PHẦN I: MỞ ĐẦU

Từ bài toán *Batch Scheduling* trong kì thi *IOI 2002*, kĩ thuật bao lồi được đưa rộng rãi vào các cuộc thi lập trình. Nhưng kĩ thuật bao lồi được biết đến nhiều nhất từ bài toán *Acquire* trong bảng Gold của USACO năm 2008.

Kĩ thuật bao lồi là kĩ thuật (hoặc cấu trúc dữ liệu) dùng để xác định cực trị của một tập các hàm tuyến tính tại một giá trị của biến độc lập. Nó không giống như thuật toán bao lồi của hình học, nhưng nó có tên như vậy vì trong kĩ thuật có phần tạo đường bao là một đường lồi để hình thành bao lồi của một tập điểm.

Kĩ thuật này có thể dùng để cải tiến các bài toán quy hoạch động thời gian xuống còn , thậm chí một số trường hợp còn xuống cho một lớp các bài toán.

Tôi nhận thấy đây là một kĩ thuật khá lạ, thú vị và có ít tài liệu về nó, nên quyết định tìm hiểu và viết chuyên đề: *"Kĩ thuật bao lồi (Convex Hull Trick) và ứng dụng"*.

PHẦN II: NỘI DUNG

I. KĨ THUẬT BAO LỒI (CONVEX HULL TRICK)

1. Bài toán 1

Cho một tập n đường thẳng trong đó và điểm trên trục : Tìm trong đó thỏa mãn:

- Dữ liệu vào: Cho trong file **CHT.INP.***
  - Dòng đầu tiên là số nguyên , số đường thẳng.
  - dòng tiếp theo mỗi dòng chứa hai số thực là hệ số của phương trình đường thẳng .
  - Dòng tiếp theo chứa số nguyên , là số lượng truy vấn.
  - dòng tiếp theo mỗi dòng chứa một số thực .
- Dữ liệu ra: In ra file **CHT.OUT.**
  - Mỗi dòng một số thực là giá trị thỏa mãn.
- Ví dụ:

CHT.INP	CHT.OUT
4 0.17 2.67 1 3 0.5 2 0.8 4 3 -2 2 0	1 3 2

- Giới hạn:
  - .
  - .

1.1. Phân tích

Với mỗi điểm , duyệt qua tất cả các đường thẳng trong tập ta tính được trong thời gian . Do vậy tổng thời gian để tìm tất cả các điểm là .

Vấn đề đặt ra là chúng ta có thể cải tiến thuật toán hay không?

Đầu tiên ta xét ví dụ với tập như sau:

Với mỗi điểm , giá trị nhỏ nhất tương ứng điểm thỏa mãn điểm thuộc phần gạch đỏ như hình dưới. Phần gạch đỏ trên hình dưới là tập các đoạn thẳng chứa tất cả các điểm cực tiểu cho mọi giá trị của . Phần gạch đỏ tạo thành một đường bao lồi của một tập điểm nên kĩ thuật này được gọi là *kĩ thuật bao lồi*.

Từ đây tôi viết tắt kĩ thuật bao lồi là CHT.

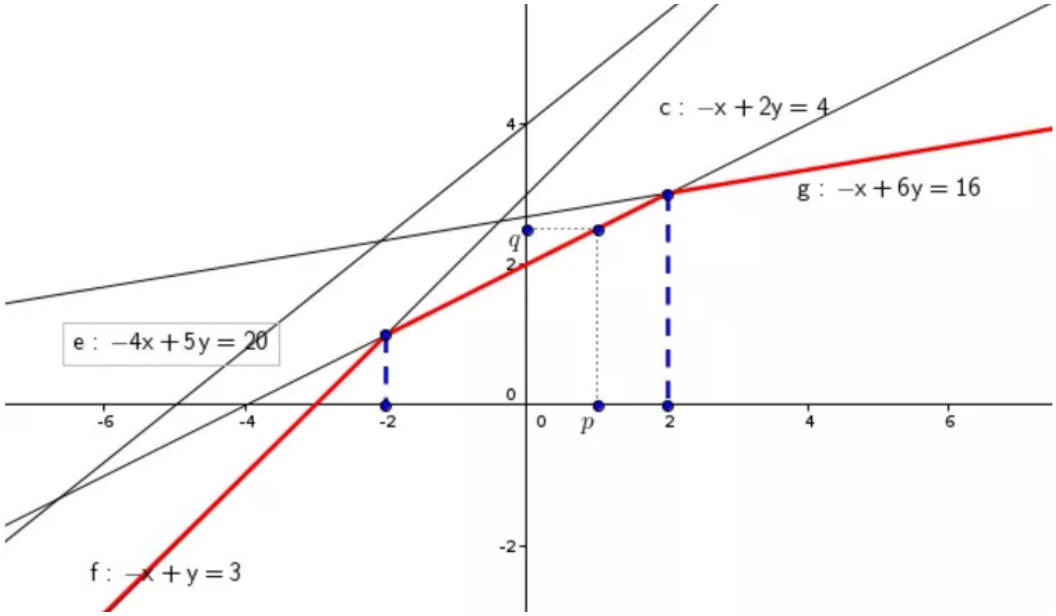
Thuật toán của bài toán có hai bước chính:

Bước 1: Xây dựng bao lồi của các đường thẳng.

Bước 2: Thực hiện các truy vấn trên bao lồi tìm được tại bước 1.

a) Thực hiện bước 1:

Dựa trên hình vẽ, ta thấy đường thẳng trong là dư thừa (tung điểm của các điểm có hoành độ trên đường thẳng này luôn lớn hơn tung độ của điểm có cùng hoành độ trên một đường thẳng khác của).



Biểu diễn bao lồi của ví dụ trên bằng tập các đoạn: và các đường thẳng tương ứng là: .

Ta nhận thấy :

Bao lồi của tập các đường thẳng cho trước có thể biểu diễn bằng m đoạn và đường thẳng tương ứng với mỗi đoạn . Hơn nữa, ta có thể tìm được biểu diễn đó trong thời gian

Giả mã: Để đơn giản ta giả sử trong không có hai đường thẳng nào song song. Ta có giả mã của thuật toán tìm bao lồi.

```
 Void   FindHull ;
{
   Sort ; // s p x p theo h  s  góc gi m d n.
   Stack ; // kh i t o ngăn x p r ng
   .Push; // Đ y đ  ng th ng  vào ngăn x p.
   ;   ;
    // m ng l u các đ  ng th ng thu c bao l i.
   For
   {
        d ←.top(); //l y ph n t  trên cùng c a ngăn x p mà không xóa nó
   đi.
        ←find_intersection_x (d, d_i);//tìm hoành đ  giao đi m 2 c a đ  ng
        th ng d và d_i
        // lo i b  đ  ng th ng d  th a.
        While (≤left())
        {
              .Pop(); //lo i b  đ  ng th ng d vì d d  th a.

              .top();
              ←;
        }
        S.push(d_i);
         ;
```

```
          ;

          ;
    }
}
```

Nhận xét : Nếu một đường thẳng được lấy khỏi Stack, nó sẽ không được kiểm tra lại nữa. Vì vậy, thời gian của thuật toán tìm bao lồi chủ yếu dành để sắp xếp các đường thẳng theo hệ số góc. Độ phức tạp: .

b) Thực hiện bước 2: Tìm tất cả các điểm thỏa mãn.

Giả sử chúng ta đã tìm được bao lồi của , với mỗi điểm trên trục hoành, sử dụng tìm kiếm nhị phân để tìm sao cho .

Ta có thể tính trong thời gian . Nên để tìm các điểm ta sẽ mất thời gian là .

Giả mã:

```
Query();
{
    FindHull(;
    For
    {
        ;
        Return ;
    }
}
```

```
;
{
     If   return
     Else
     {

           If  return ;
           Else if  return interval_search
                  Else   return interval_search;
     }
}
```

Vậy tổng thời gian của cả bài toán 1 sẽ là .

1.2. Cài đặt:

```
#include<bits/stdc++.h>
#define MAX_SIZE 1000000
#define INFTY 100000000
using namespace std;
typedef struct{ //c u trúc m t đ  ng th ng y = ax+b
    double a;
    double b;
  } double_pair;

stack<int> S;              //   stack
double_pair D[MAX_SIZE];      // m ng l u các đ  ng th ng, m i đ  ng th ng
có hai tham s  là a và b
double_pair I[MAX_SIZE];     // m ng các đo n thu c interval
double      Q[MAX_SIZE];     // m ng các đi m thu c các truy v n
int         ALines[MAX_SIZE];// t p các đ  ng th ng ch a các interval
```

```
int n, m,q;
long long         A[MAX_SIZE];
long long         B[MAX_SIZE];

void readinput(){
    freopen("CHT.INP","r",stdin);
    freopen("CHT.out","w",stdout);
    cin >> n;
    for(int i = 0 ; i < n; i++) cin>>D[i].a>>D[i].b;
    cin>>q;// s  l  ng truy v n
    for(int i = 0 ; i < q; i++) cin>> Q[i];
}
// tìm hoành đ  giao đi m c a dx và dy
double find_intersection_x(int x, int y)
{
      return ((double)(D[y].b-D[x].b))/((double)(D[x].a - D[y].a));
}
//s p x p h  s  góc gi m d n.
bool slope_compare(double_pair d1, double_pair d2) {
    return ((d1.a > d2.a) || (d1.a == d2.a && d1.b<d2.b) ) ;
}
//Tìm bao l i c a t p đi m nh  nh t.
int find_hull(int n){
    sort(D, D+n, slope_compare);//s p x p D theo h  s  góc
    S.push(0); //đ a đ  ng th ng d0 vào stack
    I[0].a = -(double)INFTY;
    I[0].b = (double)INFTY;
     m = 0;
    ALines[m] = 0;
    int i = 0;
    int d;
    double x = 0.0;
    for(i = 1; i <n ;i++){ //xét các đ  ng th ng còn l i
        d = S.top();
        x = find_intersection_x(d, i);
        while(x <= I[m].a){
            S.pop();       // lo i b  đ  ng th ng d  th a
            m--;
            d = S.top();
            x = find_intersection_x(d, i);
        }
        S.push(i);             // đ y d_i vào stack
        I[m].b = x;
        I[m+1].a = x;
        I[m+1].b = (double)INFTY;
        ALines[m+1] = i;
        m++;
    }
    return m+1;        // s  interval tìm đ  c
}
//Tìm giá tr  p thu c interval nào?
int interval_search(int x, int y, double p){
      if(y <= x){
            return x;
      } else {
            int mid = (x+y)/2;
            if((I[mid].a <= p) && (p <= I[mid].b)){
                  return mid;
            } else if (p > I[mid].b){
                  return interval_search(mid+1, y, p);
            } else {
                  return interval_search(x, mid-1, p);
            }
      }
```

```
}
//tr  l i các truy v n: tìm các q[i] thu c interval
void query(double points[], int k, int n){
    int m = find_hull(n);
    int i = 0, q = 0;
    for(i = 0; i < k; i++){
        q = interval_search(0, m-1, points[i]);
        Q[i] = D[ALines[q]].a*points[i] + D[ALines[q]].b;
    }
}
int main()
{
    readinput();
    query(Q, q, n);
    for(int i = 0 ; i < q; i++) cout<< Q[i]<<endl;
    return 0;
}
```

2. Bài toán 2

Cho phần tử khác nhau đánh số từ 1 tới . Phần tử thứ có hai tính chất . Hàm mục tiêu của bài toán có thể được biểu diễn thông qua bảng quy hoạch động một chiều thỏa mãn hệ thức sau:

(1)

Tìm giá trị .

Bài toán được giải quyết bằng thuật toán quy hoạch động với độ phức tạp .

Nếu ta đặt thì là tung độ nhỏ nhất thuộc một trong các đường thẳng tương ứng với hoành độ . Áp dụng ý tưởng của kĩ thuật bao lồi trong bài toán 1.

Áp dụng thủ tục *interval_search* để tìm đoạn chứa . Vì vậy mất cho việc tìm kiếm trong mỗi vòng lặp của thuật toán.

Việc cập nhật bao lồi trong mỗi bước khi thêm đường thẳng
mất thời gian là . Vậy, tổng thời gian của thuật toán là .

3. Bài toán 3.

Bài toán phát biểu như bài toán 2 nhưng thêm điều kiện:

Vì thì hệ số góc của các đường thẳng đã được sắp xếp theo thứ tự giảm dần. Do đó, tiết kiệm được bước sắp xếp trong thuật toán **FindHull**.

Giả mã dưới ta giả sử :

```
FASTDYNAMIC
{

   Stack ; // kh i t o ngăn x p r ng
   .Push; // Đ y đ  ng th ng  vào ngăn x p.
   ;   ;
    // m ng l u các đ  ng th ng thu c interval.
   For
   {


          // đ  ng th ng d là
```

```
          d    .top(); //l y ph n t  trên cùng c a ngăn x p mà không xóa nó
   đi.
          find_intersection_x (d, di);//tìm hoành đ  giao đi m 2 c a đ  ng
        th ng d và di
        While (  left())    // tìm đ  ng th ng d  th a.
        {
              .Pop(); //lo i b  đ  ng th ng d  th a.

              .top();
                 ;
        }
          .push(  );
         ;
         ;
          //Đ  ng th ng di đi qua đo n Im+1.
         ;
   }
   Return ;
}
```

Đánh giá độ phức tạp: Mỗi bước tìm kiếm nhị phân mất , nên tổng thời gian của thuật toán là . Trên thực tế thì số đường thẳng thuộc bao lồi ít hơn nhiều so với số đường thẳng đầu vào nên thuật toán này chạy khá nhanh.

4. Bài toán 4

Bài toán phát biểu như bài toán 3 nhưng thêm điều kiện: .

Vì nên khi tìm kiếm đoạn chứa ta không cần xét các đoạn nằm bên trái nữa. Vì vậy, trong thuật toán *FASTDYNAMIC* ta thay việc tìm kiếm nhị phân bằng tìm kiếm tuần tự bắt đầu là đoạn chứa . Trong giả mã dưới giả sử

```
 FASTDYNAMIC
{

   Stack ; // kh i t o ngăn x p r ng
   .Push; // Đ y đ  ng th ng  vào ngăn x p.
   ;   ;   ;
    // m ng l u các đ  ng th ng thu c interval.
   For
   {



        While (
        {

             If
        }

          // đ  ng th ng d là

            d ←.top();
        ←find_intersection_x (d, di);
        While (≤left())    // tìm đ  ng th ng d  th a.
        {
              .Pop();

              .top();
              ←;
        }
```

```
            .push(   );
          ;
          ;
          If  // đ  ng th ng ch a  không d  th a

          Else
                If

             Else

          ALines[m+1] = i;
           ;
    }
    Return ;
}
```

Đánh giá độ phức tạp: Ta chỉ cần xét thời gian của thủ tục tìm kiếm sao cho trong mỗi vòng lặp. Trường hợp tồi nhất là phải duyệt qua toàn bộ các đoạn. Tuy nhiên nếu đã duyệt qua đoạn, thì các vòng lặp sau không cần xét các đoạn này và các đường thẳng tương ứng với chúng nữa. Vì vậy, tổng thời gian tìm kiếm các đoạn của thuật toán là và đó cũng là thời gian của cả thuật toán.

II. BÀI TẬP ỨNG DỤNG.

Bài 1: CẮT CÂY.

1.1. Đề bài

Nguồn http://codeforces.com/problemset/problem/319/C

Có cái cây trong đó cây thứ có chiều dài . Bạn phải cắt tất cả các cây thành các đoạn có chiều dài . Bạn có một cái cưa máy, mỗi lần chỉ cưa được một đơn vị chiều dài, và mỗi lần sử dụng thì lại phải sạc pin. Chi phí để sạc pin phụ thuộc vào chi phí của cây đã bị cắt hoàn toàn (một cây bị cắt hoàn toàn có chiều dài 0). Nếu chỉ số lớn nhất của cây bị cắt hoàn toàn là thì chi phí sạc là , và khi bạn đã chọn một cây để cắt, bạn phải cắt nó hoàn toàn. Ban đầu cái cưa máy được sạc đầy pin. Giả sử và .

- *Yêu cầu:* Tìm một cách cưa cây với chi phí nhỏ nhất.
- *Dữ liệu vào:* Cho trong file **319C.INP.**
  - Dòng đầu chứa một số nguyên .
  - Dòng thứ chứa số nguyên .
  - Dòng thứ 3 chứa số nguyên .
  - Trong đó: và .
- *Dữ liệu ra:* In ra file **319C.OUT.**
  - Dòng duy nhất chứa chi phí nhỏ nhất để cắt tất cả các cây.
- *Ví dụ:*

319C.INP	319C.OUT	319C.INP	319C.OUT
5 1 2 3 4 5 5 4 3 2 0	25	6 1 2 3 10 20 30 6 5 4 3 2 0	138

- *Giới hạn:*

  Thời gian: 2 giây.

1.2. Hướng dẫn giải thuật:

Xét ví dụ 2 với:

Nếu bạn chặt cây theo thứ tự

chi phí bạn phải trả là: .

Nếu bạn chặt theo thứ tự:

chí phí bạn phải trả là:

Từ ví dụ trên, ta có nhận xét sau:

Chi phí để chặt các cây sau khi đã chặt cây thứ là .

Do đó, bài toán trên có thể được quy về bài toán: tìm chi phí nhỏ nhất để chặt cây thứ . Gọi là một thứ tự chặt cây tối ưu với . Ta có bổ đề sau: .

Dựa vào , ta có thể phát triển thuật toán quy hoạch động như sau: Gọi là chi phí nhỏ nhất để chặt cây thứ "sử dụng" các cây . Ta có công thức sau:

Dễ thấy, chi phí để chặt cây thứ là Ta thấy công thức quy hoạch động để tính với hai mảng là các mảng đã săp xếp tương ứng với bài toán 4. Do đó, có thể giải bài toán này trong thời gian .

1.3. Chương trình

```
#include<bits/stdc++.h>
#define min(x, y) (((x) < (y)) ? (x) : (y))
#define max(x, y) (((x) < (y)) ? (y) : (x))
#define is_equal(x,y) (((x) - (y) == 0) ? 1 : 0)
#define MAX_SIZE 100005
#define INFTY  1e15 + 7
#define NULL 0
using namespace std;
typedef struct{
    double a;
    double b;
  } double_pair;
stack<int> S;
double_pair I[MAX_SIZE];
long long         A[MAX_SIZE];
long long         B[MAX_SIZE];
long long   C[MAX_SIZE];
int         ALines[MAX_SIZE];// t p các đ  ng th ng I

double find_intersection_x(int x, int y);
void readinput();
void fast_fats_dynamic(long long A[], long long B[],int n);
int interval_search(int x, int y, double p);
int n;
void readinput(){
    freopen("319C.INP","r",stdin);
    freopen("319C.out","w",stdout);
      cin >> n;
      for(int i = 0 ; i < n; i++)
cin>>A[i];
      for(int i = 0 ; i < n; i++)
cin>>B[i];
}
```

```
void fast_fast_dynamic(long long A[],
long long B[], int n){
      C[0] = B[0]; // d_1 = B[0]x + C[0]
      S.push(0); // đ y d_1 vào đ nh
stack;
      I[0].a = -(double)INFTY;
      I[0].b = (double)INFTY;
      int m = 0;
      ALines[m] = 0;
      int i = 0, q = 0, l =0;
      int d;
      double x = 0.0;
      int prev_interval = 0;
      for(i = 1; i <n ;i++){
        l = prev_interval-1; //
interval ch a A[i-1]
        while(1){
            l++;
            if((I[l].a <= (double)A[i])
&& ((double)A[i]<= I[l].b)){
                  q = l;
                  break;
            }
        }
            C[i] = B[ALines[q]]*(A[i]-1)
+ C[ALines[q]] + B[i];
            d = S.top();   // Ki m tra
ph n t  trên cùng c a ngăn x p
            x = find_intersection_x(d,
i);
            while(x <= I[m].a){  // Tìm
đ  ng th ng d  th a
                  S.pop();          //
Lo i b  đ  ng th ng d  th a
                  m--;
                  d = S.top();
                  x =
find_intersection_x(d, i);
            }
            prev_interval = l;
         if(x < (double) (INFTY-1)){
               S.push(i);
               I[m].b = x;
               I[m+1].a = x;
               I[m+1].b = INFTY;
               if( l >= m){
                if ((I[m].a <=
((double)A[i]))&&(((double)A[i])<=I[m].b
)){
                     prev_interval = m;
                  }else {
                    prev_interval = m+1;
                    }
               }
               ALines[m+1] = i;
               m++;
         }
      }
cout<<C[n-1]<<endl;
}
int interval_search(int x, int y, double
p){
      if(y <= x){
            return x;
      } else {
            int mid = (x+y)/2;
            if((I[mid].a <= p) && (p <=
I[mid].b)){
                  return mid;
            } else if (p > I[mid].b){
                  return
interval_search(mid+1, y, p);
            } else {
                  return
interval_search(x, mid-1, p);
            }
      }
}
double find_intersection_x(int x, int
y){
      long long a1 = B[x], b1 = C[x];
    long long a2 = B[y], b2 = C[y];
    return ((double)(b2-
b1))/((double)(a1 - a2));
}
int main(int argc, const char * argv[]){
      readinput();
      fast_fast_dynamic(A,B,n);
      return 0;
}
```

1.4.

1.5. Test

Đường dẫn:

https://drive.google.com/file/d/0BwQZ8OxBy1tPdmJGNjJKZEJYRUE/view?usp=sharing

1.6. Cảm nhận

Đây là bài toán quy hoạch động áp dụng bài toán 4.

và

Độ phức tạp của thuật toán là .

Bài 2: ACQUIRE.

2.1. Đề bài

Nguồn: USACO 2008

Cho hình chữ nhật khác nhau về hình dạng, chiều rộng và chiều dài trên mặt phẳng.

Mục tiêu của bài toán là phải lấy được tất cả hình chữ nhật. Một tập hình chữ nhật có thể thu được với chi phí bằng tích của chiều dài dài nhất và chiều rộng dài nhất. Hình chữ nhật không thể được xoay (đổi chiều dài và chiều rộng). Ví dụ, nếu chọn hai hình chữ nhật có kích thước và 3 thì chi phí phải trả là .

- *Yêu cầu:* Cần phân hoạch tập các hình chữ nhật này một cách khôn khéo sao cho tổng chi phí là nhỏ nhất và tính chi phí này.
- *Dữ liệu vào:* Cho trong file **ACQUIRE.INP**
  - Dòng đầu tiên chứa số nguyên .
  - dòng tiếp theo mỗi dòng chứa hai số nguyên vàlà chiều rộng và chiều dài của hình chữ nhật thứ .
- *Dữ liệu ra:* In ra file **ACQUIRE.OUT**
  - Dòng duy nhất chứa số nguyên là chi phí nhỏ nhất tìm được.
- *Ví dụ:*

ACQUIRE.INP	ACQUIRE.OUT
4 100 1 15 15 20 5 1 100	500

- *Giải thích:*
  - Nhóm đầu tiên chứa một hình chữ nhật và chi phí là . Nhóm tiếp theo chứa một hình chữ nhật kích thước và chi phí . Nhóm cuối cùng chứa cả hai hình chữ nhật kích thước và và chi phí . Tổng chi phí là .
- *Giới hạn:*
  - Thời gian: 1 giây.

2.2. Hướng dẫn giải thuật

- Giả sử tồn tại hai hình chữ nhật và mà cả chiều dài và chiều rộng của hình đều bé hơn hình thì có thể để hình chung với hình , vì vậy chi phí của hình không còn quan trọng và hình B là hình dư thừa. Sau khi đã loại hết tất cả hình dư thừa đi và sắp xếp lại các hình theo chiều dài giảm dần thì chiều rộng các hình đã sắp xếp sẽ theo chiều tăng.

- Sau khi sắp xếp, nếu chọn hai hình chữ nhật ở vị trí và ở vị trí thì ta có thể chọn tất cả hình chữ nhật từ đến mà không tốn chi phí nào cả. Vậy ta có thể thấy rằng cách phân hoạch tối ưu là một cách phân dãy thành các đoạn liên tiếp và chi phí của một đoạn là bằng tích của chiều dài của hình chữ nhật đầu tiên và chiều rộng của hình chữ nhật cuối cùng.
- Vậy bài toán trờ về bài toán phân dãy sao cho tổng chi phí của các dãy là tối ưu. Đây là một dạng bài quy hoạch động hay gặp và chúng ta có thể dễ dàng nghĩ ra thuật toán .
- Sử dụng kĩ thuật bao lồi:
  - Với trong đó là chiều rộng của hình chữ nhật , là chiều dài của hình chữ nhật , là chi phí cực tiểu để lấy được hình chữ nhật đầu tiên.
  - Vậy bài toán trở về tìm hàm cực tiểu của bằng cách tìm tối ưu.
  - Nếu sắp xếp các hình chữ nhật theo chiều dài giảm dần thì các đường thẳng đã được sắp xếp giảm dần theo hệ số góc.
  - Độ phức tạp: .

II.3. Chương trình

```
#include<bits/stdc++.h>
#define mn 100010
using namespace std;
int pointer,topm=0,topb=0; //Ch  đ n đ  ng th ng t t nh t t  truy v n tr  c đó
long long M[mn],B[mn]; //m ng stack H  s  góc c a đ  ng th ng trong bao l i
//Tr  v  true n u l1 ho c l3 luôn t t h n l2.
bool bad(int l1,int l2,int l3)
{
	//giao(l1,l2) co toa do x la (b1-b2)/(m2-m1)
	//giao(l1,l3) co toa do x la (b1-b3)/(m3-m1)
	return (B[l3]-B[l1])*(M[l1]-M[l2])<(B[l2]-B[l1])*(M[l1]-M[l3]);
}
//Thêm m t đ  ng th ng m i, v i h  s  góc th p h n.
void add(long long m,long long b)
{
	//Thêm vào cu i
	topm++; M[topm-1]=m;
	topb++; B[topb-1]=b;
	//N u không t t thì lo i b  nó và l p l i.
	while (topm>=3&&bad(topm-3,topm-2,topm-1))
	{
	    M[topm-2]=M[topm-1]; topm--;
	    B[topb-2]=B[topb-1]; topb--;
	}
}
//Tr  v  t a đ  y nh  nh t c a giao đi m gi a m t đ  ng th ng d c b t kì và bao
l i.
long long query(long long x)
{
	//N u lo i b  nh ng đ  ng th ng t t nh t trong truy v n tr  c thì bây gi
chèn dòng t t nh t cho truy v n đó.
	if (pointer>=topm)
	    pointer=topm-1;
	//B t kỳ đ  ng th ng nào t t h n đ u n m bên ph i vì các giá tr  truy v n
không gi m.
	while(pointer<topm-1&&
	        M[pointer+1]*x+B[pointer+1]<M[pointer]*x+B[pointer])
	    pointer++;
	return M[pointer]*x+B[pointer];
}
int main()
{
	int M,N,i;
	pair<int,int> a[50000];
	pair<int,int> rect[50000];
	freopen("acquire.inp","r",stdin);
	freopen("acquire.out","w",stdout);
	cin>>M;
```

```
        for (i=0; i<M; i++)
            cin>>a[i].first>>a[i].second;
        // S p x p theo chi u dài, n u chi u dài b ng nhau thì s p x p theo chi u
 r ng.
        sort(a,a+M);
        for (i=0,N=0; i<M; i++)
        {
  //Hcn có chi u r ng nh  h n là d  th a vì nó đã n m trong hcn có chi u r ng
 //cao h n r i, nên lo i b  càng nhi u càng t t.
            while (N>0&&rect[N-1].second<=a[i].second) N--;
            rect[N++]=a[i]; //Thêm hcn m i.
        }
        long long cost;
        add(rect[0].second,0);
//Ban đ u đ  ng t t nh t có th  là đ  ng b t kì trong bao l i, nên đ t pointer = 0
        pointer=0;
        for (i=0; i<N; i++)
        {
            cost=query(rect[i].first);
            if (i < N-1)
                  add(rect[i+1].second,cost);
        }
        cout<<cost;
        return 0;
 }
```

2.4. Test.

Đường dẫn:

https://drive.google.com/file/d/0BwQZ8OxBy1tPeWl0VXE0dmNDanc/view?usp=sharing

2.5. Cảm nhận

Bài toán là dạng bài toán 2. Độ phức tạp là

Bài 3: COMMANDO.

3.1. Đề bài:

Nguồn: APIO 2010.

Cho một dãy có số nguyên dương và một hàm bậc với hệ số nguyên dương .

- *Yêu cầu:* Phân dãy đã cho thành các đoạn liên tiếp sao cho tổng các hàm trên các đoạn là lớn nhất (giá trị của hàm lên dãy là với là tổng dãy đó).
- *Dữ liệu vào:* Cho trong file **COMMANDO.INP**
  - Dòng đầu tiên chứa số nguyên .
  - Dòng tiếp theo chứa số nguyên là hệ số các hàm bậc .
  - Dòng cuối chứa số nguyên .
- *Dữ liệu ra:* In ra file **COMMANDO.OUT.**
  - Một dòng duy nhất là tổng nhỏ nhất tìm được.
- *Ví dụ:*

COMMANDO.INP	COMMANDO.OUT
4 -1 10 -20 2 2 3 4	9
5 -1 10 -20 1 2 3 4 5	13
8 -2 4 3 100 12 3 4 5 2 4 2	-19884

- *Giải thích:*

- Với test : và .
- Ta có thể chia dãy thành các đoạn với tổng của mỗi dãy là . Tương ứng của mỗi dãy là . Và tổng các hàm lớn nhất trên các đoạn là .

- *Giới hạn:*
  - Thời gian: giây.
  - số test có .
  - số test có .
  - số test có .

3.2. Hướng dẫn giải thuật.

- Dễ dàng tìm ra được công thức quy hoạch động :
  là tổng các số của dãy từ vị trí đến vị trí

- Biến đổi hàm "adjust":
  - Định nghĩa là Vậy với một số bất kì ta có thể viết là:

  .
- Trong đó:
  -
  -
  -
- Ta có thể thấy là đại lượng mà chúng ta muốn tối ưu hóa bằng cách chọn . sẽ bằng đại lượng đó cộng thêm với (độc lập so với). Trong đó cũng độc lập với , và và phụ thuộc vào .
- Ngược với bài "acquire" khi chúng ta phải tối thiểu hóa hàm quy hoạch động thì bài này chúng ta phải cực đại hóa nó. Chúng ta phải xây dựng một hình bao trên với các đường thẳng tăng dần về hệ số góc. Do đề bài đã cho a<0 hệ số góc của chúng ta tăng dần và luôn dương.
- Do dễ thấy , giống như bài "acquire" các truy vấn chúng ta cũng tăng dần theo thứ tự.

3.3. Chương trình.

```
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 1000001;
int T, N , a , b , c;
int x[MAXN];
long long sum[MAXN];
long long dp[MAXN],M[MAXN],B[MAXN];
int topm,topb;
bool bad(int l1,int l2,int l3)
{
        return (B[l3]-B[l1])*(M[l1]-M[l2])<(B[l2]-B[l1])*(M[l1]-M[l3]);
}
void add(long long m,long long b)
{
        topm++; M[topm-1]=m;
        topb++; B[topb-1]=b;
        while (topm>=3&&bad(topm-3,topm-2,topm-1))
        {
                M[topm-2]=M[topm-1]; topm--;
                B[topb-2]=B[topb-1]; topb--;
```

```
        }
}
int pointer;
long long query(long long x)
{
        if (pointer >=topm)
                pointer=topm-1;
        while (pointer<topm-1&&
          M[pointer+1]*x+B[pointer+1]>M[pointer]*x+B[pointer])
                pointer++;
        return M[pointer]*x+B[pointer];
}
int main(){
       freopen("commando.inp","r",stdin);
      freopen("commando.out","w",stdout);
        cin>>N;
        cin>>a>>b>>c;
        for(int n = 1 ; n <= N ; n++){
                cin>>x[n];
                sum[n] = sum[n - 1] + x[n];
        }
        dp[1] = a * x[1] * x[1] + b * x[1] + c;
        add(-2 * a * x[1] , dp[1] + a * x[1] * x[1] - b * x[1]);

        for(int n = 2 ; n <= N ; n++){
                dp[n] = a * sum[n] * sum[n] + b * sum[n] + c;
                dp[n] = max(dp[n] , b * sum[n] + a * sum[n] * sum[n] + c +
query(sum[n]));
                add(-2 * a * sum[n] , dp[n] + a * sum[n] * sum[n] - b *
sum[n]);
        }
        cout<< dp[N];
        return 0;
}
```

3.4. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPTEtSc0doS1BVNkE/view?usp=sharing

3.5. Cảm nhận

Bài toán là dạng quy hoạch động tìm giá trị cực đại có thể áp dụng kĩ thuật bao lồi như bài toán 2.
Độ phức tạp là

Bài 4: Xây dựng nhà.

4.1. Đề bài

Nguồn: http://codeforces.com/contest/91/problem/E

Mức độ: Khó.

Có con hải mã được đánh số từ đến . Mỗi con hải mã xây dựng một ngôi nhà. Ban đầu, tại thời điểm bằng , chiều cao của ngôi nhà thứ là . Mỗi phút, con hải mã thứ xây được tầng.

Có q truy vấn, mỗi truy vấn được đặc trưng bởi một nhóm ba số . Câu trả lời của mỗi truy vấn là một số , thỏa mãn:

- Số nằm trong khoảng từ đến
- Ngôi nhà của con hải mã có chiều cao tối đa trong số các ngôi nhà của các con hải mã trong khoảng vào thời điểm .

- *Yêu cầu:* Đối với mỗi truy vấn, in ra số thỏa mãn hai điều kiện trên. Nếu có nhiều câu trả lời hãy in bất kỳ câu trả lời nào.
- *Dữ liệu vào:*
  - Dòng đầu tiên chứa các số và

- dòng tiếp, mỗi dòng chứa cặp số
- dòng tiếp theo, mỗi dòng là một truy vấn bao gồm: . Tất cả các số đều là số nguyên.

- *Dữ liệu ra:*
  - Đối với mỗi truy vấn, in ra số thỏa mãn. Mỗi số trên một dòng.
  - Nếu có nhiều kết quả có thể in ra một đáp án bất kì.
- *Ví dụ:*

91E.INP	91E.OUT
5 4 4 1 3 5 6 2 3 5 6 5 1 5 2 1 3 5 1 1 0 1 5 0	5 2 1 5

- *Giới hạn:*
  -
  -
  -
  - Thời gian: 3 giây.
  - 30% test có .
  - 30% test có .
  - 30% test có .

4.2. Hướng dẫn giải thuật.

Tại thời điểm , chiều cao của mỗi ngôi nhà thứ là .

Với mỗi truy vấn , ta cần tìm giá trị .

Ta có một tập gồm đường thẳng dạng và điểm trên trục . Chúng ta cần đi tìm các giá trị trong đó mỗi thỏa mãn:

với .

Với mỗi giá trị tìm được thì ta đưa ra được giá trị tương ứng làm cho đạt .

Đây chính là bài toán 1.

4.3. Chương trình.

```
#include <bits/stdc++.h>
using namespace std;
#define mn 100010
#define mm 1000010

struct node{
    int l;
    int r;
    vector <int> id;
};

node tr[mm];

int a[mn],b[mn],d[mn],lt,n, q;

bool cmp(int l, int r){
    if(a[l] == a[r]) return b[l] >
b[r];
    return a[l] > a[r];
}

void update(int step){
    lt = 0;
    for(int i = tr[step].l; i <=
tr[step].r; i ++) d[lt ++] = i;
    sort(d, d + lt, cmp);
    int tmp = lt;
    lt = 0;
```

```
    d[lt ++] = d[0];

    for(int i = 1; i < tmp; i ++){
        if(b[d[i]] <= b[d[lt - 1]])
            continue;
        d[lt ++] = d[i];
    }
    tmp = lt;
    lt = 0;
    d[lt ++] = d[0];
    for(int i = 1; i < tmp; i ++){
         while(lt > 1 && (long long)
(a[d[lt-2]] - a[d[lt-1]]) * (b[d[i]] -
b[d[lt-1]])
        >= (long long) (a[d[lt-1]] -
a[d[i]]) * (b[d[lt-1]] - b[d[lt-2]]))
lt--;
        ;

        d[lt ++] = d[i]; //dung mang
thay stack
    }
    tr[step].id.resize(lt);
    for(int i = 0; i < lt; i ++)
tr[step].id[i] = d[i];
}


void build(int l, int r, int step){
    tr[step].l = l;
    tr[step].r = r;
    tr[step].id.clear();

    update(step);

    if(l == r) return;
    int mid = (l + r) >> 1;
    build(l, mid, step * 2);
    build(mid + 1, r, step * 2 + 1);
}

int getMax(vector <int> id, long long
t){
    int n = id.size();
    int l = 0;
    int r = n - 1;
    int ret = l;
    while (l < r){
        int mid = (l + r) >> 1;
        if ((long long)t * b[id[mid]] +
a[id[mid]]  <= (long long) t * b[id[mid
+ 1]] + a[id[mid + 1]]){
            ret = mid;
            l = mid + 1;
        }
        else{
           r = mid;
        }
    }
    return id[l];
}


int Query(int l, int r, long long t,
int step){
    if(tr[step].l >= l && tr[step].r <=
r) return getMax(tr[step].id, t);

    int mid = (tr[step].l + tr[step].r)
>> 1;

    if(r <= mid)
        return Query(l, r, t, step *
2);
    else if(l > mid)
        return Query(l, r, t, step * 2
+ 1);
    else{
        int p1 = Query(l, mid, t, step
* 2);
        int p2 = Query(mid + 1, r, t,
step * 2 + 1);
        if((long long) b[p1] * t +
a[p1] >= (long long) b[p2] * t + a[p2])
return p1;
        return p2;
    }
}

int main(){
    freopen("91E.INP", "r", stdin);
    freopen("91E.OUT", "w", stdout);
    cin >> n >> q;
    for(int i = 1; i <= n; i ++)
scanf("%d %d", &a[i], &b[i]);
    build(1, n, 1);
    int l, r, t;
    for (int i=1; i <= q ; i++) {
        scanf("%d %d %d", &l, &r, &t);
        printf("%d\n", Query(l, r, t,
1));
    }
    return 0;
}
```

4.4. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPdVdHNE4taU04UmM/view?usp=sharing

4.5. Cảm nhận.

Bài toán là trường hợp bài toán 1, trong hàm tìm bao lồi, thêm việc kiểm tra hai đường thẳng trùng nhau và hai đường song song (cùng hệ số góc). Độ phức tạp của bài toán là: .

Bài 5: Vô hiệu cài đặt.

5.1. Đề bài

Nguồn: https://www.codechef.com/problems/CLDSIN

Mức độ: Trung bình – Khó.

Có người đã cài đặt N khẩu súng tự động dọc trên đường tại các vị trí khác nhau, súng sẽ bắn vào siêu nhân MAN nếu anh ta cố tình bay lên trên vị trí của chúng.

Để vượt qua đoạn đường nguy hiểm, MAN quyết định thuê đệ tử vô hiệu hóa bản cài đặt này. Một đệ tử có thể bắt đầu công việc tại bất kỳ điểm nào trên đường và tiếp tục vô hiệu hóa các khẩu súng khác dọc theo con đường bằng cách di chuyển lần lượt; mỗi khi vô hiệu hóa được một khẩu súng đệ tử sẽ được trả số tiền là và được thưởng số tiền là bình phương khoảng cách anh ta đã đi.

- *Yêu cầu:* Hãy giúp MAN tính số tiền tối thiểu cần trả cho các đệ tử để có thể vô hiệu hóa được tất cả các khẩu súng.
- *Dữ liệu vào:* Cho trong file **CLDSIN.INP.**
  - Dòng đầu chứa số nguyên , mô tả số khẩu súng và phần thưởng bắt buộc.
  - Dòng thứ chứa số nguyên mô tả vị trí của các khẩu súng theo thứ tự tăng dần.
- *Dữ liệu ra:* In ra file **CLDSIN.OUT.**
  - In ra một số nguyên là số tiền tối thiểu MAN phải trả.
- *Ví dụ:*

CLDSIN.INP	CLDSIN.OUT
2 10 1 2	11

- *Giải thích*
  - Đệ tử vô hiệu hóa ít hơn nếu theo cách
- *Giới hạn:*
  -
  -
  -

5.2. Hướng dẫn giải thuật.

Gọi là chi phí nhỏ nhất cho cài đặt thứ .

.

Công thức này chúng ta tính tất cả cho mỗi , nhưng nó không thực sự cần thiết và quá chậm để vượt qua tất cả các test.

Sử dụng CHT có thể giải bài toán trong thời gian tuyến tính.

Các đường thẳng được thêm vào cấu trúc dữ liệu có dạng

trong đó và .

Ở cài đặt thứ ta đặt và giá trị là chi phí cần tìm.

Vì các đã sắp xếp nên chỉ cần đi từ tới là thỏa mãn.

5.3. Chương trình.

```
#include<bits/stdc++.h>
using namespace std;
#define pb push_back
#define mp make_pair
#define f first
#define s second
#define fr(i,n) for(i=0;i<n;i++)

typedef long long ll;
const int N=1000010;
ll P[N],R;
int sz=0,idx=0;
struct Line
{
    ll m,c;
    Line(){};
    Line(ll m1,ll c1){m=m1;c=c1;};
};
Line hull[N];
inline ll getval(Line a,ll x)
{
    return a.m*x+a.c;
}
inline double meet(Line &a,Line &b)
    {
        return ((b.c-
a.c)*1.0)/(double)(a.m-b.m);
    }

inline bool useless(Line &lft,Line
&mid,Line &rt)
{
    return
(meet(lft,mid)>meet(mid,rt));
}
void addline(ll slope,ll inter)
{
    Line temp(slope,inter);
    if (!sz)
    {
        hull[sz++]=temp;
        return;
    }
    while(sz>1 && useless(hull[sz-
2],hull[sz-1],temp))
        sz--;
    idx=min(idx,sz-1);
    hull[sz++]=temp;
}
ll best_result(ll x)
{
    while(idx+1<sz &&
meet(hull[idx],hull[idx+1])<(double)x)
        idx++;
    return getval(hull[idx],x);
}
int main()
{
    freopen("CLDSIN.INP","r",stdin);
    freopen("CLDSIN.OUT","w",stdout);
    int t;
    int n,i,j;
    cin>>n>>R;
    fr(i,n)cin>>P[i];
    i=0;
    ll last=0;
    for(i=0;i<n;i++)
    {
        addline(-
2*P[i],last+P[i]*P[i]);
        last=R+
P[i]*P[i]+best_result(P[i]);
    }
    cout<<last<<endl;
    return 0;
}
```

5.4.

5.5. Test.

Đường dẫn:

https://drive.google.com/file/d/0BwQZ8OxBy1tPNHpmSjBxNk9Xd3c/view?usp=sharing

5.6. Cảm nhận

Bài toán áp dụng bài toán 3. Độ phức tạp của bài toán là

Bài 6: CYCLRACE – đua xe đạp.

6.1. Đề bài

Nguồn: https://www.codechef.com/problems/CYCLRACE

Chef được mời làm giám khảo của một cuộc đua xe đạp. Để thưởng điểm cho các thí sinh, Chef muốn biết quãng đường chạy được của mỗi vận động viên tại một thời điểm bất kỳ.

Có hai loại truy vấn:

- Thay đổi vận tốc của vận động viên thứ tại thời điểm nào đó.
- Hỏi vị trí của người dẫn đầu cuộc đua tại một thời điểm nào đó.

Có tất cả là truy vấn. Mỗi truy vấn loại được mô tả bởi thời gian , số hiệu của vận động viên và vận tốc mới . Mỗi truy vấn loại được mô tả bởi thời gian .

Vận tốc ban đầu của các vận động viên là .

- *Dữ liệu vào:*
  - Dòng đầu tiên chứa hai số nguyên và cách nhau bởi khoảng trắng.
  - dòng tiếp theo chứa các truy vấn, mỗi truy vấn trên một dòng. Mỗi truy vấn gồm vài số nguyên cách nhau bởi khoảng trắng. Số nguyên đầu tiên là , có thể là hoặc chỉ loại của truy vấn.
    - Nếu loại của truy vấn là thì ba số nguyên theo sau là: và .
    - Nếu loại của truy vấn là , thì một số nguyên theo sau là: . Tất cả các truy vấn đều có thời gian khác nhau và được sắp xếp theo thời gian tăng dần.
- *Dữ liệu ra:*
  - Với mỗi bộ dữ liệu, là kết quả của truy vấn loại thứ trong bộ dữ liệu hiện tại và là tổng số lượng truy vấn loại hai. In mỗi trên một dòng.
- *Ví dụ:*

CYCLRACE.INP	CYCLRACE.OUT
5 14 1 1 1 2 1 1 2 5 1 2 3 4 1 2 4 7 2 3 2 4 1 5 5 4 2 5 2 6 1 7 5 8 2 7 2 8 2 9 2 10	10 15 21 28 35 42 49 56

- *Giới hạn:*
  -
  -
  -
  -
  -
  - Vận tốc của mỗi vân động viên không thể bị giảm xuống.
  - Subtask số test có và .
  - Subtask như ràng buộc gốc.

6.2. Hướng dẫn giải thuật.

Nếu vị trí của một người là và vận tốc là , thì tại thời điểm vị trí của anh ta là . Nó chính là đường thẳng đi qua điểm và có hệ số góc là .

Ta có thể thay đổi vận tốc tại thời điểm từ thành . Bây giờ vị trí của anh ta tại thời điểm sẽ là:

trong đó .

- Xét truy vấn 1: Tìm vị trí của người chiến thắng tại thời điểm tức là tìm giá trị của bao lồi tại thời điểm .
- Xét truy vấn 2: Thay đổi tốc độ của người thứ từ thành tại thời điểm . Tức là đường thẳng tương ứng với được thay thế bằng đường thẳng mới . Vận tốc của một người không bao giờ giảm vì vậy chúng ta xóa đường thẳng cũ hơn để cập nhật. Do đó chỉ cần bổ sung các đường thẳng trong bao lồi.

Ta có thể thêm một đường thẳng trong bao lồi bằng cách sử dụng tìm kiếm nhị phân.

Độ phức tạp:

6.3. Chương trình.

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define f first
#define s second
#define mp make_pair
#define mn 100010
pair<ll,ll> a[mn];
ll speech[mn],pos[mn],s,m,b;
int t,c,ch,n,q,p,topa=0;
vector<pair<ll,ll> >dt;
vector<int> tv;
bool
kc(pair<ll,ll>a,pair<ll,ll>b,pair<ll,l
l>c)
{
    return ((c.s-a.s)/(double)(a.f-
c.f))<((b.s-a.s)/(double)(a.f-b.f));
}
void update(ll m,ll b)
{
    int n=topa;
    pair<ll,ll> ht=mp(m,b);
    int i=topa;
    while(i>1 && kc(a[i-2],a[i-1],ht))
        {
            topa--;
            i--;
        }
    topa++; a[topa-1]=ht;
}
ll query(int x)
{
    int n=topa;
    if(p>=topa) p=topa-1;

    while(p<n-1 &&
(a[p].f*x+a[p].s)<(a[p+1].f*x+a[p+1].s
)) p++;
    return (a[p].f*x+a[p].s);
}
int main()
{
    freopen("CYCLRACE.INP","r",stdin);
freopen("CYCLRACE.OUT","w",stdout);
    cin>>n>>q;
    while(q--)
    {
        cin>>ch;
        if (ch==1)
        {           cin>>t>>c>>s;
                    m=speech[c];
                    b=pos[c];
                    speech[c]=s;
                    pos[c]=b+(m-s)*t;
dt.push_back(mp(speech[c],pos[c]));
        }
         else
                {
                    cin>>t;
                    tv.push_back(t);
                }
    }
    sort(dt.begin(),dt.end());
```

```
    for(int i=0;i<dt.size();i++)
update(dt[i].f,dt[i].s);
    for(int i=0;i< tv.size();i++)
cout<<query(tv[i])<<endl;
    return 0;
}
```

6.4.

6.5. Test

6.6. Cảm nhận

Bài toán áp dụng kĩ thuật bao lồi trong trường hợp bài toán 1. Độ phức tạp của bài toán là .

Bài 7: JUMP.

7.1. Đề bài:

Nguồn: https://www.codechef.com/problems/JUMP

Bếp trưởng đang tham gia vào một cuộc thi leo đồi. Có ngọn đồi thẳng hàng, được đánh số từ đến . Đồi thứ i có chỉ số và chiều cao .

Bếp trưởng bắt đầu ở đồi đầu tiên và muốn đi đến đồi thứ bằng cách leo qua các ngọn đồi từ trái sang phải (không được đi theo hướng khác), leo từ đồi thứ sang đồi thứ mất năng lượng.

Khi bếp trưởng leo đến đồi thứ, anh ấy phải chuẩn bị một món ăn đặc biệt cho mọi người như một lời cảm ơn, điều này khiến anh ta tốn năng lượng (một số món ăn làm bếp trưởng rất yêu thích, và việc chuẩn bị nó làm anh ta rất vui vẻ nên khi đó có thể âm).

Bếp trưởng có thể leo từ đồi thứ sang đồi thứ nếu và .

Hãy giúp bếp trưởng chọn hành trình sao cho tốn ít năng lượng nhất có thể.

Chú ý: Trong bất kỳ thời điểm nào, năng lương tiêu thụ có thể là âm.

- *Dữ liệu vào:*
  - Dòng đầu chứa số nguyên
  - Dòng thứ chứa số nguyên . là hoán vị của số nguyên đầu tiên.
  - Dòng thứ 3 chứa số nguyên .
  - Dòng thứ 4 chứa số nguyên .
- *Dữ liệu ra:*
  - In ra số nguyên là năng lượng tiêu hao tối thiểu.
- *Giới hạn:*
  - Subtask
  - Subtask
  - Subtask .
  - Subtask , .
  - Subtask Những giới hạn ban đầu.
- *Ví dụ:*

JUMP.INP	JUMP.OUT
5 1 4 3 2 5 0 1 3 0 0 1 2 3 4 5	10

7.2. Hướng dẫn giải thuật:

Thuật toán 1: O(n^2). Quy hoạch động

Gọi là số điểm đạt được khi đang đứng ở đồi thứ i. là kết quả bài toán.
Công thức :

-
-

Không thể nhảy từ ngọn đồi nào tới ngọn đồi thứ . Từ ngọn đồi thứ chúng ta có thể tìm thấy tất cả ngọn đồi mà có thể nhảy tới ngọn đồi với .

Thuật toán 2: Convex hull trick

- Viết lại công thức

- Sau khi tính được giá trị mới của , ta có thể sửa đổi tại vị trí với các tham số:
  và

- Chúng ta duy trì các hoạt động sau:
  - Thêm một hàm tuyến tính vào tập hợp.
  - Tìm với giá trị đã cho trong các hàm có trong tập hợp.
    Cả hai hoạt động đều thực hiện trong .
- Chúng ta có thể tạo ra một cây phân đoạn để lưu cấu trúc CHT. Các hoạt động trên cấu trúc dữ liệu này là:
  - Sửa đổi: Thêm các đường thẳng tới tất cả các nút có chứa vị trí của đường này. Vì đây là cây phân đoạn, sẽ không có nhiều hơn các nút như vậy. Mỗi lần thêm một nút mất . Vì vậy độ phức tạp của việc này là .
  - Truy vấn: Nó chỉ hoạt động trong cây phân đoạn chuẩn. Đoạn truy vấn ban đầu sẽ bao gồm các nút truy vấn . Và trong mỗi truy vấn việc tìm thấy giá trị tối thiểu của mỗi hàm mất . Vì vậy độ phức tạp của việc này là .

7.3. Chương trình.

```
#include <bits/stdc++.h>
#define mn 600010
using namespace std;

int i,j,m,n,p,k,vis[mn],tot,l[mn];
vector<int> cay[mn];
long long f[mn];
struct Node{
    long long p,a,h,id,F,G;
};
Node A[mn];
inline bool cmp(Node a,Node b) {
     return a.h<b.h;
}
inline bool cmp1(Node a,Node b) {
    return a.id<b.id;
}
void chen(int x,int y)
{
           if (vis[x]!=tot)
vis[x]=tot,cay[x].clear(),l[x]=0;
           int r=(int)cay[x].size()-1;
           while (l[x]<r)
           {
                    int
p=cay[x][r],p1=cay[x][r-1];
                    if ((A[y].F-
A[p].F)*(A[p].G-A[p1].G)<=(A[p].F-
A[p1].F)*(A[y].G-A[p].G)) --
r,cay[x].pop_back();
                    else break;
           }
          cay[x].push_back(y); // dung
vecto thay stack vi dung mang 2 chieu
qua l n
}
long long sqr(long long x) { return
x*x; }
int lowbit(int x) { return x&-x; }
void tl(int x,int y)
{
          if (vis[x]!=tot) return;
          int len=cay[x].size();
         while (l[x]<len-1)
         {
                    int
p=cay[x][l[x]],p1=cay[x][l[x]+1];
                    if ((A[p1].F-
A[p].F)<=A[y].h*(A[p1].G-A[p].G))
++l[x];
                    else break;
         }
         int p=cay[x][l[x]];

f[A[y].id]=min(f[A[y].id],f[A[p].id]+sq
r(A[p].h)-2*A[p].h*A[y].h);
}
void work(int l,int r)
{
  int mid=(l+r)/2;
```

```
    int i,x;
    ++tot;
    for (i=l;i<=mid;++i)
     for (x=A[i].p;x<=n;x+=lowbit(x))
chen(x,i);
    for (i=mid+1;i<=r;++i)
      for (x=A[i].p;x;x-=lowbit(x))
tl(x,i);
}
void tao(int l,int r)
{
          if (l==r)
          {
                  f[l]+=A[l].a;
                  if (l>1)
f[l]+=A[l].h*A[l].h;

A[l].F=f[l]+A[l].h*A[l].h;
                  A[l].G=2*A[l].h;
                  return;
          }
          int mid=(l+r)/2;
          tao(l,mid);
          sort(A+mid+1,A+r+1,cmp);
          work(l,r);
          sort(A+mid+1,A+r+1,cmp1);
          tao(mid+1,r);
          sort(A+l,A+r+1,cmp);
}
int main()
{
     freopen("JUMP.INP","r",stdin);
     freopen("JUMP.OUT","w",stdout);
         cin >> n;
         for (int i=1;i<=n;i++) cin >>
A[i].p;
         for (int i=1;i<=n;i++) cin >>
A[i].a;
         for (int i=1;i<=n;i++)
     {
         cin >> A[i].h;
         A[i].id=i;
     }
     memset(f,60,sizeof(f));//
f[i]=MAX_INT
     f[1]=0;
     tao(1,n);
         cout<<f[n]<<endl;
}
```

7.4.

7.5. Test:

Đường dẫn:

7.6. Cảm nhận.

Bài toán là dạng bài toán 2. Độ phức tạp là .

Bài 8: Function.

8.1. Đề bài

Nguồn: http://codeforces.com/problemset/problem/455/E

Mức độ: Trung bình – Khó.

Cho các hàm như sau:

.

Trong đó: là một mảng gồm số nguyên.

- *Yêu cầu:* Giá trị của hàm là bao nhiêu với một số điểm cho trước.
- *Dữ liệu vào:*
  - Dòng đầu tiên chứa số nguyên
  - Dòng tiếp theo chứa số nguyên .
  - Dòng tiếp theo chứa số nguyên là số truy vấn.
  - dòng tiếp theo mỗi dòng chứa hai số nguyên . Có nghĩa là cần tính .
- *Dữ liệu ra:*
  - dòng, mỗi dòng là câu trả lời cho mỗi truy vấn.
- *Ví dụ:*

455E.INP	455E.OUT
6 2 2 3 4 3 4 4 4 5 3 4 3 4 2 3	12 9 9 5
7 1 3 2 3 4 0 2 4 4 5 2 3 1 4 4 6	11 4 3 0

- *Giới hạn:*
  - Thời gian: giây.

8.2 Hướng dẫn giải thuật.

Ta có:

trong đó là số lần truy vập vào phần tử của mảng

Trên đoạn thì được truy cập lần và tất cả các số khác một lần.

Cần tìm nhỏ nhất.

là một tổng các tiền tố của mảng .

.

.

Đặt : ; ;

Nên: ;

Ta phải tìm giá trị nhỏ nhất với mọi và đã được xác định;

Ta có đường thẳng, mỗi phần tử của mảng là một đường thẳng .

.

Trong đó là đường thẳng thứ , , .

Ta sử dụng CHT với cây phân đoạn để giải bài toán. Mỗi đỉnh của cây lưu trữ tất cả các đường thẳng của đỉnh này, điều này làm mất không gian vì mỗi đường thẳng ở trên đỉnh. Ta thăm đỉnh và mỗi đỉnh thăm lần, vì vậy cần để trả lời hết các truy vấn.

8.3 Chương trình.

```
#include <bits/stdc++.h>
#define mn 100010
using namespace std;
int n,m,i,j,k,u,v,l,r,mid,aim;
int a[mn],sum[mn];
```

```
int son[mn],next[2000005],x[2000005];     }
long long ans[2000005];
int st[mn],top;

double cal(int x,int y)
{
      double res=(sum[x]-
sum[y]+(double)y*a[y]-
(double)x*a[x])/(a[x]-a[y]);
      return res;
}

int main()
{
    freopen("455E.INP","r",stdin);
    freopen("455E.OUT","w",stdout);
      cin>>n;
      for(i=1;i<=n;++i)
    {
        cin>>a[i];
        sum[i]=sum[i-1]+a[i];
    }
      cin>>m;
      for(i=1;i<=m;++i)
      {
            cin>>u>>v;

      next[i]=son[v];son[v]=i;x[i]=u;
      }
      for(i=1;i<=n;++i)
      {

      while(top&&a[i]<=a[st[top]])--
top;


while(top>1&&cal(st[top],i)>=cal(st[top
-1],i))--top; // stack chl
            st[++top]=i;
            for(j=son[i];j;j=next[j])
            {
                  l=1;r=top;aim=top;
                  while(l<=r)
                  {
                        mid=l+r>>1;

      if(st[mid]+x[j]>=i)aim=mid,r=mid-
1;
                        else l=mid+1;
                  }
                  l=aim;r=top-
1;aim=top;
                  while(l<=r)
                  {
                        mid=l+r>>1;

      if(cal(st[mid],st[mid+1])<=x[j]-
i)aim=mid,r=mid-1;
                        else l=mid+1;
                  }
                  ans[j]=sum[i]-
sum[st[aim]]+(long long)(x[j]-
i+st[aim])*a[st[aim]];
            }
      }
      for(i=1;i<=m;++i)
cout<<ans[i]<<endl;
```

8.4. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPV2laR2NWUXB2TUk/view?usp=sharing

8.5. Cảm nhận

Bài toán là dạng bài toán 2. Độ phức tạp là .

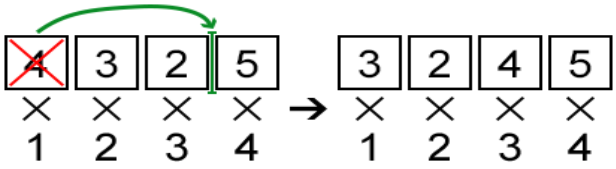
Bài 9: Product Sum

9.1. Đề bài

Nguồn: http://codeforces.com/problemset/problem/631/E

Mức độ: Khó.

Bạn được cho một mảng gồm số nguyên. Giá trị của mảng được tính bằng công thức , với phần tử thứ có giá trị . Thực hiện thao tác sau một lần: chọn một phần tử của mảng và có thể di chuyển tới vị trí bất kì với . Ngoài ra, được phép đưa phần tử đó trở lại vị trí ban đầu. Mục đích là để có được mảng với giá trị lớn nhất.



- *Yêu cầu:* Hãy thực hiện thao tác trên không quá một lần và đưa ra giá trị lớn nhất của mảng
- *Dữ liệu vào:*
  - Dòng đầu chứa số nguyên là kích thước của mảng .
  - Dòng thứ chứa n số nguyên
- *Dữ liệu ra:*
  - In một số nguyên là giá trị tối đa của mảng.
- *Ví dụ:*

631E.INP	631E.OUT
4 4 3 2 5	39
5 1 1 2 7 1	49
3 1 1 2	9

- *Giải thích*
  - Với test 1, đưa phần tử 4 đến trước phần tử 5, kết quả sẽ là:

- Với test 2, đưa phần tử 5 đến trước phần tử 3, kết quả sẽ là:

9.2. Hướng dẫn giải thuật.

*) Thuật toán

- Thao tác được mô tả trong bài toán là sự biến đổi theo chu kỳ của một số mảng con.
- Ta giải quyết riêng biệt cho việc thay đổi chu kỳ trái và chu kỳ phải.
- Gọi là giá trị chênh lệch của trước và sau khi biến đổi: , trong đó .
- Công thức cho việc biến đổi chu kỳ trái:

- Công thức biến đổi chu kỳ phải:

Ta có thể tìm giá trị này trong . Sử dụng tổng các số đầu của dãy
Suy ra:

Nếu cố định là dịch trái và là dịch phải, thì sử dụng CHT độ phức tạp của bài toán sẽ là .

9.3. Chương trình.

```
#include <bits/stdc++.h>
using namespace std;
#define mn 210000
#define ll long long
int n;
int st[mn],top;
ll ans,sum[mn],tot=0,a[mn];
ll cal1(int x,int y)
{
    return a[x]*(y-x)-sum[y]+sum[x];
}
ll cal2(int x,int y)
{
    return sum[x-1]-sum[y-1]-a[x]*(x-
y);
}
void xuli()
{
    for(int i=1;i<=n;i++)sum[i]=sum[i-
1]+a[i];
      for(int i=n;i>=1;i--)
      {
            int l=1,r=top;
            while(r-l>2)
            {
                  int
lmid=(l+l+r)/3,rmid=(l+r+r)/3;


if(cal1(i,st[lmid])>cal1(i,st[rmid]))r=
rmid;
                  else l=lmid;
            }
            for(int j=l;j<=r;j++)

      ans=max(ans,tot+cal1(i,st[j]));
            while(top>=2&&(sum[i]-
sum[st[top]])*(st[top]-st[top-
1])>=(sum[st[top]]-sum[st[top-1]])*(i-
st[top]))
            top--;
            st[++top]=i;
      }
      top=0;
      for(int i=1;i<=n;i++)
      {
            int l=1,r=top;
            while(r-l>2)
            {
                  int
lmid=(l+l+r)/3,rmid=(l+r+r)/3;


if(cal2(i,st[lmid])>cal2(i,st[rmid]))r=
rmid;
                  else l=lmid;
            }
            for(int j=l;j<=r;j++)

      ans=max(ans,tot+cal2(i,st[j]));
            while(top>=2&&(sum[i-1]-
sum[st[top]-1])*(st[top]-st[top-
1])<=(sum[st[top]-1]-sum[st[top-1]-
1])*(i-st[top]))
            top--;
            st[++top]=i;
      }
    cout << ans;
}
int main()
{
    ios_base::sync_with_stdio(0);
      freopen("631e.inp","r",stdin);
      freopen("631e.out","w",stdout);
      cin >> n;
      for(int i=1;i<=n;i++)   cin >>
a[i];
      for(int i=1;i<=n;i++)
tot+=a[i]*i;
      ans=tot;
      xuli();
}
```

9.4.

9.5. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPNjBnRGxLVmRVNjQ/view?usp=sharing

9.6. Cảm nhận

Bài toán này là ứng dụng của bài toán 2 nhưng là các giá trị để hàm đạt cực đại.
Độ phức tạp là

Bài 10: Xây dựng nhà.

10.1. Đề bài

Nguồn: http://codeforces.com/problemset/problem/377/E

Mức độ: Khó.

Tại thời điểm , số tiền Kiên có là và chưa có tòa nhà nào.

Có tòa nhà để lựa chọn: tòa nhà thứ có giá là và khi nó được chọn sẽ thu được số tiền là trong mỗi giây. Tại mỗi thời điểm, chỉ có thể chọn một tòa nhà. Tất nhiên, Kiên có thể thay đổi tòa nhà đang sử dụng mỗi giây theo ý của mình.

Kiên chỉ có thể chọn các tòa nhà mới và thay đổi tòa nhà đang sử dụng chỉ vào những thời điểm là bội số của một giây. Kiên có thể chọn tòa nhà mới và sử dụng nó cùng một lúc. Nếu Kiên bắt đầu sử dụng tòa nhà vào thời điểm , anh ta có thể kiếm được lợi nhuận đầu tiên từ nó ở vào thời điểm

- *Yêu cầu:*
  - Kiên muốn kiếm được số tiền ít nhất là bằng càng nhanh càng tốt. Xác định số giây anh ta cần để làm điều đó.
- *Dữ liệu vào:*
  - Dòng đầu tiên chứa hai số nguyên n và s tương ứng là số tòa nhà và số tiền mà Kiên muốn kiếm được.
  - Mỗi dòng trong n dòng tiếp theo chứa hai số nguyên và tương ứng là số tiền mà tòa nhà thứ i mang lại mỗi giây và giá của tòa nhà.
- *Dữ liệu ra:*
  - Chỉ ra số nguyên duy nhất là số giây tối thiểu mà Kiên cần.
- *Ví dụ:*

377E.INP	377E.OUT
3 9 1 0 2 3 5 4	6

- *Giới hạn:*
  - Thời gian: 2 giây.

10.2. Hướng dẫn giải thuật.

- Trên mặt phẳng tọa độ Decac, ta có trục Ox biểu thị thời gian, trục Oy biểu thị giá tiền.
- Ta có đồ thị y = f(x) – sum, trong đó sum là số tiền tối đa có thể thu được tại thời điểm x. Ta tiến hành xử lý lần lượt từng tòa nhà.
- Hàm này là tập hợp các đoạn thẳng với hệ số góc là v_i, và mỗi đoạn thẳng hoạt động trên một đoạn $[l_i, r_i]$ của trục Ox

10.3. Chương trình.

```
#include <bits/stdc++.h>
#define maxn 300000
#define ll long long
using namespace std;
```

```
typedef struct{
    ll v;
    ll c;
  } double_pair;
struct line { // 1 duong thang y = k*x
+b
    ll k, b;
    line () {}
    line (ll _k, ll _b) : k(_k), b(_b)
{}
    ll Solve(ll val) { // k * x + b >=
val
        ll _a = val - b, _b = k ;
        if ( _a % _b ==0 ) return _a /
_b; else return _a/_b + 1;
    }
    ll f(ll x) {return k * x + b; }
};
double_pair D[maxn];
int n;
line St[maxn];
int top;
ll X[maxn],s;


ll intersection( line a, line b) {
    ll _a = a.b - b.b, _b = b.k - a.k ;
    if ( _a % _b ==0 ) return _a / _b;
else return _a/_b + 1;
}

bool slope_compare(double_pair d1,
double_pair d2) {
    return ((d1.c < d2.c) || (d1.c ==
d2.c && d1.v > d2.v) ) ;
}

bool check(ll x, ll _s) {
    if (x > X[top-1] ) return x >=
St[top].Solve(_s);
    int pos = lower_bound(X+1, X+top,
x) - X; if(X[pos] == x) pos ++;
    return x >= St[pos].Solve(_s);
}

ll chat(ll _s) {
    ll l = 0, r = _s, mid ;
    while ( l + 1 < r) {
        mid = (l + r) >> 1;
        if ( check (mid, _s ) ) r = mid
; else l = mid;
    }
    return r;
}
void find_hull()
{
    St[ ++top ] = line(D[0].v, -
D[0].c); // D[0].c == 0
    ll i1, i2;
    for (int i = 1 ; i < n; ++ i) {
        ll x0 = chat(D[i].c) ;
        ll p = lower_bound(X+1, X+top,
x0) - X ;
        ll y0 = St[p+(X[p]==x0)].k * x0
+ St[p+(X[p]==x0)].b - D[i].c;
        line tmp (D[i].v, y0 - x0 *
D[i].v);

        while (top >= 2 && (((i1=
intersection(tmp, St[top])) <= (i2 =
intersection(St[top], St[top-1])) ||
(i1 == i2 && tmp.f(i1) >= St[top].f(i1)
)) )) -- top;
        St[ ++top ] = tmp; X[ top - 1]
= intersection(St[top], St[top-1]);
    }
}
int main()
{
    freopen("337E.INP","r",stdin);
    freopen("337E.OUT","w",stdout);
    cin >> n >> s;
    for ( int i=1; i<=n; i++) cin >>
D[i].v >> D[i].c;
    sort(D+1,D+n+1,slope_compare);
    int tmp_n = n, maxv = 0;  n = 0;
    for (int i=1; i<=tmp_n; i++) {
        if (maxv >= D[i].v) continue;
        maxv = D[i].v;
        D[n++] = D[i];
    }
    find_hull();
    cout <<chat(s);
    return 0;
}
```

10.4.

10.5. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPaW02S0hsM3FrRE0/view?usp=sharing

10.6. Cảm nhận

Bài toán này là dạng bài toán 1. Độ phức tạp là

Bài 11: Bowling.

11.1. Đề bài

Nguồn: http://codeforces.com/problemset/problem/660/F

Mức độ: Khó.

Con gấu nâu thích chơi bowling với bạn bè, hôm nay nó thực sự cố gắng để phá được kỷ lục của mình.

Khi 1 quả bóng đổ sẽ đạt được mộtt số điểm (là một số nguyên). Điểm của quả bóng thứ được nhân với và đó là điểm tổng kết. Vì vậy, với quả bóng điểm số , tổng số điểm là: . Tổng điểm là nếu không có quả bóng nào đổ.

Con gấu đổ n quả bóng và đạt a_i điểm cho quả bóng thứ i. Nó muốn đạt được tổng số điểm là tối đa và có một ý tưởng thú vị: Nó cho rằng những quả bóng đầu tiên và cuối cùng không quan trọng, vì vậy nó hủy bỏ bất kỳ tiền tố và hậu tố của chuỗi a1, a2, …, an. Nó được phép bỏ qua tất cả các cuộn hoặc không cuộn nào cả

Tổng điểm được tính như không có quả bóng nào bị hủy. Vì vậy điểm không hủy bỏ đầu tiên có điểm số nhân 1, điểm thứ 2 có điểm số nhân với 2, …cho đến cuộn không hủy cuối cùng.

- *Yêu cầu:* Tính tổng điểm tối đa mà con gấu có được.
- *Dữ liệu vào:*
  - Dòng đầu tiên chứa số nguyên n ($1<=n<=2*10^5$) tổng số quả bóng đổ
  - Dòng thứ 2 chứa n số nguyên $a_1, a_2, ..., a_n$ ($|a_i| \le 10^7$) điểm cho mỗi quả bóng con gấu làm đổ.
- *Dữ liệu ra:*
  - In ra tổng số điểm tối đa sau khi hủy bỏ một số quả bóng đổ.
- *Ví dụ:*

660F.INP	660F.OUT
6 5 -1000 1 -3 7 -8	16
5 1000 1000 1001 1000 1000	15003
3 -60 -70 -80	0

- *Giải thích*
  - Trong ví dụ đầu tiên, nên hủy bỏ hai cuộn đầu tiên, và một cuộn cuối cùng, sẽ được các cuộn 1, -3, 7, đạt được tổng số điểm: $1 \cdot 1 + 2 \cdot (-3) + 3 \cdot 7 = 1 - 6 + 21 = 16$.
- *Giới hạn:*

11.2. Hướng dẫn giải thuật.

Sử dụng các mảng cộng dồn:

Với mỗi giá trị R, chúng ta cần tìm L để là nhỏ nhất. Sử dụng tìm kiếm nhị phân và kĩ thuật bao lồi.

11.3. Chương trình.

```
#include <bits/stdc++.h>
#define maxn 200005
#define ll long long
#define F first
#define S second
using namespace std;
int n;
ll a[maxn];
ll s1[maxn];
ll s2[maxn];
ll ans;
pair < ll, ll >  S[maxn];
int top;
double find_intersection_x(pair < ll,
ll > a, pair < ll, ll > b) {
       double a1 = a.F, b1 = a.S, a2 =
b.F , b2 =b.S;
       return (b2-b1)/(a1-a2);

}
ll solve(pair < ll, ll > a, ll x) {
    //  a*x+b
      return (ll)(a.F*x+a.S);
}

void add_line(pair < ll, ll > a) {
      while(top>=2 &&
find_intersection_x(a,S[top-
1])>=find_intersection_x(S[top],S[top-
1])) top--;
      S[++top] =a;
}
ll get_max(ll x) {
      int i,l,r,mid;
      if(top==1) return
solve(S[top],x);
      if(
x<find_intersection_x(S[top],S[top-
1])) return solve(S[top],x);
      l=-1;
      r=top-1;
      while(r-l>1) {
            mid=(l+r)>>1;


if(x>=find_intersection_x(S[mid],S[mid
+1])) r=mid;
            else l=mid;
      }
      return solve(S[r],x);
}
int main()
{
    freopen("BOW.INP","r",stdin);
    freopen("BOW.OUT","w",stdout);
    cin >> n;
    for ( int i=1 ; i<=n;  i++ ) cin
>> a[i];
    s1[0]=0;
    s2[0]=0;
    ans =0;
    for ( int  i=1 ; i<=n; i++)
    {
        s1[i] = s1[i-1] + a[i];
        s2[i] = s2[i-1] + i*a[i];
        ans = max ( ans, s2[i] );
    }
    top=0;
    for(int i=1;i<=n;i++) {
            add_line(make_pair(1ll-
i,s1[i-1]*(i-1)-s2[i-1]));


ans=max(ans,s2[i]+get_max(s1[i]));
      }
      cout<<ans;
    return 0;
}
```

11.4.

11.5. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPc2V3XzdsdjV3eGs/view?usp=sharing

11.6. Cảm nhận.

Bài toán này thuộc dạng bài toán 2, độ phức tạp

Bài 12: Game.

12.1. Đề bài

Nguồn: http://codeforces.com/problemset/problem/673/E

Mức độ: Khó.

Rade chơi một Game có level, được đánh số từ tới . Level được chia thành nhóm, mỗi nhóm có một số các level liên tiếp.

Game lặp lại các bước sau:

1. Nếu tất cả các khu vực bị đánh bại thì trò chơi sẽ kết thúc ngay. Nếu không hệ thống sẽ tìm thấy khu vực đầu tiên có ít nhất một cấp độ không bị đánh. biểu thị khu vực này.
2. Hệ thống tạo ra một túi rỗng để lấy các thẻ. Mỗi mã thông báo sẽ đại diện cho một level và có thể có nhiều đồng xu đại diện cho cùng một level.
   i. Với mỗi level đã bị đánh bại trong khu vực , hệ thống sẽ thêm thẻ vào túi.
   ii. Level không bị đánh bại đầu tiên trong khu vực . Hệ thống sẽ thêm thẻ vào túi.
3. Cuối cùng, hệ thống lấy một thẻ ngẫu nhiên từ túi và người chơi bắt đầu ở level tương ứng với thẻ được lấy ra. Mỗi người chơi phải mất một giờ và đánh bại level đó, ngay cả khi level đó đã được đánh bại trước đó.

Cho và .

- *Yêu cầu:* Hãy phân vùng các level. Mỗi level phải thuộc chính xác một vùng và mỗi vùng phải chứa tập các level liên tiếp khác rỗng. Số giờ dự kiến để hoàn thành trò chơi là bao nhiêu?
- *Dữ liệu vào:*
  - Dòng đầu chứa hai số nguyên và là số lượng level và số vùng.
  - Dòng thứ 2 chứa số nguyên .
- *Dữ liệu ra:*
  - In ra một sô thực là số giờ tối thiểu dùng để hoàn thành trò chơi. Sai số .
  - Nếu câu trả lời của bạn là a và output chính xác là b thì .
- *Ví dụ:*

673E.INP	673E.OUT
4 2 100 3 5 7	5.7428571429
6 2 1 2 4 8 16 32	8.5000000000

- *Giải thích*
  - Trong test , chia level thành vùng. Vùng gồm level , vùng gồm level .
  - Trong test 2, chia thành vùng, mỗi vùng 3 level.
- *Giới hạn:*
  - .
  - .
  - .
  - Thời gian: 3 giây.

12.2. Hướng dẫn giải thuật.

Ta có hai nhận xét:

- k càng lớn thì kết quả càng nhỏ.
- Với mỗi ta có

Với mỗi i tồn tại giá trị X thực mà:

Giá trị X xác định trong khoảng (trừ trường hợp và

Dùng tìm kiếm nhị phân để tìm được X tốt nhất.

Sau khi tìm được X tốt nhất chúng ta tính giá trị tối ưu cho toàn bộ chuỗi và trừ đi .

12.3. Chương trình.

```
#include<bits/stdc++.h>
using namespace std;
const int N = 2e5+5;
double exc[N] = {0}, sum[N] = {0},
rev[N] = {0};
double t[N];
double dp[N][55];
double y(int x, int p){
    return dp[x][p] - exc[x] +
sum[x]*rev[x];
}
double g(int j, int k, int p){
    return (y(j, p)-y(k, p))/(sum[j]-
sum[k]);
}
double cal(int l, int i){
    return exc[i]-exc[l]-(rev[i]-
rev[l])*sum[l];
}
int q[N], top, tail;
void add(int i, int k){
    while(top < tail-1 && g(i, q[tail-
1], k) < g(q[tail-1], q[tail-2], k))
tail--;
    q[tail++] = i;
}
int getmax(int i, int k){
    while(top < tail-1 && g(q[top+1],
q[top], k) < rev[i]) top++;
    return q[top];
}
int main(){
    freopen("673E.INP","r",stdin);
    freopen("673E.OUT","w",stdout);
    int n, K;
    scanf("%d%d", &n, &K);
    for(int i = 1; i <= n; ++i)
scanf("%lf", t+i);
    for(int i = 1; i <= n; ++i){
        sum[i] = sum[i-1] + t[i];
        rev[i] = rev[i-1] + 1.0/t[i];
        exc[i] = exc[i-1] +
sum[i]/t[i];
    }
    for(int i = 1; i <= n; ++i)
dp[i][1] = exc[i];
    for(int k = 2; k <= K; ++k){
        top = tail = 0;
        q[tail++] = 0;
        for(int i = 1; i <= n; ++i)
add(i, k-1);
        for(int i = 1; i <= n; ++i){
            if(i >= k){
                int l = getmax(i, k-
1);
                dp[i][k] = dp[l][k-1]
+ cal(l, i);
            }
            else dp[i][k] = 1e50;
        }
    }
    printf("%.10f\n", dp[n][K]);
}
```

12.4.

12.5. Test.

Đường dẫn:
https://drive.google.com/file/d/0BwQZ8OxBy1tPUS1vZ1FESXI0ZEE/view?usp=sharing

12.6. Cảm nhận

Bài toán này thuộc dạng bài toán 2. Độ phức tạp O(nlogn).

Bài 13: VUN ĐỐNG

13.1. Đề bài

Một công ty khai thác mỏ chiết xuất terbium, một kim loại hiếm được lấy từ cát sông. Trên con sông LONG họ khai thác tại N điểm, khoảng cách của mỗi điểm được xác định từ đầu nguồn con sông tới điểm đó. Tại mỗi điểm khai thác, một lượng quặng tương đối nhỏ nhưng có giá trị cao được chiết xuất từ sông.

Để thu thập quặng, công ty đã gộp N đống nhỏ thành K đống lớn hơn. Các đống mới hình thành được vận chuyển đi bằng xe tải.

Để dồn N đống, họ sử dụng một chiếc thuyền lớn đi từ đầu nguồn đến hạ lưu của con sông. Tức là, các đống sản xuất tại điểm khai thác X có thể được đưa đến điểm khai thác Y chỉ khi Y > X. Mỗi đống được chuyển hoàn toàn đi tới điểm mới hoặc không di chuyển. Chi phí di chuyển một đống có trọng lượng W từ điểm khai thác X đến điểm khai thác Y là . Tổng chi phí cho viêc dồn đống là tổng chi phí của việc di chuyển mỗi đống. Lưu ý, một đống không di chuyển, không ảnh hưởng đến tổng chi phí.

- *Yêu cầu:* Tính tổng chi phí tối thiểu để nhóm N đống ban đầu thành K nhóm.
- *Dữ liệu vào:*
  - Dòng đầu chứa hai số nguyên N và K là số lượng đống ban đầu và số lượng đống sau khi dồn. .
  - Mỗi dòng trong N dòng tiếp theo chứa hai số nguyên X và W là vị trí và trọng lượng của một đống. .
  - Các đống được mô tả theo thứ tự khai thác tăng dần.
- *Dữ liệu ra:*
  - Dòng duy nhất chứa một số nguyên là chi phí tối thiểu tìm được.
- *Ví dụ:*

HEAP.INP	HEAP.OUT
3 1 20 1 30 1 40 1	30
3 1 11 3 12 2 13 1	8
6 2 10 15 12 17 16 18 18 13 30 10 32 1	278
6 3 10 15	86

12 17 16 18 18 13 30 10 32 1	

13.2. Hướng dẫn giải thuật.

- Gọi là chi phí tối thiểu để chuyển một đống từ vị trí i đến vị trí j.
- : là chi phí di chuyển của tất cả các đống từ k+1 tới i.
-
-
- Đặt:
-
- Nên:

Đặt:

Chúng ta có tập đường thẳng y = a*x+b.

Áp dụng thuật toán CHT.

13.3. Chương trình.

```
#include <bits/stdc++.h>

using namespace std;
#define MAXN 1000
int N,K,X[MAXN],W[MAXN];
long long f[MAXN],g[MAXN];
long long dp[2][MAXN];

struct line{
    long long y0;
    int m;

    line(){}
    line(long long _y0, int _m):
        y0(_y0), m(_m){}
};

bool check(line a, line b, line c){
    return (a.y0 - b.y0) * (c.m - a.m)
< (a.y0 - c.y0) * (b.m - a.m);
}

int nh,pos;
line H[MAXN];

void update(line l){
    while(nh >= 2 && !check(H[nh -
2],H[nh - 1],l)){
        if(pos == nh - 1) --pos;
        --nh;
    }
    H[nh++] = l;
}

long long eval(int id, int x){
    return H[id].y0 + (long
long)H[id].m * x;
}

long long query(int x){
    while(pos + 1 < nh && eval(pos,x)
> eval(pos + 1,x)) ++pos;
    return eval(pos,x);
}

int main(){
    freopen("HEAP.INP","r",stdin);
    freopen("HEAP.OUT","w",stdout);
    cin>>N>>K;
          for(int i = 0;i < N;++i)
            cin>>X[i]>>W[i];
        f[0] = W[0];
        g[0] = (long long)X[0] * W[0];
        for(int i = 1;i < N;++i){
            f[i] = f[i - 1] + W[i];
            g[i] = g[i - 1] + (long
long)X[i] * W[i];
        }
        for(int i = 0;i < N;++i)
            dp[1][i] = X[i] * f[i] -
g[i];
        for(int k = 2,r = 0;k <=
K;++k,r ^= 1){
            nh = 0; pos = 0;
```

```
            for(int i = k - 1;i <
N;++i){
                update(line(dp[r ^
1][i - 1] + g[i - 1],-f[i - 1]));
                dp[r][i] = X[i] * f[i]
- g[i] + query(X[i]);
            }
        }
        cout<<dp[K & 1][N - 1];
    return 0;
}
```

15.3.

13.4. Test.

13.5. Cảm nhận

Bài toán là dạng bài toán 2, độ phức tạp là

Bài 14: Painting Tree.

14.1. Đề bài

Nguồn: https://www.codechef.com/problems/KILLER

Mức độ: Khó.

Bạn được cho một cây không có trọng số với N nút, được đánh số từ tới Nút là nút gốc. Bạn muốn tô màu tất cả các đỉnh của cây. Có các chổi màu khác nhau trên mỗi nút. Bạn có thể chọn chổi màu tại một nút và vẽ một nét đơn đến một số nút liên tiếp trên đường đi từ nút này đến nút gốc. Bạn phải chắc chắn mỗi nút vẽ chính xác một lần.

Một đường trên cây là tốt nếu không có hai nút khác nhau trên con đường này có cùng khoảng cách đến nút gốc, một đường tốt là đường đi giữa một nút tới tổ tiên của nó. Đường đi chứa một nút coi là đường đi tốt.

- Phân vùng tất cả các nút thành những đường đi tốt sao cho tổng chi phí là nhỏ nhất. Mỗi nút nằm chính xác trong một đường đi đã chọn.

là số cạnh trên đường đi từ gốc đến . Vì vậy, , là độ sâu lớn nhất của tất cả các nút.

Chi phí của đường đi: Cho là đường đi tốt. là nút xa gốc nhất thì chi phí của con đường từ gốc đến là: với là mọi đỉnh trên .

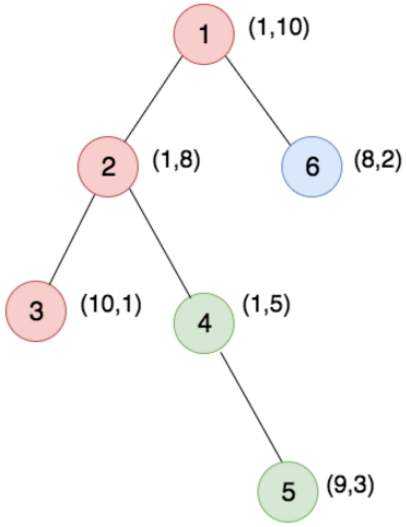
Tổng chi phí =

Tổng chi phí là tổng chi phí của mỗi đường đi đã chọn.

- *Yêu cầu:* Tính tổng chi phí tối thiểu để tô màu tất cả các nút.
- *Dữ liệu vào:*
  - Dòng đầu chứa số nguyên , là số test.
  - Mỗi test gồm:
    - Dòng đầu chứa số nguyên , là số nút trên cây.
    - dòng tiếp theo, mỗi dòng chứa hai số nguyên và .
    - dòng tiếp theo mỗi dòng chứa hai số nguyên và biểu thị có cạnh nối giữa nút và nút .
- *Dữ liệu ra:*
  - Mỗi test in ra trên một dòng một số nguyên là tổng chi phí nhỏ nhất.
- *Ví dụ:*

KILLER.INP	KILLER.OUT
2 3 4 5 2 3 2 2 1 2 1 3 6 1 10 1 8 10 1 1 5 9 3 8 2 1 2 1 6 2 3 2 4 4 5	15 11

- *Giới hạn:*
  -
  -
  -
  -
  - Subtask : số test có .
    - Cây đảm bảo là một đường đi. Nút là một trong những điểm cuối của đường đi. Nút và là kề nhau với mọi .
  - Subtask : số test có .
  - Subtask : số test có .
    - Cây đảm bảo là một đường đi. Nút là một trong những điểm cuối của đường đi. Nút và là kề nhau với mọi .
  - Subtask : số test với điều kiện như giới hạn đề bài.
- *Giải thích*
  - Với test số 2:
    - Kí hiệu giá trị trong ngặc là.
    - Giải pháp tối ưu là vẽ các nút (1, 2, 3) bằng chổi vẽ 3; các nút (4, 5) bằng chổi vẽ 5; nút 6 bằng chổi vẽ 6.



Vì vậy .

14.2. Hướng dẫn giải thuật.

Gọi là chi phí của đường từ tới trong đó là tổ tiên của trong cây ban đầu:

Cho : là khoảng cách của từ tới mức có nút sâu nhất.

Suy ra:

Trong đó là tổng của n số tự nhiên đầu tiên.
Ta có:

Ta đặt:

Thì ta có phương trình :

Vậy là một parabol trong đó các hệ số chỉ phụ thuộc vào .
Ứng dụng Convex Hull Trick (CHT) cho parabol:

- Thấy rằng chỉ phụ thuộc vào . Vì vậy có thể áp dụng CHT cho hàm chi phí trên.

Trong đó từ đó ta chứng minh được:

- là tăng dần .

- và có nhiều nhất một giao điểm .

- Từ những nhận xét trên ta thấy rằng các parabol của hàm chi phí hoạt động giống một đường thẳng nên ta hoàn toàn có thể áp dụng CHT trên các parabol. Lưu ý, các điểm trong các truy vấn CHT luôn không âm .

14.3. Chương trình.

A

```
#include <bits/stdc++.h>

using namespace std;

#define NN 100500

typedef vector<int> vi;
typedef vector<vi> vvi;
typedef pair<int, int> ii;
typedef vector<ii> vii;
typedef vector<string> vs;
typedef double D;
typedef long double LD;
typedef long long ll;
typedef pair<ll, ll> pll;
typedef vector<ll> vll;

template<class T> T abs(T x) { return
x > 0 ? x : -x;}

int n;

ll H[NN];
ll C[NN];

vector<int> v[NN];
int p[NN];

ll val1[NN];
ll val2[NN];
ll h[NN];

ll big = (ll)1e18;

int gh;

void go0(int x, int pr, int ch) {
      p[x] = pr;
      h[x] = ch;
      for (int pp=0; pp < v[x].size();
pp++) {
        int y=v[x][pp];
            if (y == pr)
                  continue;
            go0(y, x, ch + 1);
      }
}

ll gans;

ll cur1[NN];
ll cur2[NN];
ll k[NN];

const int PRO = 500;
vector<pair<ll, int> > bv[NN];

inline ll get(int x) {
      return cur1[x] * C[x] + k[x] *
C[x] * C[x] - H[x] + cur2[x];
}

void go2(int x) {
      bv[x].clear();

      cur1[x] = gh - h[x];
      cur2[x] = val2[x];
      k[x] = 1;

      ll opa = get(x);
      bv[x].push_back({opa, x});
      gans = min(gans, get(x));

      for (int pp=0; pp < v[x].size();
pp++) {
        int z=v[x][pp];
            int bu = NN;
            for (int kk=0; kk <
bv[z].size(); kk++) {
                  pair<ll, int>
o=bv[z][kk];
                  int y = o.second;

                  if (C[y] >= bu)
                        continue;

                  k[y]++;
                  cur1[y] += gh -
h[x];
                  cur2[y] += val2[x] -
val1[z];

                  ll opa2 = get(y);
                  gans = min(gans,
opa2);

      bv[x].push_back({opa2, y});

                  bu = C[y];
            }
      }

      sort(bv[x].begin(),
bv[x].end());
      if (bv[x].size() > PRO)
            bv[x].resize(PRO);
}

void go1(int x) {
      ll sum = 0;

      for (int pp=0; pp < v[x].size();
pp++) {
        int y=v[x][pp];
```

```
            go1(y);
            sum += val1[y];
      }

      val2[x] = sum;

      gans = big;
      go2(x);

      val1[x] = gans;
}

void solve() {
    cin>>n;
      for (int i=0; i<n; ++i) {
            cin>>H[i]>>C[i];
            v[i].clear();
      }
      for (int i=0; i<n-1; ++i) {
            int a, b;
            cin>>a>>b;
            a--;
            b--;
            v[a].push_back(b);
            v[b].push_back(a);
      }
      go0(0, -1, 0);
      gh = *max_element(h, h + n);
      for (int i=0; i<n; ++i) {
    for (int j=0; j<v[i].size(); ++j)
{
                  if (v[i][j] == p[i])
{
                        swap(v[i][j],
v[i][v[i].size() - 1]);

      v[i].pop_back();
                        break;
                  }
            }
      }
      go1(0);
      cout << val1[0] << endl;
}
int main() {
    freopen("killer.inp","r",stdin);
    freopen("killer.out","w",stdout);
      int tc;
  cin>>tc;
  while(tc--) {
            solve();
      }
}
```

14.4.

14.5. Test.

Đường dẫn:

14.6. Cảm nhận

Bài toán ứng dụng kĩ thuật bao lồi với trường hợp tập các đường thẳng là các parabol. Độ phức tạp là:

Bài 15: Đa giác

15.1. Đề bài

Nguồn: https://www.codechef.com/COOK68/problems/KTHCON

Cho một đa giác là một đa giác thường (các cạnh của nó không giao hay chạm vào nhau) có ít nhất góc lõm.

Có điểm trên mặt phẳng. Coi là diện tích lớn nhất của một đa giác với các đỉnh thuộc những điểm đã cho. Chú ý rằng nếu không tồn tại đa giác nào, in ra -1.

- *Yêu cầu:* Tính .
- *Dữ liệu vào:*
  - Dòng đầu tiên chứa một số nguyên – số lượng test
  - Dòng đầu tiên của mỗi test chứa – số lượng điểm
  - Theo sau là dòng chứa hai số nguyên và – tọa độ của một điểm.
- *Dữ liệu ra:*
  - Nếu không có đa giác , in ra . Ngược lại, in ra một số nguyên: hai lần diện tích đa giác lõm lớn nhất (nó đảm bảo là một số nguyên với các ràng buộc đã cho)
- *Ví dụ:*

KTHCON.INP	KTHCON.OUT
2 5 2 2 -2 -2 2 -2 -2 2 0 1 3 0 0 1 0 0 1	28 -1

- *Giới hạn:*
  -
  -
  - Không có hai điểm trùng nhau (có cả tọa độ và giống hệt nhau).
  - Không có ba điểm thẳng hàng.
  - Tổng của trong tất cả các test không vượt quá
  - .

15.2. Hướng dẫn giải thuật.

Nếu tất cả các điểm trên bao lồi thì in ra. Ngược lại xây dựng bao lồi cho tất cả các điểm. Ngoài ra, tìm một bao lồi bên trong của tất cả điểm nằm bên trong bao lồi đã xây dựng.

Tìm tam giác có diện tích nhỏ nhất với hai đỉnh thuộc cạnh đa giác lồi bên ngoài và đỉnh khác nằm bên trong đa giác lồi bên ngoài (chính xác là trên bao lồi bên trong).

Lặp với cạnh của bao lồi bên ngoài một lần theo chiều kim đồng hồ

15.3. Chương trình.

```
#include<bits/stdc++.h>                    using namespace std;
```

```
#define clr(mark)
memset(mark,0,sizeof(mark))
#define F first
#define S second
#define MP make_pair
#define PB push_back
#define ll long long
#define INF 5e18
typedef long double T;
const T EPS = 1e-7;
struct PT {
  T x, y;
  PT() {}
  PT(T x, T y) : x(x), y(y) {}
  bool operator<(const PT &rhs) const {
return make_pair(y,x) <
make_pair(rhs.y,rhs.x); }
  bool operator==(const PT &rhs) const
{ return make_pair(y,x) ==
make_pair(rhs.y,rhs.x); }
};

T cross(PT p, PT q) { return p.x*q.y-
p.y*q.x; }
T area2(PT a, PT b, PT c) { return
cross(a,b) + cross(b,c) + cross(c,a); }


void ConvexHull(vector<PT> &pts) {
  sort(pts.begin(), pts.end());
  pts.erase(unique(pts.begin(),
pts.end()), pts.end());
  vector<PT> up, dn;
  for (int i = 0; i < pts.size(); i++)
{
    while (up.size() > 1 &&
area2(up[up.size()-2], up.back(),
pts[i]) >= 0) up.pop_back();
    while (dn.size() > 1 &&
area2(dn[dn.size()-2], dn.back(),
pts[i]) <= 0) dn.pop_back();
    up.push_back(pts[i]);
    dn.push_back(pts[i]);
  }
  pts = dn;
  for (int i = (int) up.size() - 2; i
>= 1; i--) pts.push_back(up[i]);

}

int pointer[2]; //L u tr  đ  ng t t
nh t t  query tr  c.
vector<long long> M[2]; //H  s  góc c a
đ  ng trong bao l i
vector<long long> B[2]; //Giá tr  B c a
đ  ng th ng Mx+B.
//Tr  l i True n u đ  ng th ng l1 ho c
l3 luôn t t h n l2.
bool bad(int l1,int l2,int l3,bool f)
{
      /*
      hoành đ  giao đi m c a l1 và l2
là: (b1-b2)/(m2-m1)
      hoành đ  giao đi m c a l1 và l3
là: (b1-b3)/(m3-m1)
      */
      return (B[f][l3]-
B[f][l1])*(M[f][l1]-
M[f][l2])<(B[f][l2]-
B[f][l1])*(M[f][l1]-M[f][l3]);
}
//Thêm m t đ  ng m i v i h  s  góc nh
h n vào bao l i.
void add(long long m,long long b,bool
f)
{
      //Đ u tiên, thêm nó vào cu i.
      M[f].push_back(m);
      B[f].push_back(b);
      while
(M[f].size()>=3&&bad(M[f].size()-
3,M[f].size()-2,M[f].size()-1,f))
      {
            M[f].erase(M[f].end()-2);
            B[f].erase(B[f].end()-2);
      }
}
//Tr  v  t a đ  y nh  nh t trong các
giao đi m c a đ  ng th ng đ ng v i cách
canh c a bao l i
long double Query(long double x,bool f)
{
      //N u ta lo i b  đ  ng th ng t t
nh t cho truy v n tr  c
      //thì đ  ng th ng v a chèn bây
gi  là t t nh t cho truy v n đó.
      if (pointer[f]>=M[f].size())
            pointer[f]=M[f].size()-1;
//B t kỳ đ  ng th ng nào t t h n ph i
n m bên ph i vì các giá tr  truy v n
không gi m.
      while (pointer[f]<M[f].size()-1&&

M[f][pointer[f]+1]*x+B[f][pointer[f]+1]
<M[f][pointer[f]]*x+B[f][pointer[f]])
            pointer[f]++;
      return
M[f][pointer[f]]*x+B[f][pointer[f]];
}

long double ComputeSignedArea(const
vector<PT> &p) {
  long double area = 0;
  for(int i = 0; i < p.size(); i++) {
    int j = (i+1) % p.size();
    area += p[i].x*p[j].y -
p[j].x*p[i].y;
  }
  return area / 2.0;
}

long double ComputeArea(const
vector<PT> &p) {
  return fabs(ComputeSignedArea(p));
}
long double square(ll X)
{
      return X*X;
}
long double dist(ll X1,ll Y1,ll X2,ll
Y2)
{
```

```
      return sqrt(square(X2-
X1)+square(Y2-Y1));
}
ll x[100010],y[100010];
ll X[100010],Y[100010];
pair<ll,ll> arr[100010];
ll a[100010],b[100010],c[100010];
set<pair<ll,ll> > sett;
pair<long double,int> query[100010];
long double ans[100010];
int main()
{
      int t;
      int n,i;
      freopen("KTHCON.INP","r",stdin);
      freopen("KTHCON.OUT","w",stdout);
     cin>>t;
      while(t--)
      {
            M[0].clear();
            M[1].clear();
            B[0].clear();
            B[1].clear();
            pointer[0]=pointer[1]=0;
        cin>>n;
            vector<PT> v;
            v.clear();
            for(i=0;i<n;++i)
            {
                  cin>>X[i];
                  cin>>Y[i];
                  arr[i]=MP(X[i],Y[i]);
                  PT pt(X[i],Y[i]);
                  v.PB(pt);
            }
            ConvexHull(v);
            if(v.size()==n)
            {
                   cout<<-1<<endl;
                  continue;
            }
            long double
area=ComputeArea(v);
            int sz=(int)v.size();
            sett.clear();
            for(i=0;i<sz;++i)
            {
                  if(v[i].x<0)
                        x[i]=v[i].x-
0.5;
                  else

      x[i]=v[i].x+0.5;

                  if(v[i].y<0)
                        y[i]=v[i].y-
0.5;
                  else

      y[i]=v[i].y+0.5;

                  //cout<<x[i]<<'
'<<y[i]<<'\n';

      sett.insert(MP(x[i],y[i]));
            }
            for(i=0;i<sz;++i)
            {
                  a[i]=y[(i+1)%sz]-
y[i];
                  b[i]=x[i]-
x[(i+1)%sz];

      c[i]=y[i]*x[(i+1)%sz]-
y[(i+1)%sz]*x[i];


if(a[i]*x[(i+2)%sz]+b[i]*y[(i+2)%sz]+c[
i]<0)
                  {
                        a[i]=-a[i];
                        b[i]=-b[i];
                        c[i]=-c[i];
                  }
            }
            sort(arr,arr+n);
            reverse(arr,arr+n);
            ll minx=INF,maxx=-INF;
            for(i=0;i<n;++i)
            {

      if(sett.find(arr[i])==sett.end())
                  {

      add(arr[i].F,arr[i].S,0);

      minx=min(minx,arr[i].F);

      maxx=max(maxx,arr[i].F);
                  }
            }
            for(i=n-1;i>=0;--i)
            {

      if(sett.find(arr[i])==sett.end())
                  {
                        add(-arr[i].F,-
arr[i].S,1);
                  }
            }

            for(i=0;i<sz;++i)
            {
                  query[i].S=i;
                  if(b[i]==0)
                  {
                        query[i].F=INF;
                        if(a[i]<0)

      ans[i]=a[i]*maxx+c[i];
                        else

      ans[i]=a[i]*minx+c[i];
                  }
                  else

      query[i].F=((long
double)a[i])/((long double)b[i]);
            }
            sort(query,query+sz);
            for(i=0;i<sz;++i)
            {
                  if(query[i].F+1>=INF)
                        continue;
```

```
                long double val;
                if(b[query[i].S]>0)

      val=Query(query[i].F,0);
                else
                      val=-
Query(query[i].F,1);


val=val*b[query[i].S]+c[query[i].S];
                ans[query[i].S]=val;
            }
            long double min_area=INF;
            for(i=0;i<sz;++i)
            {
                  long double
area1=ans[i]/sqrt(a[i]*a[i]+b[i]*b[i]);


area1=area1*dist(x[i],y[i],x[(i+1)%sz],
y[(i+1)%sz])*.5;

      min_area=min(min_area,area1);
            }
            area-=min_area;
            area=area*2;
            ll out=area+0.5;
            cout<<out<<endl;
      }
}
```

15.4.

15.5. Test.

Đường dẫn:

15.6. Cảm nhận

Sử dụng CHT để tìm bao lồi và tính diện tích của bao lồi đó. Độ phức tạp là:

PHẦN III: KẾT LUẬN

Trong các bài toán ứng dụng kĩ thuật bao lồi đã trình bày ở phần trên các thao tác chính cần làm:

- Sắp xếp các đường thẳng theo hệ số góc.
- Loại bỏ đường thẳng dư thừa.
- Tìm bao lồi của tập các đường thẳng
- Thực hiện các truy vấn trên tập bao lồi vừa tìm được.

Ưu điểm của CHT:

- Khi cài đặt không phụ thuộc vào kích thước hoành độ x, x không bắt buộc phải là số nguyên.
- Bộ nhớ và thời gian nhỏ.

Nhược điểm của CHT:

- Chỉ ứng dụng được với đường thẳng chứ không ứng dụng được với đoạn thẳng. Nếu đường thẳng ax+b chỉ tồn tại khi x thuộc một khoảng (l, h) bao lồi sẽ không làm được.
- Thao tác thêm đường thẳng phức tạp nếu các đường thẳng không được sắp xếp theo hệ số góc.
- Hơi khó code.

Từ những ưu nhược điểm trên tôi nhận thấy CHT là kĩ thuật tốt để cải tiến các bài toán quy hoạch động. Và ngày càng có nhiều ứng dụng của kĩ thuật này.

Do thời gian hạn chế và kiến thức còn chưa được sâu, rộng nên chắc chắn chuyên đề còn nhiều thiếu sót. Tôi rất mong quý thầy cô đồng nghiệp đóng góp ý kiến để chuyên đề hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

1. http://wcipeg.com/wiki/Convex_hull_trick
2. http://www.giaithuatlaptrinh.com
3. https://www.codechef.com
4. http://codeforces.com

MỤC LỤC

PHẦN I: MỞ ĐẦU 1
PHẦN II: NỘI DUNG 2
I. KĨ THUẬT BAO LỒI (CONVEX HULL TRICK) 2
1. Bài toán 1 2
1.1. Phân tích 2
1.2. Cài đặt: 4
2. Bài toán 2 6
3. Bài toán 3. 6
4. Bài toán 4 7
II. BÀI TẬP ỨNG DỤNG. 8
Bài 1: CẮT CÂY. 8
1.1. Đề bài 8
1.2. Hướng dẫn giải thuật: 9
1.3. Chương trình 10
1.4. Test 11
1.5. Cảm nhận 11
Bài 2: ACQUIRE. 11
2.1. Đề bài 11
2.2. Hướng dẫn giải thuật 12
2.5. Cảm nhận 13
Bài 3: COMMANDO. 13
3.1. Đề bài: 13
3.2. Hướng dẫn giải thuật. 14
3.3. Chương trình. 15
3.4. Test. 16
3.5. Cảm nhận 16
Bài 4: Xây dựng nhà. 16
4.1. Đề bài 16
4.2. Hướng dẫn giải thuật. 17
4.3. Chương trình. 17
4.4. Test. 18
4.5. Cảm nhận. 18
Bài 5: Vô hiệu cài đặt. 18
5.1. Đề bài 18
5.2. Hướng dẫn giải thuật. 19
5.3. Chương trình. 19

5.4. Test. 20
5.5. Cảm nhận 20
Bài 6: CYCLRACE – đua xe đạp. 20
6.1. Đề bài 20
6.2. Hướng dẫn giải thuật. 21
6.3. Chương trình. 21
6.4. Test 22
6.5. Cảm nhận 22
Bài 7: JUMP. 22
7.1. Đề bài: 22
7.2. Hướng dẫn giải thuật: 23
7.3. Chương trình. 23
7.4. Test: 24
7.5. Cảm nhận. 24
Bài 8: Function. 24
8.1. Đề bài 24
8.2 Hướng dẫn giải thuật. 25
8.3 Chương trình. 25
8.4. Test. 26
8.5. Cảm nhận 26
Bài 9: Product Sum 26
9.1. Đề bài 26
9.2. Hướng dẫn giải thuật. 27
9.3. Chương trình. 27
9.4. Test. 28
9.5. Cảm nhận 28
Bài 10: Xây dựng nhà. 28
10.1. Đề bài 28
10.2. Hướng dẫn giải thuật. 29
10.3. Chương trình. 29
10.4. Test. 30
10.5. Cảm nhận 30
Bài 11: Bowling. 30
11.1. Đề bài 30
11.2. Hướng dẫn giải thuật. 31
11.3. Chương trình. 31

11.4. Test. 31
11.5. Cảm nhận. 32
Bài 12: Game. 32
12.1. Đề bài 32
12.2. Hướng dẫn giải thuật. 33
12.3. Chương trình. 33
12.4. Test. 33
12.5. Cảm nhận 33
Bài 13: VUN ĐỐNG 33
13.1. Đề bài 34
13.2. Hướng dẫn giải thuật. 34
13.3. Chương trình. 35
13.4. Test. 35
13.5. Cảm nhận 36
Bài 14: Painting Tree. 36
14.1. Đề bài 36
14.2. Hướng dẫn giải thuật. 38
14.3. Chương trình. 38
14.4. Test. 40
14.5. Cảm nhận 40
Bài 15: Đa giác 40
15.1. Đề bài 40
15.2. Hướng dẫn giải thuật. 41
15.3. Chương trình. 41
15.4. Test. 43
15.5. Cảm nhận 43
PHẦN III: KẾT LUẬN 44
TÀI LIỆU THAM KHẢO 44
MỤC LỤC 45