c
Python	https://ideone.com/A3n87G

Tóm tắt đề:

Cho N loại virus. Loại virus i có thời gian nuôi cấy là t_i. Biết rằng công thức tính số lượng virus thuộc loại bất kỳ sau khoảng thời gian t là 2^t. Tìm phần trăm số lượng của từng loại.

Input

Dòng đầu tiên gồm một số nguyên dương M là số lượng bộ test. Mỗi bộ test được tổ chức như sau:

- Dòng đầu chứa một số nguyên dương N là số loại virus.
- N dòng tiếp theo, mỗi dòng gồm một số nguyên t_i là thời gian nuôi cấy virus loại i.

Giới hạn:

- $1 \le M \le 50$
- $1 \le N \le 10$
- $0 \le t_i \le 10^{10}$
- Đảm bảo rằng $t_{max} - t_{min} \le 10$.

Output

Với mỗi bộ test, in ra trên một dòng là N giá trị p với p_i là phần trăm virus loại i.

Ví dụ

1 6 9 1 8 0 9 3	39.6591789311 0.1549186677 19.8295894655 0.0774593338 39.6591789311 0.6196746708

Giải thích ví dụ:

Ví dụ mẫu chỉ gồm 1 test case. Số lượng virus của từng loại như sau:

- Loại 1: 2^9 con
- Loại 2: 2^1 con
- Loại 3: 2^8 con
- Loại 4: 2^0 con
- Loại 5: 2^9 con
- Loại 6: 2^3 con

Như vậy, phần trăm tương ứng của từng loại là 39.6591789311, 0.1549186677, 19.8295894655, 0.0774593338, 39.6591789311 và 0.6196746708.

Hướng dẫn giải:

Công thức tính phần trăm số lượng của virus loại i:

$$\frac{2^{t_i}}{\sum_{j=1}^{N} 2^{t_j}}$$

Tuy nhiên giới hạn của t_i tương đối lớn (10^{10}), do đó nếu áp dụng trực tiếp công thức trên sẽ bị tràn số.

Để ý rằng công thức trên hoàn toàn có thể được rút gọn bằng cách đặt nhân tử chung là giá trị t_i nhỏ nhất:

$$\frac{2^{t_i}}{\sum_{j=1}^{N} 2^{t_j}} = \frac{2^{t_{min}} * 2^{t_i - t_{min}}}{2^{t_{min}} * \sum_{j=1}^{N} 2^{t_j - t_{min}}} = \frac{2^{t_i - t_{min}}}{\sum_{j=1}^{N} 2^{t_j - t_{min}}}$$

Với ràng buộc $t_{max} - t_{min} \leq 10$, ta hoàn toàn có thể tính toán trực tiếp được.

** Có thể sử dụng thêm mảng phụ để lưu các lũy thừa 2 giúp tiết kiệm thời gian tính toán.

Độ phức tạp: O(M * N) với M là số lượng bộ test và N là số lượng loại virus.