ANALYSIS

GOOGLE CODE JAM 2018 PRACTICE SESSION

Link contest:

https://codejam.withgoogle.com/2018/challenges/0000000000000130/dashboard

Đề thi gồm 4 bài toán cần giải quyết: gồm 3 bài đã được ra đề cho các kỳ Google Code Jam trước đó, và có 1 bài mới nhưng sẽ vô cùng quen thuộc với các bạn học thuật toán. Cụ thể như sau:

- **Number Guessing:** Bài toán quen thuộc với những ai học thuật toán, và bài này được ra nhằm test hệ thống máy chấm với những bài có dạng Interactive Problem.
- **Senate Evacuation:** Đây là bài dễ nhất của kỳ thi Google Code Jam Round 1C năm 2016.
- **Steed 2:** Cruise Control: Đây là bài dễ nhất của kỳ thi Google Code Jam Round1B năm 2017.
- **Bathroom Stalls:** Đây là bài đứng vị trí là bài khó thứ nhì của kỳ thi Google Code Jam Qualification Round năm 2017.

Lưu ý: hướng dẫn giải và phân tích đề của BigO-Coding sẽ minh họa bằng ngôn ngữ C++.

1. GUESSING NUMBER

Solution: https://ideone.com/zUW54n

Đây là một bài toán kinh điển, đề bài này được viết ra nhằm test hệ thống chấm cho những bài "interactive problem". Dạng bài "interactive problem" là dạng bài source code của thí sinh và hệ thống máy chấm sẽ chạy cùng một lúc. Khi thí sinh gửi cho hệ thống một output, hệ thống sẽ trả lại cho source code thí sinh một input và thí sinh sẽ quyết định làm gì cho việc tiếp theo, tùy vào input mà hệ thống gửi.

Bạn có thể xem thêm dạng bài "interactive problem" cụ thể hơn ở link bên dưới: https://codejam.withgoogle.com/codejam/resources/faq#what-is-interactive-problem

Đề bài:

Hệ thống sẽ cho ta một số nguyên P nằm trong đoạn (A, B] ($A < P \leq B$). Bạn có N lần thử đoán số. Mỗi lần bạn sẽ đoán một số nguyên, nếu như số nguyên đó không đúng với số P, hệ thống sẽ cho bạn biết là số P nhỏ hơn hay lớn hơn số bạn vừa đoán.

Input và Output:

Dòng đầu tiên gồm một số nguyên T là số lượng test case. Sau đó bạn cần xử lý T test case.

Với mỗi test case, chương trình bạn cần đọc 2 số A và B là cận dưới và cận trên như được mô tả ở đề bài. Sau đó, chương trình cần đọc tiếp số nguyên N là số

lần dự đoán tối đa mà bạn có thể đoán. Chương trình của bạn sẽ thực hiện tối đa N lần giao tiếp với hệ thống.

Với mỗi lần giao tiếp, bạn cần sử dụng standard output để gửi một số nguyên Q duy nhất trên một dòng: số mà bạn đoán. Sau đó, hệ thống sẽ trả ra cho bạn một từ tương ứng. Từ đó có thể là CORRECT nếu như số của bạn đoán chính xác, hoặc TOO_SMALL nếu dự đoán bạn đưa ra nhỏ hơn số P của hệ thống, hoặc TOO_BIG nếu dự đoán của bạn lớn hơn số P của hệ thống.

Nếu như trong N lần đoán mà bạn đoán ra được số đó, hệ thống sẽ trả ra cho bạn dòng CORRECT và bạn có thể chuyển sang bộ test case tiếp theo để xử lý. Còn nếu sau N lần đoán mà bạn vẫn chưa đoán được, hệ thống sẽ trả WRONG_ANSWER và bạn dừng quá trình xử lý ở đây và bài của bạn sẽ được đánh dấu là sai.

Chương trình của bạn cho đến test case cuối cùng vẫn phải nhận được dòng CORRECT, nếu không toàn bộ sẽ bị đánh dấu là WRONG_ANSWER.

Giới hạn dữ liệu:

$1 \le \mathbf{T} \le 20$.

A = 0. **N** = 30.

Time limit: 10 giây cho 1 dataset

Memory limit: 1GB.

Small Dataset:

B = 30

Large Dataset:

B = 10^9.

Đây là một đoạn mã giả minh họa:

```
t = read_int()          // reads 3 into t
a, b = read_two_int()   // reads 0 into a and 30 into b;
note that 0 30 is one line
n = read_int()          // reads 30 into n
print 30 to stdout      // guesses 30
flush stdout
string s = read()       // because 30 > 9, reads TOO_BIG
into s
print 5 to stdout       // guesses 5
flush stdout
s = read()              // reads TOO_SMALL into s since 5 <
9
print 10 to stdout      // guesses 10
flush stdout
s = readline()          // reads TOO_BIG into s since 10 > 9
print 9 to stdout       // guesses 9
flush stdout
s = read()              // reads CORRECT into s
```


Hướng dẫn giải:

Với bài toán này, ta có thể sử dụng kỹ thuật chặt nhị phân (Binary Search). Cụ thể, vì P xuất hiện trong (A, B] nên ta đặt left = A + 1, right = B đại diện cho đoạn giá trị mà P xuất hiện.

Trong lúc left ≤ right, thì ta sẽ gọi mid = (left + right) / 2. Đây chính là số mà ta cần guess. Nếu chuỗi s ta nhận được là CORRECT thì thôi, ta không làm gì nữa

cả. Nếu nhận lại là TOO_SMALL, tức là số P phải lớn hơn mid, ta gán left = mid + 1, ngược lại gán right = mid – 1.

Vì số B trong Large Dataset có giá trị là 10^9, độ phức tạp cho việc chặt nhị phân như vậy là log(B). Do đó log(B) = log(10^9) ~ 30. Do đó với N = 30 thì ta hoàn toàn có thể đoán được số P trong tối đa 30 lần đoán.

2. SENATE EVACUATION

(Google Code Jam Round 1C 2016)

Solution: https://ideone.com/98VJe5

Đề bài:

Một trận hỏa hoạn nhỏ đã xảy ra trong căn phòng và cần được sơ tán ngay.

Có một số thượng nghị sĩ đang ở trong phòng, mỗi người sẽ thuộc một trong N thành phần chính trị. Các thành phần chính trị sẽ được đánh chữ cái là N chữ cái đầu tiên trong bảng chữ cái Alphabet.

Cổng cứu hộ chỉ đủ sức chứa tối đa 2 người. Do đó với mỗi lượt, bạn phải quyết định lựa chọn tối đa 2 người để đưa vào cổng cứu hộ.

Những thượng nghị sĩ có thể bầu cử vào lúc nào cũng được, kể cả khi đang trong cuộc sơ tán. Do đó, nhiệm vụ của bạn là phải di tản như thế nào sao cho trong mỗi lần đưa người ra cổng, bạn phải đảm bảo sao cho những người còn lại ở trong phòng, không có quá nửa số người thuộc cùng một thành phần chính trị.

Bạn có thể lên kế hoạch cho một cuộc sơ tán không. Những thượng nghị sĩ đang đặt số phận vào trong tay bạn.

Input:

Dòng đầu tiên gồm một số nguyên T là số lượng test case cần phải xử lý.

Những test case sau, mỗi test case gồm 2 dòng:

- Dòng đầu tiên là số nguyên dương N là số lượng thành phần chính trị.

- Dòng thứ 2 gồm N số nguyên P_1, P_2,…,P_N với P_i là số lượng thượng nghị sĩ thuộc thành phần chính trị thứ i.

Output:

Với mỗi test case, ta sẽ xuất ra theo định dạng: Case #x: y, trong đó x là thứ tự test case (đánh số từ 1) và y là kế hoạch sơ tán. Kế hoạch sơ tán gồm nhiều chuỗi được cách bởi 1 khoảng trắng, những chuỗi chỉ gồm các ký tự trong bảng chữ cái N ký tự in hoa ban đầu thể hiện cho tên của các thành phần chính trị mà kế hoạch sơ tán có chứa thượng nghị sĩ đó.

Nếu có nhiều cách sơ tán, bạn in ra 1 cách bất kì.

Giới hạn:

$1 \le \mathbf{T} \le 50$.

$1 \le \mathbf{P_i} \le 1000$, với mọi i.

Time limit: 30 seconds cho một dataset.

Memory limit: 1GB.

Small Dataset:

$2 \le \mathbf{N} \le 3$.

Tổng các $\mathbf{P_i} \le 9$.

Large Dataset:

$2 \le \mathbf{N} \le 26$.

Tổng các $\mathbf{P_i} \le 1000$.

Ví dụ:

4 2 2 2 3 3 2 2 3 1 1 2 3 2 3 1	Case #1: AB BA Case #2: AA BC C BA Case #3: C C AB Case #4: BA BB CA

Giải thích ví dụ:

Có 4 bộ test bạn cần phải giải quyết:

Với Case #1, có 2 thành phần chính trị, và mỗi thành phần chính trị có 2 người. Kế hoạch sơ tán của ta như sau:

- Sơ tán AB, trong phòng còn lại 1 người A và 1 người B. Không có thành phần chính trị nào có số lượng người quá nửa cả nên không thể bầu cử.
- Cuối cùng sơ tán BA, như vậy là đã sơ tán được hết toàn bộ phòng.

Với Case #2, có 3 thành phần chính trị, thành phần A có 3 người, thành phần B có 2 người, thành phần C có 2 người. Kế hoạch sơ tán của ta như sau:

- Sơ tán AA, ta còn lại trong phòng 1 người thuộc thành phần chính trị A, 2 người thuộc thành phần chính trị B, 2 người thuộc thành phần chính trị C. Không có thành phần chính trị nào có số lượng thượng nghị sĩ nhiều hơn một nửa nên chưa thể bầu cử.
- Sơ tán BC, ta còn lại trong phòng 1 A, 1 B và 1 C. Không thể bầu cử.
- Sơ tán C, ta còn lại 1 A và 1 B, cũng không thể bầu cử.
- Cuối cùng ta sơ tán AB. Toàn bộ người đã được sơ tán.

Với những Case #3 và Case #4 ta cũng giải thích tương tự.

Hướng dẫn giải:

Chiến thuật sơ tán của ta sẽ như sau: Những thượng nghị sĩ ta cần sơ tán đó là những thượng nghị sĩ thuộc các thành phần chính trị có đông người nhất.

Từ đó, ta mới có một thuật toán như sau:

- Đặt sum = $P_1 + P_2 + \ldots + P_N$.
- Trong lúc sum > 0, đồng nghĩa với việc toàn bộ những thượng nghị sĩ trong phòng chưa ra khỏi, ta sẽ làm những thao tác như sau:
    - Sắp xếp những giá trị P tăng dần theo giá trị.
    - Tạo một hàm check(sum, P, n) với ý nghĩa: hỏi xem trong những giá trị P_i, có giá trị nào lớn hơn sum / 2 hay không ? Trả về 0 nếu có và 1 nếu như không có.
    - Người thượng nghị sĩ ở vị trí có giá trị P cao nhất chắc chắn phải là người đầu tiên được đưa ra khỏi phòng. Lúc đó, ta giảm sum đi 1 và giảm giá trị P_i tương ứng với người thượng nghị sĩ đó.
    - Ta sẽ quyết định đó là người thượng nghị sĩ duy nhất ta đưa ra khỏi phòng, hoặc quyết định thêm 1 người nữa. Nếu như việc đưa người thượng nghị sĩ này ra mà làm check(sum, P, n) = 1 thì người thượng nghị sĩ này hoàn toàn có thể đưa ra được. Nếu không, ta quyết định chọn người thượng nghị sĩ tiếp theo là 1 trong 2 người P[n] hoặc P[n – 1] để đưa ra.
    - Lặp lại quá trình trên cho đến khi sum = 0 thì ngừng.

Độ phức tạp: O(T * sum * N * log(N)) với sum là tổng các P_i, N là số lượng thành phần chính trị, T là số lượng test case của đề.

3. STEED 2: CRUISE CONTROL

(Google Code Jam Round 1B 2017)

Solution: https://ideone.com/QgWBrq

Đề bài:

Annie là một tài xế xe buýt với một công việc khá là căng thẳng. Do đó cô ấy quyết định làm một chuyến du lịch tại Caribbean. Nhưng chuyến du lịch này cũng khá căng thẳng, do đó cô ấy quyết định sẽ cưỡi ngựa.

Hôm nay, Annie quyết định sẽ cưỡi ngựa theo hướng từ đông sang tây, hiện tại cô ấy đang ở trên kilomet thứ 0 và đích đến của cô ấy là kilomet thứ D.

Có N con ngựa đang chạy và chúng đang ở các vị trí ở giữa đoạn từ con ngựa của Annie đến đích. Hiện tại, con ngựa thứ i đang ở vị trí K_i và đang chạy với tốc độ tối đa là S_i kilomet trên giờ.

Những con ngựa rất lịch sự, con ngựa H_1 sẽ không vượt qua mặt con ngựa H_2 đã chạy trước mặt con ngựa H_1 (2 hay nhiều con ngựa có thể ở cùng một vị trí, bạn có thể coi những con ngựa như là những điểm). Những con ngựa sẽ chạy với vận tốc tối đa của mình. Nếu như con ngựa H_1 đuổi kịp con ngựa H_2, thì con ngựa H_1 sẽ thay đổi vận tốc của nó bằng với vận tốc của con ngựa H_2.

Con ngựa của Annie không bị giới hạn vận tốc, nó có thể chạy với vận tốc mà Annie được quyền tùy chọn, tất nhiên nó không được phép vượt qua mặt những con ngựa khác. Để cho đường chạy một cách thuận lợi, Annie có thể chọn một vận tốc hằng số cho con ngựa của mình, từ vị trí hiện tại đến đích và phải đảm

bảo rằng con ngựa của Annie sẽ không vượt qua mặt những con ngựa nào khác. Hãy giúp Annie chọn vận tốc lớn nhất có thể.

Input:

Dòng đầu tiên gồm một số nguyên T là số lượng test case.

Mỗi test bắt đầu bằng 2 số nguyên D và N là điểm đích và số lượng con ngựa ở trên đường.

N dòng tiếp theo, dòng thứ i gồm 2 số nguyên K_i và S_i là địa điểm của con ngựa thứ i tại thời điểm hiện tại và vận tốc tối đa của con ngựa thứ i.

Output:

Với mỗi test case, bạn hãy in ra theo định dạng Case #x: y, với x là thứ tự test case (được đánh số từ 1), y là vận tốc tối đa mà con ngựa của Annie có thể chạy (km/h), y sẽ coi như là đúng nếu như như giá trị tuyệt đối độ chênh lệch giữa y so với đáp án không quá 10^{-6}.

Giới hạn dữ liệu:

$1 \le \mathbf{T} \le 100$.
$0 < \mathbf{K_i} < \mathbf{D} \le 10^9$, với mọi i.
$\mathbf{K_i} \ne \mathbf{K_j}$, for all $i \ne j$.
$1 \le \mathbf{S_i} \le 10000$.
Time limit: 10 seconds trên 1 test.
Memory limit: 1GB.

Small Dataset:

$1 \le \mathbf{N} \le 2$.

Large Dataset:

$1 \le \mathbf{N} \le 1000$.

Ví dụ:

3 2525 1 2400 5 300 2 120 60 60 90 100 2 80 100 70 10	Case #1: 101.000000 Case #2: 100.000000 Case #3: 33.333333

Giải thích test ví dụ:

Ở Case #1, rõ ràng với vận tốc lớn hơn 101km/h, chắc chắn con ngựa của Annie sẽ vượt qua con ngựa kia tại một vị trí nào đó trước vạch đích. Vì vậy kết quả chỉ có thể là 101km/h.

Với những case còn lại, ta giải thích cũng tương tự. Do đó Case #2 đáp án là 100 km/h, còn Case #3 đáp án là 33.3333333 km/h.

Hướng dẫn giải:

Ta có công thức vận tốc bằng quảng đường chia thời gian ($v = \frac{s}{t}$). Đề bài cho quảng đường là cố định, vậy để vận tốc tối đa thì thời gian chạy phải càng nhỏ càng tốt. Tuy nhiên, nếu vận tốc quá nhanh thì con ngựa của Annie sẽ vượt qua các con ngựa khác tại một địa điểm nào đó.

Như vậy con ngựa của Annie chỉ có thể gặp các con ngựa khác từ vị trí đích trở đi, mà để tốt nhất thì chắc chắn là sẽ gặp ngay tại vị trí đích. Trước tiên ta cần biết con ngựa về đích cuối cùng trong N con đã cho. Ta lấy quãng đường ta chạy chia cho thời gian đó.

Với con ngựa thứ i ở vị trí P_i, vận tốc là S_i, ta tính thời gian bằng cách: $t_i = \frac{D-P_i}{S_i}$.

Trong các giá trị t_i, ta chọn giá trị lớn nhất, ta tạm gọi là t_{max} là thời gian con ngựa cuối cùng về đích.

Như đã phân tích ở trên, đây cũng sẽ là thời gian tốt nhất để ngựa của Annie chạy về đích, vậy thì ta có kết quả là $\frac{D}{t_{max}}$.

Độ phức tạp: O(T * N) với T là số test case, N là số lượng con ngựa.

4. BATHROOM STALLS

(Google Code Jam Qualification Round 2017)

Solution: https://ideone.com/Pcps8n

Đề bài:

Trong phòng tắm có N + 2 cái bồn tắm được xếp trên một hàng và được đánh số từ 0 đến N + 1. Hai bồn tắm thứ 0 và thứ N + 1 đã có người sử dụng, N bồn tắm còn lại hoàn toàn có thể sử dụng.

Khi một người vào phòng tắm, họ sẽ chọn một bồn tắm mà xa những người đã sử dụng nhất có thể. Với mỗi bồn tắm chưa có người sử dụng, họ sẽ tính 2 giá trị L_S và R_S là số lượng bồn tắm rỗng nằm giữa bồn tắm hiện tại và bồn tắm gần nhất nằm về phía tay trái và tay phải. Họ sẽ lựa chọn bồn tắm sao cho giá trị $\min(L_S, R_S)$ lớn nhất có thể. Nếu chỉ có một bồn tắm như vậy, họ sẽ chọn nó. Nếu có nhiều hơn một bồn tắm thỏa mãn, họ sẽ chọn bồn tắm sao cho giá trị $\max(L_S, R_S)$ là lớn nhất có thể. Nếu có nhiều bồn tắm thỏa mãn như thế nữa, họ sẽ chọn bồn tắm nằm về phía bên trái nhất có thể.

Ta có tổng cộng K người vào phòng.

Với người cuối cùng chọn bồn tắm S, giá trị $\max(L_S, R_S)$, $\min(L_S, R_S)$ là bao nhiêu.

Input:

Dòng đầu tiên gồm số T là số lượng test case.

T test case sau, mỗi test case gồm 2 số nguyên N và K.

Output:

Với mỗi test case, bạn xuất ra theo định dạng: Case #x: y z, trong đó x là thứ tự test case (được đánh số từ 1), y là giá trị max(L_S, R_S) và z là giá trị min(L_S, R_S).

Giới hạn dữ liệu:

1 ≤ **T** ≤ 100.

1 ≤ **K** ≤ **N**.

Time limit: 30 giây cho một dataset.

Memory limit: 1GB.

Small Dataset 1: 1 ≤ **N** ≤ 1000.

Small Dataset 2: 1 ≤ **N** ≤ 10^6.

Large Dataset: 1 ≤ **N** ≤ 10^{18}.

Ví dụ:

5 4 2 5 2 6 2 1000 1000 1000 1	Case #1: 1 0 Case #2: 1 0 Case #3: 1 1 Case #4: 0 0 Case #5: 500 499

Giải thích test ví dụ:

Với Case #1, ban đầu trạng thái phòng tắm của ta là O….O. Người thứ 1 đi vào, họ sẽ chọn bồn tắm thứ 2, khi đó trạng thái của phòng tắm là: O.O..O. Người thứ 2 đi vào, họ sẽ chọn bồn tắm thứ 3, trạng thái của phòng tắm là: O.OO.O. Với trạng thái của bồn tắm thứ 3 thì giá trị L_S là 0, giá trị R_S là 1.

Với Case #2, ban đầu trạng thái phòng tắm của ta là O…..O. Người thứ 1 đi vào, họ sẽ chọn bồn tắm thứ 3, khi đó trạng thái phòng tắm thành O..O..O. Người thứ 2 đi vào, họ sẽ chọn bồn tắm thứ 1, khi đó trạng thái phòng tắm thành OO.O.O. Do đó, $L_S = 0$, $R_S = 1$.

Với Case #3, ban đầu trạng thái phòng tắm của ta là O……O. Người thứ 1 đi vào, họ sẽ chọn bồn tắm thứ 3, khi đó trạng thái của phòng tắm là: O..O…O. Người thứ 2 đi vào, họ sẽ chọn bồn tắm thứ 5, khi đó, trạng thái của phòng tắm là: O..O.O.O. Với vị trí bồn tắm số 5, ta có $L_S = 1$ và $R_S = 1$.

Với Case #4, trạng thái của phòng tắm khi người thứ 1000 vào sử dụng là toàn bộ các phòng tắm đã có người sử dụng hết. Do đó $L_S = 0$ và $R_S = 0$.

Với Case #5, trạng thái của phòng tắm ban đầu gồm 1000 bồn tắm chưa có ai sử dụng. Người thứ 1 vào sử dụng sẽ sử dụng bồn tắm thứ 500. Do đó $L_S = 499$ và $R_S = 500$.

Hướng dẫn giải:

Ta sẽ sử dụng một cấu trúc cây nhị phân tìm kiếm để lưu trữ lại với mỗi loại dãy gồm x bồn tắm liên tiếp chưa có người sử dụng (key) sẽ có số lượng là bao nhiêu (value). Tạm gọi nó là cnt thì cnt[x] nghĩa là số lượng dãy gồm x bồn tắm liên tục chưa có người sử dụng.

Ban đầu ta có N bồn tắm đều chưa có ai sử dụng, nên ta có cnt[N] = 1.

Khi một người chiếm hữu một bồn tắm, thì từ dãy gồm L bồn tắm trống liên tiếp sẽ thành 2 dãy con, một dãy có $\frac{L-1}{2}$ bồn tắm liên tiếp, dãy còn lại gồm

$L - 1 - \frac{L-1}{2}$ bồn tắm liên tiếp. Ta cần cập nhật đến khi hết K người. Từ đó, ta có một thuật giải như sau:

- cnt[N] = 1.
- Tại mỗi bước, ta tìm trong cnt phần tử có key L lớn nhất (độ dài dãy bồn tắm chưa sử dụng là lớn nhất), cập nhật K = K – cnt[L]. Nếu lúc này K ≤ 0 nghĩa là người thứ K ban đầu đã sử dụng bồn tắm, thì ta thoát khỏi vòng lặp. Ngược lại lúc này với mỗi dãy L đó, nó sẽ chia thành 2 dãy con có độ dài lần lượt là (L-1)/2 và L-1-(L-1)/2, nên ta sẽ cập nhật cnt[(L-1)/2] và cnt[(L-1)-(L-1)/2] thêm một lượng là cnt[L]. Sau khi cập nhật, ta sẽ xóa phần tử có khóa là L ra khỏi danh sách.
- Kết quả của ta chính là (L-1)/2 và (L-1) – (L-1)/2 với L = first lớn nhất trong danh sách hiện tại của mình.

Độ phức tạp: O(T * log(N)) với T là số lượng test case, N là số lượng bồn tắm trống. Lý do có thể tính được trong log(N) là vì cứ mỗi lần N sẽ được đem chia đôi, đồng thời những đoạn chia đôi chỉ được tính có một lần.