TRƯỜNG THCS HỒNG BÀNG
ĐỘI TUYỂN HỌC SINH GIỎI TIN HỌC

Đề thi có 02 trang

KIỂM TRA CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

Môn thi: TIN HỌC
Thời gian làm bài: 120 phút, không kể thời gian phát đề

TỔNG QUAN ĐỀ THI

Bài	Tên bài	Tên file chương trình	Hạn chế thời gian	Hạn chế bộ nhớ	Điểm
1	Treo bằng khen	BANGKHEN.*	1 giây	1024 MB	3
2	Sinh nhật	SINHNHAT.*	1 giây	1024 MB	4
3	Tên trộm	TENTROM.*	1 giây	1024 MB	3

Dấu * được thay thế bởi PAS hoặc CPP theo ngôn ngữ lập trình được sử dụng tương ứng với PASCAL hoặc C++
LẬP TRÌNH GIẢI CÁC BÀI TOÁN SAU

Bài 1: Treo bằng khen
Bạn Minh nhận được N bằng khen trong suốt những năm học như Học sinh giỏi, Bằng khen các kỳ thi Tin học Quốc gia và Quốc tế. Mỗi bằng khen Minh để vào một khung hình chữ nhật kích thước W × H. Minh muốn treo tất cả vào một khung hình vuông kích thước K × K, mỗi bằng khen đều có cạnh song song với khung hình vuông và không chồng lên nhau. Để tiết kiệm chi phí và diện tích, Minh cần tính toán sao cho khung hình vuông có kích thước bé nhất.
Yêu cầu: Tìm K nhỏ nhất sao cho Minh có thể treo được tất cả N bằng khen.

Dữ liệu: vào từ file văn bản **BANGKHEN.INP** gồm 3 số W, H, N ($1 <= W, H, N <= 10^9$)

Kết quả: ghi ra file văn bản **BANGKHEN.OUT** ghi số K nhỏ nhất tìm được

Ví dụ:

BANGKHEN.INP	BANGKHEN.OUT
2 3 10	9

Giải thích:

Ràng buộc:
- 25% số điểm của bài tương ứng với các test có N, W, H <= 100
- 25% số điểm của bài tương ứng với các test có N, W, H <= 10000
- 50% số điểm còn lại không có ràng buộc nào thêm.

Bài 2: Sinh nhật
Minh chuẩn bị tổ chức sinh nhật tròn 15 tuổi tại nhà. Tại nhà mình, Minh có N chiếc thìa và M chiếc dĩa. Minh muốn sắp xếp thìa và dĩa sao cho thật đẹp, theo Minh cách bố trí đẹp là mỗi vị khách sẽ được chia một thìa và 2 dĩa, trong đó độ dài thìa phải lớn hơn một chiếc dĩa và nhỏ hơn chiếc dĩa còn lại.

Yêu cầu: Hãy cho biết Minh có thể mời tối đa được bao nhiêu khách?

Dữ liệu: vào từ file văn bản **SINHNHAT.INP** gồm:
- Dòng 1: N và M ($1 <= N, M <= 10^6$)
- Dòng 2: N số nguyên $a_1, a_2, \ldots, a_n$ ($0 <= a_i <= 10^9$) là độ dài của n chiếc thìa.
- Dòng 3: M số nguyên $b_1, b_2, \ldots, b_m$ ($0 <= b_i <= 10^9$) là độ dài của m chiếc dĩa

Kết quả: ghi ra file văn bản **SINHNHAT.OUT** 1 số duy nhất là số khách tối đa Minh có thể mời dự sinh nhật.

Ví dụ:

SINHNHAT.INP	SINHNHAT.OUT
4 4 3 4 1 3 2 1 7 0	2

Giải thích:

Ràng buộc:

- 50% số điểm của bài tương ứng với các test có N, M <= 1000
- 50% số điểm còn lại không có ràng buộc nào thêm.

Bài 3: Tên trộm

Có một dãy ngôi nhà dọc theo một con đường. Trong mỗi ngôi nhà, chủ nhân đều có cất một số tiền. Một tên cướp muốn trộm tiền trong các ngôi nhà đó nhưng không dám trộm các ngôi nhà liền kề nhau vì sợ để lại dấu vết. Tên cướp có khả năng trộm số tiền tối đa từ một ngôi nhà trong số tất cả các ngôi nhà mà nó có thể trộm.

Số tiền trong các ngôi nhà được cho trong một mảng số nguyên biểu thị ngôi nhà thứ i có A[i] usd. Cho số K là số nhà tối thiểu mà tên cướp sẽ trộm. Tên cướp có thể trộm ít nhất K căn nhà.

Yêu cầu: Cho biết số tiền tối thiểu mà tên cướp có thể trộm được trong tất cả các phương án có thể trộm từ ít nhất K ngôi nhà.

Dữ liệu: vào từ file văn bản **TENTROM.INP** gồm

- Dòng đầu chứa hai số nguyên N ($1 <= N <= 10^6$) và K ($1 <= K <= (N+1)/2$)
- Dòng tiếp theo chứa N số nguyên cách nhau khoảng trắng là số tiền có trong mỗi ngôi nhà ($1 <= a_i <= 10^9$).

Kết quả: ghi ra file văn bản **TENTROM.OUT** ghi số tiền trộm it nhất trong các phương án

Ví dụ:

TENTROM.INP	TENTROM.OUT
4 2 2 3 5 9	5

Giải thích:

Có ba phương án để trộm tiền từ ít nhất 2 ngôi nhà.
Số tiền lớn nhất có thể trộm từ nhà thứ 0 đến thứ 2 là 5
Số tiền lớn nhất có thể trộm từ nhà thứ 0 đến thứ 3 là 9
Số tiền lớn nhất có thể trộm từ nhà thứ 1 đến thứ 3 là 9
Do đó, số tiền tối thiểu tên cướp có thể trộm từ 3 phương án trên là 5

Ràng buộc:

- 30 % số điểm có N <= 10.
- 30 % số điểm có N <= 1000.
- 40 % số điểm không có giới hạn nào thêm

--------- HẾT ---------

Thí sinh không sử dụng tài liệu. Cán bộ coi thi không giải thích gì thêm

HƯỚNG DẪN GIẢI

Bài 1: Treo bằng khen

Dạng bài: Tìm kiếm nhị phân

Cấu trúc dữ liệu:

- Sử dụng biến số nguyên long long

Giải thuật:

Phân tích bài toán:

- Để treo được 1 bằng khen thì kích thước cạch của khung phải bằng cạnh lớn hơn của bằng khen.
- Để chắc chắn treo đủ N bằng khen thì kích thước cạnh lớn nhất của khung phải bằng số bằng khen nhân với cạnh lớn nhất của bằng khen (treo tất cả các bằng khen thẳng 1 hàng).
- Tuy nhiên ngoài phương pháp xếp các bằng khen thẳng hàng, ta có thể xếp các bằng khen thành nhiều hàng và cột để tối ưu kích thước khung.
- Để có được kích thước khung nhỏ nhất đủ treo tất cả các bằng khen ta duyệt tìm khung có kích thước từ đủ treo một bằng khen đến khung có kích thước đủ treo n bằng khen.
    - Mục tiêu: Tìm số nguyên dương K (kích thước khung vuông K * K) nhỏ nhất
    - Điều kiện: Khung K * K phải chứa được N tấm bằng khen kích thước W * H.
    - Cách xếp: Các bằng khen không được xoay, không chồng chéo.
        - Với chiều rộng K, ta xếp được K / W bằng khen.
        - Với chiều cao K, ta xếp được K / II bằng khen.
        - Tổng số bằng khen xếp được trong khung K * K là (K / W) * (K / H)
- Bài toán trở thành tìm số nguyên dương K nhỏ nhất sao cho K * K >= (K / W) * (K / H)

Subtask 1, 2: line sreach (brute force)

- ***Phân tích:***
    - Để kích thước khung đủ treo N bằng khen có chiều ngang W và chiều cao H thì kích thước khung phải lớn nhất sẽ là max(W, H) * N (treo các bằng khen trên 1 hàng)
    - Áp dụng tìm kiếm tuần tự tìm từ kích thước khung K là 1 tăng dần cho đến khi (K/W) * (K/H) >= N (treo bằng khen nhiều hàng, nhiều cột trong khung.
    - Vì duyệt tăng dần nên kết quả đầu tiên tìm thấy là kết quả nhỏ nhất.
- Các bước thực hiện:
    - Gọi C là kích thước cạnh khung lớn nhất có thể C = max(W, H) * N
    - Duyệt K = 1 đến C
        - Nếu (K / W) * (K / H) >= N
            - Đây chính là kích thước cạnh khung nhỏ nhất đủ treo N bằng khen có cạnh là W, H
            - Xuất kết quả và thoát chương trình.

*Độ phức tạp O(max(W, H) * N)*

Subtaks 3: binary sreach

- ***Phân tích:***
    - Ta nhận thấy kích thước khung K cần tìm luôn có thứ tự tăng dần nên ta có thể áp dụng tìm kiếm nhị phân để tìm ra giá trị K nhỏ nhất.
    - Gọi L là kích thước khung đủ treo 1 bằng khen, L = max(W, H)
    - Gọi R là kích thước khung đủ treo N bằng khen thẳng hàng, R = N * max(W, H)
    - Tìm kiếm nhị phân trong đoạn L, R để tìm ra giá trị M nhỏ nhất là cạnh của khung đủ treo N bằng khen theo dạng nhiều hàng và nhiều cột.
    - Điều kiện để xác định M là (M/W) * (M/H) >= N

*Độ phức tạp O(Ln(N*max(W, H))*

Bài 2: Sinh nhật

Dạng bài: Tham lam, Tìm kiếm nhị phân

Cấu trúc dữ liệu:

```
const int nm = 1e6+5;
pair <int, bool> b[nm];
int a[nm], m, n, kq = 0;
multiset <int> B;
```

- Mảng số nguyên A lưu kích thước của các thìa
- Mảng kiểu pair B lưu kích thước của các đĩa và trạng thái đĩa đã được sử dụng hay chưa
- Hoặc sử dụng multiset để lưu kích thước các đĩa. Tận dụng tính chất được các hàm sắp xếp, tìm kiếm và xóa

Giải thuật:

Phân tích bài toán

- Bài toán ghép cặp (matching)
  - Đầu vào: Chúng ta có N chiếc thìa (mảng A) và M chiếc đĩa (mảng B).
  - Yêu cầu: Mỗi vị khách cần 1 thìa (độ dài T) và 2 chiếc đĩa (độ dài D_1 và D_2).
  - Điều kiện: Phải thỏa mãn $D_1 < T < D_2$.
  - Mục tiêu: Tìm số lượng khách tối đa, tức là tìm số lượng bộ ba (T, D_1, D_2) tối đa có thể tạo ra từ N thìa và M đĩa, mà không dùng lại bất kỳ vật phẩm nào.
- Bài toán yêu cầu tìm số lượng khách tối đa, mỗi khách được chia một thìa và hai dĩa. Điều kiện quan trọng là độ dài của thìa S phải nằm giữa độ dài của hai chiếc dĩa D_1 và D_2, tức là $D_1 < T < D_2$ hoặc $D_2 < T < D_1$. Để đơn giản hóa, chúng ta có thể sắp xếp D_1 và D_2 sao cho $D_1 < D_2$
- Để tối ưu hóa số lượng khách, chúng ta nên sử dụng các thìa và dĩa một cách hiệu quả nhất nên ta cũng cần sắp xếp lại độ dài của thìa.

Subtask 1: Tư tưởng tham lam, tìm kiếm tuần tự

- ***Phân tích:***
  - Ta cần sắp xếp mảng chứa các thìa và các đĩa tăng dần
  - Duyệt qua từng cái thìa A_i (i = 1 đến M)
  - Với mỗi thìa A_i, ta sẽ tìm cặp đĩa (B_j, B_k) chưa được sử dụng sao cho ($B_j < A_i < B_k$)
  - Để tối ưu hóa số khách mời ta cần tìm cặp Bj và Bk gần nhau nhất.
    - Tìm B_j lớn nhất thỏa $B_j < A_i$ và chưa được sử dụng
    - Tìm B_k nhỏ nhất $A_i < B_k$ và chưa được sử dụng.
  - Nếu tìm được cả 2 giá trị Bj và Bk thì
    - Tăng số khách lên
    - Đánh dấu đã sử dụng
- ***Các bước thực hiện***
  - Duyệt qua từng cái thìa A_i (i = 1 đến N)
    - Tạo biến D1 = -1 lưu vị trí đĩa lớn nhất nhỏ hơn thìa Ai
    - Tạo biến D2 = -1 lưu vị trí đĩa nhỏ nhất lớn hơn thìa Ai
    - Duyệt qua từng cái đĩa Bj (j = 1 đến M)
      - Nếu đĩa Bj nhỏ hơn thìa Ai và chưa được sử dụng thì lưu vị trí j vào D1 ngược lại thì thoát
    - Nếu D1 = -1 có nghĩa là không tìm thấy đĩa nào nhỏ hơn thìa Ai đang xét thì ta bắt đầu xét lại thìa Ai tiếp theo
    - Duyệt qua từng cái đĩa Bj (j = D1+1 đến M)
      - Nếu đĩa Bj đang xét lớn hơn thìa Ai và chưa được sử dụng thì lưu vị trí j vào D1 và thoát vòng lặp
    - Nếu vị trí D1 nhỏ hơn D2 thì

- Tăng kết quả
- Đánh dấu vị trí đĩa D1 và D2 đã được sử dụng

Subtask 2, 3: Tư tưởng tham lam, tìm kiếm nhị phân

- ***Phân tích:***
  - Ta thấy giá trị các đĩa B đang có thứ tự tăng dần nên sẽ áp dụng tìm kiếm nhị phân để tìm ra 2 vị trí đĩa Bj lớn nhất nhỏ hơn Ai và Bj nhỏ nhất lớn hơn Ai
  - Dùng 2 hàm lower_bound và upper_bound để tìm ra 2 vị trí theo yêu cầu
  - Nếu tìm được 2 vị trí thì tăng biến kết quả
  - Xóa đĩa Bj tại 2 vị trí đó ra khỏi multiset
- ***Các bước thực hiện***
  - Duyệt qua từng cái thìa A_i (i = 1 đến N)
    - Tạo biến con trỏ D2 chỉ đến vị trí đĩa nhỏ nhất có kích thước lớn hơn thìa Ai
    - Dùng hàm upper_bound để tìm ra vị trí cái đĩa nhỏ nhất lớn hơn Ai lưu vào con trỏ D2
    - Nếu con trỏ D2 nằm ở cuối multiset có nghĩa là không tìm thấy. Xét đến thìa tiếp theo
    - Tạo biến con trỏ D1 chỉ đến vị trí đĩa đầu tiên có kích thước bằng thìa Ai
    - Nếu con trỏ D1 nằm ở đầu multiset có nghĩa là không tìm thấy. Xét đến thìa tiếp theo.
    - Giảm con trỏ D1 một đơn vị để lấy phần tử lớn nhất nhỏ hơn Ai
    - Tăng kết quả
    - Xóa đĩa ở vị trí D1 và D2 ra khỏi multiset.

*Độ phức tạp O(N*ln(M))*

Bài 3: Tên trộm

Dạng bài: Tìm kiếm nhị phân dựa trên kết quả lớn nhất nhỏ nhất

Cấu trúc dữ liệu:

```
const int nmax = 1e6+5;
int h[nmax];
```

- Mảng H lưu số tiền trong các ngôi nhà.

Giải thuật:

Phân tích bài toán:

- Đầu vào: Ta có N ngôi nhà trên một con đường, được biểu diễn bằng mảng A (với A[i] là số tiền ở nhà i). Ta cần trộm ít nhất K ngôi nhà.
- Ràng buộc: Không được trộm 2 nhà liền kề nhau.
- Mục tiêu: "Cho biết số tiền tối thiểu mà tên cướp có thể trộm được trong tất cả các phương án có thể trộm từ ít nhất K ngôi nhà."
  - Xét ví dụ N = 4, K = 2 A = {2; 3; 5; 9}
  - Có ba phương án để trộm tiền từ ít nhất 2 ngôi nhà.
  - Số tiền lớn nhất có thể trộm từ nhà thứ 0 đến thứ 2 là 5
  - Số tiền lớn nhất có thể trộm từ nhà thứ 0 đến thứ 3 là 9
  - Số tiền lớn nhất có thể trộm từ nhà thứ 1 đến thứ 3 là 9
  - Do đó, số tiền tối thiểu tên cướp có thể trộm từ 3 phương án trên là 5

Subtask 1: Đệ quy quay lui (back tracking)

Subtask 2, 3: *Tìm kiếm nhị phân dựa trên kết quả*

- ***Phân tích:***
  - Đây là một bài toán dạng "minimax" (tìm nhỏ nhất của lớn nhất), nên ta áp dụng tìm kiếm nhị phân theo kết quả (Binary Search on the Answer).

- Chúng ta cần tìm giá trị X nhỏ nhất. X có thể là bất kỳ giá trị nào từ 0 đến 10^9 (giá trị tiền tối đa). Ta có thể tìm kiếm nhị phân trên không gian kết quả này.
- Với một giá trị X bất kỳ (ta gọi là MID), ta cần một hàm Check(MID) để trả lời câu hỏi: "Liệu có tồn tại cách trộm lớn hơn K ngôi nhà không liền kề, mà chỉ trộm những nhà có số tiền A[i] <= MID hay không?"
  - Nếu Check(MID) trả về true, ta biết X có thể là MID hoặc nhỏ hơn. Ta thử tìm giá trị nhỏ hơn: R = MID - 1.
  - Nếu Check(MID) trả về false, ta biết MID là quá nhỏ. Ta cần X lớn hơn: L = MID + 1.

- ***Các bước thực hiện:***
  - Xác định ngôi nhà có số tiền nhỏ nhất và số tiền lớn nhất trong quá trình nhập dữ liệu. Lưu vào 2 biến L, R. Vì các giá trị từ L đến R luôn tăng dần nên ta áp dụng tìm kiếm nhị phân dựa trên giá trị từ L đến R để tìm ra phương án trộm số tiền nhỏ nhất của một phương án trộm từ K ngôi nhà.
    - Xây dựng hàm kiểm tra khi trộm ít nhất K ngôi nhà với mỗi ngôi nhà có số tiền nhỏ hơn hoặc bằng MID (MID = (L+R)/2)
      - Hàm có 3 tham số đầu vào là N (số ngôi nhà), K (số nhà tối thiểu có thể trộm), MID (giá trị tiền tối thiểu có thể trộm từ 1 phương án)
      - Duyệt i từ ngôi nhà đầu tiên đến ngôi nhà cuối cùng.
        - Nếu A[i] <= M thì
          - K—
          - i++
        - Vì không được trộm 2 ngôi nhà liên tục nên ta tăng i 1 lần nữa
        - Nếu K = 0 thì đây là một phương án trộm khả thi
      - Nếu duyệt đến cuối các ngôi nhà mà K > 0 thì không có phương án trộm với số tiền nhỏ hơn hoặc bằng MID
  - Dựa trên kết quả của hàm kiểm tra mà ta sẽ quyết đinh tăng L hay giảm R của phạm vi tìm kiếm nhị phân để tìm ra giá trị tiền trộm nhỏ nhất từ 1 phương án.

Độ phức tạp O(Nlog(10^9))