September 17, 2021

MÔ HÌNH tìm kiếm nhị phân

Thuộc: mô hình thiết kế thuật toán chia để trị

# Giới thiệu chuyên đề

Hưởng ứng chương trình giáo dục phổ thông 2018 và chương trình chuyên sâu mới, việc tìm hiểu các yêu cầu cần đạt cũng như đổi mới phương pháp tiếp cận nhằm đạt được mục tiêu của chương trình chuyên là một yêu cầu cấp thiết.

Chuyên đề các mô hình thiết kế thuật toán là một trong những trọng tâm cơ bản của chương trình chuyên Tin học lớp 10. Căn cứ vào các yêu cầu cần đạt và thời lượng dành cho nội dung này, có thể thấy, việc thực hiện chương trình là một nhiệm vụ không đơn giản. Để có thể đáp ứng được đa số nhu cầu học tập nội dung chuyên Tin của học sinh lớp chuyên, rất cần thiết phải thay đổi quan điểm, cách tiếp cận và phương pháp dạy học để có được hiệu quả cao ngay từ các chuyên đề mang tình chất cơ bản, cốt lõi.

Chuyên đề mô hình thiết kế thuật toán chia để trị cùng với mô hình Quay lui, mô hình Tham lam, Quy hoạch động tạo thành bộ công cụ quan trọng hỗ trợ thiết kế các thuật toán giải quyết một số lượng lớn các bài tập một cách hiệu quả. Trong đó, nổi bật vẫn phải kể tới một mô hình thiết kế dựa trên nguyên lí của thuật toán Tìm kiếm Nhị phân.

Được đánh giá là một trong những thuật toán phổ biến nhất trong lĩnh vực Tin học, mô hình tìm kiếm nhị phân có hiệu quả đặc biệt về thời gian chạy cũng như sự rõ ràng về cách thức thiết kế thuật toán. Có thể nói, tìm kiếm nhị phân không chỉ đơn thuần là thuật toán tìm kiếm mà còn là một mô hình tư duy thiết kế thuật toán phổ biến. Mô hình này dựa trên nguyên lí hoạt động của thuật toán tìm kiếm nhị phân, đó là tìm kiếm một phần tử trên một dãy giá trị đã được sắp xếp bằng cách chia nhỏ miền tìm kiếm, sau mỗi lượt không gian tìm kiếm được thu gọn giảm bớt một nửa.

Chuyên đề này thường được dạy ngay sau giai đoạn học sinh học xong phần kĩ thuật lập trình cơ bản. Thời điểm này, học sinh đã có thể sử dụng ngôn ngữ lập trình cùng với các công cụ có sẵn trong thư viện để thể hiện thuật toán. Tuy nhiên, khả năng tự mình nhìn nhận, đánh giá và thiết kế được một thuật toán cho bài toán mới còn hạn chế. Do đó, mục tiêu chính của phần này là phát triển kĩ năng thiết kế thuật toán theo mô hình chia để trị, tức là giáo viên cần quan tâm kĩ tới khả năng nhận dạng bài toán, khả năng nhìn ra mô hình tìm nhị phân trong bài toán và vận dụng mô hình tìm nhị phân để đưa ra được thuật toán phù hợp. Bên cạnh đó, cần quan tâm tới yêu cầu học sinh đánh giá độ phức tạp thuật toán.

Trong quá trình giảng dạy và học tập nâng cao chuyên môn, tôi tổng hợp thành chuyên đề này chú trọng tới việc tổ chức lại một cách có hệ thống các bài tập lí thuyết, các bài tập mẫu có hướng dẫn thiết kế thuật toán, cùng hệ thống bài tập vận dụng, bài tập tự luyện tập nhằm giúp các em học sinh có thể theo đó làm tài liệu học tập để đạt mức thành thạo ở nội dung này.

Mặc dù tác giả đã có đầu tư về công sức và thời gian để hoàn thành chuyên đề, nhưng cũng không thể tránh được các thiếu sót. Rất mong nhận được đóng góp ý kiến của các thầy cô giáo, học sinh để chuyên đề ngày càng được hoàn thiện và thực sự trở thành một chuyên đề hữu ích cho việc dạy và học.

# Tổng quan về mô hình tìm nhị phân

Dựa trên thuật toán tìm kiếm nhị phân, mô hình thiết kế thuật toán dựa trên dạng này thường có phát biểu tổng quát dạng: tìm kiếm một giá trị thoả mãn điều kiện nào đó trên một miền giá trị được sắp thứ tự.

Nhận dạng bài toán: Tìm một giá trị trên miền giá trị đã được sắp xếp.

Định hướng một số công việc cụ thể khi phân tích bài toán:

* Xác định giá trị cần tìm kiếm là gì?
* Xác định miền cần tìm: là miền có giá trị đã được sắp xếp đó là miền nào?
* Quá trình thực hiện:
  + Chia đôi miền tìm kiếm:
    - Xác định vị trí chốt (giữa)
    - Dựa trên mối quan hệ giữa vị trí chốt và giá trị cần tìm, tiến hành thu hẹp miền tìm kiếm (nửa trái, nửa phải)
    - Lặp lại quá trình tìm kiếm với miền giá trị mới.

Mặc dù thuật toán tìm kiếm nhị phân có nhiều cách cài đặt, ngoài ra thư viện STLC++ cũng có các hàm hỗ trợ nhưng việc tự mình cài đặt và hiểu được mô hình là rất quan trọng trong việc vận dụng thiết kế thuật toán theo mô hình tìm kiếm nhị phân. Dưới đây là một cách cài đặt phổ biến:

int solve(int a[], int x, int L, int R)

//tìm giá trị x trên dãy a[] từ chỉ số L tới chỉ số R

{

ans = -1;

while(L<=R){

int mid = (L+R)/2;

if(a[mid]==x) {

ans = mid;

break;

}

if(a[mid]<x) L = mid+1;

else R = mid-1;

}

return ans;

}

# Nhật xét chung về các dạng bài tập tìm nhị phân

Dưới đây là một số dạng phát biểu chung của những bài toán có thể giải quyết bằng mô hình tìm nhị phân. Tương ứng với mỗi dạng này, phần tiếp theo sẽ gồm các bài toán liên quan.

* Tìm phần tử có giá trị bằng k trên dãy a[] đã sắp xếp
* Lower\_bound: tìm phần tử đầu tiên có giá trị
* Upper\_bound: tìm phần tử đầu tiên có giá trị
* Tìm trên dãy đã sắp xếp quay vòng (nửa đầu tăng, nửa sau tăng) (Rotated array)
* Tìm cực trị địa phương (nửa đầu tăng, nửa sau giảm hoặc nửa đầu giảm, nửa sau tăng) (Find the peak)
* Tìm kiếm giá trị nghiệm (nếu có) của hàm f(x) đơn điệu trên đoạn [L,R] (Bisection method)
* Tìm kiếm trên miền kết quả (Binary search the answer)

Dạng tìm một phần tử trên dãy được sắp xếp là dạng cơ bản quen thuộc. Ta có thể mở rộng việc tìm kiếm nhị phân trên miền kết quả. Bắt đầu bằng miền tìm kiếm: là miền giá trị mà chắc chắn kết quả cần tìm sẽ thuộc vào đó. Với mỗi lần tìm kiếm, chia đôi miền đó.

Giả sử ta có hàm check(x) trả về true nếu kết quả của x là có thể, ngược lại trả về false. Với loại bài toán dạng này, ta thường tìm giá trị lớn nhất, nhỏ nhất của x sao cho check(x)=true.

* Nếu muốn tìm giá trị lớn nhất của x:
  + Nếu check(x) là true thì check(y)=true với mọi
  + Nếu check(x)=false, thì check(y)=false với mọi
* Nói cách khác, ta muốn giảm không gian tìm kiếm sử dụng hàm check ở trên theo dạng sau: TrueTrueTrueTrueTrueFalseFalseFalseFalse
* Khi đó, cần tìm vị trí mà thay đổi từ True thành False bằng TKNP

# Hệ thống bài tập cơ bản

## TKNP1 - cơ bản 1 (BINSEARCH.CPP)

(Nguồn: <https://www.spoj.com/problems/BSEARCH1/> )

Cho dãy số A gồm N số nguyên đã được sắp xếp tăng dần và Q truy vấn, mỗi truy vấn là một số nguyên X. Với mỗi truy vấn, hãy tìm vị trí xuất hiện của X trong A? Nếu không tồn tại giá trị X trong A, in ra -1.

Input:

* Dòng đầu ghi N, Q
* Dòng thứ hai ghi N số nguyên
* Q dòng tiếp theo mỗi dòng ghi một số nguyên X

Output:

* Với mỗi truy vấn, hãy in kết quả trên một dòng

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5 3  1 2 3 4 5  2  8  5 | 2  -1  5 |

**Sol1:** Tìm kiếm tuần tự. Yêu cầu đánh giá độ phức tạp + cài đặt sol1

**Sol2:** Tìm vị trí phần tử x trên dãy A đã sắp xếp tăng dần. Nếu không tồn tại x trong A, in ra -1.

Tìm kiếm nhị phân: tìm x, trên dãy A, từ phần tử low tới high.

int bsearch(int a[], int l, int r, int x){

int ans = -1;

while(l <= r){

int mid = (l+r)/2;

if(a[mid]==x){

ans = mid;

break;

}

else if(a[mid] < x){

l = mid + 1;

}

else{

r = mid - 1;

}

}

return ans;

}

## TKNP2 - cơ bản 2 (LOWERBOUND.CPP)

(Nguồn: <https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/> )

Cho dãy A được sắp xếp tăng dần . Có Q truy vấn, mỗi truy vấn là một số nguyên k:

* Với mỗi k, hãy in ra số đầu tiên bé nhất có giá trị lớn hơn hoặc bằng k gọi là P

Ví dụ:

Dãy A = {1,2,2,3,4,4,4,5,6,6}

Với k = 2

🡪 số đầu tiên nhỏ nhất có giá trị lớn hơn hoặc bằng tại vị trí id=2; Dãy số: A={1,**2**,2,3,4,4,4,5,6,6}

Input:

* Dòng đầu ghi N, Q
* Dòng thứ hai ghi N số nguyên
* Q dòng tiếp theo mỗi dòng ghi một số nguyên x.

Output:

* Với mỗi truy vấn, hãy in kết quả trên một dòng là số P; nếu không tồn tại giá trị lớn hơn hoặc bằng k, in ra -1

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 10 2  1 2 2 3 4 4 4 5 6 6  2  4 | 2  5 |

**Sol1:** Tìm tuần tự

**Sol2 cải** **tiến** : Tìm x đầu tiên nhỏ nhất có giá trị >=x

* ans = -1
* Khi L<=R
  + giữa = (L+R)/2;
  + Nếu a[giữa]<x thì tìm x trong [giữa+1, R]
  + Else tìm x trong [L, giữa-1], ans = giữa



* return ans

int lower(int a[], int l, int r, int x){

//Tìm phần tử đầu tiên có giá trị >=x

int ans = -1;

while(l<=r){

int mid = (l+r)/2;

if(a[mid] >= x){

r = mid-1;

ans = mid;

}

else{

l = mid + 1;

}

}

return ans;

}

## TKNP3 cơ bản 3 – Upper\_bound

(Nguồn: <https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/>)

Cho dãy A được sắp xếp tăng dần . Có Q truy vấn, mỗi truy vấn là một số nguyên k:

* Với mỗi k, hãy in ra số đầu tiên nhỏ nhất có giá trị > k gọi là P

Ví dụ:

Dãy A = {1,2,2,3,4,4,4,5,6,6}

Với k = 2

🡪 số đầu tiên nhỏ nhất có giá trị > k=2 tại id=4 A = {1,2,2,**3**,4,4,4,5,6,6}

Input:

* Dòng đầu ghi N, Q
* Dòng thứ hai ghi N số nguyên
* Q dòng tiếp theo mỗi dòng ghi một số nguyên x.

Output:

* Với mỗi truy vấn, hãy in kết quả trên một dòng là số P cần tìm

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 10 2  1 2 2 3 4 4 4 5 6 6  2  4 | 4  8 |

**Sols tìm nhị phân:**

* Giá trị cần tìm: số nhỏ nhất có giá trị >x
* Miền tìm kiếm A[1]…A[N] được sắp tăng dần
* Khi L<=R
  + Chốt mid = (L+R)/2
  + Nếu vị trí chốt > x:
    - chốt cũng là một nghiệm tiềm năng ans = mid
    - Thu hẹp miền tìm kiếm là nửa đầu dãy tức là [L, mid-1]
  + Ngược lại:
    - Tìm kiếm trên miền [mid+1, R]

Cài đặt:

int upper(int a[], int l, int r, int x){

//Tìm phần tử có giá trị nhỏ nhất mà >x

int ans = -1;

while(l <= r){

int mid = (l+r)/2;

if(a[mid]>x){

ans = mid;

r = mid - 1;

}

else{

l = mid +1;

}

}

return ans;

}

## Tìm phần nguyên bằng TKNP

(Nguồn: <https://leetcode.com/problems/sqrtx/> )

Cho số nguyên x, tìm giá trị căn bậc hai của x lấy phần nguyên.

Yêu cầu: Không sử dụng hàm thư viện.

Input:

* Số nguyên x

Output:

* Một số nguyên là kết quả.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 50 | 7 |
| 10 | 3 |

Hướng dẫn:

* Biết rằng giá trị cần tìm sẽ trong đoạn [0,x] 🡪 L=0; R=x: Gọi phần nguyên kết quả sqrt(x) là k thoả mãn k\*k>=x; tìm k đầu tiên, nhỏ nhất thoả mãn k\*k>=x
  + Nếu mid\*mid<=x thì tìm ở [mid+1, R]; ans=mid
  + Ngược lại tìm trong [L, mid-1]

Cài đặt:

int mySqrt(int x) {

int l = 0, r = x;

int ans;

while(l<=r){

long long mid = (l+r)/2;

if(mid\*mid<=x){

ans = mid;

l = mid+1;

}

else{

r = mid-1;

}

}

return ans;

}

## Function

Cho hàm với x là số thực và hai số nguyên a, b sao cho f(a)\*f(b)<0 và biết hàm f(x) liên tục trên đoạn [a,b]. Tìm nghiệm của f(x)=0 trên đoạn [a,b] (tức là tìm x sao cho f(x)=0)

Ví dụ: thoả mãn (tức là và ngược dấu)

🡪 giá trị nghiệm là

Input:

* Một dòng ghi m, n, p, a, b

Output:

* In ra một số x là nghiệm thoả mãn, lấy 4 số phần thập phân

Ví dụ:

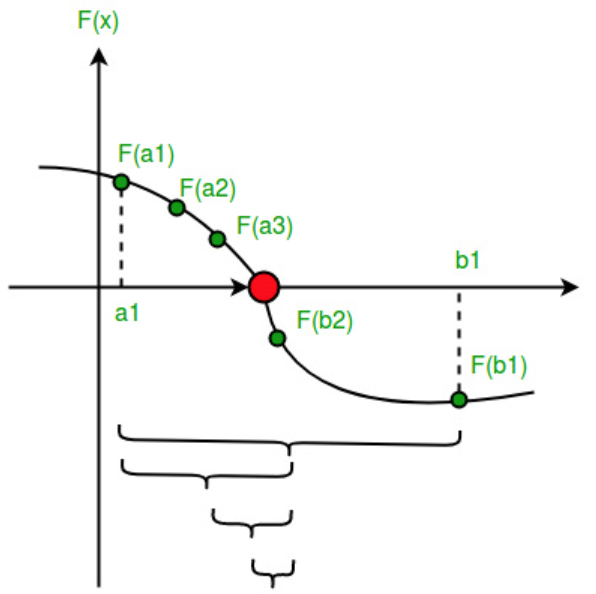
|  |  |
| --- | --- |
| Input | Output |
| 1 -1 2 -200 300 | -1.0025 |

**Sols:**

Nhận xét: f(x) liên tục trên [a,b] và f(a)\*f(b)<0

Như vậy, nghiệm của f(x) trên đoạn đó là một giá trị thuộc [a,b]

* L=a, R=b
* while(L-R<eps)
  + mid=(a+b)/2
  + Nếu f(mid)=0 thì mid là nghiệm cần tìm
  + Ngược lại:
    - Nếu f(a)\*f(mid)<0 tức là nghiệm sẽ thuộc [a, mid]
    - Ngược lại nếu f(b)\*f(c)<0 tức là nghiệm thuộc [mid, b]



Bài toán vận dụng:

[UVa 11881](https://onlinejudge.org/index.php?option=onlinejudge&Itemid=8&page=show_problem&problem=3003) - Internal Rate of Return (bisection method)

[1201C codeforces](https://codeforces.com/problemset/problem/1201/C)

## SORTEDROTATED

(Nguồn: <https://leetcode.com/problems/search-in-rotated-sorted-array/> )

Tìm một phần tử x trong dãy đã được sắp xếp và quay vòng.

Một dãy được sắp xếp và quay vòng có dạng: nửa dãy đầu có giá trị tăng dần, rồi dãy tiếp theo cũng tăng dần. Dãy này có được từ dãy tăng dần dịch trái k vị trí.

Ví dụ: A={3,4,5,1,2}; {4, 5, 1, 2, 3} là một dãy quay vòng đã được sắp xếp. Cần tìm một phần tử có giá trị x trong dãy với độ phức tạp O(logn) bằng áp dụng thuật toán tìm kiếm nhị phân.

Input:

* Dòng đầu ghi số lượng testcase T . Mỗi testcase gồm:
  + Số nguyên N (1 ≤ N ≤ 107)
  + Dòng thứ hai ghi N số nguyên cách nhau bởi dấu cách, các phần tử được đánh chỉ số từ 0 (|ai| ≤ 108, 0 ≤ i ≤ N)
  + Dòng thứ 3 ghi số nguyên x (1 ≤ x ≤ 108)
  + Dữ liệu đảm bảo dãy đã cho là dãy được sắp xếp vòng, các phần tử trong dãy khác nhau đôi một (ai ≠ aj với ∀i ≠ j)

Output: Với mỗi test, ghi một số nguyên là vị trí phần tử x trong dãy, nếu không tồn tại phần tử x, hãy in ra -1.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 3  9  5 6 7 8 9 10 1 2 3  10  3  3 1 2  1  4  3 5 1 2  6 | 5  1  -1 |

**Sols 1:**

* Tìm vị trí a[i] mà thoả mãn a[i-1]<a[i] && a[i]>a[i+1] là vị trí chia dãy thành 2 dãy tăng A[1..i] và A[i+1…N]
* Tìm chốt bằng tknp: điểm chốt là điểm duy nhất trên dãy thoả mãn a[i]<a[i-1] và a[i]<a[i+1]
  + l=1; r=n;
  + ans = -1
  + while(l<=r)
    - if(a[mid]<a[mid-1] && a[mid]<a[mid+1])
      * ans = mid; break;
    - else if(a[mid]>a[0]) tìm trong nửa sau l=mid+1
    - else tìm trong nửa trước r = mid-1
  + return ans
* Xét xem x thuộc nửa nào; áp dụng TKNP trên nửa đó
* Tìm a[i] bằng TKNP:
  + Nếu Mid thoả mãn a[mid-1]<a[mid]<a[mid+1] thì tìm x trong nửa sau; ngược lại tìm trong nửa trước.

int findX(int a[], int l, int r, int x){

//Tìm x trong dãy rotated array

int ans = -1;

int p = bsFindChot(a,l,r);

if(p==-1){ //Nếu dãy không phải dãy vòng mà là dãy

ans = bsearch(a, l, r, x);

}

else{

if(x>=a[l]){

ans = bsearch(a,l,p-1,x);

}

else{

ans = bsearch(a,p,r,x);

}

}

return ans;

}

Khi chặt nhị phân, giá trị chốt sẽ thuộc 1 trong 3 trường hợp sau:

* Chốt ở trên đoạn tăng phía trước: khi đó a[mid]>a[0] và a[mid]<a[mid+1] (hình vị trí (1))
* Chốt ở trên đoạn tăng phía sau: khi đó a[mid]<a[0] và a[mid]<a[mid+1] (hình vị trí (2))
* Chốt ở đúng vị trí cần tìm: a[mid]<a[mid-1] và a[mid]<a[mid+1]; cả dãy chỉ có một vị trí thoả mãn điều kiện này (hình vị trí (3))



Cài đặt bằng 1 lần tìm nhị phân:

int search(vector<int>& a, int target) {

int n = a.size();

int left = 0, right = n - 1;

while(left <= right) {

int mid = left + (right - left) / 2;

if(a[mid] == target) {

return mid;

}

if((target >= a[0] && (a[mid] >= target || a[mid] < a[0])) || (target < a[0] && (a[mid] < a[0] && a[mid] >= target))) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return -1;

}

## TKNP7 - PEAK cực đại địa phương

(Nguồn: <https://leetcode.com/problems/find-peak-element/> )

Phần tử cực đại địa phương trong một dãy là phần tử thỏa mãn điều kiện có giá trị không nhỏ hơn các phần tử kế bên.

Ví dụ:

* Dãy A = {5, 10, 20, 15} thì 20 là cực trị địa phương vì 20>10 và 20>15
* Dãy B = {10, 20, 15, 2, 23, 90, 67} có hai cực đại địa phương là 20 và 90

Cho dãy A gồm N phần tử . Hãy tìm vị trí phần tử là cực đại địa phương trong dãy?

Input:

* Dòng đầu ghi N
* Dòng thứ hai ghi dãy A gồm N số nguyên
* Dữ liệu đảm bảo trong dãy chỉ tồn tại **đúng 1 cực đại địa phương**.

Output:

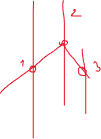
* Ghi một số nguyên là chỉ số của phần tử là cực đại địa phương trong dãy

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 2  3 4 | 1 |
| 3  1 2 3 | 2 |
| 6  1 2 5 3 1 | 2 |

Sols: Chặt nhị phân; các vị trí chốt có thể là các vị trí 1, 2, 3 trên hình

1. nếu a[mid]>a[mid-1] && a[mid]<a[mid+1]
2. Nếu a[mid]>a[mid-1] && a[mid]>a[mid+1]
3. Nếu a[mid]<a[mid-1] && a[mid]>a[mid+1]



# Hệ thống bài tập vận dụng

## TKNP4 – Count x

Cho dãy A gồm N phần tử đã được sắp xếp tăng dần và Q truy vấn, mỗi truy vấn gồm một số nguyên x yêu cầu đếm số lượng phần tử có giá trị là x?

Input:

* Dòng đầu ghi N, Q
* Dòng thứ hai ghi N số nguyên
* Q dòng cuối, mỗi dòng ghi một số nguyên x

Output:

* Với mỗi truy vấn, hãy in kết quả trên một dòng.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 10 3  1 2 2 3 4 4 4 5 6 6  2  4  9 | 2  3  0 |

**Sols:** Với mỗi truy vấn:

* Tìm id1 là chỉ số vị trí x đầu tiên, id2 là chỉ số vị trí ngay sau x cuối cùng 🡪 số lượng là id2-id1
* id1 = lower\_bound; id2=upper\_bound

int countx(int a[], int l, int r, int x){

//Đếm trong dãy a từ a[l] đến a[r] số lượng phần tử x

int id1 = lower(a, l, r, x);

int id2 = upper(a, l, r, x);

return id2-id1;

}

## TKNP5 – COUNTONE.CPP – Vận dụng 1

Cho dãy A chỉ gồm các số nguyên 0, 1; Hãy đếm số lượng phần tử có giá trị 1 trong dãy đã được sắp xếp giảm dần.

Input:

* Dòng đầu ghi số nguyên n là số lượng phần tử trong dãy (1 ≤ n ≤ 106)
* Dòng thứ hai ghi n số nguyên, mỗi số có giá trị là 0 hoặc 1.
* Dữ liệu đảm bảo dãy đã được sắp xếp giảm dần.

Output: Một số nguyên là số lượng phần tử 1 trong dãy

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5  1 1 1 0 0 0 | 3 |
| 3  1 1 1 | 3 |
| 4  0 0 0 0 | 0 |

Yêu cầu ĐPT O(logn)

Sols 1 (vận dụng các bài trên): id1 = lower(a,1,n,0); id2=upper(a,1,n,0) 🡪 kq=id2-id1

Sols 2: tìm nhị phân trực tiếp trên dãy

* Nhận xét: dãy toàn số 0, rồi toàn số 1 🡪 Tìm vị trí i thoả mãn a[i]<a[i-1] 🡪 số lượng số 0 là n-i



* Tìm nhị phân thế nào:
* L=1; R=n
* while(L<=R)
  + mid=(L+R)/2
  + Nếu a[mid]=1 && a[mid+1]=0 thì ans=mid
  + else nếu a[mid]=1 && a[mid+1]!=0 thì tìm ở [mid+1,R]
  + ngược lại nếu a[mid]=0 và a[mid+1]=0 thì tìm ở nửa đầu [L,mid-1]

int countone(int a[], int l, int r){

int ans = 0;

while(l<=r){

int mid = (l+r)/2;

if(a[mid]==1 && a[mid+1]==0){

ans = mid;

break;

}

else if(a[mid]==1 && a[mid+1]!=0){

l = mid+1;

}

else{

r = mid-1;

}

}

return ans;

}

## POSZERO – Vị trí số 0 cuối cùng

Cho một xâu kí tự 𝑆 chỉ toàn kí tự số 0 và 1. Biết rằng các kí tự số 0 đứng trước các ký tự số 1: 000....0011...11.

Yêu cầu: Hãy cho biết vị trí của ký tự số 0 cuối cùng trong dãy.

Input:

* Chứa xâu kí tự S (Độ dài của xâu 𝑆≤107 và luôn có ký tự số 0 và ký tự số 1)

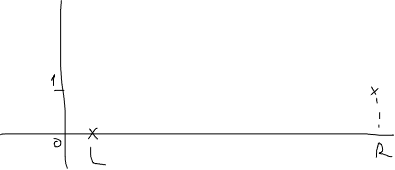
Output:

* Một số nguyên dương là vị trí của ký tự số 0 cuối cùng trong dãy.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 0000011111 | 5 |

Gợi ý: tương tự bài trước. Tìm vị trí đầu tiên có f(i)!=f(i-1)



Sols tượng tự bài countone.

## Tìm số nhỏ nhất trên dãy tăng đã dịch phải k vị trí

Bài này tương tự bài Tìm giá trị k trên dãy quay vòng (Bài 6 – phần cơ bản). Tìm nhị phân đúng vị trí (1) có giá trị nhỏ nhất thoả mãn điều kiện đây là vị trí duy nhất trên dãy.



## Khieuvu3 - Khiêu vũ 3

(Nguồn: <http://ntucoder.net/Problem/Details/5571> )

Câu chuyện tình yêu Elo Cruz và Mara ở Philippines là một minh chứng cho tình yêu đích thực, không màng đến ngoại hình. Họ khiến cư dân mạng thế giới phải khâm phục vì một tình yêu bất chấp những khác biệt về ngoại hình. Tuy nhiên admin rất lo lắng cho Elo, không biết anh chàng này sẽ phải chọn cây ghế cao thế nào để hôn vợ. Cho nên trong buổi tiệc khiêu vũ “Cơn gió đêm hè!” sắp đến đây admin muốn các cặp đôi có chiều cao chênh lệch phải đúng bằng K mới được khiêu vũ cùng nhau.

Bạn hãy tính giúp cho admin xem có thể có bao nhiêu cách sắp xếp từng cặp đôi với nhau thỏa mãn?

Input:

* Dòng đầu tiên là N - số lượng người tham gia bữa tiệc và số K (N≤10^5, K≤10^9)
* Các dòng tiếp theo là chiều cao của N người tham gia bữa tiệc – không có 2 người nào có chiều cao giống nhau. (Hi ≤ 10^9)

Output:

* Gồm một số duy nhất là số cách lớn nhất có thể sắp xếp

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 6 2  1 3 2 4 9 5 | 3 |

Tóm tắt đề: cho độ cao của N người. Đếm số cặp A[i], A[j] mà |A[i]-A[j]|=K?

Sols 1 trâu Duyệt và đếm các cặp (i,j) thoả mãn đk.

Sols 2: TKNP:

* Sắp xếp lại dãy theo độ cao của người: 1 2 3 4 5 9
* Với người i, tìm người tại j thoả mãn điều kiện A[j]-A[i]=K tức là A[i]+K=A[j]
* Với mỗi i, tìm trong dãy A phần tử A[i]+K 🡪 nếu tồn tại thì đếm

A=[1 2 3 4 5 9]; K=2

* Tìm A[1]+2=3 🡪 có
* Tìm A[2]+2=4 🡪 có
* Tìm A[3]+2=5 🡪 có
* Tìm A[4]+2=6 🡪 không có
* Tìm A[5]+2=11 🡪 không có

## DIKI - Sân điền kinh

(Nguồn: <http://ntucoder.net/Problem/Details/135> )

Trong một sân điền kinh, người ta kẻ các vạch sơn để thi đấu các cự ly khác nhau. Có tất cả n vạch sơn. Vạch sơn thứ i cách đầu sân khoảng cách là ai mét (a1 < a2 < a3 <...< an). Với một cự ly chạy thi đấu là m mét, Ban tổ chức cần tìm 02 vạch sơn để thi đấu cự ly này. Nghĩa là tìm ra vạch sơn xuất phát ai và vạch sơn kết thúc aj sao cho aj - ai = m. Bạn hãy giúp ban tổ chức tìm ra hai vạch sơn này nhé.

Input:

* Dòng đầu ghi hai số nguyên n, m
* Dòng thứ hai ghi n số nguyên

Output:

* Nếu tồn tại cách xác định hai vạch sơn, hãy in ra hai số x, y thể hiện hai vạch sơn cần tìm (y-x=m). Nếu có nhiều đáp án, hãy in ra đáp án bất kì.
* Nếu không có cách xác định hai vạch sơn, hãy in ra -1

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 4 2  1 5 7 10 | 5 7 |
| 4 3  1 5 7 10 | 7 10 |
| 4 5  1 5 7 10 | 5 10 |
| 4 2  1 4 7 10 | -1 |

Tóm tắt đề bài: Tìm các cặp số a[i], a[j] sao cho a[i]+m=a[j]

**Sols trâu:** theo đề bài, dễ thấy dãy a tăng dần. Với mỗi vạch sơn i, tìm vạch sơn thoả mãn a[i]+m

**Cải tiến** tìm j thoả mãn a[j]=a[i]+m bằng tìm kiếm nhị phân:

* Với mỗi i: tìm a[i]+m trên dãy a[i]…a[n]

## NAUAN - Thi nấu ăn

(Nguồn: <http://ntucoder.net/Problem/Details/1177>)

Có n bạn sinh viên đang tham gia dự thi nấu ăn nhân dịp năm mới và được đánh số báo danh từ 1 đến n, bạn sinh viên thứ i tham dự với số lượng là ai món ăn. Ban tổ chức sẽ đánh số các món ăn dự thi như sau: các món ăn của thí sinh thứ nhất đánh số từ 1 đến a1, các món ăn của thí sinh thứ hai đánh số từ a1+1 đến a1+a2.... và tương tự như vậy cho đến món cuối cùng.

Sau khi chấm thi, Ban tổ chức chọn trao giải cho m món ăn với các số hiệu là p1, p2, ..., pm.

Hãy cho biết các món ăn đạt giải đó thuộc về các bạn sinh viên nào?

Input:

* Dòng thứ nhất là số nguyên n (1 ≤ n ≤ 105) là số thí sinh tham gia dự thi.
* Dòng thứ hai là n số nguyên a1, a2, ..., an (1 ≤ ai ≤ 104) là số lượng món ăn của từng thí sinh, mỗi số cách nhau một khoảng trắng.
* Dòng thứ ba là số nguyên m (1 ≤ m ≤ 104) là số lượng món ăn đạt giải.
* Dòng thứ tư là m số nguyên p1, p2, ..., pm là số hiệu của m món ăn đạt giải, mỗi số cách nhau một khoảng trắng.

Output:

* Gồm m số là số báo danh thí sinh của từng món ăn đạt giải (món ăn là của thí sinh có số báo danh )

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5  5 4 1 2 3  3  5 6 12 | 1 2 4 |

Tóm tắt đề:



**Sols:**

* Tính lại các b[i] là tổng các a[1]+…+a[i] (b[i] = a[i-1]+a[i])
* Với mỗi p[i]: tìm vị trí j mà b[j] đầu tiên có giá trị >=p[i]

## MOBI - Đặt trạm phủ sóng (OLP 30/4 Khối 10)

(Nguồn: <http://ntucoder.net/Problem/Details/4408> )

Nhà cung cấp dịch vụ viễn thông Mobi đã khảo sát số lượng người sẽ dùng dịch vụ trên một con đường thẳng mới được xây dựng và đánh dấu lại những vị trí trên con đường này. Đầu con đường được đánh tọa độ bắt đầu từ 0. Tại vị trí có tọa độ X (X nguyên dương) có số lượng người sẽ sử dụng dịch vụ là Y. Trước mắt, nhà cung cấp dịch vụ cần đặt một trạm phát sóng có bán kính phủ sóng là K đơn vị chiều dài để phủ sóng cho một số người sử dụng dịch vụ trên con đường này.

Yêu cầu: Bạn hãy xác định vị trí đặt trạm phát sóng (tọa độ nguyên dương) sao cho trạm có thể phục vụ được số lượng người sử dụng nhiều nhất có thể.

Input:

* Dòng đầu ghi N, K với N là số điểm dân cư đã được đánh dấu, K là bán kính phủ sóng của trạm
* Trong N dòng tiếp theo, dòng thứ i ghi hai số nguyên X, Y cho biết tại vị trí X có Y người dùng

Output:

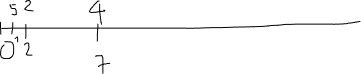
* In ra một số nguyên là số lượng người dùng nhiều nhất sẽ được phục vụ

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 4 3  7 4  15 10  2 2  1 5 | 11 |

Chọn vị trí đặt trạm phủ sóng tại X=4. Như vậy có thể phủ sóng đến các vị trí có toạ độ 1, 2, 7. Số lượng người sử dụng lớn nhất là 11.

Tóm tắt:



**Sols 1**: (Sliding windows – prefix sum) ĐPT O(n^2)

* Tính sẵn các A[i] là số lượng người được phục vụ tại toạ độ i 🡪 test mẫu

A[0]=0; a[1]=5; a[2]=2; a[3]=0; a[4]=0;…; a[7]=4;… a[15]=10

* Tìm vị trí i đặt trạm sao cho tổng các x từ i-K tới i+K là lớn nhất (đoạn có độ dài 2K)
* Xét từng vị trí i: tính tổng số người trong vùng phủ sóng (i-K, i+k) 🡪 sliding windows kích thước 2K; tổng sau = tổng trước – biên trái + biên phải 🡪 tìm max

Cải tiến bằng chặt nhị phân kiểu gì? cải tiến yêu cầu tìm kiếm ở trên (màu vàng)

* Tính lại a[i] là tổng số người được phục vụ trong đoạn từ x=0 tới x=i (mảng cộng dồn)
* Với mỗi vị trí i, tìm vị trí j lớn nhất thoả mãn điều kiện: j<=i+2k. Tổng số người được phục vụ trong đó là a[j]-a[i-1]



## TIVI2 – Truyền hình 2 (OLPCĐ 2011)

(Nguồn: <http://ntucoder.net/Problem/Details/66> )

Khi có quá nhiều kênh truyền hình với rất nhiều chương trình giải trí thú vị, bạn sẽ chọn lựa xem những chương trình nào? Đây quả là một câu hỏi khó.

Có n chương trình giải trí, chương trình thứ i (1 ≤ i ≤ n) có thời điểm bắt đầu là và thời điểm kết thúc là . Chương trình giải trí thứ i và chương trình giải trí thứ j (với 1 ≤ i < j ≤ n) được gọi là không phù hợp với nhau về lịch phát sóng nếu người xem không thể xem trọn vẹn nội dung của cả hai chương trình giải trí này. Nếu thời điểm kết thúc của chương trình là thời điểm bắt đầu của chương trình thì hai chương trình này vẫn được xem là có lịch phát sóng phù hợp với nhau.

Ví dụ: Có 3 chương trình giải trí như sau: Chương trình 1 , chương trình 2 , chương trình 3 . Chương trình 1 và chương trình 2 có lịch phát sóng phù hợp với nhau. Tương tự, chương trình 1 và chương trình 3 cũng được xem là có lịch phát sóng phù hợp với nhau. Tuy nhiên, chương trình 2 và chương trình 3 có lịch phát sóng không phù hợp với nhau.

Yêu cầu: Cho biết kế hoạch phát sóng của N chương trình giải trí, hãy xác định có bao nhiêu cặp chương trình có lịch phát sóng không phù hợp với nhau?

Input:

* Dòng đầu tiên chứa một số nguyên dương n (với n ≤ 50.000).
* Dòng thứ i trong số n dòng tiếp theo (1 ≤ i ≤ n), mỗi dòng gồm hai số nguyên dương si và ti là thời điểm bắt đầu và thời điểm kết thúc của chương trình giải trí thứ i (với ). Các số trên cùng một dòng được ghi cách nhau bởi 1 khoảng trắng.

Output:

* In ra một số nguyên là số lượng cặp chương trình có lịch phát sóng không phù hợp với nhau.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 3  7 10  12 15  10 20 | 1 |
| 3  1 3  3 5  5 7 | 0 |

**Sols duyệt trâu: O(n^2)**

* Sắp xếp lại các chương trình theo tăng dần thời gian bắt đầu
* Với mỗi chương trình có thời gian bắt đầu là a[i], b[i]; Tìm các chương trình có thời gian bắt đầu a[j]>a[i] 🡪 mỗi a[j] thoả mãn là một chương trình không phù hợp với a[i]



**Sols cải tiến – Tìm nhị phân:**

* Với mỗi i, tìm a[j] xa nhất thoả mãn a[j]<b[i] 🡪 số lượng chương trình không phù hợp với a[i] là (j-i)
* Tìm j bằng tìm kiếm nhị phân trên dãy tăng dần

## Chênh lệch

(Nguồn: <http://ntucoder.net/Problem/Details/5574>)

Các bài tập về dãy số rất hay và thường xuyên gặp trong các kỳ thi của các NTUcoder. Hôm nay bài tập thử thách cho các bạn là tìm độ chênh lệch như sau:

Cho một dãy số gồm N số nguyên dương A1, A2, … An. Với một cặp số (u,v) bạn hãy tìm độ chênh lệch bé nhất khi chia đoạn con [Au .. Av] thành hai phần.

Bạn hãy xem ví dụ:

* dãy A: 3 1 4 2 5
* Với cặp (2,5) bạn cần tìm chênh lệch nhỏ nhất khi chia đoạn này ra hai phần. Ví dụ có các cách chia sau: (1) và (4,2,5); (1,4) và (2,5); (1,4,2) và (5); (1,4,2,5) và (); Khi đó chênh lệch bé nhất là 2 - Tương ứng với cặp (1,4) (2,5).

Input:

* Dòng đầu tiên là hai số nguyên dương N, Q (1 ≤ N, K ≤ 10^5). Q là số lượng truy vấn.
* Dòng tiếp theo là N số nguyên dương trong dãy A (Ai ≤ 10^9)
* Q dòng tiếp theo mỗi dòng là 2 số nguyên dương (u,v). Mỗi cặp số là một truy vấn cần bạn trả lời.

Output:

* Gồm Q dòng, dòng thứ i là giá trị chênh lệch nhỏ nhất tìm được ứng với truy vấn i.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5 1  3 1 4 2 5  2 5 | 2 |

Tóm tắt đề: cho dãy số, với mỗi cặp (u,v) tìm độ chênh lệch nhỏ nhất khi chia đoạn con a[u]…a[v] thành hai phần.

Yêu cầu: HS biết tính tổng đoạn con bằng mảng cộng dồn.

**Sols1:** Trâu + mảng cộng dồn

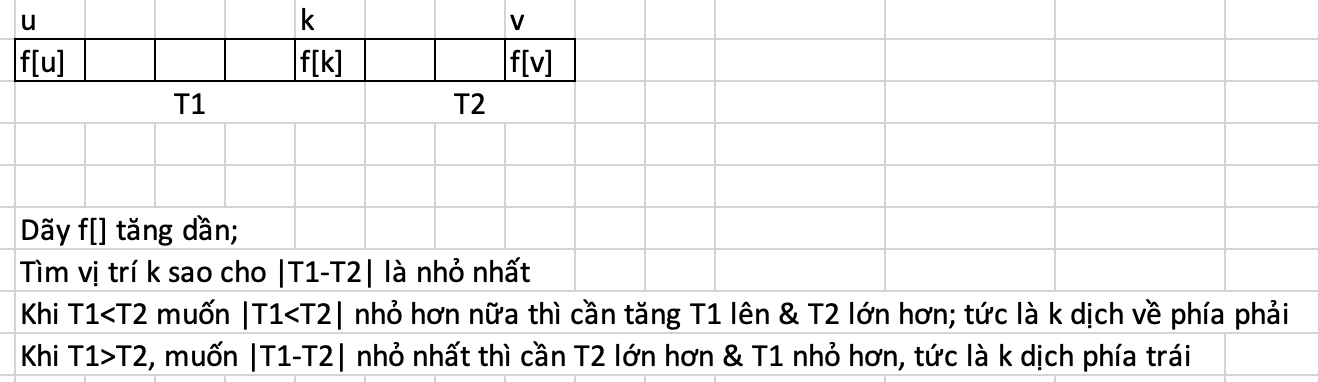
* Với mỗi cặp (u,v) thử chia ở vị trí có chỉ số k (u<=k<=v):
  + Tính tổng phần trước từ (u,k) là T1
  + Tính tổng phần sau từ (k+1,v) là T2
  + Tìm chênh lệch |T1-T2| lớn nhất

ĐPT O(Q\*N) nếu đã biết dùng mảng cộng dồn tính tổng đoạn con. Nếu không ĐPT là P(Q\*N\*N)

**Sols 2:** Tìm nhị phân – Cải tiến: thay vì duyệt từng cách chia (vị trí k): ta tìm giá trị k thuộc [u,v] thoả mãn điều kiện sum(A[u],…,a[k])-sum(a[k+1],…,a[v]) là lớn nhất.

Tại sao tìm nhị phân lại đúng??







## TAHO2 - Tặng hoa 8/3

(Nguồn: <http://ntucoder.net/Problem/Details/3260>)

Nhân ngày Quốc tế Phụ nữa 8/3 sắp tới, bạn An quyết định mua một bó hoa thật đẹp tặng cho Vân - cô gái mà cậu luôn thầm thương trộm nhớ và cũng là cơ hội để bày tỏ tình cảm của mình. An đi đến một cửa hàng đồng giá, ở đây tất cả các loại hoa đề được bán với cùng một giá, tuy nhiên với mỗi loại hoa, bạn chỉ có thể mua tối đa 1 bông hoa để tránh việc các loại hoa quý và đẹp bị mua hết. Vân có những quan niệm rất đặc biệt, theo cô thì mỗi loại hoa đều tượng trưng cho một con số may mắn nào đó và bạn An cũng chúng ta đã bằng một cách nào đó có được danh sách các loại hoa và con số tương ứng này. Vân cũng có một con số yêu thích S của mình, cá tính của cô ấy rất đặc biệt, nếu bất kì ai tặng quà cho cô ấy mà không có liên quan đến con số này thì cô ấy sẽ giận ngay. Tất nhiên là An không muốn điều này xảy ra vì như thế thì xem như cơ hội bày tỏ tình cảm cũng không còn, như thế thì thà không tặng còn hơn.

An quyết định sẽ tặng cho Vân một bó hoa có tổng các con số may mắn của các đóa hoa đúng bằng S. Nhưng vì đang là sinh viên, không có nhiều tiền nên An muốn tìm ra cách mua bó hoa sao cho tổng chi phí phải trả là thấp nhất. Bạn hãy giúp An thực hiện việc này nhé.

Input:

* Dòng đầu tiên chứa 3 số nguyên N, S, P. N là số lượng hoa có trong cửa hàng (1 ≤ N ≤ 40), S là con số yêu thích của Vân (0 ≤ S ≤ 109) và P là giá của mỗi đóa hoa (1 ≤ P ≤ 106).
* Dòng tiếp theo chứa N số nguyên dương Ai không vượt quát 109 là con số may mắn tương ứng với loại hoa thứ i.

Output: Nếu có cách mua hoa thỏa yêu cầu thì in ra giá của bó hoa mà An phải chi trả. Nếu không tìm ghép được bó hoa ưng ý thì in ra -1.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 10 390 2  200 10 20 20 50 50 50 50 100 100 | 10 |

Tóm tắt đề: Tìm tập con (mỗi phần tử lấy tối đa 1 lần) sao cho tổng đúng bằng S

## Hàng cây – Cưa máy

Trong khu vườn người ta trồng một hàng cây chạy dài gồm N cây, mỗi cây có độ cao là a1, a2,…, aN.

Người ta cần lấy M mét gỗ bằng cách đặt cưa máy sao cho lưỡi cưa ở độ cao H mét để cưa các cây có độ cao lớn hơn H.

Ví dụ: Nếu hàng cây có độ cao 20, 15, 10, 18; cần lấy M=7 mét. Khi đó lưỡi cưa nên đặt ở độ cao H=15 là hợp lý; khi đó độ cao các cây còn lại sẽ là 15, 15, 10, 15 🡪 Tổng số gỗ thu được là 8 mét (dư 1 mét so với yêu cầu)

Yêu cầu:

* Tìm vị trí đặt cưa hợp lý (số H lớn nhất) sao cho lấy được M mét gỗ và số gỗ dư ra là ít nhất.

Input:

* Dòng đầu ghi N, M
* Dòng thứ hai ghi N số nguyên

Output:

* Ghi ra một số nguyên là giá trị H lớn nhất cần tìm

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 4 7  20 15 10 18 | 15 |

**Sols1 – trâu:**

* + Thử mọi độ cao H đặt cưa:
  + Tính lượng gỗ thu được ở mỗi độ cao H; tìm giá trị tối ưu

**Sols2 tìm nhị phân:**

* Đặt cưa từ đỉnh lùi dần xuống dưới sẽ có được số gỗ dư tăng dần 🡪 H thử từ 0 tới 🡪 Cải tiến đoạn tìm H bằng tknp. L=0; R=
  + Giả sử đặt cưa tại vị trí Mid=(L+R)/2
  + Tính lại tổng gỗ thu được nếu đặt ở vị trí Mid: S = (tổng các a[Mid]-H>0)
  + Nếu tổng S tại đây <M 🡪 Tìm vị trí đặt cưa trong ở nửa dưới [L=0;R=Mid-1]
  + Nếu tổng tại S >M 🡪 Tìm vị trí đặt cưa ở nửa trên [L=Mid+1; R]

## Tìm phần tử nhỏ thứ k trong hai dãy A, B đã sắp xếp tăng dần

Cho hai dãy kích thước N, M phần tử đều là số nguyên và được sắp xếp tăng dần A, B và số nguyên k. Hãy tìm phần tử nhỏ thứ k trong hai dãy đó?

Input:

* Dòng đầu ghi N, M, k
* Dòng thứ hai ghi N số nguyên thuộc dãy A
* Dòng thứ ba ghi M số nguyên thuộc dãy B

Output:

* In ra một số nguyên là số thứ k cần tìm

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5 4 5  2 3 6 7 9  1 4 8 10 | 6 |

Giải thích: hai dãy sau khi gộp lại là 1 2 3 4 6 7 8 9 10 🡪 Phần tử thứ 5 là 6

**Sols1: trâu**

* Nối hai dãy đã sx thành dãy C đã sx, giá trị cần tìm là c[k]
* ĐPT O(n) do khi gộp hai dãy thành 1 dãy C, mỗi phần tử trong dãy được duyệt 1 lần
* Bộ nhớ cần dùng N+M (kích thước mảng C)

**Sols 2: cải tiến**

Nhận xét: với sol1 khi nối thành dãy C, kích thước của C quá lớn 🡪 tốn nhiều bộ nhớ 🡪 Tìm phần tử nhỏ thứ k mà không cần tạo dãy C?

* Trong quá trình duyệt A, B để nối thành C: tìm và lưu phần tử nhỏ nhất của hai phần đã được trộn 🡪 Khi nào tìm tới phần tử có giá trị nhỏ thứ k thì dừng
* ĐPT O(k)

**Sols3: TKNP 1** – không cần tạo dãy trộn C, chia đôi dãy A, B – ĐPT O(logn + logm)

* Do dãy A, B đã sx tăng dần
* Gọi BS(L1, L2, R1, R2, k) là giá trị phần tử nhỏ thứ k trong hai đoạn:
  + A[L1,R1]
  + B[L2,R2]
* Dễ thấy: nếu dãy chia nhỏ còn 1 phần tử thì giá trị cần tìm sẽ chính là A[k] hoặc B[k]
* Gọi mid1 là phần tử giữa dãy A[L1, R1]; mid2 phần tử giữa dáy B[L2,R2]
  + mid1 = (R1-L1)/2; mid2 = (R2-L2)/2
  + Nếu mid1+mid2<k tức là hai nửa đầu của cả A, B không đủ k phần tử 🡪 sẽ phải tìm k ở nửa sau của A hoặc nửa sau của B
    - Nếu A[mid1]>B[mid2] tức là dãy A dài hơn dãy B 🡪 tìm ở A[L1, R1] và B[L2+mid2+1,R2]



* + - Nếu a[mid1]<=b[mid2] 🡪 tìm ở A[L1+mid1+1,R1] và B[L2,R2]



* + Ngược lại: tìm k ở nửa trước của A hoặc B
    - Nếu a[mid1] > b[mid2]) thì tìm phần tử thứ k trong A[L1,L1+mid1); B[L2,R2]
    - Ngược lại tìm trong A[L1,R1] và B[L2,L2+mid2]

Sol3 ở trên tìm nhị phân bằng cách chia đôi dãy theo n/2 và m/2

**Sol4 – TKNP 2**, chia dãy thành các đoạn có độ dài k/2 và gọi đệ quy. ĐPT O(logk)

## Tìm phần tử giữa (mean) của các phần tử gộp từ hai dãy cùng kích thước

Bài này giống bài trên chỉ khác n=m; và nếu n chẵn k=(A[n/2]+A[n/2-1])/2; ngược lại k=A[n/2]

Sols1: Sau khi gộp hai dãy thành 1 dãy, in ra phần tử tại c[k] – ĐPT O(n)

Sols2: So sánh phần tử giữa của hai dãy với nhau ĐPT: O(logn)

* Gọi m1, m2 là phần tử giữa của dãy A, B.
* Nếu m1=m2 🡪 giá trị cần tìm là A[m1] hoặc B[m2]; Lúc này phần tử giữa dãy chính là giữa dãy A hoặc B.
* Bài toán con: khi mỗi dãy con còn 2 phần tử giá trị trung bình cần tìm là trung bình của 4 phần tử thuộc hai dãy đó 🡪 kq=max(A[1],B[1]) + min(A[1],B[2])/2
* Nếu m1>m2 🡪 phần tử giữa cần tìm sẽ thuộc một trong hai dãy con sau
  + Nửa đầu dãy A[1,m1]
  + Nửa cuối dãy B[m2,n]
* Nếu m1<m2 🡪 phần tử cần tìm sẽ thuộc một trong hai dãy con sau:
  + Nửa cuối dãy A[m1,n]
  + Nửa đầu dãy B[1,m2]

## Tìm phần tử nhỏ thứ k trên ma trận được sx theo hàng, cột

Bài này giống BNHAN2 Ntucoder <http://ntucoder.net/Problem/Details/2242>

Cho ma trận kích thước n\*n các phần tử trên hàng, cột đều được sắp xếp tăng dần theo chiều trái🡪phải; trên🡪dưới. Hãy tìm phần tử nhỏ thứ k trên ma trận đó?

Input:

* Dòng đầu ghi n, k
* n dòng tiếp theo mỗi dòng ghi n số nguyên thể hiện ma trận

Output:

* Một số nguyên là số nhỏ thứ k cần tìm

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 4 3  10 20 30 40  15 25 35 45  24 29 37 48  32 33 39 50 | 20 |

**Sol1:** Dùng min-heap:

* Dựng heap min O(n)
* Heapify k lần O(klogn)

**Sol2:** TKNP trên đoạn giá trị

Nhận xét: kết quả là một giá trị A[0][0]<=T<=A[n-1][n-1] 🡪 tìm nhị phân trên đoạn này A[0][0] tới A[n-1][n-1]:

Chặt nhị phân:

* Tìm giá trị nguyên trong miền tăng dần A[0][0]🡪A[n-1][n-1] thoả mãn số lượng phần tử trong ma trận A có giá trị ĐPT O(log|A[0][0]-A[n-1][n-1]|)

**Cài đặt:**

L=A[0][0]; R=A[n-1][n-1]

while(L<=R){

mid = (L+R)/2

X = số lượng phần tử trong ma trận <= mid (ĐPT O(nlogn)

Nếu X>=k thì phần tử cần tìm ở dãy phía trước tìm trong [L,mid-1]

Ngược lại: phần tử cần tìm trong đoạn phía sau [mid+1,R]

}

return L;

## Tìm k phần tử có giá trị gần với X nhất

Cho dãy A gồm N phần tử đều là số nguyên và một số nguyên X. Hãy tìm k phần tử trong A có giá trị gần với X nhất.

Input:

* Dòng đầu ghi N, k, x
* Dòng thứ hai ghi N số nguyên thể hiện dãy A

Output:

* In ra k số nguyên cách nhau bởi dấu cách là các số thoả mãn yêu cầu; các giá trị được in theo thứ tự tăng dần.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 13 4 35  12 16 22 30 35 39 42 45 48 50 53 55 56 | 30 39 42 45 |

**Sols1 trâu:** Tìm vị trí trên dãy A gần với X nhất (lớn hơn, bằng, nhỏ hơn); giả sử ta tìm phần tử lớn nhất mà có giá trị <= X 🡪 Tìm xung quanh điểm này k điểm thoả mãn yêu cầu

ĐPT O(n)

**Sols2 cải tiến** O(logn+k): cải tiến đoạn tìm vị trí gần X nhất bằng TKNP

## Tổng dãy đối xứng

Cho dãy số nguyên không âm a1, a2,…, aN được gọi là dãy tổng đối xứng nếu ta có thể tách dãy đó làm 2 dãy có tổng các giá trị bằng nhau. Nghĩa là tồn tại một số nguyên k sao cho

Yêu cầu: Cho một dãy gồm N số nguyên không âm. Tìm dãy con gồm các phần tử liên tiếp dài nhất mà cũng là dãy tổng đối xứng?

Input:

* Dòng đầu ghi N
* Dòng thứ hai ghi các phần tử thuộc dãy A

Output:

* Một dòng ghi độ dài lớn nhất của dãy con gồm các phần tử liên tiếp dài nhất là dãy tổng đối xứng. Nếu không tồn tại dãy đó, in ra 0

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 6  2  10  3  2  5  1 | 4 |

**Sols1 trâu:** Kiểm tra mọi đoạn [i,j] xem có phải là dãy đối xứng không, độ dài dãy là j-i+1 🡪 Tìm độ dài dãy đối xứng dài nhất.

ĐPT O(n^2) (xét mọi cặp i,j là O(n^2); kiểm tra dãy con tổng đối xứng O(n)

* Thuật toán kiểm tra dãy đối xứng A[L]…A[R] có phải là dãy đối xứng không
  + Gọi B[i] là tổng các phần tử dãy A từ chỉ số 1 tới i;
  + Tổng dãy con từ chỉ số L đến R sẽ bằng B[R]-B[L-1]
  + Xét đoạn [L,R], kiểm tra xem đoạn này có là dãy tổng đối xứng không:
    - Nếu tồn tại k sao cho Tổng[L,K]=Tổng[K+1,R] thì đoạn là dãy đối xứng

**Sols2: cải tiến TKNP**

* Cải tiến phần kiểm tra đoạn là tổng đối xứng bằng TKNP
  + nửa tổng S = (B[R]-B[L-1])/2
  + Tìm k trong [L,R] sao cho B[k]-B[L-1]=S

## Bước nhảy xa nhất

Cho dãy A gồm N số nguyên không âm . Một bước nhảy tử được gọi là bước nhảy xa nhất nếu thoả mãn các điều kiện sau:

* là lớn nhất

Khi đó là độ dài bước nhảy dài nhất của dãy.

Yêu cầu: Tìm độ dài bước nhảy xa nhất của A

Input:

* Dòng đầu ghi N, P
* Dòng thứ hai ghi dãy A

Output:

* Gồm một số nguyên dương là độ dài bước nhảy xa nhất của dãy A, nếu không có bước nhảy thoả mãn, in ra 0

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 6 3  4 3 7 2 6 4 | 3 |

**Sols1 trâu**: nhận xét

🡪 Với mỗi i, tìm j xa nhất thoả mãn

Xét mọi cặp (i,j) mà i<j. Tìm cặp (i,j) thoả mãn điều kiện trên

ĐPT O(n^2)

**Sols2 cải tiến – ĐPT O(n^2)**

* Gọi L[i] là giá trị nhỏ nhất trong dãy A[1]…A[i]
* Với mỗi j (từ 2->n):
  + Tìm i trong [1,j-1] nhỏ nhất sao cho A[j]-P>=L[i] khi đó đoạn (j-i) tìm được sẽ thoả mãn với mỗi j
  + Tìm giá trị (j-i) tối ưu

**Sol3: TKNP** – có thể cải tiến phần tìm i nhỏ nhất trên dãy L không tăng ĐPT tổng quát O(nlogn)

* Nhận xét: do L[i] là dãy lưu giá trị nhỏ nhất của các phần tử trong A[1]…A[i] nên ta luôn có dãy A là dãy không tăng.
* Với mỗi j:
  + Tìm i nhỏ nhất thoả mãn điều kiện L[i]<=A[j]-P 🡪 ĐPT chỗ này là O(logn)
  + Tìm kq tối ưu

## Aggressive cows – chú bò hung hăng – spoj

(Nguồn <https://www.spoj.com/problems/AGGRCOW/> )

Bờm được nhận vào làm việc cho nhà Phú Ông, nhiệm vụ của cậu ta là chăn dắt đàn bò. Với bản tính ham chơi nên Bờm đã đóng N cái cọc và cột các con bò vào đó, vì thế cậu thoải mái chơi mà không lo các con bò đi mất.

N cái cọc được đặt trên một đường thẳng ở các vị trí có toạ độ . Phú ông giao cho Bờm chăn thả C con bò. Những con bò này không thích bị buộc vào những chiếc cọc gần các con bò khác. Chúng sẽ trở nên hung dữ khi bị buộc gần nhau.

Để tránh các con bò làm đau nhau, Bờm muốn buộc mỗi con vào một cái cọc sao cho khoảng cách nhỏ nhất giữa hai con bò bất kì là lớn nhất có thể?

Yêu cầu: Tìm giá trị lớn nhất này.

Input:

* Dòng đầu ghi N, C
* N dòng tiếp theo mỗi dòng chứa số nguyên thể hiện vị trí của mỗi cọc. Không có hai cọc nào cùng vị trí.

Output:

* In ra một số nguyên là giá trị lớn nhất của khoảng cách nhỏ nhất giữa hai con bò bất kì.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5 3  1  2  8  4  9 | 3 |

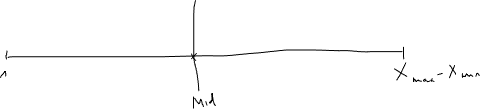
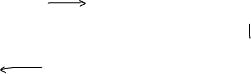
**Sol1:** thử mọi cách xếp C con bò vào N cột, tính độ dài khoảng cách nhỏ nhất có thể 🡪 tìm giá trị lớn nhất của các khoảng cách nhỏ nhất đó.

**Sols2 - TKNP:** Sắp xếp lại các cột theo toạ độ x tăng dần.

Nhận xét: giá trị lớn nhất của khoảng cách nhỏ nhất giữa hai con bò bất kì là một giá trị trong đoạn từ 1 tới : thử mỗi k thuộc [1,x] và kiểm tra xem với khoảng cách k này, có thể buộc đủ C con bò vào N cột không?

* L=1; R=x
* while(L<=R)
  + mid=(L+R)/2
  + Nếu có thể đặt C bò thoả mãn khoảng cách nhỏ nhất >=k thì tìm ở đoạn sau [mid,R]
  + Ngược lại thì tìm ở đoạn trước [L,mid-1]





**Bài toán bổ trợ**: Kiểm tra xem có thể đặt C con bò vào N cột sao cho khoảng cách nhỏ nhất giữa hai con bò >=k không?

bool check(int k): thử xếp C con bò vào các cột, bắt đầu từ cột đầu tiên i=1

* pos = a[0]: là vị trí ban đầu thử đặt 1 bò; dem=1 (1 con bò được sắp)
* Xét các cột khác i từ 1 🡪 n-1:
  + Nếu đặt được bò vào cột i tức là a[i]-pos>=k 🡪 số con bò được sắp xếp thoả mãn là dem++; pos = i (để tiếp tục kiểm tra con bò tiếp theo)
* Nếu đến cuối mà dem=c thì kết luận có cách đặt c bò vào N cột thoả mãn yêu cầu

## BOOKS1 - Copying Books – SPOJ

(Nguồn: <http://www.spoj.com/problems/BOOKS1/>)

Trước khi máy photo sách được phát minh ra, việc tạo ra một bản photo cho một quyển sách là vô cùng khó khăn. Tất cả nội dung phải được ghi chép lại bằng tay. Một người thợ được đưa cho quyển sách và anh ta phải hoàn thành nó sau một vài tháng. Một trong những người thợ nổi tiếng sống ở thế kỉ 15, và tên anh ta là Tèo. Nhưng dù sao thì công việc cũng vô cùng chán và buồn tẻ. Và chỉ có một cách để cải thiện tình hình đó là thuê những người thợ khác.

Hồi đó, có một đoàn múa hát muốn trình diễn một vở kịch rất nổi tiếng. Kịch bản được chia ra làm nhiều cuốn. Và diễn viên cần có nhiều bản copy của chúng. Tưởng tượng rằng bạn có m cuốn sách (đánh số từ 1, 2,…, m) có số trang lần lượt là p[1], p[2], …, p[m]. Và bạn phải tạo ra bản copy của mỗi cuốn đó. Nhiệm vụ của bạn là chia m cuốn sách đó cho k người thợ, k <= m. Mỗi cuốn chỉ phép giao cho một người thợ duy nhất. Và mỗi người thợ phải nhận những cuốn sách liên tiếp nhau. Có nghĩa là tồn tại một dãy số tăng dần thoả mãn 0 = b[0] < b[1] < b[2] < … < b[k] = m sao cho người thợ thứ i lấy những cuốn sách được đánh số giữa b[i-1] + 1 và b[i]. Thời gian để tạo ra bản copy của tất cả những cuốn sách được xác định dựa trên người thợ nhận nhiều việc nhất. Vì vậy, nhiệm vụ đặt ra là tối thiểu hoá số lượng trang sách lớn nhất được giao cho mỗi người thợ. Hãy tìm ra giải pháp cho vấn đề này.

Input:

* Dòng đầu ghi số testcase N . Mỗi test gồm:
  + Dòng đầu ghi hai số nguyên m, k
  + Dòng thứ hai ghi dãy

Output:

* Với mỗi testcase, in kết quả trên một dòng gồm dãy được chia làm k phần sao cho tổng lớn nhất của mỗi phần là nhỏ nhất. Ưu tiên cho những người thợ đầu tiên trước: tức là người thợ đầu tiên làm ít hơn người thợ thứ hai,… Sử dụng dấu ‘/’ để phân tách các phần. Có 1 dấu cách duy nhất giữa hai số và giữa số với ‘/’

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 2  9 3  100 200 300 400 500 600 700 800 900  5 4  100 100 100 100 100 | 100 200 300 400 500 / 600 700 / 800 900  100 / 100 / 100 / 100 100 |

Giải thích:

* Test 1, tổng lớn nhất của mỗi phần là 1700
* Test 2, Tổng lớn nhất của mỗi phần là 200

Tóm tắt đề: Chia dãy thành k đoạn con (không giao nhau) sao cho tổng lớn nhất của các đoạn là nhỏ nhất

**Sols: TKNP** – bài này khá giống bài Aggressive Cows

* Nhận xét: Tổng nhỏ nhất là 1, lớn nhất là tổng cả dãy. Tiến hành chặt nhị phân trên đoạn giá trị tăng này [1,N] (xem xét mọi tổng có thể);
* Kiểm tra giá trị X là tổng lớn nhất của các dãy con của A không? check(X)
  + mid = (L+R)/2
  + Nếu (check(mid)==true) nếu mid là tổng

## Đếm cặp đôi (HSG'20)

(Nguồn: <https://lqdoj.edu.vn/problem/capdoi>)

Cho dãy số 𝐴 gồm 𝑛 phần tử nguyên dương 𝐴1, 𝐴2,…, 𝐴𝑛. Mỗi phần tử có giá trị không vượt quá 109 và 𝑛≤105. Một cặp số được gọi là cặp tương đồng với 𝑥, nếu cặp số này có tổng bằng số 𝑥 cho trước nào đó.

Yêu cầu: Hãy đếm xem trong dãy số 𝐴 có bao nhiêu cặp số (𝐴𝑖; 𝐴𝑗) tương đồng với 𝑥 (có nghĩa là 𝐴𝑖+𝐴𝑗=𝑥) với 𝑖<𝑗.

Input:

* Dòng đầu tiên chứa dãy số 𝑛,𝑥 (𝑛≤105,𝑥≤106).
* Dòng thứ 2 chứa 𝑛 phần tử của dãy số 𝐴 (𝐴𝑖≤109).

Output:

* Ghi ra một số nguyên là cặp đôi tương đồng của dãy số.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 7 6  1 2 4 3 4 5 3 | 4 |

Tóm tắt đề: Đếm các cặp số (A[i], A[j]) thoả mãn A[i]+A[j]=x

**Sols1 trâu**: xét mọi cặp (i,j); kiểm tra điều kiện và đếm O(n^2)

**Sols2 cải tiến – TKNP**: O(nlogn)

* Sắp xếp lại dãy tăng dần
* Với mỗi vị trí i, tìm vị trí j thoả mãn a[j]=x-a[i] bằng TKNP

**Sols3** – đếm phân phối

## CSP – Tập xe

(Nguồn: <https://lqdoj.edu.vn/problem/csp>)

Cô giáo trường tiểu học 𝑋 đang dạy 𝑛 học sinh tập xe đạp, các học sinh được đánh số từ 1 tới 𝑛, học sinh thứ 𝑗 có trọng lượng là 𝑎𝑗. Có một xe đạp duy nhất với tải trọng là 𝑚, hai học sinh chỉ có thể cùng lên xe nếu tổng trọng lượng của hai học sinh không vượt quá 𝑚.

Cô giáo tự hỏi có bao nhiêu cách chọn hai học sinh khác nhau cho cùng lên xe, sau nhiều giờ tính toán không có kết quả, cô quyết định hỏi các chuyên gia lập trình giải bài toán Counting Student Pairs (CSP)

Yêu cầu: Đếm số cặp chỉ số 𝑖,𝑗 trong đó 𝑖<𝑗 và 𝑎𝑖+𝑎𝑗≤𝑚

Input:

* Dòng 1 chứa hai số nguyên dương 𝑛,𝑚 (𝑚≤106)
* Dòng 2 chứa 𝑛 số nguyên dương 𝑎1,𝑎2,…𝑎𝑛 (∀𝑖:𝑎𝑖≤106)

Output:

* Ghi một số nguyên duy nhất là đáp số

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5 6  1 2 3 4 5 | 6 |

Giới hạn:

* SubTask1: 𝑛≤104. (60% test)
* SubTask2: 𝑛≤105. (20% test)
* SubTask3: 𝑛≤106. (20% test)

Sols: Tìm nhị phân

* Với mỗi a[i], tìm vị trí j cuối cùng (lớn nhất) thoả mãn điều kiện a[j]<=m-a[i] (upper\_bound, lower\_bound)

## Pair 2 – Chia cặp 2

(Nguồn: <https://lqdoj.edu.vn/problem/pair2>)

Cho 𝑛 học sinh có năng lực 𝑎1, 𝑎2,..., 𝑎𝑛. Thầy giáo muốn chọn ra 𝑘 cặp học sinh rời nhau. Gọi 𝑥 là chênh lệch nhỏ nhất giữa hai bạn trong một nhóm. Thầy muốn 𝑥 càng lớn càng tốt. Hãy in ra max 𝑥.

Input:

* Dòng đầu có 2 số nguyên 𝑛,𝑘 (2≤𝑛≤3×105,1≤𝑘≤𝑛/2).
* Dòng thứ hai có 𝑛 số nguyên 𝑎1,𝑎2,...,𝑎𝑛 (1≤𝑎𝑖≤109)

Output

* In ra một số nguyên là kết quả bài toán

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 6 2  1 4 3 7 12 9 | 8 |
| 6 1  1 4 3 7 11 9 | 10 |

**Giải thích:**

Trong test ví dụ 1, chúng ta chia cặp như sau: (1,9),(3,12). Cặp có khoảng cách nhỏ nhất là (1,9) và 9−1=8.

Trong test 2, chia cặp (1,11).

Sols: bài này dễ hơn bài Aggressive Cows.

* Nhận xét: khoảng các nhỏ nhất giữa các cặp có thể là giá trị trong [0, a[max]-a[min]]
* Tìm một giá trị X trong đoạn trên thoả mãn điều kiện là khoảng cách nhỏ nhất có thể có giữa các cặp.

# Kết luận

Trên đây là nội dung chuyên đề thiết kế thuật toán theo mô hình Tìm kiếm nhị phân, là một phần cơ bản thuộc chuyên đề mô hình thiết kế thuật toán chia để trị. Rất mong nhận được những góp ý của các thầy cô cùng các em học sinh để chuyên đề thực sự là một tài liệu giảng dạy, học tập hữu ích. Góp phần nâng cao chất lượng dạy học môn chuyên theo chương trình Tin học chuyên sâu mới.

Link test: <https://drive.google.com/drive/folders/10NnmTk3-XVkOdfed_84aTcZcNYjJjxK4?usp=sharing>