

MỤC LỤC

VR50. NGÀY SINH NHẬT	<i>Tên chương trình: BIRTHDAY.CPP</i>	2
VS04. ĐỘ TƯƠNG ĐỒNG	<i>Tên chương trình: SIMILARITY.CPP</i>	5
VS05. CON ĐƯỜNG GÓM SỨ	<i>Tên chương trình: CERAMIC.CPP</i>	9
VS07. ĐIỀU CHỈNH LƯƠNG	<i>Tên chương trình: SALARY.CPP</i>	12
VS02. SỐ NGUYÊN MỚI	<i>Tên chương trình: NEWNUM.CPP</i>	18
VS09. ẢNH HOA	<i>Tên chương trình: FLOWERS.CPP</i>	21
VS08. CASIO	<i>Tên chương trình: CASIO.CPP</i>	24
VS10. LẬP TRÌNH NHANH	<i>Tên chương trình: FASTPROG.CPP</i>	26
VS11. KẾT CẤU KÉP	<i>Tên chương trình: DUALSTRUCT.CPP</i>	29
VS12. MÃ KIỂM TRA	<i>Tên chương trình: NUMBERS.CPP</i>	32
VS13. QUAN SÁT	<i>Tên chương trình: OBSERVE.CPP</i>	35
VS14. QUÀ TẶNG	<i>Tên chương trình: GIFTS.CPP</i>	42
VS15. CHUNG KẾT GIẢI ĐÁ CẦU	<i>Tên chương trình: FINAL.CPP</i>	45
VS16. TRẠM THU PHÍ	<i>Tên chương trình: STATIONS.CPP</i>	48
VS17. TIẾNG ANH CỎ	<i>Tên chương trình: ENGLISH.CPP</i>	52
VS19. SỐNG SỐT	<i>Tên chương trình: SURVIVE.CPP</i>	55
VS20. CƠ SỞ	<i>Tên chương trình: BASIC.CPP</i>	60
VS21. ĐÀN ORGAN	<i>Tên chương trình: ORGAN.CPP</i>	63
VS22. DUNG KÉ	<i>Tên chương trình: MEASURE.CPP</i>	66
VS23. SỬA CHỮA	<i>Tên chương trình: REPAIR.CPP</i>	69
VS24. MÔ HÌNH MÁY BAY	<i>Tên chương trình: AIRPLANS.CPP</i>	72
VS25. SỐ LỚN NHẤT	<i>Tên chương trình: MAXNUMBER.CPP</i>	75
VS26. PHẦN THƯỞNG	<i>Tên chương trình: PRIZES.CPP</i>	79
VS27. HÀNH TINH ĐỎ	<i>Tên chương trình: MARS.CPP</i>	82
VS28. XÂU ĐỌC ĐÁO	<i>Tên chương trình: STRANGE.CPP</i>	85
VS29. DU LỊCH BẰNG TÀU HỎA	<i>Tên chương trình: TRAINS.CPP</i>	89
VS30. KHU BẢO TỒN	<i>Tên chương trình: FOREST.CPP</i>	97
VS31. CỜ VÂY	<i>Tên chương trình: DRAUGHTS.CPP</i>	100
VS32. SỐ DỄ CHỊU	<i>Tên chương trình: PLEASANT.CPP</i>	104
VS33. CÓ HẬU	<i>Tên chương trình: GOODNUM.CPP</i>	108
VS34. MỞ KHÓA	<i>Tên chương trình: LOCKER.CPP</i>	110
VS35. HỎI – ĐÁP	<i>Tên chương trình: ANSWER.CPP</i>	113

VR50. NGÀY SINH NHẬT

Tên chương trình: BIRTHDAY.CPP

Alice chuyển đến trường mới. Tôm và Bob đều rất muốn làm quen với Alice và hỏi ngày sinh nhật của người bạn mới. Alice lần lượt rỉ tai Tôm và Bob rồi tinh nghịch nói: “Mình đã nói cho mỗi bạn một phần sự thật, Tôm biết được ngày sinh của mình, còn Bob – biết tháng sinh. Các cậu phải giữ bí mật như đã hứa. Chiều nay lên facebook ta sẽ trao đổi thêm”.

Buổi chiều Alice đưa lên trang facebook của mình danh sách n ngày tháng và cho biết một trong số các ngày đã nêu là ngày sinh nhật của mình.

Dĩ nhiên các bạn Tôm và Bob không bỏ sót thông tin nào trên trang facebook của Alice và sau đây là nội dung thông tin trao đổi giữa 2 người.

Tôm: - *Mình không xác định được ngày sinh nhật của Alice và tin rằng cậu cũng không thể xác định được.*

Bob: - *Ban đâu mình cũng không xác định được ngày sinh nhật của Alice, nhưng sau khi nghe cậu nói như vậy bây giờ mình đã biết chính xác ngày sinh nhật của Alice!*

Tôm: - *Nếu vậy thì bây giờ mình cũng biết được rồi!*

Hãy đưa ra ngày sinh nhật (ngày và tháng) của Alice.

Dữ liệu: Vào từ file văn bản BIRTHDAY.INP:

- ➡ Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 366$),
- ➡ Dòng thứ i trong n dòng tiếp theo chứa 2 số nguyên d_i và m_i xác định một ngày trong năm ($1 \leq d_i \leq 31$, $1 \leq m_i \leq 12$). Các ngày tháng được đưa theo thứ tự tăng dần trong năm.

Dữ liệu đảm bảo tồn tại nghiêm theo tinh thần đổi thoại đã nêu.

Kết quả: Đưa ra file văn bản BIRTHDAY.OUT 2 số nguyên – ngày sinh nhật của Alice.

Ví dụ:

BIRTHDAY.INP
11
29 2
5 5
16 5
31 5
17 6
18 6
14 7
16 7
5 12
14 12
17 12

BIRTHDAY.OUT
17 6



Giải thuật: Mô phỏng hoạt động ô tô mát.

Nhận xét:

- ⊕ Theo điều kiện đầu bài tồn tại 3 thao tác lọc dữ liệu:
 - ⊖ Thao tác I : Loại bỏ các dữ liệu đơn trị theo ngày (**d**) và theo tháng (**m**),
 - ⊖ Thao tác II : Lọc giữ lại các dữ liệu đơn trị theo **m**,
 - ⊖ Thao tác III: Lọc giữ lại dữ liệu đơn trị theo **d**,
- ⊕ Số lượng dữ liệu cần xử lý nhỏ (không quá 366) vì vậy việc loại bỏ có thể thực hiện bằng cách đánh dấu.

Tổ chức dữ liệu:

- ❖ Các mảng **int a[367], b[367]**: lưu trữ dữ liệu vào,
- ❖ Mảng **int f[367]** – đánh dấu xóa dữ liệu,
- ❖ Các mảng **int m[13], d[32]** – lưu tần số xuất hiện tháng và ngày,
- ❖ Mảng **int fd[32]** – đánh dấu ngày bị loại bỏ.

Xử lý:

- Nhập dữ liệu và thống kê tần số xuất hiện ngày và tháng,
- Đánh dấu loại bỏ các dữ liệu có tần số ngày hoặc tháng bằng 1,
- Chỉnh lý tần số tháng và bảng đánh dấu loại bỏ dựa theo các dữ liệu đã đánh dấu loại bỏ hoặc các ngày đã bị loại,
- Thống kê lại tần số xuất hiện của ngày trong các dữ liệu chưa bị loại bỏ,
- Tìm và đưa ra dữ liệu có tần số xuất hiện ngày và tháng bằng 1.

Dộ phức tạp của giải thuật: O(n).

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "birthday."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,d[32]={0},m[13]={0},f[367]={0},t1,t2,fd[32]={0};
int a[367],b[367];

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=0;i<n;++i)
 {
     fi>>t1>>t2; a[i]=t1; b[i]=t2;
     ++d[t1];++m[t2];
 }
 for(int i=0;i<n;++i)if(d[a[i]]==1 || m[b[i]]==1)f[i]=1,fd[a[i]]=1;
 for(int i=0;i<n;++i)if(f[i]==1 || fd[a[i]]==1)--m[b[i]],f[i]=1;
 for(int i=1;i<=31;++i)d[i]=0;
 for(int i=0;i<n;++i)if(f[i]==0 && m[b[i]]==1)++d[a[i]];
 for(int i=0;i<n;++i)
     if(f[i]==0 && m[b[i]]==1&& d[a[i]]==1){t1=a[i];t2=b[i];break;}
 fo<<t1<<' '<<t2;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS04. ĐỘ TƯƠNG ĐỒNG

Tên chương trình: SIMILARITY.CPP

Trong quá trình tìm kiếm các hành tinh có khả năng tồn tại sự sống người ta phải xác định các tham số đặc trưng cho hành tinh.

Với 2 hành tinh mới phát hiện người ta xác định được bộ giá trị đặc trưng cho mỗi hành tinh là $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$ và $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m)$.

Mức độ tương đồng của 2 hành tinh là đoạn k lớn nhất các phần tử liên tiếp nhau trong mỗi dãy trùng nhau với độ chính xác hoán vị. Nói một cách khác độ giống nhau là độ dài lớn nhất của đoạn các phần tử liên tiếp nhau trong \mathbf{A} mà bằng cách đổi chỗ các phần tử trong đó ta được đoạn các phần tử liên tiếp trong \mathbf{B} .

Hãy xác định k và vị trí đầu của dãy con tìm được trong \mathbf{A} và \mathbf{B} .

Dữ liệu: Vào từ file văn bản SIMILARITY.INP:

- ✚ Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 1\,000$),
- ✚ Dòng thứ 2 chứa n số nguyên $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ ($1 \leq \mathbf{a}_i \leq 10^5$, $i = 1 \div n$),
- ✚ Dòng thứ 3 số nguyên m ($1 \leq m \leq 1\,000$),
- ✚ Dòng thứ 4 chứa m số nguyên $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m$ ($1 \leq \mathbf{b}_j \leq 10^5$, $j = 1 \div m$).

Kết quả: Đưa ra file văn bản SIMILARITY.OUT trên một dòng 3 số nguyên \mathbf{k}, \mathbf{p} và \mathbf{q} , trong đó \mathbf{k} là độ dài lớn nhất của dãy con tìm được, \mathbf{p} – vị trí đầu của dãy con trong \mathbf{A} , \mathbf{q} – vị trí đầu của dãy con trong \mathbf{B} . Nếu không tồn tại dãy con giống nhau thì đưa ra kết quả 0 và cặp giá trị -1, -1.

Ví dụ:

SIMILARITY.INP
3
1 2 3
3
2 1 3

SIMILARITY.OUT
3 1 1



Giải thuật: Úng dụng tổng tiền tố và hàm băm.

Nhận xét:

- + Cần thiết phải so sánh hai đoạn các phần tử liên tiếp nhau, vì vậy trong phần lớn các trường hợp – việc sử dụng tổng tiền tố sẽ giảm đáng kể độ phức tạp của giải thuật,
- + Trình tự các phần tử trong đoạn so sánh không đóng vai trò quan trọng, điều cho phép xây dựng giải thuật làm việc với bảng tần số giá trị các phần tử trong đoạn,
- + Sự khác biệt các bảng tần số dễ dàng nhận biết bằng kỹ thuật hàm băm.

Tổ chức dữ liệu:

- + Các mảng:
 - int64_t a[1001], b[1001] – lưu tổng tiền tố nhận được từ ánh xạ 2 mảng tần số xuất hiện của dữ liệu ban đầu sang tập số nguyên bằng kỹ thuật hàm băm,
 - int64_t h[100001] – mảng giá trị phục vụ tính hàm băm, $h_i = hb^i$, $i=0 \div 10^5$,
- + Tập set<int64_t> s – lưu các giá trị băm đã xét.

Các bước xử lý:

- + Tính giá trị hệ số h_i của hàm băm,
- + Ánh xạ các dữ liệu ban đầu sang tổng tiền tố các giá trị băm:

```
fi>>n; a[0]=0;
for(int i=1;i<=n;++i){fi>>t;a[i]=a[i-1]+h[t];}
fi>>m; b[0]=0;
for(int i=1;i<=m;++i){fi>>t;b[i]=b[i-1]+h[t];}
```

- + Xây dựng hàm kiểm tra đoạn khớp nhau dài nhất:

```
void check_ab(const int64_t*x,const int64_t*y, int nx,int ny)
```

Cách truyền tham số mảng
theo địa chỉ

- + Hoạt động của hàm **check_ab**:

- Để giảm độ phức tạp của chương trình và độ phức tạp của giải thuật cần gọi hàm với kích thước mảng **x** nhỏ hơn hoặc bằng kích thước mảng **y** ($\mathbf{nx} \leq \mathbf{ny}$),
- Lần lượt kiểm tra các đoạn có độ dài **k = nm** tới 1, trong đó $\mathbf{nm} = \min\{\mathbf{n}, \mathbf{m}\}$,
- Với mỗi **k**:
- Xóa tập **s**,
- Nạp các giá trị băm tương ứng với đoạn độ dài **k** các phần tử liên tiếp nhau của **x** vào **s**,
- Xác định giá trị băm tương ứng với đoạn độ dài **k** các phần tử liên tiếp nhau của **y** và kiểm tra xem có trong **s** hay không, nếu có – ghi nhận điểm đầu của đoạn và thoát khỏi chu trình kiểm tra,
- Nếu tìm thấy điểm đầu trong **y** – duyệt và tìm điểm đầu trong **x**,
- + Tô chúc đưa ra kết quả theo đúng yêu cầu của đề bài.

Độ phức tạp của giải thuật: O(nlogn) (do sử dụng tập hợp).

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "similarity."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int hb=3107;
int64_t a[1002],b[1002],h[100001],t;
int n,m,k,p,q,c,nm;
set<int64_t>s;

void check_ab(const int64_t*x,const int64_t*y, int nx,int ny)
{int64_t t,tp;
p=-1;q=-1;k=0;
for(int i=nm;i>0;--i)
{
    s.clear();
    for(int j=1;j<=nx-i+1;++j)
    {
        t=x[j+i-1]-x[j-1];s.insert(t);
    }
    for(int j=1;j<=ny-i+1;++j)
    {
        t=y[j+i-1]-y[j-1];
        if(s.find(t)!=s.end()) {p=j;tp=t; k=i; break;;}
    }
    if(p>0)break;
}
if(p>0)
{
    for(int i=1;i<=nx-k+1;++i)
    {
        t=x[i+k-1]-x[i-1];
        if(tp==t) {q=i;break;}
    }
}
}

int main()
{clock_t aa=clock();
h[0]=1; for(int i=1;i<=100000;++i)h[i]=h[i-1]*hb;
fi>>n; a[0]=0;
for(int i=1;i<=n;++i){fi>>t;a[i]=a[i-1]+h[t];}
fi>>m; b[0]=0;
for(int i=1;i<=m;++i){fi>>t;b[i]=b[i-1]+h[t];}
nm=min(n,m);if(n<=m){c=1;check_ab(a,b,n,m);}
else {c=2;check_ab(b,a,m,n);}
if(k>0){if(c==1){t=p;p=q;q=t;}}
fo<<k<<'\n'; fo<<p<<' ' <<q;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS05. CON ĐƯỜNG GÓM SỨ

Tên chương trình: CERAMIC.CPP

Sau khi bê tông hóa đê chống lụt, thành phố quyết định cho khảm lên tường bê tông của đê tranh ghép tạo bởi các mảnh gốm sứ lấy từ các lò gốm nổi tiếng trong nước. Toàn bộ con đê được chia thành n phần có độ rộng giống nhau, mỗi phần gọi là một lô. Mỗi bức tranh khảm trên đó đều phải có độ rộng giống nhau, tức là bao gồm một số như nhau các lô liên tiếp và toàn bộ tường phải được phủ kín tranh từ đầu đến cuối, mỗi lô phải được tạo màu chủ đạo (gọi là màu của lô) từ một loại gốm đặc trưng lấy từ một lò gốm nào đó trong nước, ví dụ gốm màu xanh Cô ban từ lò gốm Ánh Hồng Quảng Ninh, gốm da lươn – từ Bát Tràng Hà Nội, gốm mộc hồng nhạt – từ Biên Hòa Đồng Nai, . . . Các loại gốm này được đánh số từ 1 đến 50 000.



Hướng dẫn viên du lịch giới thiệu với khách tham quan là có 2 nhóm nghệ nhân được giao việc tạo hình và khảm tranh. Với mỗi nhóm các bức tranh của đều được đặc trưng bởi dãy số (c_1, c_2, \dots, c_k), trong đó k là độ rộng của tranh, c_i – màu của lô, $i = 1 \div k$, các bức tranh khác nhau có thể khác nhau ở trình tự xuất hiện màu của các lô, ví dụ với dãy số đặc trưng (2, 6, 2, 9), trình tự màu trong tranh có thể là (9, 2, 2, 6) hoặc (6, 9, 2, 2) nhưng không thể là (6, 9, 2, 3). Dãy đặc trưng của 2 nhóm là khác nhau, tức là không thể bằng phép hoán vị trình tự màu của lô để đưa một dãy về dãy kia. Các bức tranh được ghép với nhau rất hài hòa và khách tham quan không nhận biết được sự chuyển tiếp từ tranh này sang tranh khác. Tuy vậy nhiều khách tham quan vẫn muốn biết có bao nhiêu bức tranh đã tạo ra và trong đó số bức tranh của mỗi nhóm là bao nhiêu.

Hãy xác định số lượng tranh có thể có và số lượng tranh mỗi nhóm đã làm. biết rằng nhóm nào cũng có tranh của mình.

Dữ liệu: Vào từ file văn bản CERAMIC.INP:

- ✚ Dòng đầu tiên chứa một số nguyên n – số lượng lô của con đê ($2 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n – màu của các lô ($1 \leq a_i \leq 50\,000$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản CERAMIC.OUT, dòng đầu tiên chứa số nguyên m – số lượng phương án khác nhau chia con đường thành các bức tranh, nếu không có cách phân chia để đảm bảo phân biệt tranh của đúng 2 nhóm thì đưa ra số -1. Nếu có cách phân biệt thì ở mỗi dòng tiếp theo đưa ra 3 số nguyên k, p và q – độ rộng bức tranh, số tranh do nhóm 1 thực hiện và số tranh do nhóm 2 thực hiện, thông tin đưa ra theo thứ tự tăng dần của k và ở mỗi dòng có $p \geq q > 0$.

Ví dụ:

CERAMIC.INP
9
1 2 3 6 4 9 3 1 2

CERAMIC.OUT
1
3 2 1



Giải thuật: Tổng tiền tố và hàm băm.

Nhận xét:

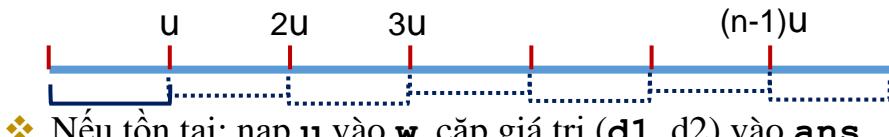
- + Việc nhận dạng 2 dãy số cùng độ dài **x** và **y** có trùng nhau với độ chính xác hoán vị được thực hiện tương tự như phương pháp đă nêu ở bài VS04.
Độ tương đồng,
- + Cần xác định có đúng 2 loại dãy con khác nhau,
- + Phải duyệt mọi dãy con độ dài m là ước của n để tìm tất cả các nghiệm.

Tổ chức dữ liệu:

- + Các mảng:
 - ▣ **int64_t a[100001]** – lưu tổng tiền tố nhận được từ ánh xạ mảng tần số xuất hiện của dữ liệu ban đầu sang tập số nguyên bằng kỹ thuật hàm băm,
 - ▣ **int64_t h[50001]** – mảng giá trị phục vụ tính hàm băm, **h_i = hbⁱ, i=0 ÷ 10⁵,**
- + Các véc tơ:
 - ▣ **vector<int> w** – lưu số lượng nphương án tìm được,
 - ▣ **vector<pair<int, int> >ans** – lưu các cặp giá trị **p** và **q**,
 - ▣ Các biến **d1** và **d2** – lưu số lượng tranh mỗi nhóm thực hiện.

Các bước xử lý:

- Tính các hệ số **h[i]** của hàm băm, **i = 0 ÷ 500 000**,
- Tính các tổng tiền tố giá trị hàm băm ứng với dữ liệu vào,
- Với ước **u** của **n**:
 - ❖ Kiểm tra tồn tại đúng 2 loại tranh độ rộng **u**,



- ❖ Nếu tồn tại: nạp **u** vào **w**, cặp giá trị (**d1**, **d2**) vào **ans**,

- Đưa ra kết quả tìm được.

Độ phức tạp của giải thuật: $\approx O(n \log n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "ceramic."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
uint64_t hb=5017,h[50001],x,y,a[100001];
int n,m,k,p,q,t,r;
vector<pair<int,int> >ans;
vector<int>w;

void check_m(int u)
{int t1=0,t2=0,d1=1,d2=0;
 uint64_t tg1,tg2,tg;
 tg1=a[u];tg2=0;
 for(int i=u;i<=n-u;i+=u)
 {
     tg=a[i+u]-a[i];
     if(tg==tg1)++d1;
     else
     {
         if(d2==0)tg2=tg,++d2;
         else {if(tg==tg2)++d2; else {r=-1;return;}}
     }
 }
 if(d1!=0 && d2!=0)
     {ans.push_back(make_pair(d1,d2)); w.push_back(u);}
}

int main()
{clock_t aa=clock();
 h[0]=1;
 for(int i=1;i<=50000;++i)h[i]=h[i-1]*hb;
 fi>>n;a[0]=0;
 for(int i=1;i<=n;++i){fi>>t;a[i]=a[i-1]+h[t];}
 for(int i=2;i<=n/2;++i)
     if(n%i==0)check_m(i);
 if(w.empty())fo<<-1;
 else
     {fo<<w.size()<<'\n';
      for(int i=0;i<w.size();++i)
          fo<<w[i]<<' '<<ans[i].first<<' '<<ans[i].second<<'\n';
     }

 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS07. ĐIỀU CHỈNH LƯƠNG

Tên chương trình: SALARY.CPP

Một công ty đa quốc gia có cơ cấu tổ chức như sau: nhân viên thường được hưởng lương sàn – bậc 0, bậc thấp nhất, Ban Giám đốc được hưởng mức lương bậc $n+1$ ứng với lương trần cao nhất trong công ty. Giữa nhân viên thường và Ban Giám đốc có n bậc đánh số từ 1 đến n . Người ở bậc i nhận lương tháng là s_i và với $i > j$ không nhất thiết s_i phải lớn hơn s_j . Hiện tại công ty có n nhân viên, nhân viên thứ i có bậc lương là i , $i = 1 \div n$.

Hàng năm Tiêu ban khen thưởng và Tiêu ban kỷ luật của công ty sẽ độc lập xem xét, đánh giá công việc của mỗi nhân viên. Người i ($i = 1 \div m$) nhận được quyết định tăng p_i bậc lương từ Tiêu ban khen thưởng và giảm q_i bậc lương từ Tiêu ban kỷ luật ($0 \leq p_i, q_i \leq n$).

Việc *tăng một bậc lương* được thực hiện như sau: nếu người đó đang ở bậc i thì sẽ được chuyển sang bậc $k > i$ gần nhất có $s_k > s_i$. Việc tăng nhiều bậc lương được thực hiện bằng cách tăng liên tiếp một bậc lương nhiều lần. Nếu lương đã đạt tới mức trần thì việc tăng lương tiếp không làm thay đổi lương.

Việc *giảm một bậc lương* được thực hiện như sau: nếu người đó đang ở bậc i thì sẽ được chuyển sang bậc $r < i$ gần nhất có $s_r < s_i$. Việc giảm nhiều bậc lương được thực hiện bằng cách giảm liên tiếp một bậc lương nhiều lần. Nếu lương đã đạt tới mức sàn thì việc giảm lương tiếp không làm thay đổi lương.

Mỗi người trong số những người được xét có quyền kích hoạt hệ thống xử lý quyết định của Tiêu ban khen thưởng trước rồi mới kích hoạt hệ thống xử lý quyết định của Tiêu ban kỷ luật hoặc ngược lại. Mỗi người đều muốn có lương cao nhất sau khi xử lý.

Hãy xác định bậc của mỗi người nếu họ biết xử lý tối ưu để đạt được mục đích của mình. Người đạt mức lương sàn có bậc là 0, mức lương trần – bậc $n+1$.

Dữ liệu: Vào từ file văn bản SALARY.INP:

- ✚ Dòng đầu tiên chứa 2 số nguyên n ($1 \leq n \leq 5 \times 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên s_1, s_2, \dots, s_n ($0 < s_i \leq 10^9$, $i = 1 \div n$),
- ✚ Dòng thứ 3 chứa n số nguyên p_1, p_2, \dots, p_n ($0 < p_i \leq 10^9$, $i = 1 \div n$),
- ✚ Dòng thứ 4 chứa n số nguyên q_1, q_2, \dots, q_n ($0 < q_i \leq 10^9$, $i = 1 \div n$),

Các số trên một dòng ghi cách nhau ít nhất một dấu cách.

Kết quả: Đưa ra file văn bản SALARY.OUT một dòng chứa n số nguyên, số thứ i xác định bậc mới của người thứ i trong danh sách. Các số ghi cách một dấu cách.

Ví dụ:

SALARY.INP
10
6 2 4 8 5 1 4 9 8 7
3 4 2 1 4 2 3 1 2 2
2 1 4 2 3 3 1 3 1 1

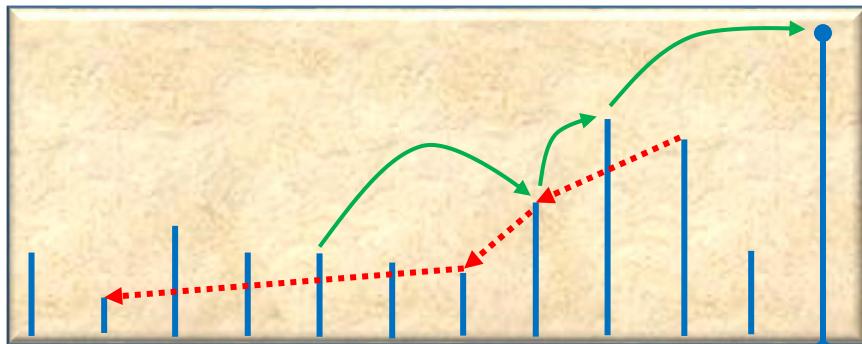
SALARY.OUT
8 11 4 3 11 4 11 1 11 11



Giải thuật: Tổ chức và xử lý Heap.

Nhận xét:

- ⊕ Khi tăng lương, các bậc lương tương ứng với việc liên tiếp tăng một bậc lương tạo thành dãy số tăng dần và khoảng cách giữa 2 bậc liên tiếp phải là nhỏ nhất,



- ⊕ Khi giảm lương, các bậc lương tương ứng với việc liên tiếp giảm một bậc lương tạo thành dãy số giảm dần và khoảng cách giữa 2 bậc liên tiếp phải là nhỏ nhất,
- ⊕ Như vậy ta phải tìm giá trị lớn hơn (nhỏ hơn) gần nhất bên trái (bên phải),
- ⊕ Khi xét tăng lương, dãy số tăng dần đã có sê hô trợ cho việc khéo dài dãy số này sang trái vì vậy phải xét từ người cuối cùng lùi dần đến người đầu tiên,
- ⊕ Ngược lại, với dãy số giảm dần ta có thể mở rộng sang phải, vì vậy phải xét từ người đầu tiên về tới người cuối cùng,
- ⊕ Khi xét tăng lương các bậc lương mà một người có thể tối tạo thành một *Heap min* mà giá trị đầu của Heap là bậc lương người đang xét, tương tự như vậy, khi xét giảm lương các bậc lương mà một người có thể tối tạo thành một *Heap max* mà giá trị đầu của Heap là bậc lương người đang xét,
- ⊕ Việc nạp vào Heap thực hiện như nạp vào Stack (nhưng không được dùng Stack vì ta phải truy nhập vào các giá trị nằm dưới vị trí đầu),
- ⊕ Bài toán ban đầu có thể đưa về 2 bài toán:
 - Tiến hành tăng lương cho mọi người, sau đó thực hiện việc giảm lương,
 - Tiến hành giảm lương cho mọi người, sau đó thực hiện việc tăng lương,
- ⊕ So sánh kết quả cuối cùng ở 2 lần xử lý, chọn kết quả lớn hơn,
- ⊕ Lưu ý là ban đầu mỗi người ở một vị trí khác nhau, nhưng sau khi điều chỉnh tăng (hoặc giảm) lương có thể **nhiều người ở cùng một vị trí!**

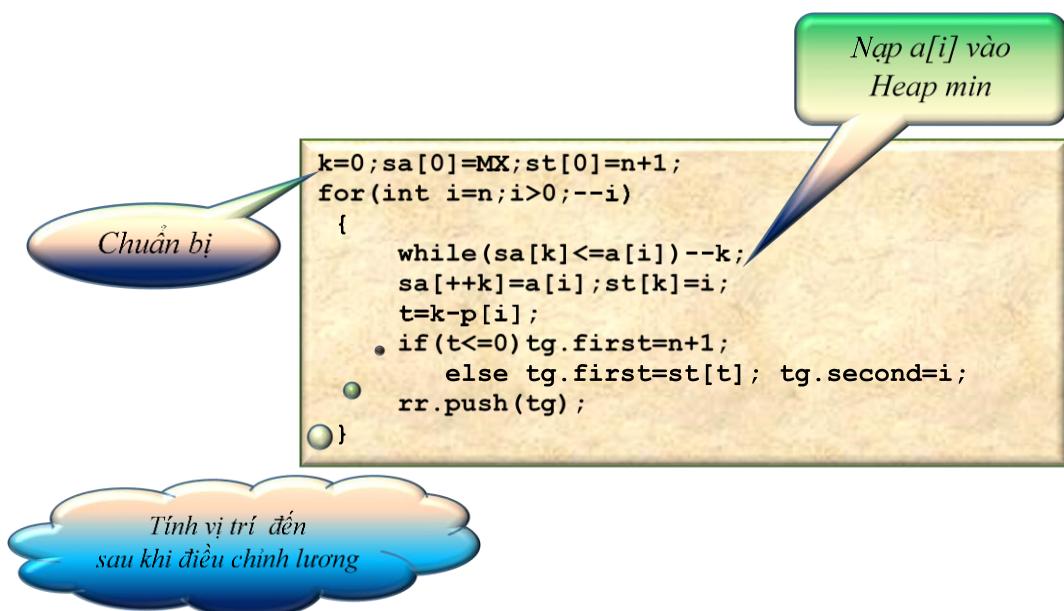
Tổ chức dữ liệu:

- Các mảng nguyên **s,p,q** – chứa dữ liệu vào,

- Mảng nguyên **sa** – dùng để tạo *Heap min* và *Heap max* trong quá trình xử lý,
- Mảng **st** – chứa vị trí của đỉnh *Heap* trong dữ liệu ban đầu,
- Mảng **ans** – chứa vị trí của mỗi người khi duyệt theo trình tự tăng lương, sau đó giảm,
- Mảng **r** – chứa vị trí của mỗi người khi duyệt theo trình tự giảm lương, sau đó tăng,
- Hàng đợi ưu tiên **rr** kiểu cặp dữ liệu nguyên – lưu cặp dữ liệu (*vị trí mới, vị trí ban đầu*) của mỗi người khi xử lý tăng lương, dữ liệu sắp xếp *tăng dần* theo vị trí mới,
- Hàng đợi ưu tiên **rl** kiểu cặp dữ liệu nguyên – lưu cặp dữ liệu (*vị trí mới, vị trí ban đầu*) của mỗi người khi xử lý giảm lương, dữ liệu sắp xếp *giảm dần* theo vị trí mới.

Xử lý:

- Nhập dữ liệu,
- Tính vị trí mới của mỗi người sau khi tăng lương:



- Tương tự: tính vị trí mới của mỗi người sau khi giảm lương,
- Xử lý giảm lương sau khi tăng:
 - Duyệt từ trái sang phải,
 - Nạp các bậc lương vào *Heap max* cho đến khi gặp bậc lương có người cần xử lý,

```

k=0;sa[0]=0;st[0]=0;ti=0;
// # -> L -> R
for(int i=1;i<=n;++i)
{
    tg=rr.top(); rr.pop();
    while(ti<tg.first)
    {
        while(sa[k]>=a[ti])if(a[ti]==0)break;else --k;
        sa[++k]=a[ti];st[k]=ti; ++ti;
    }
    if(st[k]<tg.first)
    {
        ti=tg.first;
        while(sa[k]>=a[ti])if(a[ti]==0)break;else --k;
        sa[++k]=a[ti];st[k]=ti;
    }
    t=k-q[tg.second];
    if(t<=0)ans[tg.second]=0; else ans[tg.second]=st[t];
}

```

Nạp bậc lương vào
Heap max với bậc cần
xử lý

Nạp bậc lương vào
Heap max bỏ cho tới
khi gặp bậc lương cần

Tính vị trí bậc lương mới

- Đưa ra lương có giá trị lớn trong 2 bậc lương tìm được.

Độ phức tạp của giải thuật: O(n).

Ghi chú: Kích thước khai trong chương trình ứng với tường hợp $n \leq 10^6$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "salary."
using namespace std;
typedef pair<int,int> pii;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
const int MX=1000000010;
int n,a[1000002],p[1000001],q[1000001],ans[1000002],r[1000002];
int sa[1000002],st[1000002],k,t,ti;
pii tg;
priority_queue<pii> rl;
priority_queue<pii,vector<pii>,greater<pii> >rr;

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=1;i<=n;++i)fi>>a[i];a[n+1]=MX;
 for(int i=1;i<=n;++i)fi>>p[i];
 for(int i=1;i<=n;++i)fi>>q[i];
 k=0;sa[0]=1000000010;st[0]=n+1;
 for(int i=n;i>0;--i)
 {
     while(sa[k]<=a[i])--k;
     sa[++k]=a[i];st[k]=i; t=k-p[i];
     if(t<=0)tg.first=n+1; else tg.first=st[t]; tg.second=i;
     rr.push(tg);
 }
 k=0;sa[0]=0;st[0]=0;
 for(int i=1;i<=n;++i)
 {
     while(sa[k]>=a[i])--k;
     sa[++k]=a[i];st[k]=i; t=k-q[i];
     if(t<=0)tg.first=0; else tg.first=st[t]; tg.second=i;
     rl.push(tg);
 }
 k=0;sa[0]=0;st[0]=0;ti=0;
 // # -> L -> R
 for(int i=1;i<=n;++i)
 {
     tg=rr.top(); rr.pop();
     while(ti<tg.first)
     {
         while(sa[k]>=a[ti])if(a[ti]==0)break;else --k;
         sa[++k]=a[ti];st[k]=ti; ++ti;
     }
     if(st[k]<tg.first)
     {
         ti=tg.first;
         while(sa[k]>=a[ti])if(a[ti]==0)break;else --k;
         sa[++k]=a[ti];st[k]=ti;
     }
     t=k-q[tg.second];if(t<=0)ans[tg.second]=0; else
     ans[tg.second]=st[t];
 }
 // # -> R -> L
 k=0;sa[0]=a[n+1];st[k]=n+1;ti=n+1;
```

```

for(int i=n;i>0;--i)
{
    tg=rl.top();rl.pop();
    while(ti>tg.first)
    {
        while(sa[k]<=a[ti])if(a[ti]==MX)break;else --k;
        sa[++k]=a[ti];st[k]=ti; --ti;
    }
    if(st[k]>tg.first)
    {
        ti=tg.first;
        while(sa[k]<=a[ti])if(a[ti]==n+1)break;else --k;
        sa[++k]=a[ti];st[k]=ti;
    }
    t=k-p[tg.second];if(t<=0)t=n+1;else t=st[t];r[tg.second]=t;
}
for(int i=1;i<=n;++i)
{
    if(a[ans[i]]>a[r[i]])t=ans[i];else t=r[i];fo<<t<<' ';
}
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}

```

VS02. SỐ NGUYÊN MỚI

Tên chương trình: NEWNUM.CPP

Cho số nguyên dương n có không quá 100 chữ số. Hãy xác định số nguyên lớn nhất m chia hết cho 3 và khác n ở đúng một chữ số.

Ví dụ, $n = 123$ thì m sẽ là 723.

Dữ liệu: Vào từ file văn bản NEWNUM.INP gồm một dòng chứa số nguyên n có không quá 100 chữ số và không chứa các số 0 không có nghĩa.

Kết quả: Đưa ra file văn bản NEWNUM.OUT số nguyên m tìm được.

Ví dụ:

NEWNUM.INP	NEWNUM.OUT
123	723



Giải thuật: Phân tích tình huống.

Nhận xét:

- + Một số nguyên chia hết cho 3 khi và chỉ khi tổng các chữ số chia hết cho 3,
- + Gọi d là tổng các chữ số của số đang xét, ta có thể tăng (điều chỉnh tăng) hoặc giảm (điều chỉnh giảm) một chữ số nào đó để nhận được số mới chia hết cho 3,
- + Bài toán luôn có nghiệm,
- + Cần xét điều chỉnh tăng trước, nếu không tìm thấy chữ số có thể tăng thì tiến hành điều chỉnh giảm,
- + Khi điều chỉnh tăng: Có thể tăng nhiều lần, số gia cho chữ số cần tăng sẽ là $d1=3-d \% 3 + 3*k$, $k=0, 1, \dots$ chừng nào $d1$ còn nhỏ hơn 10,
- + Khi điều chỉnh giảm, số gia cho chữ số cần giảm sẽ là $d1=-d \% 3$ và chỉ giảm một lần!

Tổ chức dữ liệu: Lưu trữ số nguyên n ban đầu dưới dạng xâu.

Xử lý:

- Nhập dữ liệu và tính d ,
- Kiểm tra khả năng điều chỉnh tăng:
 - Tính $d1=3-d \% 3$,
 - Duyệt các chữ số từ trái sang phải tìm chữ số x_i đầu tiên thỏa mãn $x_i+d1 \leq 9$, tiến hành tăng xi chừng nào kết quả tăng còn nhỏ hơn 10,
- Nếu không tìm thấy chữ số có thể tăng thì điều chỉnh giảm: tính lại $d1$, tìm chữ số x_i đầu tiên từ phải sang trái thỏa mãn điều kiện $x_i + d1 \geq 0$ và chỉnh lý (một lần) chữ số này,
- Đưa ra kết quả tìm được.

Độ phức tạp của giải thuật: $O(m)$, trong đó m – số chữ số của n .

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "newnum."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int m,d=0,d1,flg=0;
string s;

int main()
{clock_t aa=clock();
 fi>>s; m=s.size();
 for(int i=0;i<m;++i)d+=s[i]-48;
 d%=3; d1=3-d;
 for(int i=0;i<m;++i)if(s[i]-48+d1<=9)
     {flg=1;while(s[i]+d1<=57)s[i]+=d1,d1=3;break;}
 if(flg==0)for(int i=m-1;i>=0;--i)
     if(s[i]-48-d>=0){s[i]-=d;break;}
 fo<<s;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS09. ẢNH HOA

Tên chương trình: FLOWERS.CPP

Steve có nhiệm vụ trang trí lối vào vườn thực vật của trường. Với sự lựa chọn công phu và chăm sóc chu đáo cả n bụi hoa chạy dọc theo đường thẳng dẫn tới vườn đều sống và nở hoa đồng thời, trông rất hấp dẫn. Bụi thứ i tính từ đầu đường là loại hoa a_i , $i = 1 \div n$. Các bạn trong trường thường đến đứng trước luồng hoa chụp ảnh đưa lên facebook. Theo kinh nghiệm của Steve, muốn được nhiều người like thì phần luồng hoa nền ở đằng sau phải không chứa 3 bụi hoa liên tiếp cùng loại và Steve đưa lên trang Web của trường điểm đầu và cuối đoạn dài nhất nên đưa vào khung hình khi chụp ảnh.



Hãy xác định điểm đầu và cuối được nêu trong trang Web. Nếu có nhiều đoạn cùng độ dài thì đưa ra đoạn có điểm đầu nhỏ nhất.

Dữ liệu: Vào từ file văn bản FLOWERS.INP:

- ✚ Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 2 \times 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản FLOWERS.OUT trên một dòng 2 số nguyên – các điểm đầu và cuối của đoạn dài nhất tìm được.

Ví dụ:

FLOWERS.INP
6
5 6 6 6 23 9

FLOWERS.OUT
3 6

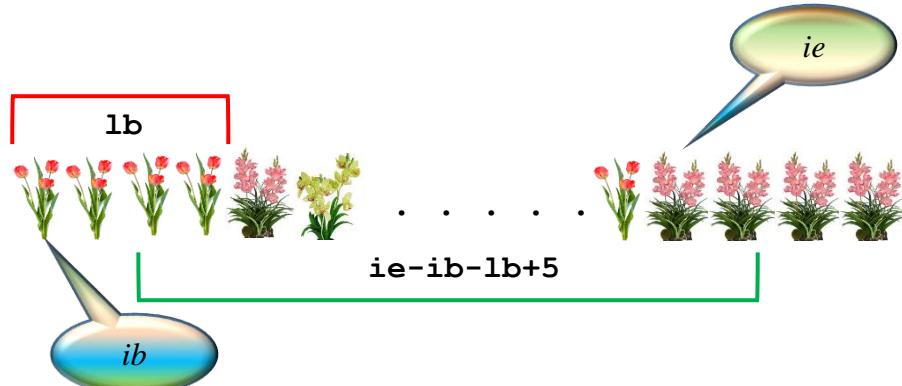


RKO 2015/16 1212 A

Giải thuật: Kỹ thuật duyệt dữ liệu tìm các đoạn giống nhau và khác nhau.

Nhận xét:

- ⊕ Phân biệt 4 tình huống:
 - ▣ Không có 3 cụm hoa liên tiếp nào giống nhau, độ dài cần tìm sẽ là n,
 - ▣ Đoạn không có 3 cụm hoa liên tiếp nhau bắt đầu từ cụm hoa đầu tiên,
 - ▣ Đoạn không có 3 cụm hoa liên tiếp nhau kết thúc ở cụm hoa cuối cùng,
 - ▣ Trường hợp còn lại,
- ⊕ Gọi **ib** là vị trí đầu đoạn có **1b** cụm hoa liên tiếp giống nhau ($1b \geq 3$),
- ⊕ Gọi **ie** là vị trí đầu đoạn có **1e** cụm hoa liên tiếp giống nhau ($1e \geq 3$) tiếp sau đoạn nói trên,
- ⊕ Từ các giá trị **ib**, **ie**, **1b** ta có thể dễ dàng tính ra độ dài đoạn có thể đưa lên ảnh,



- ⊕ Như vậy toàn bộ giải thuật xoay quanh hàm **next3()** tìm **ie** và **1e**,
- ⊕ Mỗi cụm hoa phải so sánh ít nhất với 2 cụm hoa tiếp theo, vì vậy cần bổ sung các giá trị hàng rào a_{n+1} và a_{n+2} với giá trị khác a_n .

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "flowers."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a[200003],ib,ie,lb,le,lx=0,ixb,i xe,t;
void next3()
{
    le=0; ie=n;
    for(int i=ib+lb;i<=n;++i)
        if(a[i]==a[i+1]&&a[i]==a[i+2])
        {
            ie=i;le=3;
            while(a[i+le]==a[i])++le;
            break;
        }
}
int main()
{clock_t aa=clock();
    fi>>n;
    for(int i=1;i<=n;++i) fi>>a[i];a[n+1]=-1;a[n+2]=-2;
    ib=1;lb=0;
    next3();ib=ie;lb=le;
    ixb=1;lx=ib+1; if(lx>n){lx=n; fo<<ixb<<' '<<n; return 0;}
    while(lb)
    {
        next3();
        if(le==0)t=n-ib-lb+3; else t=ie-ib-lb+5;
        if(t>lx){lx=t; ixb=ib+lb-2;if(le==0)ixe=n;else ixe=ie+1;}
        if(ie==n)break;
        ib=ie;lb=le;
    }
    fo<<ixb<<' '<<ixe;

    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS08. CASIO

Tên chương trình: CASIO.CPP

Steve bị ốm và phải nghỉ học. Bạn của Steve gửi Mail đề bài tập thực hành trên máy tính bấm tay Casio. Email mà Steve nhận được có nội dung: "Hãy tính tổng của n số nguyên: " và sau đó là dãy số nguyên a_1, a_2, \dots, a_n .

Steve đang ngạc nhiên về tính đơn giản của bài tập thì nhận được Email mới: "Minh quên mất cách soạn thảo số mũ, ở mỗi số nguyên đã gửi, chữ số cuối cùng là số mũ, ví dụ số 213 có nghĩa là phải tính 21 lũy thừa 3 . Thông cảm nhé ☺"

Hãy tính và đưa ra tổng cần tìm.

Dữ liệu: Vào từ file văn bản CASIO.INP:

- ➡ Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 10$),
- ➡ Dòng thứ i trong n dòng sau chứa số nguyên a_i ($10 \leq a_i \leq 9999$).

Kết quả: Đưa ra file văn bản CASIO.OUT tổng tìm được.

Ví dụ:

CASIO.INP
2
212
1253

CASIO.OUT
1953566



cocitruongtrinh.com

Giải thuật: Thực hành với các phép tính số học, tính đa thức theo sơ đồ Horne.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "casio."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,t,p,q;
int64_t r,ans=0;

int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=0;i<n;++i)
 {
     fi>>t; p=t%10; q=t/10;
     r=1; for(int j=0;j<p;++j) r=r*q;
     ans+=r;
 }
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS10. LẬP TRÌNH NHANH

Tên chương trình: FASTPROG.CPP

Một môn mới được đưa vào chương trình *Ngày hội thể thao* của nhà trường: thi lập trình nhanh đôi kháng. Những người tham dự được chia thành từng cặp đấu loại, mỗi người trong cặp phải lần lượt giải các bài số 0, số 1, số 2, ... trong kho dữ liệu đè cực lớn. Điểm của mỗi bài được cho trong phạm vi từ 0 đến 15 và được ghi nhận dưới dạng một chữ số hệ 16: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Càng lên cao bài càng khó vì vậy kết quả được ghi nhận với trọng số, bài thứ i có trọng số 16^i . Điểm số tích lũy của người chơi sau khi giải bài thứ i sẽ là điểm số cũ cộng với $16^i \times \text{điểm bài } i$. Điểm số cuối cùng của mỗi người tham dự sẽ được ghi nhận dưới dạng một số hệ 16. Đây là thi đấu đôi kháng nên kết quả thi đấu của mỗi người được xác định theo quy tắc sau: Hệ thống so sánh kết quả từng bài của 2 người trong cặp, lần lượt từ bài số 0 trở đi, nếu 2 bài tương ứng có số điểm khác nhau thì bài của người thấp điểm hơn sẽ bị đánh dấu loại bỏ. Nếu một người làm được nhiều bài hơn người kia thì điểm số của những bài làm nhiều hơn được tự động giữ lại. Kết quả cuối cùng của mỗi người tham dự là một số hệ 16 nhận được bằng cách gạch bỏ trong điểm số các chữ số bị đánh dấu loại bỏ.

Ví dụ, người thứ nhất giải được 5 bài với điểm số là **3406A**, người thứ 2 giải được 4 bài với điểm số **1569** thì kết quả thi của 2 người tương ứng sẽ là **346A** và **56**.



Cho biết điểm số của 2 người trong cặp đấu, hãy đưa ra kết quả của 2 người trong cặp đấu đó dưới dạng số hệ 16 không có các chữ số 0 không có nghĩa. Nếu với một người nào đó tất cả các điểm từng bài đều bị gạch bỏ thì đưa ra thông báo kết quả “**Lose**”.

Dữ liệu: Vào từ file văn bản FASTPROG.INP gồm 2 dòng, mỗi dòng chứa một số nguyên hệ 16 không quá 5×10^5 chữ số và không có các chữ số 0 không có nghĩa xác định điểm số của mỗi người trong cặp.

Kết quả: Đưa ra file văn bản FASTPROG.OUT 2 dòng chứa kết quả của cặp đấu.

Ví dụ:

FASTPROG.INP	FASTPROG.OUT
3406A	346A
1569	56



Giải thuật: Xử lý xâu.

Nhận xét:

- + Kết quả của mỗi người được biểu diễn dưới dạng số Hexa, số lượng chữ số lớn vì vậy cần lưu trữ dưới dạng xâu,
- + Cần đảo ngược vị trí các ký tự trong mỗi xâu để ký tự thứ i tương ứng với trọng số 16^i ,
- + So sánh ký tự thứ i trong 2 xâu, $i = 0 \div \text{độ dài của xâu ngắn hơn}$,
- + Đánh dấu xóa: thay ký tự bé hơn bằng ký tự không phải là chữ số hexa (ví dụ bằng ký tự ‘L’), như vậy sự tương ứng trọng số của các chữ số tiếp theo không bị thay đổi,
- + Dẫn xuất kết quả: tạo xâu mới từ các ký tự không bị đánh dấu xóa,
- + Chú ý phân biệt có một xâu kết quả bị rỗng.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "fastprog."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s,x,y,r1,r2;
int lx,ly,m;

int main()
{clock_t aa=clock();
 fi>>s; lx=s.size(); x="";
 for(int i=lx-1;i>=0;--i)x+=s[i];
 fi>>s; ly=s.size(); y="";
 for(int i=ly-1;i>=0;--i)y+=s[i];
 m=lx;if(m>ly)m=ly;
 for(int i=0;i<m;++i)
    if(x[i]>y[i])y[i]='L';else if(y[i]>x[i])x[i]='L';
 r1="";
 for(int i=lx-1;i>=0;--i)if(x[i]!='L')r1+=x[i];
 r2="";
 for(int i=ly-1;i>=0;--i)if(y[i]!='L')r2+=y[i];
 while(r1[0]=='0' && r1.size()>1)r1.erase(0,1);
 while(r2[0]=='0' && r2.size()>1)r2.erase(0,1);
 if(r1!="")fo<<r1<<'\n';else fo<<"Lose\n";
 if(r2!="")fo<<r2<<'\n';else fo<<"Lose\n";
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS11. KẾT CẤU KÉP

Tên chương trình: DUALSTRUCT.CPP

Để chống tác động của động đất nền của đường tàu hỏa cao tốc được thiết kế theo kiểu cấu trúc kép gồm 2 lớp bê tông ứng lực đặt chồng lên nhau. Khi có động đất các lớp bê tông nền sẽ trượt trên nhau, triệt tiêu lực tác động lên đường ray, đảm bảo an toàn cho đoàn tàu và cho cả con đường. Hiện tại có n loại tấm bê tông chịu ứng lực đúc sẵn cùng kích thước, phù hợp với yêu cầu xây dựng đường trong điều kiện bình thường. Tấm bê tông thứ i có trọng lượng a_i , $i = 1 \div n$. Kết quả tính toán cho thấy độ an toàn của con đường sẽ là cao nhất khi tổng trọng lượng của 2 tấm dùng để ghép không vượt quá x và giá trị tuyệt đối hiệu trọng lượng 2 tấm không nhỏ hơn y .

Hãy chỉ ra 2 tấm phù hợp với độ an toàn tối ưu.

Dữ liệu: Vào từ file văn bản DUALSTRUCT.INP:

- ✚ Dòng đầu tiên chứa ba số nguyên n , x và y ($1 \leq n \leq 10^5$, $1 \leq y \leq x \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^5$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản DUALSTRUCT.OUT trên một dòng 2 số nguyên – các chỉ số của những tấm bê tông cần chọn. Nếu không tồn tại cách chọn để đạt độ an toàn tối ưu thì đưa ra một số 0. Trong trường hợp tồn tại nhiều cách chọn – đưa một nghiệm tùy chọn.

Ví dụ:

DUALSTRUCT.INP
5 5 2
1 2 3 4 5

DUALSTRUCT.OUT
1 4



Giải thuật: Nguyên lý cực trị, phương pháp xin cấp phát bộ nhớ động, kỹ thuật tổ chức chu trình.

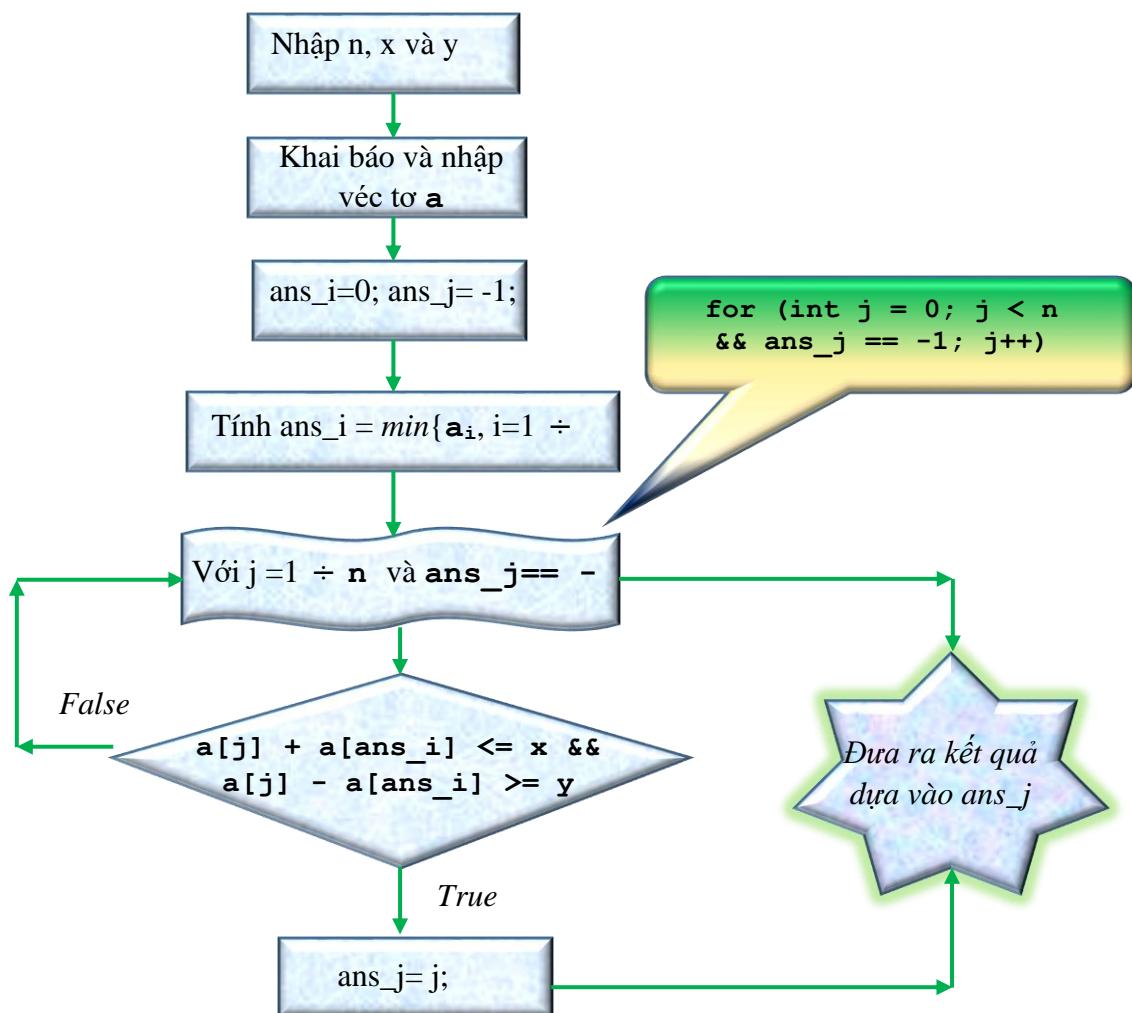
Nhận xét:

Hai tám bê tông i và j ghép được khi thỏa mãn điều kiện $y \leq a_i + a_j \leq x$, nếu tám bê tông có trọng lượng nhỏ nhất không ghép được với tám nào để thỏa mãn các điều kiện đã nêu thì bài toán vô nghiệm.

Tổ chức dữ liệu:

`vector<int> a(n)` – lưu trữ dữ liệu ban đầu, tham số n trong khai báo phục vụ cho hệ thống cấp phát một lần n phần tử, việc xử lý a sẽ có tốc độ tương đương với khai báo xin cấp phát tĩnh.

Xử lý:



Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "dualstruct."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");

int main()
{clock_t aa=clock();

    int n, x, y;
    fi >> n >> x >> y;

    vector<int> a(n);
    for (int i = 0; i < n; i++)
        fi >> a[i];

    int ans_i = 0, ans_j = -1;

    for (int i = 1; i < n; i++)
        if (a[ans_i] > a[i])
            ans_i = i;

    for (int j = 0; j < n && ans_j == -1; j++)
        if(a[j] + a[ans_i] <= x && a[j] - a[ans_i] >= y)
            ans_j = j;

    if (ans_j == -1)
        fo << 0 << endl;
    else
        fo << ans_i + 1 << ' ' << ans_j + 1 << endl;
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS12. MÃ KIỂM TRA

Tên chương trình: NUMBERS.CPP

Steve xây dựng một trang WEB giới thiệu các kiến thức về kỹ thuật lập trình. Tuy không có ý định hạn chế nhưng Steve cũng không muốn người dùng quá tùy tiện truy nhập sâu vào các thông tin trên trang WEB này, nhất là những đối tượng không đủ trình độ chuyên môn. Để đọc hoặc tải thông tin từ trang WEB này người dùng phải chứng tỏ có đủ trình độ chuyên môn và kiên nhẫn. Ngoài ra, việc từ chối cho phép truy nhập cũng nên làm một cách tế nhị để không làm tổn thương tính tự trọng của người truy nhập.

Có 2 số nguyên **a** và **b** được hiển thị trên màn hình. Hướng dẫn kèm theo cho biết thao tác mở khóa là chọn 2 chữ số liên tiếp của **a**. Khi đó 2 chữ số này được thay bằng số dư của tổng 2 chữ số đó khi chia cho 10. Thao tác này được lặp lại nhiều lần cho đến khi đưa được **a** về **b**. Lưu ý là sau mỗi thao tác độ dài của số **a** bị rút ngắn và không có chê độ Lùi về trạng thái trước (Undo) vì vậy bạn chỉ có một cơ hội và phải thao tác chính xác! Với những người bị đánh giá là không đủ điều kiện truy nhập số **a** được đưa ra với giá trị không thể đưa về **b**.

Với 2 số nguyên cho trước hãy xác định người dùng này có thể tiếp cận được các nội dung ở mức sâu hơn hay không và đưa ra thông báo **YES** hoặc **NO**.

Dữ liệu: Vào từ file văn bản NUMBERS.INP gồm 2 dòng, dòng thứ nhất chứa số nguyên **a**, dòng thứ 2 – số nguyên **b**, mỗi số có không quá 2×10^5 chữ số.

Kết quả: Đưa ra file văn bản NUMBERS.OUT thông báo xác định được.

Ví dụ:

NUMBERS.INP	NUMBERS.OUT
123456 326	YES



Giải thuật: Nguyên lý cực trị.

Nhận xét:

- ⊕ Tham số trong các bài toán tin học đều có tính hữu hạn về kích thước và giá trị,
- ⊕ Những giá trị biên (*lớn nhất, nhỏ nhất, vị trí trái nhất, phải nhất, phần tử xuất hiện đầu tiên, cuối cùng, . . .*) là những **cực trị**,
- ⊕ Một hoặc một vài cực trị có các tính chất cho phép nhận dạng, phân lập và xử lý,
- ⊕ Sau khi phân lập và xử lý, các phần tử này có thể được đánh dấu và loại bỏ (tường minh hoặc không tường minh), kích thước bài toán ban đầu bị thu nhỏ,
- ⊕ Quá trình phân lập, xử lý và loại bỏ được thực hiện nhiều lần cho đến khi kích thước bằng 0.

Tổ chức dữ liệu: 2 xâu **a** và **b** chứa dữ liệu ban đầu, **la**, **lb** – độ dài các xâu tương ứng.

Xử lý:

- Nếu xâu **a** có thể đưa về xâu **b** bằng phép biến đổi đã cho thì tồn tại phần đầu của **a** có thể biến đổi về ký tự đầu tiên của **b**, loại bỏ phần đầu của **a** và ký tự đầu tiên của **b**, ta có bài toán ban đầu với kích thước nhỏ hơn. Quá trình trên được thực hiện cho đến khi các xâu **a** và **b** đều đồng thời có kích thước bằng 0 (kết quả **YES**) hoặc phát hiện không thể loại bỏ ký tự đầu của **b** (kết quả **NO**).
- Lưu ý: để giảm thời gian xử lý việc loại bỏ được tiến hành không tường minh: đánh dấu các điểm đầu tiếp theo của **a** và **b**.
- Đánh dấu và loại bỏ:

```
int j = 0, t = 0;
for (int i = 0; i < la; i++)
{
    t = (t + (a[i] - '0')) % 10;
    if (j < lb && t == b[j] - '0')
    {
        t = 0; j++;
    }
}
```

- Ghi nhận kết quả:

```
if (t == 0 && j == lb) fo << "YES" << endl;
else fo << "NO" << endl;
```

Dộ phức tạp của giải thuật: O(n).

Chương trình:

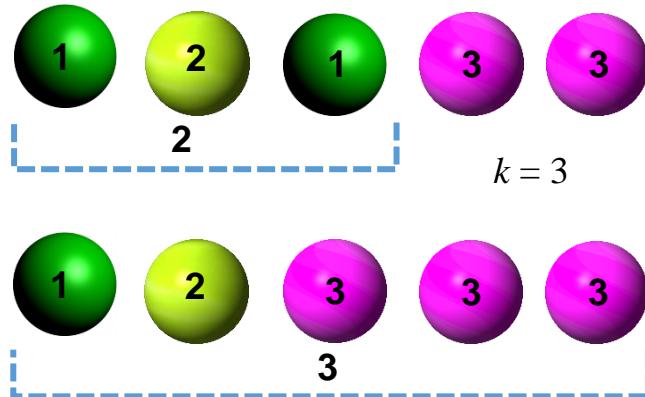
```
#include <bits/stdc++.h>
#define NAME "numbers."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");

int main()
{clock_t aa=clock();
 int la,lb;
 string a, b;
 while (fi >> a >> b)
 {la=a.size(); lb=b.size();
  int j = 0;
  int t = 0;
  for (int i = 0; i < la; i++) {
   t = (t + (a[i] - '0')) % 10;
   if (j < lb && t == b[j] - '0') {
    t = 0;
    j++;
   }
  }
  if (t == 0 && j == lb) fo << "YES" << endl;
  else fo << "NO" << endl;
 }
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
 return 0;
}
```

VS13. QUAN SÁT

Tên chương trình: OBSERVE.CPP

Để rèn luyện trí nhớ và khả năng quan sát nhanh các thành viên trong đội đặc nhiệm thường xuyên phải vượt qua bài tập quan sát. Đó là một mục huấn luyện gian khổ không kém các bài tập thể lực. Trên màn hình hiện lên một dãy n quả cầu màu sắc khác nhau, quả cầu thứ i có màu a_i , $i = 1 \div n$. Thỉnh thoảng một quả cầu nào đó thay đổi màu. Cần phải quan sát, xác định số màu khác nhau trong khoảng k quả cầu liên tiếp. Khi thông báo 2 số l và r , người được kiểm tra phải nói nhanh số lượng màu khác nhau lớn nhất trong các đoạn độ dài k các quả cầu liên tiếp nằm trong phạm vi từ l đến r . Sau khi có câu trả lời trên màn hình sẽ hiện lên kết quả đúng đắn để người học tự đánh giá. Số k là cố định trong suốt quá trình kiểm tra.



Trong buổi học này có m lần xuất hiện sự kiện cầu đổi màu hoặc câu hỏi dành cho người được kiểm tra. Hãy xác định các kết quả hiển thị trên màn hình.

Dữ liệu: Vào từ file văn bản OBSERVE.INP:

- ✚ Dòng đầu tiên chứa 3 số nguyên n , m và k ($1 \leq n, m \leq 10^5$, $1 \leq k \leq n$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^6$, $i = 1 \div n$),
- ✚ Mỗi dòng trong m dòng tiếp theo chứa thông tin một trong 2 dạng:
 - ❖ **1 i x** – đổi màu quả cầu thứ i thành màu x ($1 \leq i \leq n$, $1 \leq x \leq 10^6$),
 - ❖ **2 l r** – câu hỏi dành cho người được kiểm tra ($r-l+1 \geq k$).

Kết quả: Đưa ra file văn bản OBSERVE.OUT các số tương ứng hiển thị trên màn hình theo trình tự xuất hiện câu hỏi, mỗi số trên một dòng.

Ví dụ:

OBSERVE.INP
5 3 3
1 2 1 3 3
2 1 3
1 3 3
2 1 5

OBSERVE.OUT
2
3



Giải thuật: Cấu trúc dữ liệu: kỹ thuật tạo mốc nối, tổng tiền tố, cây BIT.

Nhận xét:

- + Cần xét các đoạn **k** phần tử liên tục vì vậy cần sử dụng tổng tiền tố để tính giá trị của các đoạn,
- + Có **n-k+1** đoạn **k** phần tử liên tiếp do đó cần lưu trữ **n-k+1** giá trị của đoạn này,
- + Với mỗi truy vấn dạng **1 x y :**
 - Xác định việc loại bỏ **a_x** tác động tới những khoảng nào,
 - Cập nhật lại mảng tổng tiền tố,
 - Xác định việc gán **a_x=y** tác động tới những khoảng nào,
 - Cập nhật lại mảng tổng tiền tố,
- + Cập nhật cây BIT,
- + Với mỗi truy vấn dạng **2 l r :** tìm *max* trong đoạn **[l, r]** trên cây BIT.

Tổ chức dữ liệu:

- + **vector <int> a** kích thước 100 001 lưu trữ mảng màu,
- + **vector <int> b** kích thước 400 001 – cây BIT,
- + **vector <int> c** kích thước 1 000 001 – lưu trữ tàn số xuất hiện các trị trong một khoảng độ dài **k** các phần tử liên tiếp của **a**,
- + Các tập **set<int>p[NX+1]** (**NX=10⁶**) quản lý các mốc nối, tập **p_i** quản lý các vị trí xuất hiện của giá trị **i** trong véc tơ **a**.

Xử lý:

- Nhập dữ liệu và tạo tập quản lý vị trí xuất hiện của các phần tử giống nhau:

```
int t;
for(int i=0;i<n;++i)
    {fi>>t;p[t].insert(i);a[i]=t;}
```

- Xác định vị trí các nút lá trong cây BIT:

Gọi **q** – chỉ số nút lá trái nhất trong cây BIT **b**, **b_{i+q}** – số màu khác nhau trong đoạn độ dài **k** bắt đầu từ vị trí **i**, **i = 0 ÷ n-1**:

```
void calc_q()
{int t=n;
 q=1;
 while(t) q<<=1,t>>=1;
}
```

Xác định giá trị các nút lá của BIT:

Giá trị nút lá trái nhất (b_q): tính tần số xuất hiện của các giá trị a_i , $i = 0 \div k-1$, đếm số tần số khác 0:

```
for(int i=0;i<k;++i)
    {t+=(c[a[i]]==0);++c[a[i]];}
b[q]=t;
```

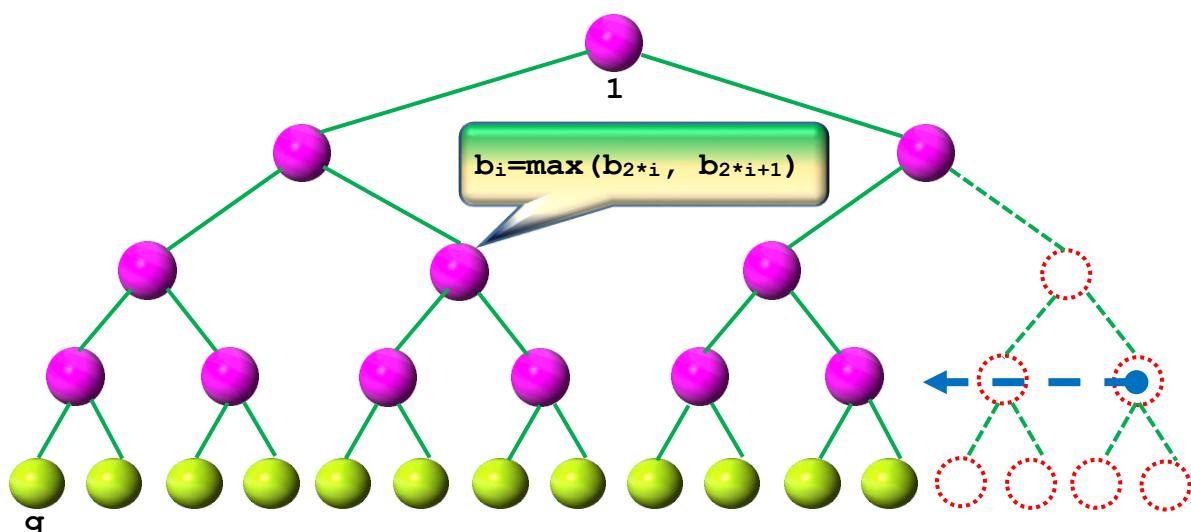
Giá trị nút lá thứ j ($j = 1 \div n-k+1$):

Giảm tần số xuất hiện của a_{j-1} và tăng tần số xuất hiện của a_{j+k-1} , điều chỉnh số lượng tần số khác 0:

```
for(int i=k;i<n;++i)
{
    --c[a[i-k]];if(c[a[i-k]]==0)--t;
    ++c[a[i]];if(c[a[i]]==1)++t;
    b[q+i-k+1]=t;
}
```

Tính giá trị các nút còn lại của BIT:

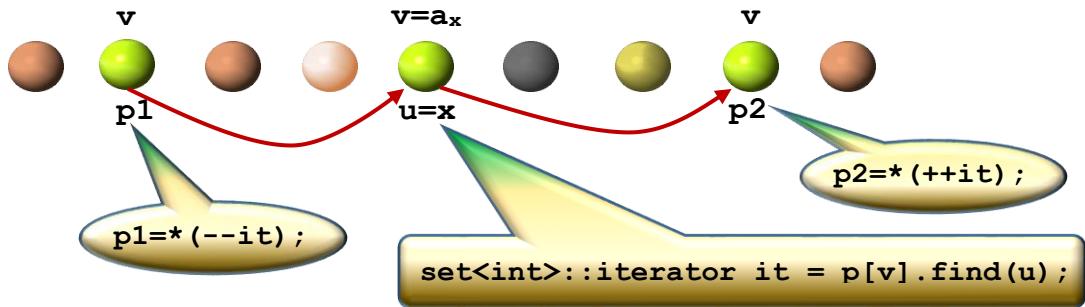
```
for(int i=q-1;i>0;--i)
    b[i]=max(b[2*i],b[2*i+1]);
```



➤ Xử lý truy vấn $1 \times y$:

Gọi $p1$ và $p2$ là vị trí quả cầu gần nhất bên trái và gần nhất ở bên phải cùng màu với quả cầu ở vị trí x ,

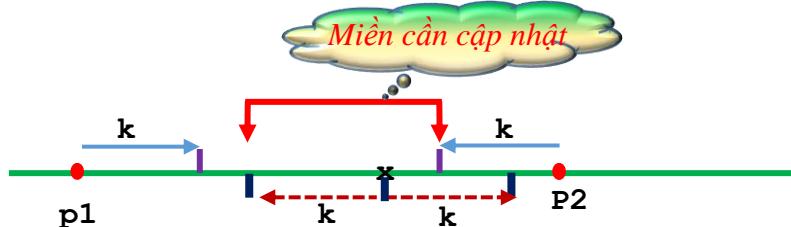
Cách tính các vị trí này được nêu ở hình dưới. Nếu không tồn tại quả cầu cùng màu ở bên trái thì $p1 = -\infty$, nếu không tồn tại quả cầu cùng màu ở bên phải thì $p2 = +\infty$.



Phạm vi ảnh hưởng khi xóa quả cầu ở vị trí $u=x$ là các đoạn độ dài k bắt đầu từ vị trí $l = \max(u-k+1, p1+1)$,

Phạm vi kết thúc ảnh hưởng khi xóa quả cầu ở vị trí $u=x$ là các đoạn độ dài k bắt đầu từ vị trí $r = \min(u, p2-k)$,

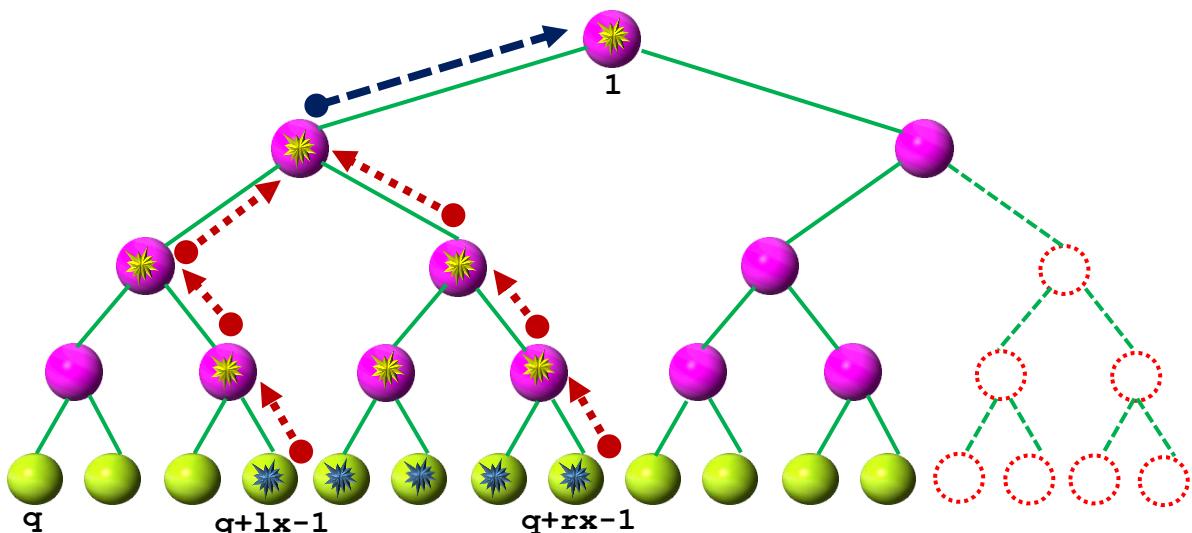
Nếu $l \leq r$ thì số màu khác nhau trong các đoạn độ dài k có điểm đầu trong đoạn $[l, r]$ bị giảm 1.



Tương tự như vậy, tìm miền cần cập nhật (số lượng màu khác nhau tăng lên 1) khi làm cho quả cầu ở vị trí x có màu y .

Cập nhật cây BIT:

Gọi $[lx, rx]$ là đoạn nhỏ nhất chứa các nút bị cập nhật. Việc cập nhật cây bao

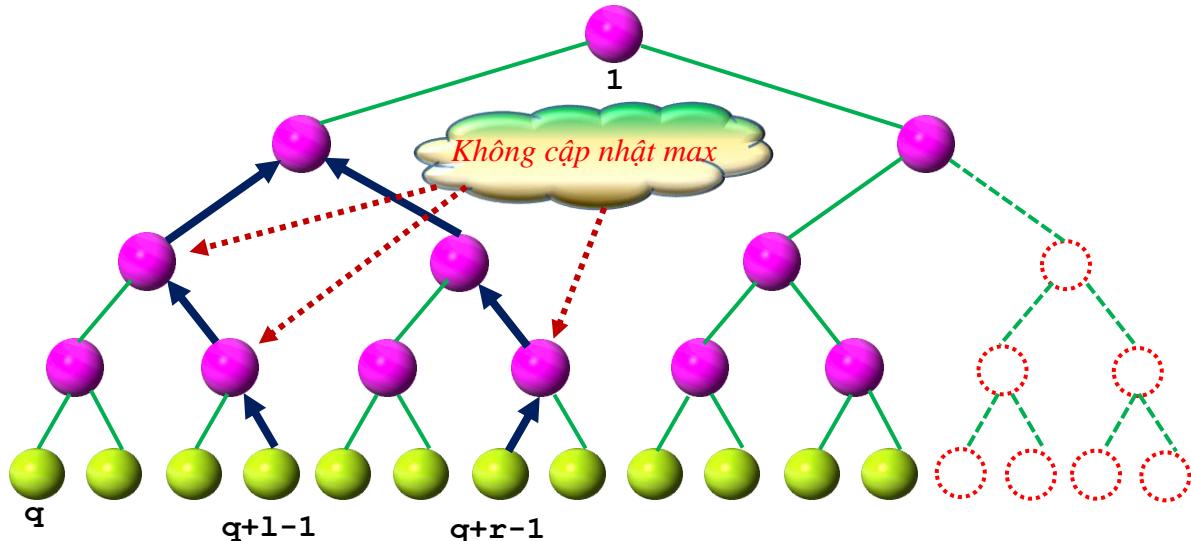


gồm cập nhật các nút nằm trên và giữa 2 con đường đi từ $q+lx-1$ và từ $q+rx-1$ về tới nút cha chung gần nhất và cập nhật các nút trên đường đi từ nút cha chung

đó tới nút gốc của cây BIT (nút 1). Ở hình trên các nút bị cập nhật được đánh dấu sao.

➤ Xử lý truy vấn **2 l r**:

Tìm max trên các nút thuộc đường đi từ $q+l-1$ và từ $q+r-1$ về nút cha chung gần nhất (đường bên trái và đường bên phải). Lưu ý là với đường bên trái, ở các nút được tới từ nút có số thứ tự trong cây là lẻ giá trị max trung gian không thay đổi, với đường bên phải, ở các nút được tới từ nút có số thứ tự trong cây là chẵn giá trị max trung gian không thay đổi.



Hàm xử lý:

```
void get_ans(int u,int v)
{int rl,rr;
    u+=q-1; v+=q-1; rl=b[u]; rr=b[v];
    while(v-u>1)
    {
        if((u&1)==0 )rl=max(rl,b[u+1]);
        if((v&1)==1 )rr=max(rr,b[v-1]);

        u>>=1;v>>=1;
    }
    rl=max(rl,rr);
    fo<<rl<<'\n';
}
```

Độ phức tạp của giải thuật: bậc $O(m \log n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "observe."
using namespace std;
const int NX=1000000;
const int NS=100001;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,k,x,y,rx,lx,rx,l1,l2,r1,r2,q;
set<int>p[NX+1];
vector<int>a(NS),b(4*NS,0),c(NX+1,0);
void calc_q()
{
    int t=n;
    q=1;
    while(t) q<<=1, t>>=1;
}

void upd_b();
void get_ans(int u,int v);
void add(int u,int v,int vl)
{
    for(int i=q+u;i<=q+v;++i)b[i]+=vl;
}

void get_newb(int u,int v)
{
    int u1,v1;
    u+=q;v+=q;
    u1=u>>1; v1=v>>1;
    while(u1!=v1)
    {
        for(int i=u1;i<=v1;++i)b[i]=max(b[2*i],b[2*i+1]);
        u1>>=1; v1>>=1;
    }
    while(u1>0)
    {
        b[u1]=max(b[2*u1],b[2*u1+1]);
        u1>>=1;
    }
}

void get_newb1()
{
    for(int i=q-1;i>0;--i)b[i]=max(b[2*i],b[2*i+1]);
}

int main()
{
    clock_t aa=clock();
    fi>>n>>m>>k;
    calc_q();
    int t;
    for(int i=0;i<n;++i){fi>>t;p[t].insert(i);a[i]=t;}
    t=0;
    for(int i=0;i<k;++i){t+=(c[a[i]]==0);++c[a[i]];}
    b[q]=t;
    for(int i=k;i<n;++i)
    {
        --c[a[i-k]];if(c[a[i-k]]==0)--t;
    }
}
```

```

        ++c[a[i]]; if(c[a[i]]==1)++t;
        b[q+i-k+1]=t;
    }
    for(int i=q-1;i>0;--i)b[i]=max(b[2*i],b[2*i+1]);
    for(int i=0;i<m;++i)
    {
        fi>>t>>x>>y;
        if(t==1){--x;upd_b();} else get_ans(x,y-k+1);
    }
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}

pair<int,int> getseg(int u,int v)
{
    set<int>::iterator it = p[v].find(u);
    int p1=-1000000, p2=2*NX;
    if (it != p[v].begin()) p1 = *(--it), ++it;
    if (++it != p[v].end()) p2 = *it;
    int l = max(0, u - k + 1), r = u;
    l = max(l, p1 + 1);
    r = min(r, p2 - k);
    return make_pair(l,r);
}

void upd_b()
{
    if(a[x]==y) return;
    pair<int,int> seg=getseg(x,a[x]); l1=seg.first;r1=seg.second;
    lx=2*NX; rx=-NX;
    if(l1<=r1) {add(l1,r1,-1);lx=l1;rx=r1;}
    p[a[x]].erase(x);
    p[y].insert(x);
    seg=getseg(x,y);
    l2=seg.first; r2=seg.second;
    if(l2<=r2) {add(l2,r2,1); if(l2<lx)lx=l2; if(r2>rx)rx=r2;}
    a[x]=y;
    if(lx<=rx)get_newb(lx,rx);
}

void get_ans(int u,int v)
{int rl,rr;
u+=q-1; v+=q-1; rl=b[u]; rr=b[v];
while(v-u>1)
{
    if((u&1)==0 )rl=max(rl,b[u+1]);
    if((v&1)==1 )rr=max(rr,b[v-1]);

    u>>=1;v>>=1;
}
rl=max(rl,rr);
fo<<rl<<'\n';
}

```

VS14. QUÀ TẶNG

Tên chương trình: GIFTS.CPP

Nhân dịp năm mới Steve quyết định mua tặng 2 người bạn thân của mình mỗi người một món quà. Trong cửa hàng lưu niệm có n mặt hàng khác nhau, mặt hàng thứ i có giá a_i , $i = 1 \div n$. Với tổng số tiền trong túi là x , Steve quyết định sẽ mua 2 món quà khác nhau có tổng giá trị lớn nhất và tất nhiên – không vượt quá khả năng chi trả của mình.



Ví dụ, có 6 mặt hàng với giá nêu ở trên và số tiền có thể chi tối đa là 18, Steve sẽ chọn các món quà thứ nhất và thứ ba. Tổng số tiền cần chi sẽ là $5 + 10 = 15$.

Hãy xác định tổng số tiền Steve cần chi trả.

Dữ liệu: Vào từ file văn bản GIFTS.INP:

- ─ Dòng đầu tiên chứa một số nguyên n và x ($2 \leq n \leq 10^5$, $2 \leq x \leq 10^9$),
- ─ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản GIFTS.OUT một số nguyên – số tiền cần chi trả.

Ví dụ:

GIFTS.INP
6 18
5 3 10 2 4 9

GIFTS.OUT
15



Giải thuật: Quy hoạch động.

Giải thuật độ phức tạp $O(n^2)$:

- ⊕ Với mỗi mặt hàng i ($i = 1 \div n-1$) tìm $\max\{a_i + a_j\}$ với mọi $j = i+1 \div n$ và thỏa mãn điều kiện $a_i + a_j \leq x$, lưu \max tìm được vào b_i ,
- ⊕ Đưa ra kết quả $\max\{b_i, i = 1 \div n\}$.

Lưu ý: Nếu tìm thấy một $b_i = x$ thì không cần tính các giá trị còn lại của mảng b .

Giải thuật độ phức tạp $O(nlnn)$:

Nhận xét:

- ▣ Chỉ cần thực hiện mở rộng sơ đồ tìm kiếm \max một lần khi chuyển từ chọn một mặt hàng sang chọn 2 mặt hàng,
- ▣ Không cần duyệt hết danh sách các mặt hàng còn lại nếu giá của chúng đã được sắp xếp,
- ▣ Không cần thiết phải lưu các giá trị \max trung gian trong quá trình tìm kiếm.

Xử lý:

- Nhập dữ liệu,
- Sắp xếp mảng a theo thứ tự tăng dần,
- Chuẩn bị: $ans = -1$ (lưu \max), $j = n$ (mặt hàng thứ 2),
- Với mọi $i = 1 \div n-1$:
 - ✓ Chừng nào có $j > i$: Lùi j cho đến khi gặp mặt hàng có thể mua cùng mặt hàng i ,
 - ✓ Chính lý ans ,
- Đưa ra kết quả.

Chương trình: Độ phức tạp O(n²)

```
#include <bits/stdc++.h>
#define NAME "gifts."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a[100001],b[100001],t,r;

int main()
{clock_t aa=clock();
 int n, x;
 fi>>n>>x;
 for (int i = 0; i < n; i++) fi>>a[i];
 for(int i=0;i<n-1;++i)
 {
     t=a[i];r=-1;
     for(int j=i+1;j<n;++j) if(t+a[j]<=x && t+a[j]>r) r=t+a[j];
     b[i]=r; if(r==x) break;
 }
 if(r<x) for (int i=0;i<n-1;i++) if(r<b[i]) r=b[i];
 fo<<r;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

Chương trình: Độ phức tạp O(nlogn)

```
#include <bits/stdc++.h>
#define NAME "gifts."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a[100001];

int main()
{clock_t aa=clock();
 int n, x;
 fi>>n>>x;
 for (int i = 0; i < n; i++) fi>>a[i];
 sort(a, a + n);
 int ans = -1;
 for (int i = 0, j = n - 1; i < n; i++)
 {
     while (j > i && a[i] + a[j] > x) --j;
     if (j > i) ans = max(ans, a[i] + a[j]);
 }
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS15. CHUNG KẾT GIẢI ĐÁ CẦU

Tên chương trình: FINAL.CPP

Hai đội *Bóng Súng* và *Hoa Ban* lọt vào chung kết giải Đá cầu Toàn quốc làm các bạn ở trường có đội *Bóng súng* hết sức phấn khích. Nhưng do là các tinh nguyện viên phục vụ giải đấu nên các bạn ấy không được xem trực tiếp trận chung kết mà phải theo dõi qua băng ghi hình phát sau khi giải đã kết thúc.



Quy tắc của trận chung kết là hai đội phải đấu liên tiếp nhiều hiệp, mỗi hiệp đều có phân thắng thua (tức là không có kết quả hòa), đội thắng trong hiệp được một điểm. Đội nào đạt n điểm trước sẽ là đội vô địch. Thông tin đầu băng ghi hình cho biết cuộc chiến dành chức vô địch đã diễn ra trong k hiệp đấu. Dựa vào các thông tin này có thể dễ dàng xác định được tỷ số cuối cùng. Ví dụ $n=3$ và $k = 4$ thì tỷ số *Bóng Súng* : *Hoa Ban* sẽ là 3:1 hoặc 1:3 vì nếu tỷ số là 3:0 hoặc 0:3 thì chỉ cần 3 hiệp đấu, ngược lại nếu kết quả là 3:2 hoặc 2:3 thì cần phải đấu tới 5 hiệp. Sau khi cùng xem một số hiệp đấu đầu tiên thì có thể xác định được kết quả một số hiệp đấu khác trong số các hiệp chưa xem và có thể sau khi xem m hiệp đấu tiên thì có thể biết đội nào giành chức vô địch.

Với n , k và a_1, a_2, \dots, a_k cho trước, trong đó $a_i \in \{1, 2\}$ xác định đội thắng trong hiệp thứ i , $i = 1 \div k$ hãy xác định m – số hiệp đấu cần xem để biết ai là đội vô địch và xác định những hiệp đấu nào có thể biết trước kết quả mà không cần xem.

Dữ liệu: Vào từ file văn bản FINAL.INP:

- ✚ Dòng đầu tiên chứa hai số nguyên n và k ($1 \leq n \leq 100$, $1 \leq k \leq 2 \times n - 1$),
- ✚ Dòng thứ hai chứa k số nguyên a_1, a_2, \dots, a_k .

Kết quả: Đưa ra file văn bản FINAL.OUT:

- Dòng thứ nhất chứa số nguyên m ,
- Dòng thứ 2 chứa k số nguyên b_1, b_2, \dots, b_k , trong đó $b_i = 0$ – không đoán được kết quả hiệp thứ i , $b_i = 1$ – có thể xác định được kết quả hiệp i , $i = 1 \div k$.

Ví dụ:

FINAL.INP
3 4
1 1 2 1

FINAL.OUT
2
0 0 1 1



Giải thuật: Phân tích tình huống lô gic.

Nhận xét:

- + Gọi điểm số hiện tại của 2 đội tương ứng là \mathbf{a} và \mathbf{b} ,
- + Số hiệp đấu mà đội đạt chức vô địch bị thua là $\mathbf{sc} = \mathbf{n} - \mathbf{k}$,
- + Nếu ở hiệp đấu tiếp theo đội ít điểm thắng và làm cho số điểm của 2 đội đều lớn hơn \mathbf{sc} thì sẽ dẫn đến vô lý và như vậy ở hiệp đấu tiếp theo này đội ít điểm sẽ bị thua,
- + Nếu ở hiệp đấu tiếp theo một đội nào đó thắng và làm cho điểm của mình vượt quá \mathbf{n} thì cũng sẽ mâu thuẫn với điều kiện đầu bài, như vậy đội nhiều điểm hơn sẽ bị thua,
- + Nếu ở hiệp đấu tiếp theo một đội nào đó thắng và đạt mốc \mathbf{n} điểm trong khi tổng số các hiệp (kể cả hiệp tiếp theo đang xét) chưa bằng \mathbf{k} thì cũng mâu thuẫn với điều kiện đầu bài, như vậy đội nhiều điểm hơn sẽ bị thua,
- + Đội đầu tiên có số điểm vượt quá \mathbf{sc} sẽ là đội vô địch và xem xong hiệp đấu này sẽ biết kết quả cuối cùng của trận đấu!

Lưu ý: Không cần thiết lưu trữ mảng giá trị $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n$ ở dữ liệu vào.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "final."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int r[200]={0},sc,n,k,a = 0,b = 0,ans = -1,x;

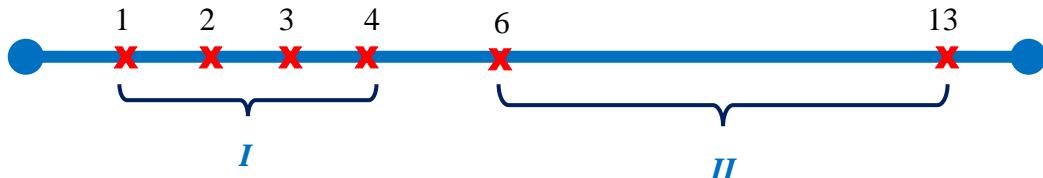
bool can(int x, int y)
{
    if (x > sc && y > sc) return true;
    if (x > n || y > n) return true;
    if ((x == n || y == n) && (x + y < k)) return true;
    return false;
}

int main()
{clock_t aa=clock();
 fi>>n>>k;
 sc = k - n;
 for (int i = 0; i < k; i++)
 {
    fi>>x;
    if (can(a + 1, b) || can(a, b + 1))r[i] = 1;
    if (x == 1) ++a; else ++b;
    if (a > sc || b > sc) if (ans < 0) ans = i;
 }
 fo<<ans + 1<<'\n';
 for (int i = 0; i < k; i++) fo<<r[i]<<' ';
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS16. TRẠM THU PHÍ

Tên chương trình: STATIONS.CPP

Dọc theo đường cao tốc được xây dựng theo phương thức BOT có n trạm thu phí, trạm thứ i ở vị trí x_i tính từ đầu con đường. Để quản lý hoạt động, các trạm thu phí được chia thành k đơn vị, mỗi đơn vị bao gồm các trạm liên tiếp nhau và có từ a đến b trạm, mỗi trạm phải thuộc một đơn vị quản lý. Bán kính quản lý của mỗi đơn vị là khoảng cách từ trạm đầu đến trạm cuối trong đơn vị này. Những người làm công tác quản lý thường xuyên phải đi duyệt tất cả các trạm trong đơn vị quản lý của mình, vì vậy việc phân chia các trạm vào đơn vị quản lý phải đảm bảo sao cho bán kính quản lý lớn nhất phải là nhỏ nhất.



Ví dụ có 6 trạm thu phí ở các vị trí 1, 2, 3 ,4, 6, 13 và cần chia thành 2 đơn vị quản lý, mỗi đơn vị có từ 2 đến 4 trạm. Với việc phân chia tối ưu, bán kính quản lý lớn nhất sẽ là $13 - 6 = 7$.

Hãy đưa ra bán kính quản lý lớn nhất trong phương án phân chia tối ưu.

Dữ liệu: Vào từ file văn bản STATIONS.INP:

- ✚ Dòng đầu tiên chứa 4 số nguyên n , k , a và b ($1 \leq n \leq 200$, $1 \leq k \leq n$, $1 \leq a \leq b \leq n$, $a \times k \leq n \leq b \times k$),
- ✚ Dòng thứ 2 chứa n số nguyên x_1, x_2, \dots, x_k ($0 \leq x_1 < x_2 < \dots < x_k \leq 10^9$).

Kết quả: Đưa ra file văn bản STATIONS.OUT một số nguyên - bán kính quản lý lớn nhất trong phương án phân chia tối ưu.

Ví dụ:

STATIONS.INP
6 2 2 4
1 2 3 4 6 13

STATIONS.OUT
7



StPb20151214 E

Giải thuật: Quy hoạch động.

Nhận xét:

Lời giải của bài toán có thể nhận được bằng cách mở rộng dần số đơn vị quản lý. Xuất phát từ không có đơn vị quản lý nào, xác định những trạm nào có thể đưa vào đơn vị quản lý thứ nhất và bán kính lớn nhất trong mỗi trường hợp nếu việc phân phối vào đơn vị quản lý kết thúc ở trạm này. Đây là một việc làm đơn giản: trạm quản lý đầu tiên bao gồm các trạm từ 1 đến a , 1 đến $a+1$, 1 đến $a+2$, ..., 1 đến b , bán kính quản lý trong mỗi trường hợp là khoảng cách từ trạm cuối tới trạm đầu (trạm 1).

Dấu hiệu trạm đã thuộc một đơn vị quản lý nào đó là bán kính quản lý ứng với nó khác $+\infty$.

Giả thiết đã phân phối các trạm có thể vào j đơn vị quản lý, xét trường hợp có thêm một đơn vị quản lý mới.

Gọi $d_{i,j}$ – bán kính lớn nhất của đơn vị quản lý thứ j có trạm cuối là i (trong cách phân chia tối ưu các trạm cho j đơn vị quản lý), các trạm e có thể đưa vào làm trạm cuối của đơn vị quản lý mới là $e = i+a, i+a+1, i+a+2, \dots, i+b$ và đương nhiên cần có $e \leq n$. Duyệt những e thỏa mãn các điều kiện nêu trên và ghi nhận bán kính quản lý nhỏ hơn nếu có.

Kết quả bài toán sẽ có sau khi mở rộng số đơn vị quản lý đến k và kết quả cần tìm lưu ở $d_{n,k}$.

Tổ chức dữ liệu:

Mảng **int dp[201][201]** – lưu bảng giá trị phục vụ mở rộng lời giải, thực tế chỉ cần 2 cột, nhưng với kích thước n không lớn ($n \leq 200$) việc lưu cả bảng sẽ dễ lập trình hơn. **Ý nghĩa của $d_{i,j}$ đã được nêu ở trên.**

Xử lý:

- Khởi tạo giá trị đầu cho dp : $dp_{0,0} = 0$, các phần tử còn lại $= +\infty$,
- Sơ đồ lặp mở rộng số trạm quản lý:
 - ❖ Xét lần lượt từ trái sang phải tất cả các trạm,
 - ❖ Với mỗi trạm – xét khả năng đưa vào đơn vị quản lý j , $j = 0 \div k$, (trường hợp xét $j = 0$ phục vụ khởi tạo cột đầu cho bảng dp),
 - ❖ Công thức lặp:

```
int val = dp[i][j];
if (val == INF) continue;
for (int e = i + a; e <= n && e <= i + b; e++)
{
    int cur = max(val, x[e - 1] - x[i]);
    if (dp[e][j + 1] > cur)
        dp[e][j + 1] = cur;
}
```

Độ phức tạp của giải thuật: $O(n^2)$.

Lưu ý: Có thể dùng 2 cột **int dp_old[201], dp_new[201]**, hoặc **int dp[2][201]** và áp dụng kỹ thuật con lắc để biến giá trị mới thành giá trị cũ, với cách sau – độ phức tạp của giải thuật không thay đổi.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "stations."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int const INF = 1 << 30;
int dp[201][201];
int x[201];

int main()
{clock_t aa=clock();
 int n, k, a, b;
 fi>>n>>k>>a>>b;
 for (int i = 0; i < n; i++) fi>>x[i];
 for (int i = 0; i <= n; i++)
 for (int j = 0; j <= k; j++)
 dp[i][j] = INF;
 dp[0][0] = 0;
 for (int i = 0; i < n; i++)
 for (int j = 0; j < k; j++)
 {
 int val = dp[i][j];
 if (val == INF) continue;
 for (int e = i + a; e <= n && e <= i + b; e++)
 {
 int cur = max(val, x[e - 1] - x[i]);
 if (dp[e][j + 1] > cur)
 dp[e][j + 1] = cur;
 }
 }
 fo<<dp[n][k];
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS17. TIẾNG ANH CỔ

Tên chương trình: ENGLISH.CPP

Steve không thích môn Ngôn ngữ Anh cổ lăm, các bài tập ở môn này luôn bị để lại sau cùng, khi đã giải hết các bài tập của những môn khác. Hôm nay cũng vậy, chỉ có điều là bây giờ không còn đủ thời gian nghiên cứu về các quy tắc ngữ pháp! Nội dung của bài tập là dịch **n** từ tiếng Anh hiện đại sang ngôn ngữ tiếng Anh cổ. Steve quyết định cứu vãn tình thế bằng cách sử dụng các quy tắc thay thế chung, có thể không đạt kết quả cao nhưng điểm số chắc cũng không đến nỗi tệ!

Các quy tắc thay thế là như sau:

- Các ký tự **s** không xuất hiện đầu từ và sau nó không phải là **h** thì được thay bằng nhóm ký tự **th**,
- Nếu **e** là ký tự đầu tiên của từ thì sẽ được thay bằng **æ**,
- Cặp ký tự **oo** được thay bằng **ou**, nếu có một dãy nhiều ký tự **o** liên tiếp nhau thì chỉ có cặp ký tự đầu tiên bị thay thế.

Các từ cần dịch chỉ chứa ký tự la tinh thường, ngoại trừ chữ cái đầu tiên có thể là chữ hoa, độ dài mỗi từ không quá 30.

Hãy xác định kết quả dịch theo các quy tắc thay thế đã nêu.

Dữ liệu: Vào từ file văn bản ENGLISH.INP:

- Dòng đầu tiên chứa một số nguyên **n** ($1 \leq n \leq 100$),
- Mỗi dòng trong **n** dòng sau chứa một từ cần dịch.

Kết quả: Đưa ra file văn bản ENGLISH.OUT các từ đã được dịch ra tiếng Anh cổ, mỗi từ trên một dòng.

Ví dụ:

ENGLISH.INP	ENGLISH.OUT
3	soun
soon	Aenglisch
English	thith
this	



Giải thuật: Nhận dạng và xử lý xâu.

Nhận xét:

- ✚ Để nhận cần chuẩn hóa đối tượng: Lưu ký tự đầu của xâu và nếu là ký tự hoa – thay bằng ký tự thường tương ứng,
- ✚ Ký tự ‘**e**’ xử lý theo vị trí xuất hiện,
- ✚ Ký tự ‘**s**’ xử lý theo vị trí xuất hiện và phụ thuộc vào ký tự tiếp theo sau,
- ✚ Nhóm ký tự “**oo**”: Dùng biến đánh dấu để phân biệt nhóm đầu tiên với các nhóm sau.
- ✚ Khôi phục lại trạng thái chữ hoa/thường dựa vào ký tự đầu đã lưu giữ.

Độ phức tạp của giải thuật: $O(m)$, trong đó m – tổng độ dài các xâu cần xử lý.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "english."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s,ans;
char c;
int n,ls;
void trans_s()
{int ib=0,fl=0;
    ls=s.size(); ans="";
    c=s[0]; if(c<97)s[0]+=32;
    if(s[0]=='e')ans+="ae",ib=1;
    while(ib<ls)
    {
        if(ib>0 && s[ib]=='s' &&(ib==ls-1 ||ib<ls-1 &&
s[ib+1]!='h'))ans+="th",++ib; else
            if(ib<ls-1 && s[ib]=='o' && s[ib+1]=='o'&&
fl==0)ans+="ou",fl=1,ib+=2;
            else ans+=s[ib],++ib;
    }
    if(c<97)ans[0]-=32;
    fo<<ans<<'\n';
}

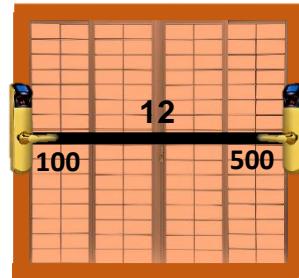
int main()
{clock_t aa=clock();
 fi>>n;
 for(int i=0;i<n;++i)
 {
     fi>>s; trans_s();
 }
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS19. SÓNG SÓT

Tên chương trình: SURVIVE.CPP

Trong chương trình truyền hình *Sóng sót* của Discovery, Bear Gryll có nhiệm vụ dẫn dắt một người vượt qua hành trình với các thử thách tới hạn của con người. Một trong các thử thách lần này là rèn luyện khả năng giữ bình tĩnh và suy nghĩ nhạy bén trong tình huống nguy hiểm.

Hai người đi vào một đường hầm mỏ bị bỏ hoang và tới trước cánh cửa ra đang đóng. Trời mưa tầm tã và nước ở con suối bên ngoài tràn vào mỏ. Nếu không nhanh chóng mở cửa thoát ra ngoài thì cả hai sẽ bị chết chìm trong làn nước đục ngầu đang dâng lên rất nhanh. Để mở được cửa cần rút thanh chấn bị khóa ở 2 đầu bằng 2 khóa số. Cạnh khóa bên trái có ghi số nguyên a , cạnh khóa bên phải – số nguyên b ($a \leq b$) và ở giữa cửa – số nguyên x . Khóa bên trái sẽ mở khi xoay các ô số tạo thành số nguyên nhỏ nhất lớn hơn hoặc bằng a , bé hơn hoặc bằng b và có tổng các chữ số là x . Khóa bên phải sẽ mở khi xoay các ô số tạo thành số nguyên lớn nhất nhỏ hơn hoặc bằng a , bé hơn hoặc bằng b và có tổng các chữ số là x .



Hãy xác định các số cần xác lập trên 2 ô khóa để mở được cửa.

Dữ liệu: Vào từ file văn bản SURVIVE.INP:

- ✚ Dòng đầu tiên chứa số nguyên a ,
- ✚ Dòng thứ 2 chứa số nguyên b ,
- ✚ Dòng thứ 3 chứa số nguyên x .

Dữ liệu đảm bảo tồn tại cách mở khóa, $1 \leq a \leq b \leq 10^9$.

Kết quả: Đưa ra file văn bản SURVIVE.OUT hai số cần chọn để mở khóa, mỗi số trên một dòng, số thứ nhất tương ứng với số cần chọn ở ô khóa bên trái, số thứ 2 – tương ứng với ô khóa bên phải.

Ví dụ:

SURVIVE.INP
100
500
12

SURVIVE.OUT
128
480



Crotia2016_5 A

Giải thuật: Phân tích tình huống lô gic, nguyên lý cực trị.

Nhận xét:

- ⊕ Cần xét 2 bài toán con độc lập:
 - ❖ Tìm số $r1$ nhỏ nhất có tổng các chữ số bằng x và $r1 \geq a$,
 - ❖ Tìm số $r2$ lớn nhất có tổng các chữ số bằng x và $r2 \leq b$,
- ⊕ Với mỗi toán con: phân biệt 3 trường hợp:
 - ❖ Tổng các chữ số của a (b) bằng x ,
 - ❖ Tổng các chữ số của a (b) nhỏ hơn x ,
 - ❖ Tổng các chữ số của a (b) lớn hơn x ,
- ⊕ Dữ liệu đảm bảo tồn tại nghiệm vì vậy không cần nhận trường hợp vô nghiệm với mỗi bài toán con.

Tổ chức dữ liệu:

- ⊕ Các mảng nguyên c và d lưu trữ các chữ số của a và b , đồng thời chứa kết quả để tính $r1$ và $r2$,
- ⊕ Các biến nguyên ta và tb – lưu tổng các chữ số của a và b , na , nb – số lượng chữ số của a và b .

Xử lý:

- Hàm **calc_mn()** – tính $r1$ (bài toán con thứ nhất),
- Hàm **calc_mx()** – tính $r2$ (bài toán con thứ 2),
- Tìm $r1$ khi $ta \neq x$:
 - ✓ Trường hợp $u = x - ta > 0$: Duyệt từ hàng đơn vị trở đi ($i = 0 \div na - 1$):
 - Chừng nào u còn lớn hơn $9 - c_i$ – thay c_i bằng 9 và giảm u một lượng $9 - c_i$,
 - Trong trường hợp ngược lại: tăng c_{i+1} lên thêm u , kết thúc biến đổi.
 - ✓ Trường hợp $u = ta - x > 0$:
 - Chừng nào u còn lớn hơn $c_i \rightarrow$ thay c_i bằng 0 và tương ứng giảm u ,
 - Thay c_0 bằng $c_i - u$, $c_i = u - 1$, tăng c_{i+1} lên 1.
- Lưu ý: *số chữ số có thể tăng, cần chú ý khi đưa ra kết quả.*
- Tìm $r2$ khi $tb \neq x$:
 - ✓ Trường hợp $u = tb - x > 0$: giảm các chữ số từ hàng đơn vị một cách thích hợp tương tự như ở bài toán con thứ nhất,
 - ✓ Trường hợp $tb < x$:
 - Tìm i nhỏ nhất thỏa mãn các điều kiện:

- $\mathbf{d}_i > 0$,
- $x - \sum_{j=i}^{nb-1} d_j + 1 \geq i \times 9$
- Giảm 1 ở \mathbf{d}_i , biến phần còn lại thành số có dạng $99\dots900\dots0m$ ($0 \leq m \leq 9$) sao cho tổng các chữ số từ 0 đến $nb-1$ bằng x .

Độ phức tạp của giải thuật: O(max{na, nb})

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "survive."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a,b,r1=0,r2=0,x,na=0,nb=0,ta=0,tb=0,c[12]={0},d[12]={0},t,tg;

void calc_mn()
{int u,v;
    if(ta==x){r1=a;return;}
    if(ta<x)
    {
        u=x-ta;
        for(int i=0;i<12 && u>0;++i)
        {
            v=u+c[i]-9;
            if(v>=0)u=v,c[i]=9;
            else {c[i]+=u;u=0;break;}
        }
    }
    else
    {
        u=ta-x;
        for(int i=0;i<na && u>0;++i)
            if(u>c[i])u-=c[i],c[i]=0;
            else {u-=c[i],++c[i+1];++u;c[0]-=u;c[i]=0;}
    }
    for(int i=11;i>=0;--i)r1=r1*10+c[i];
}

void calc_mx()
{int u,v;
    if(tb==x){r2=b;return;}
    if(tb>x)
    {
        u=tb-x;
        for(int i=0;i<nb && u>0;++i)
            if(u>d[i])u-=d[i],d[i]=0;
            else d[i]-=u,u=0;
    }
    else
    {
        u=x;
        for(int i=nb-1;i>=0;--i)
        {
            u-=d[i];
            if(d[i]>0 && (u<=i*9+1))v=i;
        }
        --d[v];++u;
        for(int i=v-1;i>=0;--i)if(u>9)d[i]=9,u-=9; else d[i]=u,u=0;
    }
    for(int i=nb;i>=0;--i)r2=r2*10+d[i];
}
```

```
}

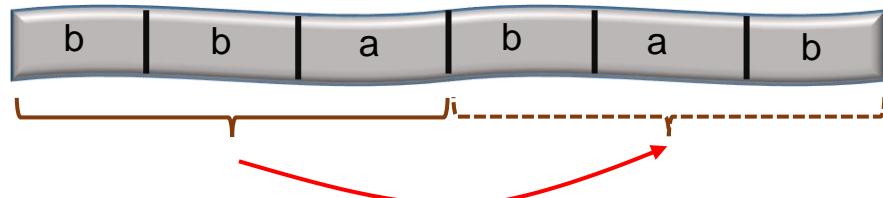
int main()
{clock_t aa=clock();
 fi>>a>>b>>x;
 t=a;
 while(t>0)
 {
    tg=t%10;c[na]=tg;++na;ta+=tg;t/=10;
 }
 t=b;
 while(t>0)
 {
    tg=t%10;d[nb]=tg;++nb;tb+=tg;t/=10;
 }
 calc_mn(); calc_mx();
 fo<<r1<<'\\n'<<r2;

clock_t bb=clock();
fo<<"\\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS20. CƠ SỞ

Tên chương trình: BASIC.CPP

Con đường cao tốc mới hoàn thành được chia thành n đoạn, mỗi đoạn do một trong số 26 công ty đảm nhiệm việc lát asphalt mặt đường, có loại chống trơn trượt, loại chống nước bắn, loại tro với hóa chất rửa đường v.v... Mỗi công ty chỉ sử dụng một loại asphalt. Các asphalt được ký hiệu bằng một trong số các chữ cái la tinh thường.



Việc phân chia công ty nào thi công lát đoạn đường nào là công việc khá khó khăn và tốn kém vì liên quan tới điều hòa quan hệ giữa đơn vị chủ quản với các công ty. Để đơn giản hóa công tác đấu thầu người ta lấy m đoạn đầu tiên, tổ chức đấu thầu, các đoạn gồm m phần tiếp sau đó cũng sẽ do các công ty đã trúng thầu đảm nhiệm với số lượng in hệt như số đoạn họ đã giành được ở đoạn đầu tiên, tuy vậy trình tự loại asphalt có thể thay đổi theo đặc thù của địa hình. Như vậy, ở mỗi khoảng gồm m đoạn đường tiếp theo nếu ta thay đổi vị trí các đoạn đường trong đó ta có thể có khoảng giống hệt khoảng m đoạn đường đầu tiên. Dĩ nhiên m được chọn sao cho khoảng cuối cùng cũng phải có đúng m đoạn.

Khoảng m đoạn đường đầu tiên được gọi là *đoạn cơ sở*. Hồ sơ kỹ thuật xác định cách lát đường được ghi nhận dưới dạng xâu s độ dài n chỉ chứa các ký tự la tinh thường.

Hãy xác định xâu tương ứng với đoạn cơ sở ngắn nhất có thể.

Dữ liệu: Vào từ file văn bản BASIC.INP gồm một dòng chứa xâu s độ dài không quá 10^5 .

Kết quả: Đưa ra file văn bản BASIC.OUT đưa ra xâu tương ứng với đoạn cơ sở ngắn nhất hoặc số -1 nếu toàn thể con đường là đoạn cơ sở ngắn nhất.

Ví dụ:

BASIC.INP	BASIC.OUT
bbbabab	bba



Crotia2016_5 B

Giải thuật: Hàm băm và tổng tiền tố.

Nhận xét:

- Việc so khớp các xâu con với độ chính xác hoán vị có thể thực hiện nhanh chóng bằng cách so sánh giá trị tương ứng của hàm băm (xem VS04. Độ tương đồng),
- Để có giá trị băm của xâu con các ký tự liên tục – dùng tổng tiền tố,
- Trước khi tính giá trị băm nên ánh xạ ‘a’ → 0, ‘b’ → 1, ‘c’ → 2, . . .
- Nên kiểm tra riêng trường hợp tất cả các ký tự của xâu đẽ giống nhau,
- Gọi n là độ dài của xâu s, việc kiểm tra cơ sở thực hiện với các ước của n, từ nhỏ đến lớn và dừng khi tìm được cơ sở đầu tiên.

Tổ chức dữ liệu và xử lý: Tương tự như đã nêu ở VS04. Độ tương đồng.

Độ phức tạp của giải thuật: $\approx O(n \log n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "basic."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int64_t h[26],p=100001,q=1000000007,a[100001];
string s,sr;
char c;
int n;

void check_b(int x)
{int64_t t,flg=1;
 t=a[x];
 for(int i=x;i<=n-x;i+=x)
    if(t!=a[i+x]-a[i]) {flg=0;break;}
 if(flg)sr=s.substr(0,x);
}
int main()
{clock_t aa=clock();
 fi>>s; n=s.size();
 h[0]=1;a[0]=0;
 for(int i=1;i<26;++i)h[i]=h[i-1]*p%q;
 for(int i=0;i<n;++i)a[i+1]=a[i]+h[s[i]-97];
 c=s[0];sr=c;
 for(int i=0;i<n;++i)if(s[i]!=c){sr="-1"; break;}
 if(sr=="-1")
 {
    for(int i=2;i<=n/2;++i)
    {
        if(n%i==0)check_b(i);
        if(sr!="-1")break;
    }
 }
 fo<<sr;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS21. ĐÀN ORGAN

Tên chương trình: ORGAN.CPP

Khi chơi đàn organ nếu nghệ sĩ bấm đồng thời nhiều phím một lúc sẽ tạo được âm thanh sống động hơn. Nhưng con người chỉ có 10 ngón tay nên số phím đồng thời có thể bấm là không quá 10. Vì vậy người ta chế tạo rô bốt chơi nhạc có thể bấm k phím đồng thời. đương nhiên phải chế tạo riêng đàn organ đặc biệt cho rô bốt này. Đàn có n phím. Các phím khác nhau ngoài sự khác biệt về cao độ còn khác biệt về cường độ và trường độ (tức là thời gian ngắn khi bấm phím). Phím thứ i có trường độ là a_i , $i = 1 \dots n$.

Để thử nghiệm tổ hợp đàn và rô bốt người ta cài vào rô bốt chương trình theo đó lần lượt tắt cả các tổ hợp k phím khác nhau sẽ được bấm. Tổ hợp phím tiếp theo sẽ được nhấn khi âm thanh của phím có trường độ dài nhất ở lần nhấn trước tắt. Tổng thời gian từ khi bắt đầu bấm tổ hợp phím đầu tiên cho đến khi âm thanh ở lần bấm cuối cùng tắt là thời gian thử nghiệm.



Ví dụ, với $n = 5$, $k = 3$ và trường độ các phím là 2, 4, 2, 3, 4 sẽ có 10 lần nhấn tổ hợp các phím (1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 3, 4), (1, 3, 5), (1, 4, 5), (2, 3, 4), (2, 3, 5), (2, 4, 5) và (3, 4, 5). Thời gian thử nghiệm sẽ là $4 + 4 + 4 + 4 + 3 + 4 + 4 + 4 + 4 + 4 = 39$.

Cho n , k và các giá trị a_1, a_2, \dots, a_n . Hãy xác định thời gian thử nghiệm và đưa ra theo mô đun 10^9+7 .

Dữ liệu: Vào từ file văn bản ORGAN.INP:

- ✚ Dòng đầu tiên chứa hai số nguyên n và k ($1 \leq n \leq 10^5$, $1 \leq k \leq 50$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$, $i = 1 \dots n$).

Kết quả: Đưa ra file văn bản ORGAN.OUT một số nguyên – thời gian thử nghiệm (theo mô đun 10^9+7).

Ví dụ:

ORGAN.INP
5 3
2 4 2 3 4

ORGAN.OUT
39



Giải thuật: Tam giác Pascal.

Nhận xét:

- ⊕ Thời gian thử không phụ thuộc vào trình tự các phím vì vậy dữ liệu nên sắp xếp theo thứ tự tăng dần,
- ⊕ Thời gian thử một tổ hợp sẽ bằng giá trị trường độ của phím phải nhất trong tổ hợp,
- ⊕ Thời gian thử các tổ hợp có phím i là phím phải nhất bằng $C_{i-1}^{k-1} \times a_i$,
- ⊕ Các giá tổ hợp chập $k-1$ là các phần tử của tam giác Pascal,
- ⊕ Bằng phương pháp cập nhật tại chỗ có thể tính $C_1^{k-1}, C_2^{k-1}, \dots, C_{n-1}^{k-1}$, ở mỗi dòng chỉ cần tính lại $k-1$ giá trị (từ 1 đến $k-1$),
- ⊕ Tích lũy thời gian thử nghiệm song song với việc cập nhật dòng dữ liệu của tam giác Pascal.

Tổ chức dữ liệu:

- ▣ Mảng **int64_t p[51]={0}** – chứa k phần tử đầu của một dòng trong tam giác Pascal,
- ▣ Mảng **int a[1000001]** – chứa dữ liệu ban đầu.

Xử lý:

- Nhập và sắp xếp dữ liệu theo thứ tự tăng dần,
- Xuất phát từ $\mathbf{p} = (1, 0, 0, \dots, 0)$ cập nhật lại \mathbf{p} $n-1$ lần, song song với việc tích lũy kết quả.

```
p[0]=1;
for(int i=1;i<n;++i)
{
    for(int j=min(i,k);j>0;--j)p[j]=(p[j]+p[j-1])%q;
    ans=(ans+p[k]*a[i])%q;
}
```

Dộ phức tạp của giải thuật: $O(n \ln n)$ (sử dụng hàm sắp xếp).

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "organ."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int64_t p[51]={0},q=1000000007,ans=0;
int m=5;
int n,k,a[100001];

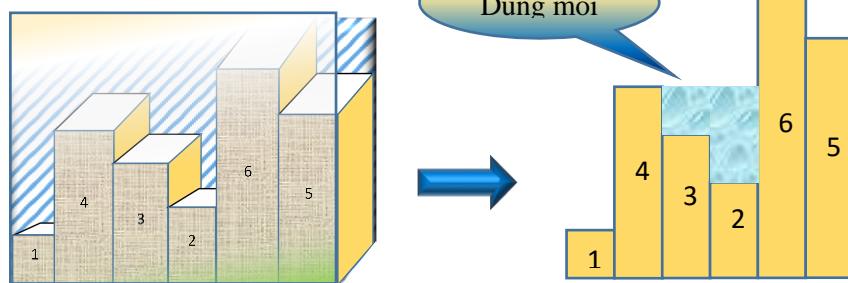
int main()
{clock_t aa=clock();
 fi>>n>>k;--k;
 for(int i=0;i<n;++i)fi>>a[i];
 sort(a,a+n);
 p[0]=1;
 for(int i=1;i<n;++i)
 {
     for(int j=min(i,k);j>0;--j)p[j]=(p[j]+p[j-1])%q;
     ans=(ans+p[k]*a[i])%q;
 }
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS22. DUNG KÉ

Tên chương trình: MEASURE.CPP

Steve sáng chế ra một dung kê cho phép điều chỉnh và lấy chính xác ra một lượng dung môi thể tích x . Cấu tạo của dung kê đơn giản đến mức không ngờ. Dung kê bao gồm n hình lăng trụ đứng đáy là hình vuông diện tích 1. Các lăng trụ có độ cao nguyên, khác nhau từng đôi một và nằm trong phạm vi từ 1 tới n . Các lăng trụ được ghép sát với nhau thành một hàng. Hai mặt bên là 2 tấm mi ca là thành chắn.



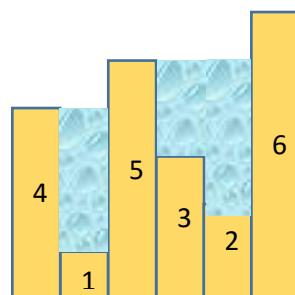
Khi nhúng dung kê vào thùng dung môi rồi kéo thẳng lên, dung môi sẽ đọng lại các chỗ lõm bị chắn 2 bên (sát nó hoặc xa hơn) bởi những cột cao hơn. Ở hình trên, lượng dung môi đọng lại là 3 đơn vị thể tích.

Thay đổi vị trí các cột ta có thể điều chỉnh để tổng thể tích dung môi đọng lại thay đổi. Với sự bố trí các cột như ở hình dưới lượng dung môi đọng lại sẽ là 8.

Cho n và x . Hãy chỉ ra một cách bố trí các cột lăng trụ để thu được lượng dung môi đọng lại đúng bằng x .

Nếu không có cách bố trí thì đưa ra số -1.

Dữ liệu: Vào từ file văn bản MEASURE.INP gồm một dòng chứa 2 số nguyên n và x ($1 \leq n \leq 10^6$, $1 \leq x \leq 10^{15}$).



Kết quả: Đưa ra file văn bản MEASURE.OUT trên một dòng n số nguyên xác định cách bố trí các cột hoặc số -1 nếu không tồn tại cách bố trí.

Ví dụ:

MEASURE.INP
6 3

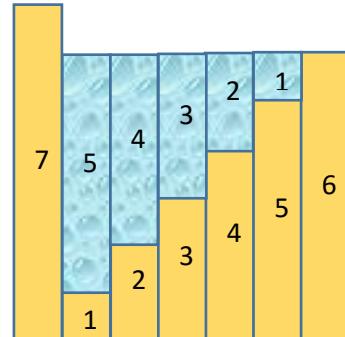
MEASURE.OUT
1 4 3 2 6 5



Giải thuật: Cấp số cộng.

Nhận xét:

- ✚ Lượng dung môi (gọi tắt là nước) lớn nhất có thể giữ lại được tương ứng với trường hợp khi 2 cột cao nhất (độ cao n và $n-1$) làm vách chắn, các cột còn lại được bố trí ở giữa 2 cột này,
- ✚ Việc thay đổi vị trí các cột trong phạm vi chứa nước không làm thay đổi lượng nước được chứa,
- ✚ Như vậy lượng nước lớn nhất có thể chứa là $1 + 2 + 3 + \dots + n-2 = \frac{(n-2)(n-1)}{2} = m$,
- ✚ Nếu $x > m$ – bài toán vô nghiệm, với $x \leq m$ – bài toán luôn có nghiệm,
- ✚ Trường hợp $x \leq m$:

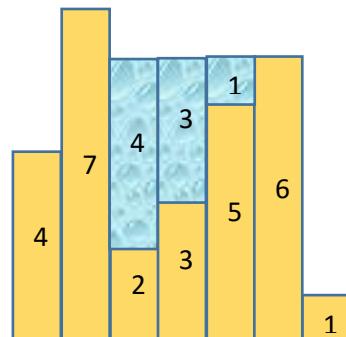


- ─ Tìm y nguyên và nhỏ nhất thỏa mãn điều kiện $z = y \times (y+1)/2 \geq x$,
- ─ Nếu $z = x$: bố trí các cột độ cao $n-2, n-3, \dots, n-1-y$ giữa 2 cột độ cao n và $n-1$,
- ─ Nếu $z > x$: bố trí tương tự như trên nhưng bỏ đi cột độ cao $k = (n-1)-(z-x)$, cột độ cao k có thể bố trí trước cột độ cao n , các cột còn lại: bố trí sau cột độ cao $n-1$ theo thứ tự giảm dần của độ cao.

Ví dụ, với $n = 7$ và $x = 8$ ta tìm được $y = 4$, $z = 4 \times 5/2 = 8$, như vậy cần bỏ đi cột $k = (7-1)-2 = 4$.

Tổ chức dữ liệu: chỉ cần sử dụng các biến đơn kiểu `int64_t` lưu từ các đại lượng đã nêu ở trên.

Độ phức tạp của giải thuật: Nếu không tính thời gian vào/ra dữ liệu độ phức tạp tính toán là $O(1)$.



Lưu ý: Với mỗi $x \leq m$ có thể tồn tại nhiều cách bố trí khác nhau.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "measure."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n;
int64_t x,y,z,m,k;
double u;

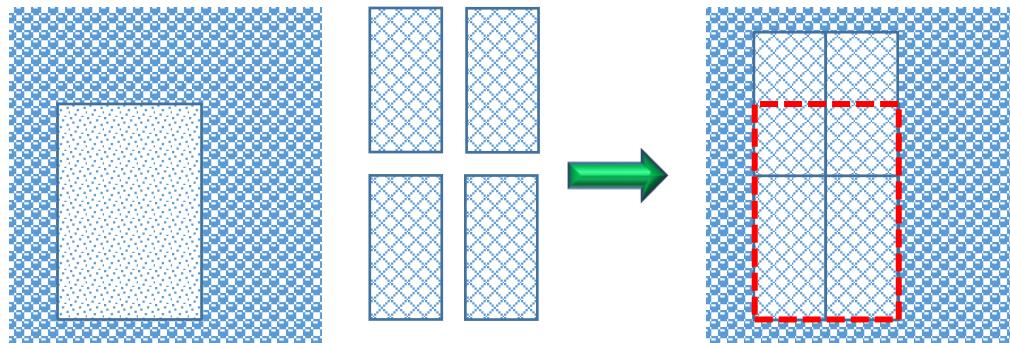
int main()
{clock_t aa=clock();
 fi>>n>>x;
 m=(int64_t) (n-1)*(n-2)/2; if(x>m){fo<<-1; return 0;}
 u=(double) (1+8*x);
 y=(int64_t) ((-1+sqrt(u))/2)+1;
 if(y*(y-1)/2==x)--y,k=0;
 else z=y*(y+1)/2-x, k=n-1-z;
 if(k>0) fo<<k<<' ' ; fo<<n<<' ' ;
 for(int i=n-2;i>=n-y-1;--i) if(i!=k) fo<<i<<' ' ;
 fo<<n-1<<' ' ;
 for(int i=n-y-2;i>0;--i) fo<<i<<' ' ;

clock_t bb=clock();
fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}
```

VS23. SỬA CHỮA

Tên chương trình: REPAIR.CPP

Viện nghiên cứu Đại dương học được xây dựng dưới đáy biển. Tường tòa nhà bằng nhựa trong có dấu hiệu hư hỏng trên một vùng hình chữ nhật độ cao h và độ rộng w . Người ta quyết định gia cố vùng yếu này bằng cách ghép đè lên đó các tấm nhựa, mỗi tấm có chiều cao a và độ rộng b . Các tấm nhựa này được thiết kế đặc biệt với các khe lồng vào nhau khi lắp ráp và bè mặt lượn sóng thích hợp để giảm thiểu lực tác động của dòng chảy đại dương. Chính vì vậy không được xoay ngang khi lắp ráp. Các tấm được gắn gia cố tạo thành hình chữ nhật phủ vừa khít hoặc rộng hơn vùng cần gia cố.



Hãy xác định số tấm ít nhất cần thiết để gia cố.

Dữ liệu: Vào từ file văn bản REPAIR.INP gồm một dòng chứa 4 số nguyên a , b , h , w ($1 \leq a, b, h, w \leq 1000$).

Kết quả: Đưa ra file văn bản REPAIR.OUT một số nguyên – số tấm nhựa cần dùng.

Ví dụ:

REPAIR.INP	REPAIR.OUT
2 1 3 2	4



Giải thuật: Luyện tập cơ sở.

Nhận xét:

- + Do không thể xoay tấm nhựa nên xử lý chiều cao và chiều rộng là độc lập,
- + Số hàng ngang các tấm nhựa mang ghép sẽ là phần nguyên là tròn lên của $\frac{h}{a}$,
- + Số hàng dọc các tấm nhựa mang ghép sẽ là phần nguyên là tròn lên của $\frac{w}{b}$,
- + Kết quả cần đưa ra là tích 2 đại lượng tìm thấy ở trên.

Độ phức tạp của giải thuật: O(1).

Lưu ý:

- Hướng dẫn cách tính làm tròn lên,
- Hướng dẫn cách khai báo, làm việc với file kiểu *stream*,
- Hướng dẫn cách tính và đưa ra thời gian thực hiện chương trình.

Chương trình:

```
#include <bits/stdc.h>
#define NAME "repair."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int a,b,h,w,u,v;

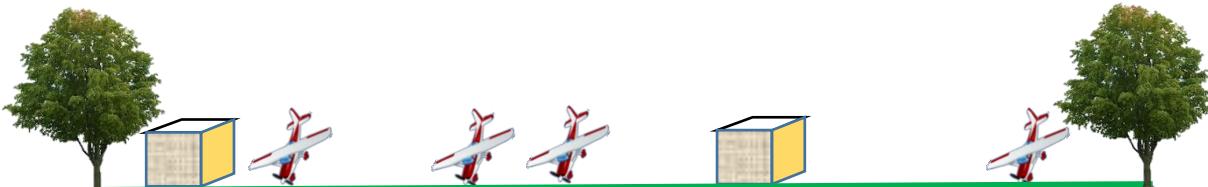
int main()
{clock_t aa=clock();
 fi>>a>>b>>h>>w;
 u=(h+a-1)/a; v=(w+b-1)/b;
 fo<<u*v;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}
```

VS24. MÔ HÌNH MÁY BAY

Tên chương trình: AIRPLANS.CPP

Câu lạc bộ mô hình máy bay chuẩn bị chương trình biểu diễn với nhiều máy bay cùng tham gia hoạt động đồng bộ. Hoạt động của các máy bay được điều khiển từ xa bằng chương trình.

Các máy bay đặt trước vạch xuất phát. Hai bên vạch xuất phát là 2 cây lớn. Ở vị trí xuất phát có n chỗ, một số chỗ có máy bay đỗ (mỗi chỗ không quá 1 máy bay) ký hiệu bằng ký tự ‘D’, một số chỗ có đê hòm đựng cụ ký hiệu ‘#’, các chỗ còn lại – ký hiệu ‘.’.



Tất cả các máy bay đồng thời hoạt động theo một chương trình điều khiển. Việc đầu tiên phải dẫn các máy bay rời khỏi vị trí xuất phát. Thay vì các lệnh tiến, lùi nhóm lập trình đưa dãy m câu lệnh di chuyển sang trái ‘L’ và sang phải ‘R’. Nếu thực hiện theo chương trình này, một số máy bay sẽ bị va vào cây hoặc hòm đựng cụ và sẽ bị vỡ nát. Những máy bay hỏng không ảnh hưởng đến hoạt động theo chương trình của các máy bay còn lại.

Rất may là giáo viên hướng dẫn đã kịp phát hiện ra sai sót này. Ông rất tức giận giáo huấn cả tổ hơn 20 phút và sau đó, dưới hình thức kỹ luật, bắt mọi người tính số máy bay còn lại nếu chương trình được kích hoạt, thực hiện hết và vị trí ban đầu của các máy bay đó. Các vị trí trên vạch xuất phát được đánh số bắt đầu từ 1.

Hãy đưa ra số máy bay còn lại và vị trí ban đầu của chúng

Dữ liệu: Vào từ file văn bản AIRPLANS.INP:

- ✚ Dòng đầu tiên chứa 2 số nguyên n và m ($1 \leq n, m \leq 10^5$),
- ✚ Dòng thứ 2 chứa xâu độ dài n gồm các ký tự từ tập { ., D, # },
- ✚ Dòng thứ 3 chứa xâu độ dài m chỉ chứa các ký tự từ tập { L, R }.

Kết quả: Đưa ra file văn bản AIRPLANS.OUT dòng thứ nhất số nguyên k – số máy bay còn lại, dòng thứ 2 chứa k số nguyên theo thứ tự tăng dần xác định vị trí ban đầu của những máy bay còn lại.

Ví dụ:

AIRPLANS.INP
11 5
#D.DD..#..D
RRRLL

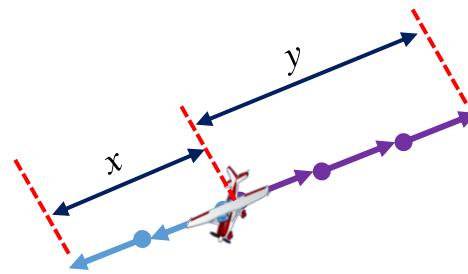
AIRPLANS.OUT
2
2 4



Giải thuật: Xử lý xâu và tổng tiền tố.

Nhận xét:

- ⊕ Theo chương trình, mỗi máy bay sẽ dao động sang trái tối đa \mathbf{x} vị trí, dao động sang phải tối đa \mathbf{y} vị trí, như vậy nếu máy bay ở vị trí \mathbf{i} thì vị trí trái nhất sẽ tới là $\mathbf{pl} = \mathbf{i}-\mathbf{x}$, vị trí phải nhất sẽ tới là $\mathbf{pr} = \mathbf{i}+\mathbf{y}$,
- ⊕ Nếu trong khoảng $[\mathbf{pl}, \mathbf{pr}]$ không có vật cản (cây hoặc hòn dựng cụ) máy bay sẽ nguyên vẹn,
- ⊕ Để xác định khoảng $[\mathbf{pl}, \mathbf{pr}]$ có vật cản hay không: dựa vào hàm tổng tiền tố các vật cản,
- ⊕ Lưu ý: các ký tự của xâu đánh số từ 0, vị trí máy bay – đánh số từ 1.



Tổ chức dữ liệu:

- ▣ Các xâu **sa**, **sp** – lưu dữ liệu ban đầu (trạng thái vạch xuất phát và chương trình điều khiển),
- ▣ Mảng **int a[1000002]** – lưu tổng tiền tố các vật cản, *cản chuẩn bị a[0]=1*,
- ▣ Mảng **int r[1000001]** – lưu vị trí ban đầu các máy bay còn nguyên vẹn, việc sử dụng bộ nhớ phân phối tĩnh sẽ giảm thời gian thực hiện chương trình.

Xử lý:

- Tính tổng tiền tố:

```
a[0]=1;
for(int i=0;i<n;++i)
    if(sa[i]=='#')a[i+1]=a[i]+1;else a[i+1]=a[i];
a[n+1]=a[n]+1;
```

- Tính biên độ dao động:

```
for(int i=0;i<m;++i)
{
    if(sp[i]=='L')--p; else ++p;
    if(p<pl)pl=p; if(p>pr)pr=p;
}
```

- Xác định máy bay còn nguyên vẹn:

```
for(int i=1;i<=n;++i)
    if(sa[i-1]=='D')
        if(i+pl>0 && i+pr<=n && a[i+pr]-a[i+pl-1]==0)
            {++k;r[k]=i;}
```

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "airplans."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,k,pl=0,pr=0,p=0,a[100002],r[100001];
string sa,sp;

int main()
{clock_t aa=clock();
 fi>>n>>m;
 fi>>sa>>sp;
 a[0]=1;
 for(int i=0;i<n;++i)
    if(sa[i]=='#')a[i+1]=a[i]+1;else a[i+1]=a[i];
 a[n+1]=a[n]+1;
 for(int i=0;i<m;++i)
 {
    if(sp[i]=='L')--p; else ++p;
    if(p<pl)pl=p; if(p>pr)pr=p;
 }
 for(int i=1;i<=n;++i)
    if(sa[i-1]=='D')
        if(i+pl>0 && i+pr<=n && a[i+pr]-a[i+pl-1]==0)
            {++k;r[k]=i;}
 }

fo<<k<<'\n';
for(int i=1;i<=k;++i)fo<<r[i]<<' ';

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS25. SỐ LỚN NHẤT

Tên chương trình: MAXNUMBER.CPP

Sau các tiết học căng thẳng thầy giáo ra một bài tập để cả lớp cùng tham gia. Một dãy số nguyên a_1, a_2, \dots, a_n được viết trên bảng. Mỗi người khi được gọi có thể thực hiện một trong số các hành động sau:

- Xóa một số chẵn tùy chọn và thay nó bằng 2 số nguyên, mỗi số bằng nữa số bị xóa,
- Xóa 2 số giống nhau trong dãy và ghi vào dãy một số bằng tổng 2 số đã xóa,
- Tuyên bố trò chơi kết thúc vì trong dãy đã có số lớn nhất có thể đạt được bằng 2 phép biến đổi trên.

Hãy xác định giá trị của số mà khi xuất hiện sẽ làm trò chơi kết thúc.

Dữ liệu: Vào từ file văn bản MAXNUMBER.INP:

- Dòng đầu tiên chứa một số nguyên n ($1 \leq n \leq 10^5$),
- Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 2 \times 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản MAXNUMBER.OUT một số nguyên – số lớn nhất có thể đạt được trong quá trình biến đổi.

Ví dụ:

MAXNUMBER.INP
6
2 2 2 2 4 4

MAXNUMBER.OUT
16

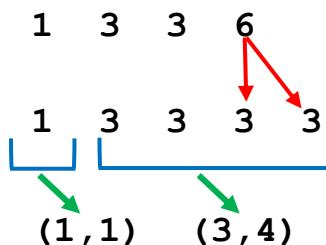


Giải thuật: Làm việc với cấu trúc dữ liệu **Map**.

Nhận xét:

- ⊕ Nếu a_i là một số chẵn, $a_i=2\times x$ ta có thể thay số này bằng 2 số x , điều này không ảnh hưởng đến kết quả vì khi cần thiết có thể gộp chúng trở lại,
- ⊕ Quá trình phân chia nói trên được thực hiện chừng nào còn có thể, kết quả ta được một dãy số với nhiều số giống nhau,
- ⊕ Dãy số mới này có thể lưu trữ dưới dạng các cặp (x, m) , trong đó x – số trong dãy, m – số lần xuất hiện của x trong dãy, các cặp khác nhau có x khác nhau,
- ⊕ Số lượng cặp sẽ không vượt quá n (số phần tử trong dãy dữ liệu vào).

Ví dụ, với dãy số 1, 3, 3, 6 ta có:



- ⊕ Quá trình gộp 2 số thành một số lớn hơn chỉ có thể thực hiện được giữa các số xuất xứ từ một nhóm và không phụ thuộc vào các nhóm còn lại,
- ⊕ Với nhóm (x, m) số lớn nhất có thể đạt được khi gộp là $y = x \times 2^k$, trong đó k là số nguyên lớn nhất thỏa mãn điều kiện $2^k \leq m$,
- ⊕ Trong cặp (x, m) có thể gọi x là khóa và m – giá trị của khóa,
- ⊕ Như vậy chỉ cần duyệt tất cả các khóa khác nhau, so sánh các giá trị max đạt được ở mỗi khóa,
- ⊕ Các cặp (x, m) có thể trữ bằng cấu trúc dữ liệu **map**,
- ⊕ Đặc trưng cấu trúc dữ liệu **map**:
 - ✓ Khi truy cập tới khóa không tồn tại hệ thống sẽ **tạo ra phần tử mới** có giá trị bằng 0 ứng với khóa truy cập,
 - ✓ Thông tin được lưu trữ dưới dạng cây nhị phân.

Tổ chức dữ liệu:

- ▣ Mảng **int b[100001]** – lưu giá trị các khóa khác nhau, có thể dùng vector, nhưng sẽ mất nhiều thời gian xử lý hơn,
- ▣ **map<int, int64_t> sm** – lưu trữ khóa của các cặp (x, m) ,
- ▣ Các biến **int mm** lưu trữ số lượng khóa khác nhau, **int64_t res** – lưu kết quả.

Xử lý:

- Tính và lưu khóa, giá trị khóa:

```
for (int i=0; i<n; i++)
{
    fi>>x;
    t=x; while((t&1)==0) t>>=1;
    if(sm.find(t)==sm.end()) b[mm++]=t;
    sm[t]+=x/t;
}
```

Nhận dạng khóa mới

- Duyệt các khóa, tìm giá trị biến đổi lớn nhất:

Tìm 2^t thích hợp

```
for(int i=0;i<mm;++i)
{
    t=sm[b[i]];
    for(int j=60;j>=0;--j) if(t&(t1<<j)){t=t1<<j;break;}
    res=max(res,t*b[i]);
}
```

Giá trị max khi biến đổi cặp dữ liệu

Độ phức tạp của giải thuật: $O(n \ln n)$ vì sử dụng cấu trúc **map**.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "maxnumber."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,mm=0,b[100001],x;
map<int,int64_t> sm;
int64_t res=0,t,t1=1;

int main()
{ clock_t aa=clock();
    fi>>n;
    for (int i = 0; i < n; i++)
    {
        fi>>x;
        t=x; while((t&1)==0)t>>=1;
        if(sm.find(t)==sm.end())b[mm++]=t;
        sm[t]+=x/t;
    }

    for(int i=0;i<mm;++i)
    {
        t=sm[b[i]];
        for(int j=60;j>=0;--j)if(t&(t1<<j)){t=t1<<j;break;}
        res=max(res,t*b[i]);
    }
    fo<<res;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS26. PHẦN THƯỞNG

Tên chương trình: PRIZES.CPP

Trong buổi sinh hoạt toàn trường tuần đầu tiên của học kỳ II có tổ chức một cuộc thi vui với n phần thưởng được trưng bày trên sân khấu. Những người dự thi sẽ bị loại dần cho đến khi còn một người ở vòng cuối. Ở vòng cuối này nếu người dự thi được k điểm với $2 \leq k \leq n$ thì sẽ được chọn một phần thưởng trong số k phần thưởng đầu tiên, nhưng trước đó người dân chương trình sẽ cắt đi một trong số k phần thưởng đầu tiên này theo số ngẫu nhiên trong phạm vi từ 1 đến k hiển thị trên màn hình ở sân khấu, như vậy người thắng cuộc chỉ còn có $k-1$ lựa chọn.

Steve lọt được đến vòng cuối và bắt đầu nhẩm tính giá trị tối đa chắc chắn của phần thưởng trong mọi trường hợp được thưởng. Với Steve, phần thưởng thứ i có giá trị a_i , $i = 1 \div n$.

Dữ liệu: Vào từ file văn bản PRIZES.INP:

- ✚ Dòng đầu tiên chứa một số nguyên n ($2 \leq n \leq 10^5$),
- ✚ Dòng thứ 2 chứa n số nguyên a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản PRIZES.OUT trên một dòng $n-1$ số nguyên, số thứ i tương ứng với giá trị tối đa chắc chắn của phần thưởng trong mọi trường hợp Steve đạt được $i+1$ điểm, $i = 1 \div n-1$.

Ví dụ:

PRIZES.INP
5
1 3 4 2 5

PRIZES.OUT
1 3 3 4



Giải thuật: Các giải thuật tìm số lớn thứ 2 và làm quen với các loại cấu trúc dữ liệu trong C++.

Nhận xét:

- ⊕ Gọi **x** và **y** là số lớn nhất và lớn thứ 2 trong các số **a₁, a₂, ..., a_i**,
- ⊕ Khi xét tiếp **z = a_{i+1}**: cặp giá trị (**x, y**) chỉ thay đổi khi **z > y**,
- ⊕ Như vậy chỉ cần lưu và xử lý 3 số **x, y** và **z** với **z** lần lượt nhận các giá trị của dãy số đã cho,
- ⊕ Để *làm quen* với các cấu trúc dữ liệu khác nhau của hệ thống lập trình có thể dùng **multiset<int> m** hoặc **priority_queue<int> q** để lưu 3 giá trị này:
 - ⊖ Không sử dụng **set** vì các giá trị lưu trữ có thể trùng nhau,
 - ⊖ Các giá trị được lưu theo thứ tự tăng dần,
 - ⊖ Khi nhập và nạp vào giá trị mới, trình tự mốc nối dữ liệu tự động thay đổi, đảm bảo phần tử có giá trị nhỏ nhất đứng đầu danh sách mốc nối,
 - ⊖ Vấn đề còn lại: loại bỏ phần tử đứng đầu và đưa ra phần tử đứng đầu mới,
- ⊕ Chương trình mô phỏng các cấu trúc dữ liệu nói trên sẽ làm việc nhanh hơn việc trực tiếp sử dụng các cấu trúc dữ liệu đó.

Dộ phức tạp của giải thuật: O(n).

Chương trình: Mô phỏng hoạt động nạp và loại bỏ phần tử trong hàng đợi (tập hợp)

```
#include <bits/stdc++.h>
#define NAME "prizes_q."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,x,y=0,z;

int main()
{clock_t aa=clock();
 fi>>n>>x;
 for(int i=1;i<n;++i)
 {
 fi>>z;
 if(y<z) y=z;
 if(x<y) swap(x,y);
 fo<<y<<' ';
 }

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

Chương trình: Phương án sử dụng tập hợp.

```
#include <bits/stdc++.h>
#define NAME "prizes_s."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,x,y=0;
multiset<int> m;

int main()
{clock_t aa=clock();
 fi>>n>>x>>y;
m.insert(x);m.insert(y);
for(int i=2;i<n;++i)
{
    fo<<*m.begin()<<' ';
    fi>>y;
    m.insert(y);
    m.erase(m.begin());
}
fo<<*m.begin()<<' ';
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS27. HÀNH TINH ĐỎ

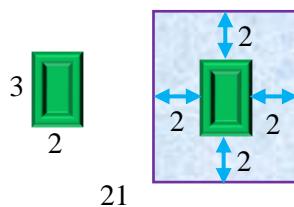
Tên chương trình: MARS.CPP

Để chuẩn bị chinh phục sao Hỏa người ta quyết định xây dựng một cơ sở nghiên cứu khoa học. Nơi đặt cơ sở nghiên cứu là một mảnh đất phẳng hình chữ nhật kích thước $w \times h$, trên đó phải bố trí n mô đun sinh sống và làm việc, mỗi mô đun có dạng hình chữ nhật kích thước $a \times b$. Để tăng độ an toàn người ta viền thêm bên ngoài một lớp bảo vệ độ dày d . Như vậy mỗi mô đun sẽ là một hình chữ nhật kích thước $(a+2d) \times (b+2d)$. Các mô đun phải được đặt theo cùng một hướng như nhau và có cạnh song song với cạnh của mảnh đất nơi chọn làm cơ sở nghiên cứu khoa học.

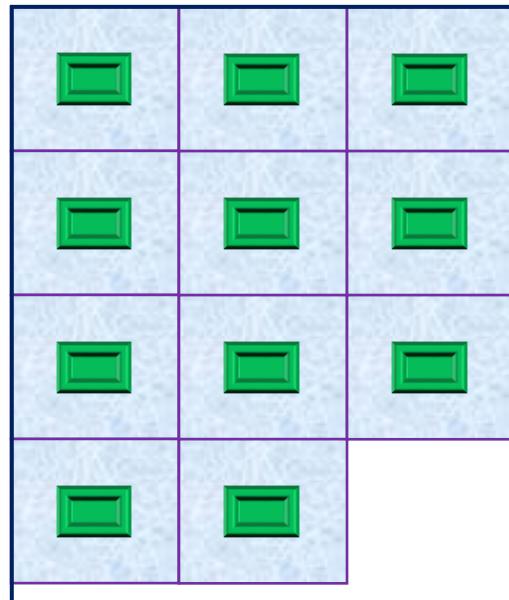
Hãy xác định độ dày d nguyên lớn nhất có thể, biết rằng với $d = 0$ có thể bố trí được n mô đun trong vùng đã xác định.

Dữ liệu: Vào từ file văn bản MARS.INP gồm một dòng chứa 5 số nguyên n, a, b, w và h ($1 \leq n, a, b, w, h \leq 10^{18}$).

Kết quả: Đưa ra file văn bản MARS.OUT một số nguyên – độ dày d lớn nhất tìm được.



21



25

Ví dụ:

MARS.INP
11 2 3 21 25

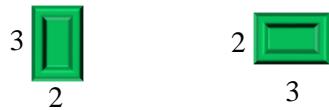
MARS.OUT
2



Giải thuật: Tìm kiếm nhị phân.

Nhận xét:

- Xét riêng 2 trường hợp: không xoay và xoay ngang mô đun sinh sống,



- Ở mỗi trường hợp: bằng phương pháp tìm kiếm nhị phân xác định d lớn nhất có thể,
- Ở các hệ thống lập trình với chế độ ngầm định có bẫy sự kiện overflow (ví dụ Freepascal) cần đưa chỉ thị dịch bỏ bẫy sự kiện nói trên hoặc lưu ý kiểm tra và vòng tránh trong quá trình xử lý.

Độ phức tạp của giải thuật: $O(\ln n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "mars."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int64_t n,a,b,w,h,ans,t;

int64_t calc_aw()
{int64_t l,r,m,aw,bh;
 l=0; r=min(w,h)+1;
 while(l+1<r)
 {
     m=(l+r)/2;
     aw=w/(a+2*m); bh=h/(b+2*m);
     if(aw*bh>=n) l=m; else r=m;
 }
 return l;
}

int64_t calc_ah()
{int64_t l,r,m,ah,bw;
 l=0; r=min(w,h)+1;
 while(l+1<r)
 {
     m=(l+r)/2;
     ah=h/(a+2*m); bw=w/(b+2*m);
     if(ah*bw>=n) l=m; else r=m;
 }
 return l;
}

int main()
{clock_t aa=clock();
 fi>>n>>a>>b>>w>>h;
 ans=calc_aw();
 t=calc_ah();
 if(ans<t) ans=t;
 fo<<ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS28. XÂU ĐỘC ĐÁO

Tên chương trình: STRANGE.CPP

Xét xâu s chỉ chứa các ký tự la tinh thường.

Gọi một hoặc nhiều ký tự liên tiếp nhau của s là *xâu con*. Gọi $W(s)$ là tập các xâu con khác nhau từng đôi một của s . Ví dụ, với $s = "abba"$, $W(s) = \{"a", "ab", "abb", "abba", "b", "bb", "bba", "ba"\}$.

Dãy con của s là xâu nhận được từ s bằng cách xóa một số (có thể là 0) ký tự của s . Gọi $Y(s)$ là tập các dãy con khác nhau từng đôi một nhận được từ s . Ví dụ, với s nêu trên $Y(s) = \{"a", "b", "ab", "aa", "bb", "ba", "abb", "aba", "bba", "abba"\}$.

Xâu s được gọi là *độc đáo* nếu $W(s) = Y(s)$.

$s = "abba"$ không phải là xâu độc đáo, còn $"abb"$ – là xâu độc đáo.

Độ độc đáo của một xâu là số lượng xâu con độc đáo khác nhau của nó. Ví dụ, độ độc đáo của $"abba"$ là 7.

Cho xâu s . Hãy xác định độ độc đáo của s .

Dữ liệu: Vào từ file văn bản STRANGE.INP gồm một dòng chứa xâu s có độ dài không vượt quá 2×10^5 .

Kết quả: Đưa ra file văn bản STRANGE.OUT một số nguyên – độ độc đáo của xâu đã cho.

Ví dụ:

STRANGE.INP	STRANGE.OUT
abba	7



Giải thuật: Đếm cấu hình.

Nhận xét:

Xâu **z** là độc đáo khi thỏa mãn các tính chất:

- ✚ Chứa không quá 2 ký tự khác nhau,
- ✚ Tất cả các ký tự giống nhau đứng liên tiếp cạnh nhau.

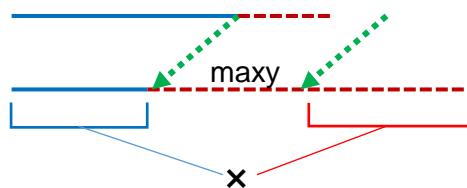
Thật vậy, nếu ký tự **a** xuất hiện **k** lần trong **z**, nhưng giữa 2 ký tự **a** nào đó có ký tự **b** thì xâu **k** ký tự **a** là dãy con của **z** nhưng không phải là xâu con.

Nếu **z** chứa **k** ký tự **a** đứng liên tiếp nhau, sau đó **m** ký tự **b** liên tiếp nhau thì cả **w(z)** và **Y(z)** đều chứa các xâu phần đầu có không quá **k** ký tự **a** và phần còn lại – chứa không quá **m** ký tự **b**.

Như vậy, để xác định độ độc đáo của một xâu cần tìm tất cả các đoạn thỏa mãn 2 tính chất nêu trên và đưa bài toán về việc *xét độc lập các đoạn* tìm được chỉ chứa cặp ký tự (**a, b**).

Giả thiết tìm được **m** đoạn, đoạn thứ **i** chứa **p_i** ký tự **a** ở đầu và sau đó là **q_i** ký tự **b**. Như vậy đoạn này tương ứng với cặp giá trị (**p_i, q_i**). Độ độc đáo của các xâu con chứa cả 2 ký tự **a** và **b** là số lượng cặp (**u, v**) thỏa mãn $1 \leq u \leq p_i, 1 \leq v \leq q_i$.

Số lượng cặp cần tìm phụ thuộc chủ yếu vào giá trị lớn nhất của **p_i** và giá trị lớn nhất của **q_i**. Sắp xếp (**p_i, q_i**) theo thứ tự giảm dần của **p_i**, bổ sung vào kết quả tích 2 số trong cặp đầu tiên, sau đó bổ sung tiếp các cặp còn thiếu dựa vào các giá trị **q_i** lớn hơn gấp trong quá trình duyệt dãy cặp số đã sắp xếp. Gọi **maxy** là giá trị lớn nhất của số thứ 2 trong các cặp đã duyệt. Nếu gặp một giá trị thứ 2 **q_j** lớn hơn thì bổ sung tiếp vào kết quả các cặp số (**u, v**) có **v** chạy từ **maxy+1** tới **q_j** và cập nhật lại **maxy**.



Trong kết quả còn có số lượng các xâu con chỉ chứa một ký tự một loại. Các giá trị cần bổ sung vào kết quả là độ dài lớn nhất các ký tự liên tiếp giống nhau. Có bao nhiêu ký tự khác nhau trong xâu ban đầu thì cần bổ sung bấy nhiêu lần.

Tổ chức dữ liệu:

vector<int> one(26) – lưu độ dài lớn nhất xâu con liên tục chỉ chứa ký tự một loại,

vector<pair<char, int>> a – lưu độ dài các đoạn chứa ký tự giống nhau,

`vector<pair<int, int> > two[26][26]` – lưu thông tin về các cặp (p_i, q_i) cho từng cặp 2 ký tự khác nhau.

Xử lý:

Tính độ dài các đoạn chứa ký tự giống nhau:

```
int n = s.length();
a.push_back(make_pair(s[0], 1));
for (int i = 1; i < n; i++)
    if (a.back().first == s[i]) a.back().second++;
    else a.push_back(make_pair(s[i], 1));
```

Xác định độ dài lớn các ký tự liên tiếp giống nhau cho từng loại ký tự:

```
for (int i = 0; i < a.size(); i++)
{
    int pos = a[i].first - 'a';
    one[pos] = max(one[pos], a[i].second);
}
```

Tính và phân loại (p_i, q_i) theo từng cặp ký tự khác nhau:

```
for (int i = 1; i < a.size(); i++)
    two[a[i-1].first-'a'][a[i].first-
    'a'].push_back(make_pair(a[i-1].second, a[i].second));
```

Duyệt tất cả các cặp ký tự khác nhau và tính theo giải thuật nêu trên, lưu ý bổ sung vào kết quả các độ dài lớn nhất xâu con chỉ chứa một loại ký tự.

Sắp xếp các cặp (p_i, q_i) phục vụ duyệt p_i theo thứ tự giảm dần: có thể chỉ ra tiêu chuẩn sắp xếp (hàm `greater()`), nhưng cũng có thể thực hiện theo cách sau:

```
sort(two[i][j].begin(), two[i][j].end());
reverse(two[i][j].begin(), two[i][j].end());
```

Độ phức tạp của giải thuật: Xấp xỉ $O(n)$, trong đó n – độ dài xâu s .

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "strange."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s;
vector<pair<char, int> > a;
vector<int> one(26);
vector<pair<int, int> > two[26][26];
int64_t ans = 0;

int main()
{clock_t aa=clock();
 fi >> s;
 int n = s.length();
 a.push_back(make_pair(s[0], 1));
 for (int i = 1; i < n; i++)
 if (a.back().first == s[i]) a.back().second++;
 else a.push_back(make_pair(s[i], 1));

 for (int i = 0; i < a.size(); i++)
 {
 int pos = a[i].first - 'a';
 one[pos] = max(one[pos], a[i].second);
 }

 for (int i = 1; i < a.size(); i++)
 two[a[i-1].first-'a'][a[i].first-
 'a'].push_back(make_pair(a[i - 1].second, a[i].second));

 for (int i=0;i<26;++i)ans += one[i];

 for (int i = 0; i < 26; i++)
 for (int j = 0; j < 26; j++)
 {
 sort(two[i][j].begin(), two[i][j].end());
 reverse(two[i][j].begin(), two[i][j].end());
 if (two[i][j].size() == 0) continue;
 ans += (int64_t)two[i][j][0].first*two[i][j][0].second;
 int64_t maxy = two[i][j][0].second;
 for (int k = 1; k < two[i][j].size(); k++)
 if (two[i][j][k].second > maxy)
 {
 ans += (two[i][j][k].second-maxy)* two[i][j][k].first;
 maxy = two[i][j][k].second;
 }
 }

 fo << ans;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double) (bb-aa)/1000<<" sec";
}
```

VS29. DU LỊCH BẰNG TÀU HỎA

Tên chương trình: TRAINS.CPP

Để thu hút khách ngành đường sắt tổ chức một tuyến du lịch bằng tàu hỏa. Có thể coi tuyến du lịch này như một đường thẳng. Tuyến có n ga đánh số từ 1 đến n , tàu chạy từ ga 1 đến ga n . Trên tàu có k ghế, đánh số từ 1 đến k . Vé được bán qua mạng.

Steve được nghỉ hè và quyết định thực hiện chuyến đi du lịch trong đó có một đoạn sẽ đi bằng chuyến tàu du lịch nói trên. Steve vạch ra q phương án thực hiện, ở phương án thứ i sẽ đi bằng tàu hỏa từ ga f_i đến ga d_i , $i = 1 \dots q$. Thông tin trên mạng cho thấy có m vé đã được bán, vé thứ j khách mua ngồi ở ghế a_j , đi từ ga s_j đến ga t_j , $j = 1 \dots m$. Ở mỗi phương án du lịch của mình, Steve cố gắng tìm cách mua một vé cho suốt chặng đường trên tàu, nhưng nếu điều đó là không được thì có thể mua vài vé và chuyển chỗ trong chặng đường. Chuyển chỗ là bất tiện nên số vé mua phải càng ít càng tốt.

Với mỗi phương án du lịch hãy xác định số vé tối thiểu cần mua. Nếu không có cách nào mua vé thì đưa ra số -1.

Dữ liệu: Vào từ file văn bản TRAINS.INP:

- ✚ Dòng đầu tiên chứa 3 số nguyên n , m và k ($2 \leq n \leq 2 \times 10^5$, $0 \leq m \leq 2 \times 10^5$, $1 \leq k \leq 2 \times 10^5$),
- ✚ Dòng thứ j trong m dòng sau chứa 3 số nguyên s_j , t_j và a_j ($1 \leq s_j < t_j \leq n$, $1 \leq a_j \leq k$),
- ✚ Dòng tiếp theo chứa số nguyên q ($1 \leq q \leq 2 \times 10^5$),
- ✚ Dòng thứ i trong q dòng sau chứa 2 số nguyên f_i và d_i ($1 \leq f_i < d_i \leq n$).

Kết quả: Đưa ra file văn bản TRAINS.OUT q số nguyên xác định số vé tối thiểu cần mua cho các phương án, mỗi số trên một dòng.

Ví dụ:

TRAIN.S.INP	TRAIN.S.OUT
5 4 3	
1 4 1	
2 5 3	
2 3 2	
4 5 2	
3	
1 5	
3 5	
4 5	
	-1
	2
	1



Giải thuật: Tổ chức dữ liệu, quảng lý khoảng, tiệm cận nhị phân.

Nhận xét:

- ⊕ Các vé đã bán tạo những khoảng bận và trống của các chỗ ngồi,
- ⊕ Để lên phương án mua vé cần quản lý các khoảng trống liên tục của từng chỗ ngồi ,
- ⊕ Với các khoảng trống có giao khác rỗng chỉ cần phân biệt khoảng trống ứng với ghế khác nhau, không cần xác định rõ là ghế nào,
- ⊕ Hướng giải bài toán:
 - ▣ Xây dựng mảng **to_i** cho biết từ ga **i** và chỉ mua một vé ta có tới ga nào xa nhất, **to_i** = -1 nếu không thể có cách mua vé rời khỏi ga,
 - ▣ Xử lý truy vấn:
 - ❖ Xây dựng mảng chứa giá trị **binup_{j,i}** cho biết ga xa nhất có thể tới từ ga **i** nếu mua không quá 2^j vé,
 - ❖ Gọi **ans** là kết quả cần tìm, sử dụng mảng nói trên để tìm các bít bằng 1 của **ans**.

Tổ chức dữ liệu:

- ▣ **vector<pair<int, int> > poss** – chứa thông tin cặp ga của ghế đã bán,
- ▣ **vector<pair<int, int> >free_pos** – chứa thông tin các cặp ga còn trống ứng với ghế đã bán,
- ▣ **vector<int> next_pos** – ga gần nhất tiếp theo xuất hiện ghế trống mới,
- ▣ **vector<bool>used** – đánh dấu chỗ trống,
- ▣ **vector<vector<int> >binup** – chứa các giá trị **binup_{j,i}** đã nói ở trên,
- ▣ **vector<int> to** – xác định ga xa nhất có thể tới bắt đầu từ ga **i** *với chỉ một vé* và nhận giá trị bằng số thứ tự của khoảng trống trong **free_pos** chứa **i** và ga xa nhất có thể tới,
- ▣ **vector<tuple<int,int,int> >scanline** lưu trữ các nhóm 3 dữ liệu (**x, type, id**), trong đó **x** – ga, **type** = 0 ứng với trường hợp **x** là ga đầu của vé được bán, **type** = 1 ứng với trường hợp **x** là ga cuối của vé được bán, **id** – số thứ tự của cặp dữ liệu trong **free_pos** từ đó thông tin được trích ra ([cấu trúc dữ liệu kiểu tuple được vào từ C++11](#)),
- ▣ **set<pair<int,int> >now** chứa cặp dữ liệu (**x, id**) trong đó **x** – ga cuối ở đó ghế tương ứng với bản ghi **id** trong **free_pos** là trống.

Xử lý: Để thuận tiện làm việc với vector, ga và ghế được đánh số lại bắt đầu từ 0.

Các từ khóa sử dụng nhiều được viết tắt theo định nghĩa ở đầu chương trình.

Lưu thông tin vé bán theo từng ghế:

```
fi>>n>>m>>k;
for (int i = 0; i < m; ++i)
{
    int a, b, c;
    fi>>a>>b>>c;
    --a;--b;--c;
    poss[c].puba(mapa(a, b));
}
```

Với mỗi ghế: Sắp xếp các cặp ga (đi, đến) theo ga đi và tạo mảng quản lý các đoạn ga trống ứng với ghế đó:

Một cách tổ chức chương trình trong C++11

Bật option C++11/ISO

```
for (int i = 0; i < k; ++i)
{
    sort(bend(poss[i]));
    int prev = 0;
    for (auto p: poss[i])
    {
        if(prev < p.ff) free_pos.puba(mapa(prev, p.ff));
        prev = p.ss;
    }
    if (prev < n - 1) free_pos.puba(mapa(prev, n - 1));
}
```

Tạo mảng quản lý có thay đổi trạng thái ghế và sắp xếp theo ga:

Tạo nhóm 3

```
vector<tuple<int, int, int>> scanline;
for (int i = 0; i < szof(free_pos); ++i)
{
    scanline.puba(make_tuple(free_pos[i].ff, 0, i));
    scanline.puba(make_tuple(free_pos[i].ss, 1, i));
}
sort(bend(scanline));
```

Xác định đồng thời:

- ✓ **to_i** – ga xa nhất tới được, xuất phát từ ga i và chỉ với một vé,
- ✓ **next_pos_i** - ga gần nhất tiếp theo xuất hiện ghế trống mới.

Sử dụng tập hợp **set<pair<int, int>> now**, truy nhập vào **now** *từ cuối về đầu* để lấy ra ga cuối cùng (ở thời điểm xét) có ghế trống ứng với vé đang tham gia xử lý, duyệt các bản ghi của **scanline**, nếu gặp bản ghi ứng với ga đầu của vé – nạp ga cuối tương ứng và tập hợp, trong trường hợp ngược lại – xóa điểm

cuối đó khởi tạo hợp (vì tiếp theo nó sẽ không được tham gia vào xử lý), nếu tập **now** khác rỗng thì ga gần nhất tiếp theo xuất hiện ghê trống mới là ga ứng với bản ghi cuối của tập của cùng một bản ghi trong mảng **free_pos**.

Khi ga đang xét nhỏ hơn ga đọc được thì ga tiếp theo xác định trong tập **now**, nếu tập là rỗng – không đi tiếp được!

Lưu ý trường hợp khi đã đọc hết **scanline** nhưng vẫn chưa xử lý đến ga cuối cùng thì với những ga còn lại – xác định thông tin trong tập **now**.

```

int c = 0;
for (int i = 0; i < szof(scanline); ++i)
{
    int x, type, id;
    tie(x, type, id) = scanline[i];
    while (c < x)
    {
        if (szof(now)) to[c] = now.rbegin() -> ss; else to[c] = -1;
        ++c;
    }
    if (type == 0) now.insert(mapa(free_pos[id].ss, id));
    else
    {
        now.erase(mapa(free_pos[id].ss, id));
        if (szof(now)) next_pos[id] = now.rbegin() -> ss;
        else next_pos[id] = id;
    }
}
while (c < n)
{
    if (szof(now)) to[c] = now.rbegin() -> ss;
    else to[c] = -1;
    ++c;
}

```

Trải dữ liệu từ nhóm ra
các biến đơn (C++11)

Tính bảng giá trị **binup_{j,i}**:

Chỉ cần làm việc với $j < \text{bd}$, trong đó **bd** là giá trị nhỏ nhất thỏa mãn $2^{\text{bd}} \geq n$,

Số lượng dòng: bằng số khoảng trống (kích thước của **free_pos**),

Hàm **topsort(int v)** chuẩn bị các giá trị cho bảng cần tính,

Xét truy vấn đi từ l đến r:

Nếu bằng 1 vé không thể rời được l → ans = -1,

Trong trường hợp ngược lại: bằng 1 vé đi được tới ga xác định bởi khoảng pos và tiếp theo, với không quá 2^j vé đi được tới ga xác định bởi temp=binup[j][pos].

Nếu $\text{ga free_pos}[\text{temp}] < r$ thì tăng ans thêm 2^j và tiếp tục xét tiếp từ đó với j nhỏ hơn. Nếu đã xét với mọi $j \geq 0$ vẫn không tới được r : $\text{ans} = -1$.

Trong trường hợp tới được: tăng ans lên 1 và đưa ra kết quả (ban đầu sđã sử dụng một vé).

Độ phức tạp của giải thuật: nhỏ hơn $O(n^2)$.

Lưu ý: Để tiết kiệm bộ nhớ, phù hợp với ràng buộc bộ nhớ cho phép, nên nêu lại yêu cầu tổ chức bộ nhớ phù hợp kích thước vừa đủ ở từng khâu xử lý. Điều này gần như không ảnh hưởng đến thời gian thực hiện chương trình và *không đòi hỏi người lập trình phải tự lưu kích thước phục vụ cho xử lý*.

```

#include "bits/stdc++.h"
#define puba push_back
#define mapa make_pair
#define ff first
#define ss second
#define bend(_x) (_x).begin(), (_x).end()
#define szof(_x) ((int) (_x).size())
#define NAME "trains."

using namespace std;
typedef long long ll;
typedef pair <int, int> pii;
const int MAXN = 2e5 + 5;
int bd = 30;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");

int n, m, k;
vector <pii> poss[MAXN];
vector <pii> free_pos;
vector <int> next_pos;

vector <bool> used;
vector <vector <int>> binup;

void topsort(int v)
{
    used[v] = true;
    if (!used[next_pos[v]]) topsort(next_pos[v]);
    binup[0][v] = next_pos[v];
    for (int i = 1; i < szof(binup); ++i)
        binup[i][v] = binup[i - 1][binup[i - 1][v]];
}

int main()
{clock_t aa=clock();
fi>>n>>m>>k;
for (int i = 0; i < m; ++i)
{
    int a, b, c;
    fi>>a>>b>>c;
    --a;--b;--c;
    poss[c].puba(mapa(a, b));
}
for (int i = 0; i < k; ++i)
{
    sort(bend(poss[i]));
    int prev = 0;
    for (auto p: poss[i])
    {
        if (prev < p.ff) free_pos.puba(mapa(prev, p.ff));
        prev = p.ss;
    }
    if (prev < n - 1) free_pos.puba(mapa(prev, n - 1));
}

vector <tuple<int, int, int>> scanline;

```

```

for (int i = 0; i < szof(free_pos); ++i)
{
    scanline.puba(make_tuple(free_pos[i].ff, 0, i));
    scanline.puba(make_tuple(free_pos[i].ss, 1, i));
}
sort(bend(scanline));

set <pii> now;
next_pos.resize(szof(free_pos));
vector <int> to(n);
int c = 0;
for (int i = 0; i < szof(scanline); ++i)
{
    int x, type, id;
    tie(x, type, id) = scanline[i];
    while (c < x)
    {
        if (szof(now))to[c] = now.rbegin() -> ss; else to[c] = -1;
        ++c;
    }
    if (type == 0)now.insert(mapa(free_pos[id].ss, id));
    else
    {
        now.erase(mapa(free_pos[id].ss, id));
        if (szof(now))next_pos[id] = now.rbegin() -> ss;
        else next_pos[id] = id;
    }
}
while (c < n)
{
    if (szof(now))to[c] = now.rbegin() -> ss;
    else to[c] = -1;
    ++c;
}
bd = 0;
for (int j = 1; j <= szof(free_pos); j *= 2, ++bd);
binup.resize(bd);
for (auto& v: binup)v.resize(szof(free_pos));
used.resize(szof(free_pos));
for (int i = 0; i < szof(free_pos); ++i)
    if (!used[i])topsort(i);

int q;
fi>>q;
for (int i = 0; i < q; ++i) {
    int l, r;
    fi>>l>>r;
    --l;--r;
    int ans = 0;
    int pos = to[l];
    if (pos == -1)
    {
        fo << "-1\n";
        continue;
    }
    for (int j = bd - 1; j >= 0; --j)

```

```

{
    int tmp = binup[j][pos];
    if (free_pos[tmp].ss < r)
    {
        ans += 1 << j;
        pos = tmp;
    }
}
if (free_pos[pos].ss < r)
{
    pos = binup[0][pos];
    ++ans;
}
if (free_pos[pos].ss < r) fo << "-1\n";
else fo << ans + 1 << '\n';
}

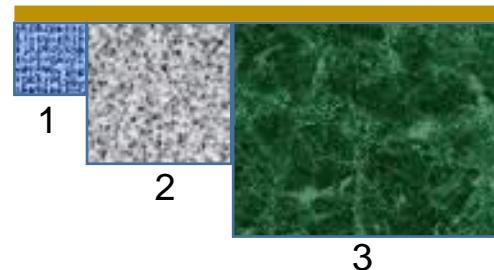
clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}

```

VS30. KHU BẢO TỒN

Tên chương trình: FOREST.CPP

Đường cao tốc độ dài n km chạy thẳng chia vùng đất thành 2 phần, một bên là đất đai trồng trọt còn bên kia là cánh rừng bạt ngàn rộng lớn. Để bảo vệ thiên nhiên và khai thác hợp lý tài nguyên rừng người ta chia cho 3 công ty quản lý và khai thác 3 lô rừng mỗi lô có hình vuông, cạnh của các lô tương ứng là a , b và c . Các công ty có tiềm lực khác nhau, công ty có tiềm lực mạnh sẽ nhận được lô rộng hơn vì vậy điều kiện chia lô là $a < b < c$. Ngoài ra các lô này phải kè với toàn bộ con đường, tức là $a+b+c = n$. Phần rừng còn lại sẽ được dùng làm khu bảo tồn sinh thái.



Với mục đích có diện tích bảo tồn lớn người ta tìm cách xác định kích thước các lô rừng sao cho tổng diện tích rừng được chia cho các công ty khai thác là nhỏ nhất.

Hãy xác định kích thước các lô và đưa ra theo trình tự tăng dần.

Dữ liệu: Vào từ file văn bản FOREST.INP gồm một dòng chứa số nguyên n ($6 \leq n \leq 10^9$).

Kết quả: Đưa ra file văn bản FOREST.OUT một dòng chứa 3 số nguyên – kích thước các lô.

Ví dụ:

FOREST.INP
6

FOREST.OUT
1 2 3



Giải thuật: Duyệt có chọn lọc.

Nhận xét:

Trên quan điểm toán học ta có bài toán *quy hoạch nguyên cầu phương*:

$$a^2 + b^2 + c^2 \rightarrow \min$$

với các ràng buộc: $a + b + c = n$, $0 < a < b < c$, a, b, c – nguyên.

Các giá trị cần tìm phải càng gần $n/3$ bao nhiêu càng tốt bấy nhiêu.

Nếu đã chọn được một số, ví dụ c , thì 2 số còn lại phải thỏa mãn $a + b = m = n - c$ và $b - a \leq 2$.

Thật vậy, nếu $b - a > 2$ thì $a + 1 \neq b - 1$. Ta có

$$\begin{aligned} (a+1)^2 + (b-1)^2 &= a^2 + 2a + 1 + b^2 - 2b + 1 \\ &= a^2 + b^2 + 2(a-b) + 2 \\ &< a^2 + b^2 - 4 + 2 \\ &< a^2 + b^2 \end{aligned}$$

Và ta có thể chọn a, b mới với tổng bình phương nhỏ hơn!

Như vậy, với m lẻ ta cần chọn $a = (m-1)/2$, $b = (m+1)/2$, còn khi m chẵn: $a = m/2 - 1$ và $b = m/2 + 1$.

Để giải bài toán ta cần xuất phát từ giá trị $a = n/3$, tính các giá trị b, c tương ứng và kiểm tra điều kiện $a < b < c$. Nếu điều này được thỏa mãn thì các giá trị tìm được là nghiệm bài toán, trong trường hợp ngược lại – giảm a và tính b, c mới.

Độ phức tạp của giải thuật: không vượt quá $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "forest."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,a,b,c;

int main()
{clock_t aa=clock();
 fi>>n;
 a=n/3;
 while(a>0)
 {
     b = (n - a) / 2 - ((n - a) % 2 == 0);
     c = n - a - b;
     if (a < b && b < c)break;
     --a;
 }
 fo<<a<<' '<<b<<' '<<c;
 clock_t bb=clock();
 fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS31. CỜ VÂY

Tên chương trình: DRAUGHTS.CPP

Steve là trọng tài giải vô địch cờ vây của trường. Mỗi ván có 3 người chơi. Điểm số thu được của mỗi người là một số nguyên. Nếu khi ván cờ kết thúc người nhất được a điểm, người thứ 2 được b điểm và người thứ 3 được c điểm thì tỷ số của ván đó là $a:b:c$. Theo luật chơi điểm số giữa 2 người bất kỳ không chênh lệch quá k lần.

Steve chuẩn bị sẵn n biến số, biến thứ i ghi số x_i , $i = 1 \div n$ để gắn các biến số điểm lên bảng công bố kết quả. Như vậy, nếu Steve chuẩn bị 5 tấm biển với các số 1, 1, 2, 2, 3 và $k=2$ thì có thể công bố 9 kết quả khác nhau: 1:1:2, 1:2:1, 2:1:1, 1:2:2, 2:1:2, 2:2:1, 2:2:3, 2:3:2, 3:2:2.

Cho n , k và các số x_1, x_2, \dots, x_n . Hãy xác định số lượng kết quả khác nhau có thể công bố.

Dữ liệu: Vào từ file văn bản DRAUGHTS.INP:

- ➡ Dòng đầu tiên chứa 2 số nguyên n và k ($3 \leq n \leq 10^5$, $1 \leq k \leq 10^9$),
- ➡ Dòng thứ 2 chứa n số nguyên x_1, x_2, \dots, x_n ($1 \leq x_i \leq 10^9$, $i = 1 \div n$).

Kết quả: Đưa ra file văn bản DRAUGHTS.OUT một số nguyên – số lượng kết quả khác nhau có thể công bố.

Ví dụ:

DRAUGHTS.INP
5 2
1 1 2 2 3

DRAUGHTS.OUT
9



Roi_Reg20160201 F

Giải thuật: Thống kê số lượng cấu hình.

Nhận xét:

Có 3 nhóm cấu hình:

- ✚ Điểm số 3 người bằng nhau,
- ✚ Điểm số 3 người khác nhau từng đôi một,
- ✚ Có 2 người điểm số giống nhau và khác điểm số người còn lại.

Để thuận tiện phân tích và xử lý dữ liệu vào cần sắp xếp theo thứ tự tăng dần và lưu trữ dưới dạng nén bằng mảng a kiểu cặp số nguyên, mỗi cặp chứa giá trị biến số và tần số xuất hiện của nó trong dãy ban đầu.

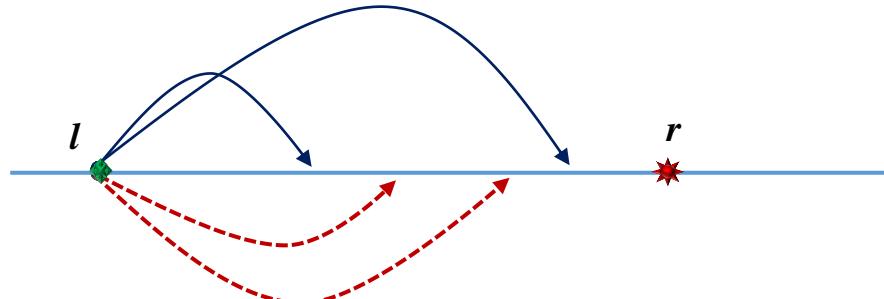
Tính số lượng cấu hình nhóm 1: Số lượng cấu hình nhóm 1 bằng số lượng cặp trong a có tần số lớn hơn hoặc bằng 3.

Tính số lượng cấu hình nhóm 2:

Với mỗi $a[1].first$ tìm r lớn nhất thỏa mãn

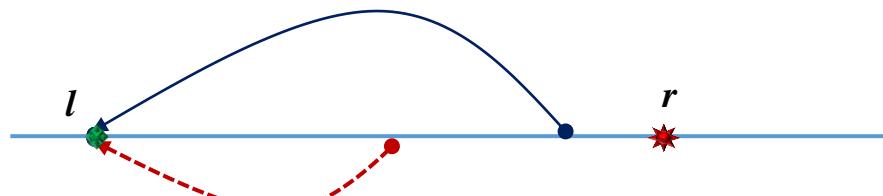
$$a[1].first * k \leq a[r].first,$$

Mỗi biến số $a[1].first$ có thể kết hợp với 2 biến số trong phạm vi từ phần tử $l+1$ đến r để có một bảng kết quả, 3 biến số có thể hoán vị cho nhau và tạo thành **6 bảng** kết quả khác nhau có thể công bố.



Tính số lượng cấu hình nhóm 2:

Nếu biến số $a[1].first$ có số lượng lớn hơn hoặc bằng 2 thì có thể dùng 2 biến số đó kết hợp với một biến số trong khoảng từ $l+1$ đến r và tạo ra **3 bảng** kết quả mới.



Tổ chức dữ liệu:

Mảng nguyên x : chứa dữ liệu vào,

Mảng a kiểu cặp dữ liệu nguyên chứa thông tin về tần số xuất hiện các biến với số khác nhau,

Biến kết quả và một số biến trung gian: cần khai báo kiểu `int64_t`!

Xử lý: Chỉ cần duyệt mảng `a` một lần, điều chỉnh `l`, `r` song song với các tham số cần thiết còn lại.

Độ phức tạp của giải thuật: $O(n \ln n)$.

Lưu ý: trong chương trình có phép thay đổi giá trị `n`. Đây là ví dụ minh họa điều này không làm ảnh hưởng tới bộ nhớ động đã phân phối theo giá trị cũ của `n`.

Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
int n, k;
int64_t t=0, t2=0, ans = 0;
vector<pair<int, int> > a;
int main()
{
    clock_t aa=clock();
    ifstream fi("draughts.inp");
    ofstream fo("draughts.out");
    fi >> n >> k;
    vector<int> x(n);
    for (int i = 0; i < n; i++) fi >> x[i];
    sort(x.begin(), x.end());
    a.push_back(make_pair(x[0], 1));
    for (int i = 1; i < n; i++)
    {
        if (a.back().first == x[i]) a.back().second++;
        else a.push_back(make_pair(x[i], 1));
    }
    n = a.size();
    int r = 1;
    for (int l = 0; l < n; l++)
    {
        while (r < n && (int64_t)a[l].first * k >= a[r].first)
        {
            if (a[r].second > 1) t2++;
            t++;
            r++;
        }
        if (a[l].second > 2) ans++;
        if (a[l].second >= 2) ans += 3 * t;
        ans += 3 * t2 + 6 * t * (t - 1) / 2;
        t--;
        if (a[l + 1].second > 1) t2--;
    }
    fo << ans;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS32. SỐ DỄ CHỊU

Tên chương trình: PLEASANT.CPP

Jimmy nổi tiếng trong lớp là một người gon gàng, ngăn nắp. Đồ đạc trong phòng của Jimmy cũng được phân loại, sắp xếp có trật tự. Chính vì vậy, khi làm các bài tập số học, Jimmy rất không hài lòng nếu gặp các số mà các chữ số không theo một trình tự nào, lúc tăng lúc giảm. Jimmy chỉ thích các số có các chữ số xuất hiện theo trình tự không giảm, ví dụ 1111, 123, 88999, . . . và gọi đó là những số dễ chịu.

Jimmy cũng hiểu rằng trong cuộc sống những điều dễ chịu không nhiều và trong thế giới số cũng vậy! Để kiểm tra, so sánh xem thế giới thực và thế giới số nơi nào tỷ lệ điều dễ chịu cao hơn Jimmy bắt tay vào việc tính số dễ chịu xuất hiện trong đoạn $[a, b]$.

Cho 2 số nguyên a và b ($0 < a \leq b \leq 10^{100}$). Hãy xác định số lượng số dễ chịu trong đoạn $[a, b]$. Kết quả có thể rất lớn vì vậy chỉ cần đưa ra theo mô đun $10^9 + 7$.

Dữ liệu: Vào từ file văn bản PLEASANT.INP:

- ✚ Dòng đầu tiên chứa số nguyên a ,
- ✚ Dòng thứ 2 chứa số nguyên b .

Kết quả: Đưa ra file văn bản PLEASANT.OUT một số nguyên – số số lượng số dễ chịu tìm được theo mô đun $10^9 + 7$.

Ví dụ:

PLEASANT.INP
1
100

PLEASANT.OUT
54



Giải thuật: Tổng tiền tố, quy hoạch động, bảng phương án.

Nhận xét:

Cần tính số lượng số dễ chịu trong một khoảng liên tục vì vậy cần phải dựa trên hàm $f(x)$ cho số lượng các số dễ chịu trong đoạn từ 1 đến x ,

Về nguyên tắc, kết quả cần tìm sẽ là $f(b) - f(a-1)$, tuy vậy ta sẽ xét trên đoạn tương đương để tránh việc phải tính $a-1$,

Việc tích lũy kết quả được thực hiện theo mô đun 10^9+7 vì vậy không cần xử lý số lớn,

Cơ sở để tính giá trị hàm $f(x)$ là bảng giá trị $d[10][101]$, trong đó $d_{i,j}$ là số lượng số dễ chịu có đúng j chữ số, bắt đầu bằng chữ số i , $i = 1 \div 9$, $j = 1 \div 100$, $d_{0,j}$ là tổng số lượng các số dễ chịu có đúng j chữ số có nghĩa,

Để thuận tiện ta tổ chức thêm mảng $tf[101]$, trong đó tf_j lưu trữ số lượng số dễ chịu có không quá j chữ số có nghĩa,

Dễ dàng thấy rằng:

$$d_{9,j} = 1 \quad \forall j,$$

$$d_{i,j} = d_{i,j-1} + d_{i+1,j}, \quad i = 8 \div 1, \quad j = 2, 3, 4, \dots$$

Cách tính giá trị $f(x)$ – số lượng số dễ chịu trong đoạn $[1, x]$:

	1	2	3	4
0	9	45	165	495
1	1	9	45	165
2	1	8	36	120
3	1	7	28	84
4	1	6	21	56
5	1	5	15	35
6	1	4	10	20
7	1	3	6	10
8	1	2	3	4
9	1	1	1	1

Phần đầu của các bảng d và tf

tf	9	54	219	714
----	---	----	-----	-----

Việc tính toán sẽ đơn giản hơn nhiều nếu x là một số dễ chịu. Trong trường hợp x không phải là số dễ chịu ta có thể dễ dàng tìm y là số dễ chịu nhỏ nhất lớn hơn x , khi đó $f(x) = f(y) - 1$.

Tổ chức dữ liệu:

Các mảng `int d[10][101] và tf[101]` với ý nghĩa sử dụng đã nêu ở trên.

Xử lý:

Lưu trữ dữ liệu vào dưới dạng xâu và đảo ngược vị trí các ký tự để đảm bảo tương ứng ký tự thứ i trong xâu có trọng số 10^i ,

Tính bảng d và tf để tra cứu thông tin phục vụ cho việc tính $f(y)$:

- Chuẩn giá trị đầu cho mỗi cột: $d[9][j]=1$,

- Tính $d[i][j]$ với $i = 8 \div 1$,
- Tính $d[0][j]$,
- Tính giá trị tích lũy $tf[j] = tf[j-1] + d[0][j]$, $j = 1, 2, 3, \dots$

Tổ chức hàm `int norm_s(string &y, int n)` biến đổi xâu **y** thành xâu tương ứng với số dẽ chịu nhỏ nhất lớn hơn hoặc bằng **y**. Hàm trả về giá trị 1 nếu **y** thay đổi và 0 trong trường hợp ngược lại,

Tính giá trị hàm **f(y)**:

- Các ký tự của **y** được lưu trữ theo trình tự y_0 – tương ứng với hàng đơn vị, y_1 – hàng chục, y_2 – hàng trăm, ...
- Khi chuyển từ ký tự $y[j-1]$ sang ký tự $y[j]$ nếu có chênh lệch giá trị thì phải tích lũy tác động của chênh lệch dựa vào cột $j-1$ của bảng **d**,
- Ở ký tự cuối cùng của xâu: tích lũy tác động của chênh lệch từ ký tự đó về 1,
- Nếu độ dài của xâu là **n** và **n** > 1 thì cần tính thêm tổng số các số dẽ chịu có số lượng chữ số không vượt quá **n-1** (cộng thêm tf_{n-2}).

Dẫn xuất kết quả:

- Khi tính **f(y)** đối với **a** theo cách trên kết quả lớn hơn 1 so với giá trị thực của **f(a-1)**,
- Giá trị **f(y)** đối với **b** có thể trùng hoặn lớn hơn 1 so với **f(b)**,
- Hiệu 2 giá trị tính được có thể âm, trong trường hợp này cần chỉnh kết quả về giá trị dương tương ứng.

*Độ phức tạp của giải thuật: O(n), trong đó n – độ dài xâu **b**.*

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "pleasant."
using namespace std;
const int pp=1000000007;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string a,b,x;
int na,nb,d[10][102],tf[102],da,db,ans;
int64_t t;
void calc_d()
{
    d[0][0]=9;
    for(int i=9;i>0;--i)d[i][0]=1;
    for(int j=1;j<=100;++j)
    {
        d[9][j]=1;
        for(int i=8;i>0;--i){t=d[i][j-1]+d[i+1][j]; d[i][j]=t%pp;}
        t=0;
        for(int i=1;i<=9;++i)t+=d[i][j];
        d[0][j]=t%pp;
    }
    tf[0]=9; for(int i=1;i<=100;++i)tf[i]=(tf[i-1]+d[0][i])%pp;
}

int norm_s(string &y,int n)
{int t=0;
    for(int i=n-1;i>0;--i)
        if(y[i-1]<y[i])
        {
            t=1; for(int j=i-1;j>=0;--j)y[j]=y[i];
            break;
        }
    return t;
}
int f(string &y,int &n)
{int64_t tg=1;
    for(int j=1;j<n;++j)
    {
        t=y[j-1]-y[j];
        if(y[j-1]-y[j]>0)
            for(int i=y[j]-48;t>0;++i,--t)tg=(tg+d[i][j-1])%pp;
    }
    for(int i=1;i<y[n-1]-48;++i)tg=(tg+d[i][n-1])%pp;
    if(n>1)tg=(tg+tf[n-2])%pp;
    return tg;
}
int main()
{clock_t aa=clock();
    fi>>x; na=x.size();a="";
    for(int i=na-1;i>=0;--i)a+=x[i];
    fi>>x; nb=x.size();b="";
    for(int i=nb-1;i>=0;--i)b+=x[i];
    calc_d();
    da=norm_s(a,na); db=norm_s(b,nb);
    ans=f(b,nb)-f(a,na)-db+1; if(ans<0)ans+=pp;
    fo<<ans;
    clock_t bb=clock();
    fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS33. CÓ HẬU

Tên chương trình: GOODNUM.CPP

Jim thích các phim và truyện kết thúc có hậu. Một lần gặp may với số n và từ đó Jim rất có cảm tình với tất cả các số nguyên kết thúc bằng n , coi chúng là những số có hậu. Ví dụ với $n = 25$ thì các số 625, 553325, 1025 là những số có hậu, còn 3255 – không có hậu!

Jim không thích các ràng buộc và hạn chế. Nhưng trong thế giới thực của chúng ta ràng buộc và hạn chế là điều tất yếu. Một người bạn của Jim khuyên chỉ nên quan tâm đến các số nguyên không vượt quá m và dĩ nhiên, Jim muốn biết có bao nhiêu số có hậu không vượt quá m .

Dữ liệu: Vào từ file văn bản GOODNUM.INP gồm một dòng chứa 2 số nguyên n và m ($1 \leq n \leq m \leq 2 \times 10^9$).

Kết quả: Đưa ra file văn bản GOODNUM.OUT một số nguyên – số lượng số có hậu tìm được.

Ví dụ:

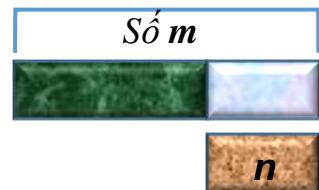
GOODNUM.INP	GOODNUM.OUT
23 122	1



Giải thuật: Số học đơn giản.

Nhận xét:

- + Gọi **k** là số chữ số của **n**,
- + Nếu trong **m** bỏ đi **k** chữ số cuối cùng thì giá trị phần còn lại sẽ là số lượng cách gắn các số vào đầu của **n** để được số mới thỏa mãn các tính chất cần có theo bài,
- + Nếu phần bị loại bỏ lớn hơn hoặc bằng **n** – kết quả sẽ tăng thêm 1.



Độ phức tạp của giải thuật: O(k).

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "goodnum."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int n,m,t10=1,tg,ans;

int main()
{clock_t aa=clock();
 fi>>n>>m;
 tg=n;
 while(tg>0)tg/=10,t10*=10;
 ans=m/t10+(m%t10 >= n);
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS34. MỞ KHÓA

Tên chương trình: **LOCKER.CPP**

Steve định tạo mật khẩu vào Icloud của mình dưới dạng xâu ký tự palindrome hoặc gần palindrome. Steve coi một xâu là gần palindrome nếu chỉ cần thay đổi chỗ không quá 2 ký tự thì sẽ được xâu palindrome.

Trước khi lưu mật khẩu vào kho dữ liệu cá nhân cần phải kiểm tra kỹ xem xâu tạo ra đã thực sự đáp ứng yêu cầu đã nêu hay chưa.

Hãy xác định xâu **s** cho trước có thỏa mãn điều kiện làm mật khẩu hay không.

Dữ liệu: Vào từ file văn bản LOCKER.INP gồm một dòng chứa xâu **s** độ dài không quá 10^5 và chỉ chứa các ký tự la tinh thường.

Kết quả: Đưa ra file văn bản LOCKER.OUT thông báo **YES** hoặc **NO** theo kết quả kiểm tra.

Ví dụ:

LOCKER.INP	LOCKER.OUT
abab	YES



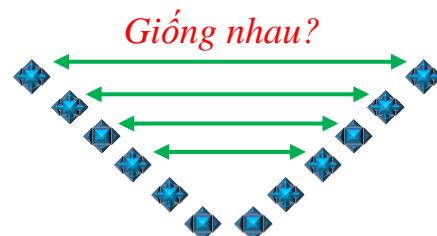
Giải thuật: Phân tích tình huống lô gic.

Nhận xét:

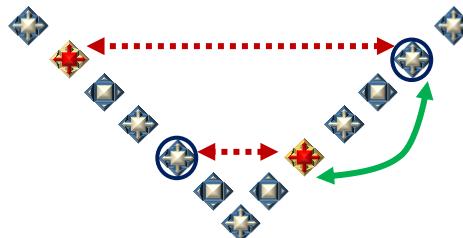
Để khôi phục tính palindrome của xâu cần xác định các cặp ký tự “xâu” phá vỡ tính đối xứng của xâu,

Xét các tình huống:

- ✚ Không có cặp ký tự “xâu” – tính chất palindrome được thỏa mãn,
- ✚ Có nhiều hơn 2 cặp xâu: không thể biến đổi về palindrome chỉ bằng phép đổi chỗ một cặp ký tự,
- ✚ Trường hợp có một cặp xâu:



- ✚ Chỉ có thể biến đổi được khi:
 - ❖ Độ dài xâu là lẻ,
 - ❖ Ký tự giữa xâu trùng với một trong 2 ký tự ở cặp xâu.
- ✚ Trường hợp có 2 cặp xâu: chỉ có thể biến đổi được nếu thỏa mãn điều kiện: 4 ký tự tròn 2 cặp này tạo thành 2 cặp, mỗi cặp chứa 2 ký tự giống nhau.



Tổ chức dữ liệu:

- ❖ Xâu x: lưu ký tự của các cặp xâu,
- ❖ Mảng int p[4] – vị trí cặp xâu.

Xử lý:

Trường hợp 2 cặp xâu: Sắp xếp các ký tự của x theo thứ tự tăng dần và kiểm tra sự tồn tại 2 cặp ký tự giống nhau.

Độ phức tạp của giải thuật: $O(n)$.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "locker."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
string s,x,ans;
int n,m=0,k,p[4];
char c;

int main()
{clock_t aa=clock();
 fi>>s; n=s.size(); x="";
 for(int i=0;i<n/2;++i)
 if(s[i]!=s[n-i-1])
 {
 ++m; x+=s[i];x+=s[n-i-1];p[m]=i;
 if(m>2){ans="NO";break;}
 }
 if(m<3)switch(m)
 {
 case 0:ans="YES"; break;
 case 1: if((n&1)==0)ans="NO";
 else {k=n/2+1;
 if(s[k]==x[0] || s[k]==x[1])ans="YES";
 else ans="NO";}
 break;
 case 2: for(int i=0;i<3;++i)
 for(int j=i+1;j<4;++j)
 if(x[i]>x[j]){c=x[i];x[i]=x[j];x[j]=c;}
 ans=(x[0]==x[1] && x[2]==x[3])? "YES" : "NO" ;
 }
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

VS35. HỎI – ĐÁP

Tên chương trình: ANSWER.CPP

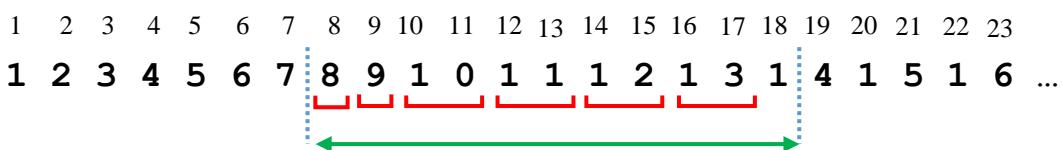
Steve tải về một trò chơi mới rất thú vị. Trò chơi có nhiều mức. Hết mỗi mức chương trình lai đưa ra một câu hỏi. Nếu người chơi trả lời đúng và đủ nhanh thì sẽ được chuyển sang mức mới ngay, trong trường hợp ngược lại – phải tạm dừng 3 ngày!

Nội dung câu hỏi là xét xâu độ dài vô hạn chỉ bao gồm các ký tự số thập phân tạo thành từ việc viết liên tục các số tự nhiên, phần đầu của xâu đó là

12345678910111213141516....

Cho 2 số nguyên a và b ($0 < a \leq b$). Hãy cho biết đoạn các ký tự từ vị trí a đến vị trí b phủ kín lên bao nhiêu số tạo nên đoạn đó.

Ví dụ, với $a = 8$, $b = 18$ đoạn $[a, b]$ phủ kín 6 số là 8, 9, 10, 11, 12, 13. Số 14 có tham gia tạo đoạn nói trên, nhưng nó không bị đoạn đó phủ kín.



Dữ liệu: Vào từ file văn bản ANSWER.INP gồm một dòng chứa 2 số nguyên a và b ($1 \leq a \leq b \leq 10^{18}$).

Kết quả: Đưa ra file văn bản ANSWER.OUT một số nguyên – số lượng số tìm được.

Ví dụ:

ANSWER.INP
8 18

ANSWER.OUT
6



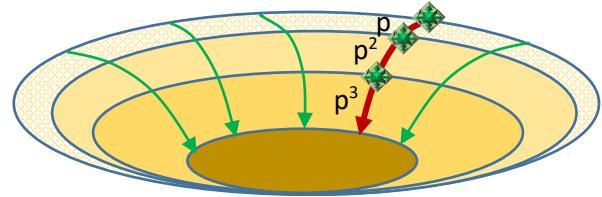
io20160207 C

Giải thuật: Phương pháp tiến nhanh dần tới đích (*accelerating descent method*).

Nhận xét:

Các giải thuật tìm kiếm nhị phân cho phép sau mỗi bước thu hẹp khoảng cách tìm kiếm 2 lần, mang lại hiệu quả cao cho phần lớn các bài toán tìm kiếm,

Trong một số trường hợp, có thể áp dụng chiến lược tìm kiếm cho phép rút ngắn khoảng cách tới đích một đoạn gấp p lần so với khoảng cách được rút ngắn ở bước trước, phương pháp tìm kiếm này được gọi là *Tiến nhanh dần tới đích* (*accelerating descent method*).



Với $p = 2$ phương pháp này có hiệu quả tương đương Tìm kiếm nhị phân,

Với $p > 2$, Tiến nhanh dần tới đích sẽ cho hiệu quả cao hơn nhiều vì việc rút ngắn khoảng cách sẽ tăng dần theo lũy thừa của p (p^1, p^2, p^3, \dots),

Thay vì làm mịn dần và cuối cùng – tìm được kết quả như ở Tìm kiếm nhị phân, phương pháp Tiến nhanh dần tới đích đưa ta đến một miền có thể dùng công thức (tức là với độ phức tạp $O(1)$) để dẫn xuất kết quả cần tìm,

Ở bài toán này ta có thể cục bộ hóa miền chứa nghiệm bằng cách số chữ số của giá trị nhỏ nhất và của giá trị lớn nhất trong tập số cần xác định,

Lực lượng của tập hợp (số lượng số trong tập) có thể dễ dàng xác định theo công thức,

Dễ dàng nhận thấy rằng độ dài phần xâu tạo bởi các số có 1 chữ số là 9, độ dài phần xâu tạo bởi các số có 2 chữ số là $9 \times 10^1 \times 2$, độ dài phần xâu tạo bởi các số có 3 chữ số là $9 \times 10^2 \times 3, \dots$. Trong trường hợp tổng quát độ dài phần xâu tạo bởi các số có d chữ số là $9 \times 10^{d-1} \times d$. Tạo sẵn bảng các giá trị nói trên xác định số chữ số của phần tử bé nhất trong tập cần tìm và từ đó – tính lực lượng của tập cần xác định,

Cần lưu ý rằng việc tạo ra tổng tiền tố của các đoạn cần bỏ qua sẽ dẫn đến việc phải xử lý số lớn vì vậy cần thu hẹp dần khoảng cách thay vì tích lũy nó như trong cách dùng tổng tiền tố trực tiếp.

Tổ chức dữ liệu:

- ➡ Mảng `int64_t p[18] - pi = 10i-1,`
- ➡ Mảng `int64_t q[18] - qi = 9 × 10i-1 × i`.

Xử lý:

- Tính giá trị các bảng p và q ,
- Nhập dữ liệu, giảm a để xác định độ dài phần đầu cần bỏ qua của xâu,

- Xác định số chữ số của số lớn nhất tạo ra đoạn cuối của phần xâu bị loại bỏ, đồng thời chỉnh lý vị trí điểm cuối của miền cần tìm trong xâu đã loại bỏ phần đầu:

```
k=1;
while(a>q[k]) a-=q[k], b-=q[k], ++k;
```

- Tính số lượng số trong tập cần tìm:

Các số đầu tiên có k chữ số
nằm ngoài vùng tìm kiếm

Nếu số lớn nhất trong tập
có không quá k chữ số

```
t1=(a+k-1)/k; c=p[k]-t1;
if(b<=q[k]) ans=(b-t1*k)/k;
else
{
    ans=c; b-=q[k]; ++k;
    while(b>q[k]) ans+=p[k], b-=q[k], ++k;
    ans+=b/k;
}
```

Nếu số lớn nhất trong tập có
nhiều hơn k chữ số

Dộ phức tạp của giải thuật: O(k), $1 \leq k \leq 17$ – số chữ số của số lớn nhất trong tập.

Chương trình:

```
#include <bits/stdc++.h>
#define NAME "answer."
using namespace std;
ifstream fi (NAME"inp");
ofstream fo (NAME"out");
int64_t a,b,p[18],q[18],c,t1,k,ans;

int main()
{clock_t aa=clock();
 fi>>a>>b;--a;
 k=1;
 p[1]=9;q[1]=9;
 for(int i=2;i<=17;++i) p[i]=p[i-1]*10, q[i]=p[i]*i;
 while(a>q[k]) a-=q[k],b-=q[k],++k;
 t1=(a+k-1)/k; c=p[k]-t1;
 if(b<=q[k]) ans=(b-t1*k)/k;
 else
 {
     ans=c; b-=q[k];++k;
     while(b>q[k]) ans+=p[k],b-=q[k],++k;
     ans+=b/k;
 }
 fo<<ans;

clock_t bb=clock();
fo<<"\nTime: "<<(double)(bb-aa)/1000<<" sec";
}
```

MỤC LỤC

VR50. NGÀY SINH NHẬT	<i>Tên chương trình: BIRTHDAY.CPP</i>	2
VS04. ĐỘ TƯƠNG ĐỒNG	<i>Tên chương trình: SIMILARITY.CPP</i>	5
VS05. CON ĐƯỜNG GÓM SỨ	<i>Tên chương trình: CERAMIC.CPP</i>	9
VS07. ĐIỀU CHỈNH LƯƠNG	<i>Tên chương trình: SALARY.CPP</i>	12
VS02. SỐ NGUYÊN MỚI	<i>Tên chương trình: NEWNUM.CPP</i>	18
VS09. ẢNH HOA	<i>Tên chương trình: FLOWERS.CPP</i>	21
VS08. CASIO	<i>Tên chương trình: CASIO.CPP</i>	24
VS10. LẬP TRÌNH NHANH	<i>Tên chương trình: FASTPROG.CPP</i>	26
VS11. KẾT CÂU KÉP	<i>Tên chương trình: DUALSTRUCT.CPP</i>	29
VS12. MÃ KIỂM TRA	<i>Tên chương trình: NUMBERS.CPP</i>	32
VS13. QUAN SÁT	<i>Tên chương trình: OBSERVE.CPP</i>	35
VS14. QUÀ TẶNG	<i>Tên chương trình: GIFTS.CPP</i>	42
VS15. CHUNG KẾT GIẢI ĐÁ CẦU	<i>Tên chương trình: FINAL.CPP</i>	45
VS16. TRẠM THU PHÍ	<i>Tên chương trình: STATIONS.CPP</i>	48
VS17. TIẾNG ANH CỎ	<i>Tên chương trình: ENGLISH.CPP</i>	52
VS19. SÓNG SÓT	<i>Tên chương trình: SURVIVE.CPP</i>	55
VS20. CƠ SỞ	<i>Tên chương trình: BASIC.CPP</i>	60
VS21. ĐÀN ORGAN	<i>Tên chương trình: ORGAN.CPP</i>	63
VS22. DUNG KÉ	<i>Tên chương trình: MEASURE.CPP</i>	66
VS23. SỬA CHỮA	<i>Tên chương trình: REPAIR.CPP</i>	69
VS24. MÔ HÌNH MÁY BAY	<i>Tên chương trình: AIRPLANS.CPP</i>	72
VS25. SỐ LỚN NHẤT	<i>Tên chương trình: MAXNUMBER.CPP</i>	75
VS26. PHẦN THƯỞNG	<i>Tên chương trình: PRIZES.CPP</i>	79
VS27. HÀNH TINH ĐỎ	<i>Tên chương trình: MARS.CPP</i>	82
VS28. XÂU ĐỌC ĐÁO	<i>Tên chương trình: STRANGE.CPP</i>	85
VS29. DU LỊCH BẰNG TÀU HỎA	<i>Tên chương trình: TRAINS.CPP</i>	89
VS30. KHU BẢO TÒN	<i>Tên chương trình: FOREST.CPP</i>	97
VS31. CỜ VÂY	<i>Tên chương trình: DRAUGHTS.CPP</i>	100
VS32. SỐ DỄ CHỊU	<i>Tên chương trình: PLEASANT.CPP</i>	104
VS33. CÓ HẬU	<i>Tên chương trình: GOODNUM.CPP</i>	108
VS34. MỞ KHÓA	<i>Tên chương trình: LOCKER.CPP</i>	110
VS35. HỎI – ĐÁP	<i>Tên chương trình: ANSWER.CPP</i>	113