

Contents

-Sinh kế tiếp	8
Hello World	8
XÂU AB CÓ ĐỘ DÀI N	9
XÂU NHỊ PHÂN CÓ K BIT 1	11
THUẬT TOÁN SINH	13
XÂU NHỊ PHÂN TRƯỚC	14
XÂU NHỊ PHÂN KẾ TIẾP	15
XÂU AB ĐẶC BIỆT (K kí tự A liên tiếp.)	16
SINH TỒ HỢP	18
TẬP CON KẾ TIẾP	21
TẬP CON LIỀN KẾ PHÍA TRƯỚC	24
TẬP QUÂN SỰ	25
ĐẶT TÊN	27
SINH HOÁN VỊ	29
HOÁN VỊ XÂU KÝ TỰ	31
HOÁN VỊ KẾ TIẾP	32
HOÁN VỊ TIẾP THEO CỦA CHUỖI SỐ	34
HOÁN VỊ NGƯỢC	36
PHÂN TÍCH SỐ	38
PHÂN TÍCH SỐ 2	41
MÃ GRAY 1	42
MÃ GRAY 2 (NP-GRAY)	44
MÃ GRAY 3 (GRAY- NP)	45
CHỌN SỐ TỪ MA TRẬN VUÔNG CẤP N	46
TÌM BỘI SỐ (bội của N chỉ chứa hai chữ số 0 và 9)	48
MÁY ATM	50
HAHAHA	51
Quay lui , nhánh cận	53

DÃY SỐ 1	53
DÃY SỐ 2	55
DI CHUYÊN TRONG MÊ CUNG 1 (DOWN RIGHT)	56
DI CHUYÊN TRONG MÊ CUNG 2 (D-L-R-U)	58
DÃY CON TỔNG BẰNG K.....	60
Tổng dãy con = K	62
Tổng dãy con = K 2.....	64
TỔ HỢP SỐ CÓ TỔNG BẰNG X.....	65
TẬP CON BẰNG NHAU	68
ĐỒI CHỖ CÁC CHỮ SỐ	70
CHIA MẢNG (thành k tập con bằng nhau).....	72
DI CHUYÊN TRONG MA TRẬN	73
SỐ NGUYÊN TỐ	75
TỪ ĐIỀN	77
BÀI 32. LOẠI BỎ DẤU NGOẶC chưa làm	79
SẮP XẾP QUÂN HẬU 1.....	79
SẮP XẾP QUÂN HẬU 2.....	81
TẬP HỢP (đếm số tập hợp số lượng =k tổng =s)	82
BIỂU THỨC TOÁN HỌC (=23)	84
ĐƯỜNG ĐI DÀI NHẤT.....	85
SỐ NHỎ NHẤT CÓ N ƯỚC SỐ	88
KÝ TỰ ĐẶC BIỆT	89
NGƯỜI DU LỊCH	90
Giải thuật tham lam.....	92
ĐỒI TIỀN	92
NHÀM CHỮ SỐ.....	93
TÌM MAX.....	94
TỔNG NHỎ NHẤT	95
CHIA MẢNG THÀNH HAI MẢNG CON CÓ TỔNG LỚN NHẤT	97
SẮP XẾP THAM LAM	98

NỐI DÂY 1.....	99
NỐI DÂY 2.....	100
SẮP ĐẶT XÂU KÝ TỰ 1	102
SẮP ĐẶT XÂU KÝ TỰ 2	103
MUA LƯƠNG THỰC	104
SẮP XẾP CÔNG VIỆC 1	106
GIÁ TRỊ NHỎ NHẤT CỦA XÂU	107
SẮP XẾP CÔNG VIỆC 2	109
Chia để trị	110
LŨY THÙA.....	110
GẬP ĐÔI DÃY SỐ.....	112
ĐÉM DÃY	113
DÃY CON LIÊN TIẾP CÓ TỔNG LỚN NHẤT	114
PHÂN TỬ THỨ K.....	115
TÌM KIẾM NHỊ PHÂN	116
TÍNH FLOOR(X)	117
ĐÉM SỐ 0	118
LŨY THÙA ĐẢO	119
DÃY XÂU FIBONACI.....	120
HỆ CƠ SỐ K	122
SẮP XẾP KANGURU.....	122
TÍCH ĐA THỨC	124
TÍCH HAI SỐ NHỊ PHÂN	125
DÃY XÂU NHỊ PHÂN	126
LŨY THÙA MA TRẬN 1	127
LŨY THÙA MA TRẬN 2	129
LŨY THÙA MA TRẬN 3	131
LŨY THÙA MA TRẬN 4	133
LŨY THÙA MA TRẬN 5	135
LŨY THÙA MA TRẬN 6	137

DÉM SỐ BÍT 1	139
Quy hoạch động	140
XÂU CON CHUNG DÀI NHẤT.....	140
DÃY CON TĂNG DÀI NHẤT.....	142
DÃY CON CÓ TỔNG BẰNG S.....	143
DÃY CON DÀI NHẤT CÓ TỔNG CHIA HẾT CHO K	144
TỔ HỢP C(n, k).....	145
XÂU CON ĐỒI XỨNG DÀI NHẤT	146
BẬC THANG	148
HÌNH VUÔNG LỚN NHẤT	149
SỐ CÓ TỔNG CHỮ SỐ BẰNG K.....	151
ĐI NHỎ NHẤT	152
CATALAN NUMBER chưa làm	154
TÍNH P(N, K)	156
SỐ UGLY	158
DÃY CON LẶP LẠI DÀI NHẤT	159
TỔNG LỚN NHẤT CỦA DÃY CON TĂNG DÀN	161
SỐ BƯỚC ÍT NHẤT	163
TỔNG LỚN NHẤT CỦA DÃY CON KHÔNG KÈ NHAU	164
XEM PHIM	165
CÁI TÚI.....	167
GIẢI MÃ	169
CATALAN NUMBER.....	170
DÃY CON CHUNG DÀI NHẤT CỦA BA XÂU	172
DÃY SỐ BI-TONIC	173
KÝ TỰ GIỐNG NHAU	175
TỔNG CÁC XÂU CON	176
TỔNG BẰNG K	177
BIẾN ĐỒI XÂU	179
TỔNG BÌNH PHƯƠNG	180

NHÀ KHÔNG KÈ NHAU	181
XÂU ĐỒI XỨNG 1 chưa.....	183
XÂU ĐỒI XỨNG 2 chưa.....	183
CẶP SỐ	184
DI CHUYÊN VỀ GÓC TỌA ĐỘ.....	185
XÂU ĐỒI XỨNG 1	186
Sắp xếp - Tìm kiếm.....	188
SẮP XẾP ĐỒI CHỖ TRỰC TIẾP	188
SẮP XẾP CHỌN	189
SẮP XẾP CHÈN	191
SẮP XẾP NỘI BỘT	192
SẮP XẾP XEN KẼ	193
SẮP XẾP THEO GIÁ TRỊ TUYỆT ĐỒI	194
HỢP VÀ GIAO CỦA HAI DÃY SỐ 1	195
HỢP VÀ GIAO CỦA HAI DÃY SỐ 2	196
SẮP XẾP [0 1 2]	197
SẮP XẾP DÃY CON LIÊN TỤC	198
CẶP SỐ TỒNG BẰNG K	200
SẮP XẾP CHỮ SỐ	201
TỒNG GẦN 0 NHẤT	202
PHẦN TỬ LỚN NHẤT	203
SỐ LẦN XUẤT HIỆN	203
TỒNG CẶP SỐ NGUYÊN TỐ	204
MERGE SORT	205
TÍCH LỚN NHẤT - NHỎ NHẤT	206
TRỘN HAI DÃY	208
BỒ SUNG PHẦN TỬ	209
SẮP XẾP THEO SỐ LẦN XUẤT HIỆN	210
TÌM KIẾM	211
TÌM KIẾM TRONG DÃY SẮP XẾP VÒNG	212

SỐ NHỎ NHẤT VÀ NHỎ THỨ HAI.....	213
XÓA DỮ LIỆU TRONG DSLK ĐƠN	214
LỌC DỮ LIỆU TRÙNG TRONG DSLK ĐƠN	215
GIAO CỦA BA DÃY SỐ	215
SẮP XÉP CHẨN LỀ.....	217
ĐỒI CHỖ ÍT NHẤT	218
Ngăn xếp.....	219
NGĂN XẾP 1	220
NGĂN XẾP 2	222
ĐẢO TỪ	223
DÃY NGOẶC ĐÚNG DÀI NHẤT.....	224
KIỂM TRA BIÊU THỨC SỐ HỌC	226
ĐÊM SỐ DÂU NGOẶC ĐỒI CHIỀU	228
BIÊN ĐỒI TIỀN TỐ - TRUNG TỐ	230
BIÊN ĐỒI TIỀN TỐ - HẬU TỐ	231
BIÊN ĐỒI TRUNG TỐ - HẬU TỐ	233
BIÊN ĐỒI HẬU TỐ - TIỀN TỐ	235
BIÊN ĐỒI HẬU TỐ - TRUNG TỐ	237
TÍNH GIÁ TRỊ BIÊU THỨC HẬU TỐ	239
TÍNH GIÁ TRỊ BIÊU THỨC TIỀN TỐ.....	240
HÌNH CHỮ NHẬT LỚN NHẤT	241
BIÊU THỨC TĂNG GIẢM.....	245
BIÊU THỨC TƯƠNG ĐƯƠNG	246
PHẦN TỬ BÊN PHẢI NHỎ HƠN	248
SO SÁNH BIÊU THỨC	251
PHẦN TỬ CÓ SỐ LẦN XUẤT HIỆN NHIỀU HƠN BÊN PHẢI	253
Hàng đợi	255
CÂU TRÚC DỮ LIỆU HÀNG ĐỢI 1.....	255
CÂU TRÚC DỮ LIỆU HÀNG ĐỢI 2.....	257
HÀNG ĐỢI HAI ĐẦU (DEQUEUE)	258

GIÁ TRỊ NHỎ NHẤT CỦA XÂU	260
SỐ 0 VÀ SỐ 9	261
SỐ NHỊ PHÂN TỪ 1 ĐẾN N	263
SỐ BDN 1 (nhị phân).....	264
SỐ BDN 2 (chia hết cho N).....	265
SỐ LỘC PHÁT 1 (gồm 6 và 8)	267
SỐ LỘC PHÁT 2	268
SỐ LỘC PHÁT 3	269
BIÊN ĐỒI S – T (thao tác : nhân , trừ).....	270
TÌM SỐ K THỎA MÃN ĐIỀU KIỆN (cs khác nhau và ≤ 5)	272
Đồ thị và đồ thị nâng cao:	274
CHUYỂN DANH SÁCH CẠNH SANG DANH SÁCH KÈ.....	274
CHUYỂN TỪ DANH SÁCH KÈ SANG DANH SÁCH CẠNH	276
CHUYỂN DANH SÁCH KÈ SANG MA TRẬN KÈ	277
BIỂU DIỄN ĐỒ THỊ CÓ HƯỚNG.....	279
CHUYỂN MA TRẬN KÈ SANG DANH SÁCH KÈ	280
DFS TRÊN ĐỒ THỊ VÔ HƯỚNG.....	281
BFS TRÊN ĐỒ THỊ VÔ HƯỚNG	286
TÌM ĐƯỜNG ĐI THEO DFS VỚI ĐỒ THỊ VÔ HƯỚNG	288
ĐƯỜNG ĐI THEO BFS TRÊN ĐỒ THỊ VÔ HƯỚNG	291
KIỂM TRA ĐƯỜNG ĐI.....	293
Đếm số thành phần liên thông (theo DFS).....	295
Tìm số thành phần liên thông với BFS	298
Kiểm tra tính liên thông mạnh.....	301
Kiểm tra tính liên thông mạnh với BFS b buộc ds kề	303
LIỆT KÈ ĐỈNH TRỤ (với dfs)	305
LIỆT KÈ ĐỈNH TRỤ VỚI BFS	309
LIỆT KÈ CẠNH CẦU	312
Kiểm tra chu trình	316
Kiểm tra chu trình với BFS	317

KIỂM TRA CHU TRÌNH SỬ DỤNG DISJOIN SET	319
KIỂM TRA CHU TRÌNH TRÊN ĐỒ THỊ CÓ HƯỚNG VỚI DFS	320
ĐƯỜNG ĐI VÀ CHU TRÌNH EULER VỚI ĐỒ THỊ VÔ HƯỚNG.....	322
CHU TRÌNH EULER TRONG ĐỒ THỊ CÓ HƯỚNG	323
KIỂM TRA ĐỒ THỊ CÓ PHẢI LÀ CÂY HAY KHÔNG.....	325
CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN DFS	327
CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN BFS	329
ĐƯỜNG ĐI HAMILTON.....	331
DIJKSTRA	332
KRUSKAL	334
Cây nhị phân	336
KIỂM TRA NODE LÁ	336
CÂY BIÊU THỨC 1.....	339
CÂY BIÊU THỨC 2.....	341
DUYỆT CÂY 1	343
DUYỆT CÂY 2	345
DUYỆT CÂY THEO MỨC	348
DUYỆT CÂY NHỊ PHÂN TÌM KIẾM 1	350
DUYỆT CÂY THEO MỨC ĐẢO NGƯỢC.....	352
DUYỆT CÂY KIỀU XOẮN ÔC (duyệt cây 7)	354
CÂY NHỊ PHÂN TỔNG	357

-Sinh kế tiếp

Hello World

In ra màn hình dòng chữ:

Hello PTIT.

Input

Không có dữ liệu vào

Output

Hello PTIT.

Giải:

Cách Python 3:

```
print('Hello PTIT.')
```

XÂU AB CÓ ĐỘ DÀI N

Giống sinh nhị phân

Xâu ký tự str được gọi là xâu AB nếu mỗi ký tự trong xâu hoặc là ký tự 'A' hoặc là ký tự 'B'. Ví dụ xâu str="ABBABB" là xâu AB độ dài 6. Nhiệm vụ của bạn là hãy liệt kê tất cả các xâu AB có độ dài n.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên n.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq n \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Mỗi xâu cách nhau 1 khoảng trắng.

Input	Output
2	AA AB BA BB
2	AAA AAB ABA ABB BAA BAB BBA
3	BBB

Giải:

Cách 1 :

//1: khởi tạo và in ra cấu hình đầu tiên

//2: đưa ra pow(2,n)-1 cấu hình tiếp theo so vs cấu hình đầu tiên

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n; cin>>n;
    string s;
    for(int i=0;i<n;i++) s[i]='A';//1
    for(int i=0;i<n;i++) cout<<s[i];cout<<' ';
/*2*/for(int i=0;i<pow(2,n)-1;i++){
        int j=n;
        while(j--&&s[j]=='B') s[j]='A';
        if(j>=0) s[j]='B';
        for(int i=0;i<n;i++) cout<<s[i];cout<<' ';
    }
}
```

```

    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
return 0;
}

```

Cách 2:

```

//1: khởi tạo và in ra câu hình đầu tiên
//2: xem là vòng lặp đã đạt đến câu hình cuối cùng chưa
//3: duyệt ngược đổi các chữ B thành A đến khi xuất hiện A
//4: Tại vị trí chữ A thì đổi thành B và dừng
#include <bits/stdc++.h>
using namespace std;
int n;
string s;
void out(){
    for(int i=0;i<n;i++) cout<<s[i];cout<<' ';
}
bool check(){
    for(int i=0;i<n;i++)
        if (s[i]=='A') return 0; return 1;
}
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++) s[i]='A';//1
    for(int i=0;i<n;i++) cout<<s[i];cout<<' ';
/*2*/while(check()==0){
    int j=n;
    while(j--&&s[j]=='B') s[j]='A';//3
    if(j>=0) s[j]='B';//4
    out();
}
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();

```

```

return 0;
}

Cách 3 : Quay lui :
//1 tìm giá trị của phần tử thứ i
//2 : số lựa chọn mà phần tử đó có thể được chọn
#include <bits/stdc++.h>
using namespace std;
string s;
int n;
void out(){
    for(int i=0;i<n;i++) cout<<s[i]; cout<<" ";
}
void Try(int i){//1
    for(int j=0;j<=1;j++){//2
        s[i]=j+'A';
        if(i==n-1) out();
        else Try(i+1);
    }
}
int main() {
    int t;cin>>t;
    while(t--){
        cin>>n;
        Try(0);
        cout<<endl;
    }
    return 0;
}

```

XÂU NHỊ PHÂN CÓ K BIT 1

Nội dung bài tập

Hãy in ra tất cả các xâu nhị phân độ dài N, có K bit 1 theo thứ tự từ điển tăng dần.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 20$). Mỗi test gồm 2 số nguyên N, K ($1 \leq K \leq N \leq 16$).

Output: Với mỗi test, in ra đáp án tìm được, mỗi xâu in ra trên một dòng.

Ví dụ:

Input	Output
-------	--------

2	0011
4 2	0101
3 2	0110
	1001
	1010
	1100
	011
	101
	110

Giải :

// 1: thực hiện duyệt các xâu nhị phân độ dài n
 //2: sau khi tạo ra 1 câu hình hoàn chỉnh thì kiểm tra
 //3 : kiểm tra xem xâu đó có chứa k số 1 hay không nếu có thì in ra

```
#include<bits/stdc++.h>
int n,k,a[17];
using namespace std;
void check(){//3
    int count=0;
    for(int i=1;i<=n;i++) if(a[i]==1) count++;
    if(count==k){
        for(int i=1;i<=n;i++) cout<<a[i];cout<<endl;
    }
}
void Try(int i){
    for(int j=0;j<=1;j++){
        a[i]=j;
        if(i==n) check();//2
        else Try(i+1);
    }
}
int main(){
    int t;cin>>t;
    while(t--){
        cin>>n>>k;
        Try(1);//1
    }
    return 0;
}
```

}

THUẬT TOÁN SINH

Một xâu nhị phân độ dài n được gọi là thuận nghịch hay đối xứng nếu đảo ngược xâu nhị phân đó ta vẫn nhận được chính nó. Cho số tự nhiên n (n nhập từ bàn phím). Hãy viết chương trình liệt kê tất cả các xâu nhị phân thuận nghịch có độ dài n .

- Dòng đầu tiên ghi lại số K là số các xâu thuận nghịch có độ dài n tìm được;
- K dòng kế tiếp ghi lại mỗi dòng một xâu nhị phân thuận nghịch có độ dài n . Hai phần tử khác nhau của xâu thuận nghịch được ghi cách nhau một vài khoảng trắng.

Ví dụ với $n = 4$ ta tìm được 4 xâu nhị phân thuận nghịch như dưới đây.

0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int n,a[1000];
void check(){
    int l=1,r=n;
    while(l<r){
        if(a[l]!=a[r])return;
        l++;r--;
    }
    for(int i=1;i<=n;i++)cout<<a[i]<<"\t";
    cout<<endl;
}
void Try(int i){
    for(int j=0;j<=1;j++){
        a[i]=j;
        if(i==n) check();
        else Try(i+1);
    }
}
int main(){
    cin>>n;
```

```

Try(1);
    return 0;
}

```

XÂU NHỊ PHÂN TRƯỚC

Cho xâu nhị phân $X[]$, nhiệm vụ của bạn là hãy đưa ra xâu nhị phân trước của $X[]$. Ví dụ $X[] = "111111"$ thì xâu nhị phân trước của $X[]$ là “111110”. Với xâu $X[] = "000001"$ thì xâu nhị trước của $X[]$ là “000000”. Chú ý: nếu xâu dữ liệu trong input là xâu đầu tiên thì trước nó sẽ là xâu cuối cùng.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu nhị phân X.
- T, X[] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(X) \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	010100
010101	111110
111111	

Giai:

```

#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s;
    cin>>s;
    int n=s.length();
    while(n--){
        if(s[n]=='1'){
            s[n]='0';
            break;
        }
        else s[n]='1';
    }
    cout<<s<<endl;
}
int main(){

```

```

int n;
cin>>n;
while(n--) initwsolve();
return 0;
}

```

XÂU NHỊ PHÂN KẾ TIẾP

Ôn lại

Nội dung bài tập

Cho xâu nhị phân $X[]$, nhiệm vụ của bạn là hãy đưa ra xâu nhị phân tiếp theo của $X[]$. Ví dụ $X[] = "010101"$ thì xâu nhị phân tiếp theo của $X[]$ là " 010110 ".

Input:

- Dòng đầu tiên đưa vào số lượng test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu nhị phân X .
- $T, X[]$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 1 \leq \text{length}(X) \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	
010101	010110
111111	000000

Giải :

Cách 1:c++ rất ngắn:

//1: Duyệt ngược đổi các chữ số 1 thành các số 0 cho đến khi nào gặp số 0 thì thôi
//2 : tại vị trí đó đổi số 0 thành số 1

```

#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s; cin>>s;
    int i=s.length();
/*1*/while(i--&&s[i]=='1') s[i]='0';
/*2*/if(i>=0) s[i]='1';
    cout<<s<<endl;
}
int main() {
    int t; cin>>t;
    while(t--) initwsolve();
}

```

```
return 0;
```

```
}
```

Cách 2:c++

//1: Tìm chỉ số đầu tiên là số 0 theo thứ tự từ phải sang trái và gán cho số đó là 1, các số đứng sau số đó (tức nằm bên phải) thì gán hết cho bằng 0.

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s;cin>>s;
    int n=s.length();
/*1*/while (n--)
    {
        if(s[i]=='1') s[i]='0';
        else {
            s[i]='1';break;
        }
    }
    cout<<s<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
return 0;
}
```

Cách 3 Python 3:

```
t=int(input())
while t>0:
    s=list(input())
    i=len(s)-1
    while i>=0 and s[i]=='1':
        s[i]='0'
        i-=1
    if i>=0 :s[i]='1'
    for x in s: print(x,end="")
    print()
    t=t-1
#chú ý : python không có t++,t--
```

XÂU AB ĐẶC BIỆT (K kí tự A liên tiếp.)

Ôn lại

Nội dung bài tập

Một xâu kí tự $S = (s_1, s_2, \dots, s_n)$ được gọi là xâu AB độ dài n nếu với mọi $s_i \in S$ thì si hoặc là kí tự A hoặc s_i là kí tự B . Ví dụ xâu $S = "ABABABAB"$ là một xâu AB độ dài 8. Cho số tự nhiên N và số tự nhiên K ($1 \leq K < N \leq 15$ được nhập từ bàn phím), hãy viết chương trình liệt kê tất cả các xâu AB có độ dài N chứa duy nhất một dãy K kí tự A liên tiếp.

Dữ liệu vào chỉ có một dòng ghi hai số N và K.

Kết quả ghi ra màn hình theo khuôn dạng:

- Dòng đầu tiên ghi lại số các xâu AB thỏa mãn yêu cầu bài toán;
- Những dòng kế tiếp, mỗi dòng ghi lại một xâu AB thỏa mãn. Các xâu được ghi ra theo thứ tự từ điển.

Ví dụ:

INPUT	OUTPUT
5 3	5 AAABA AAABB ABAAA BAAAB BBAAA

Cách 1 : quay lui:

//1 : sinh nhị phân như bình thường
//2 : đến cấu hình có n chữ cái rồi xét

//3 : xét cấu hình có bao nhiêu dãy con gồm k chữ 'A' , nếu là 1 thì đưa vào kết quả

```
#include<bits/stdc++.h>
using namespace std;
int n,k;
string a;
vector <string> res;
void check(){//3
    int ok=0;
    for(int i=0;i<n-k+1;i++){
        int count=0;
        for(int j=i;j<i+k;j++){
            if(a[j]=='A') count++;
        }
    }
}
```

```

        if(count==k) ok++;
    }
    if(ok==1) res.push_back(a);
}
/*1*/void Try(int i){
    for(char j='A';j<='B';j++){
        a[i]=j;
        if(i==n-1) check();//2
        else Try(i+1);
    }
}
int main(){
    cin>>n>>k;
    a.resize(n); //note
    Try(0);
    cout<<res.size()<<endl;
    for(int i=0;i<res.size();i++) cout<<res[i]<<endl;;
    return 0;
}

```

Cách 2 : phương pháp sinh :

```

void Sinh(){
    int i=n;
    while(a[i]=='B'&&i>0) {
        a[i]='A';
        i--;
    }
    if(i>0) a[i]='B';
    else check=false;
}

```

SINH TỔ HỢP

Nội dung bài tập

Cho hai số nguyên dương N và K. Nhiệm vụ của bạn là hãy liệt kê tất cả các tập con K phần tử của 1, 2, .., N. Ví dụ với N=5, K=3 ta có 10 tập con của 1, 2, 3, 4, 5 như sau: {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}, {3, 4, 5}.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một cặp số tự nhiên N, K được viết trên một dòng.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq k \leq n \leq 15$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2 4 3 5 3	123 124 134 234 123 124 125 134 135 145 234 235 245 345

Giải :

Cách 1:

//1: khởi tạo và in câu hình đầu tiên
//2: kiểm tra xem đến câu hình cuối để dừng chưa
//3: tạo và in ra câu hình kế tiếp

```
#include <bits/stdc++.h>
using namespace std;
int n,k,a[15];
void initwsolve(){
    cin>>n>>k;
    for(int i=1;i<=k;i++) a[i]=i;//1
    for(int i=1;i<=k;i++) cout<<a[i]; cout<<' ';
/*2*/while(1){
    int i=k;//3
    while(a[i]==n-k+i&&i>0) i--;
    if(i==0) break;
    else{
        a[i]++;
        for(int j=i+1;j<=k;j++) a[j]=a[j-1]+1;
        for(int i=1;i<=k;i++) cout<<a[i]; cout<<' ';
    }
}
cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

}

Cách 2:

```
//1: khởi tạo và in câu hình đầu tiên  
//2: kiểm tra xem đến câu hình cuối để dừng chưa  
//3: tạo và in ra câu hình kế tiếp  
#include <bits/stdc++.h>  
using namespace std;  
int n,k,a[15];  
bool check(){  
    for(int i=k;i>=1;i--)  
        if (a[i]!=n-k+i) return 0; return 1;  
}  
void initwsolve(){  
    cin>>n>>k;  
    for(int i=1;i<=k;i++) a[i]=i;//1  
    for(int i=1;i<=k;i++) cout<<a[i]; cout<<' ';  
/*2*/ while(check()==0){  
    int i=k;//3  
    while(a[i]==n-k+i&&i>0) i--;  
    a[i]++;  
    for(int j=i+1;j<=k;j++) a[j]=a[j-1]+1;  
    for(int i=1;i<=k;i++) cout<<a[i]; cout<<' ';  
    }  
    cout<<endl;  
}  
int main() {  
    int t;cin>>t;  
    while(t--)initwsolve();  
return 0;  
}
```

Cách 3 : quay lui:

```
#include <bits/stdc++.h>  
using namespace std;  
int a[16],n,k;  
void out(){  
    for(int i=1;i<=k;i++) cout<<a[i]; cout<<" ";  
}
```

```

void Try(int i){
    for(int j=a[i-1]+1;j<=n-k+i;j++){
        a[i]=j;
        if(i==k) out();
        else Try(i+1);
    }
}
int main() {
    int t;cin>>t;
    while(t--){
        cin>>n>>k;
        Try(1);
        cout<<endl;
    }
    return 0;
}

```

TẬP CON KẾ TIẾP

Ôn lại

Nội dung bài tập

Cho hai số N, K và một tập con K phần tử $X[] = (X_1, X_2, \dots, X_K)$ của 1, 2, .., N. Nhiệm vụ của bạn là hãy đưa ra tập con K phần tử tiếp theo của $X[]$. Ví dụ N=5, K=3, $X[] = \{2, 3, 4\}$ thì tập con tiếp theo của $X[]$ là $\{2, 3, 5\}$.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là hai số N và K; dòng tiếp theo đưa vào K phần tử của $X[]$ là một tập con K phần tử của 1, 2, .., N.
- T, K, N, $X[]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K \leq N \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	
5 3	
1 4 5	2 3 4
5 3	1 2 3
3 4 5	

Key :

Cách 1 :rất ngắn:

```
//1 :duyệt ngược tìm vị trí đầu tiên a[i] khác (n-k+i)
//2: nếu i=0 thì in câu hình đầu tiên còn khác 0 thì cộng 1 tại a[i], các số sau bằng
//số trước cộng thêm 1
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k; cin>>n>>k;
    int a[k+1],i=k;
    for(int i=1;i<=k;i++) cin>>a[i];
    while(a[i]==n-k+i&&i>0) i--; //1
/*2*/if(i==0) for(int i=1;i<=k;i++) a[i]=i;
    else {
        a[i]++;
        for(int j=i+1;j<=k;j++) a[j]=a[j-1]+1;
    }
    for(int i=1;i<=k;i++) cout<<a[i]<<' ';
    cout<<endl;
}
int main() {
    int t;/**/cin>>t;
    while(t--) initwsolve();
return 0;
}
```

Cách 2:

```
//1 check : kiểm tra xem tổ hợp có phải là cuối hay chưa
//2 :duyệt ngược để tìm vị trí số cần thay đổi . khi tìm được vị trí số đó Thay thế
// $a_i = a_i + 1$ , Thay  $a_j = a_i + j - i$ , với  $j = i + 1, i + 2, \dots, m$ .
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k;/**/ cin>>n>>k;
    int a[k+1],check=0;//1
    for(int i=1;i<=k;i++)/**/ cin>>a[i];
    for(int i=k;i>0;i--)
        if(a[i]!=n-k+i){//2
            a[i]++;
            for(int j=i+1;j<=k;j++)

```

```

        a[j]=a[j-1]+1;
        check=1;
        break;
    }
    if(check==1) for(int i=1;i<=k;i++) cout<<a[i]<<' ';
    else for(int i=n-k+1;i<=k;i++) cout<<i<<' ';
    cout<<endl;
}
int main() {
    int t;/**/cin>>t;
    while(t--) initwsolve();
return 0;
}

```

\cách 3 : python 3: chú ý mảng a chạy từ 0 đến k-1

```

a=[]
def out():
    for i in a : print(i,end=' ')
def initwsolve():
    n, k = map(int, input().split())
    a = [int(i) for i in input().split()[:k]] # nhập trên 1 dòng
    i = k-1;
    while i > -1 and a[i] == n - k + i+1 : i-=1
    if i== -1:
        for i in range(0,k ): a[i] = i+1
    else :
        a[i]+=1
        for j in range(i+1,k ) : a[j]=a[j-1]+1
    for i in a: print(i, end=' ')
    print()
t=int(input())
while(t>0):
    initwsolve();
    t-=1

```

cách 4 : python 3 : mảng a chạy từ 1 đến k (mới đầu học lầm bug code 2 kiểu để dò lỗi sai)

```

a=[]
def out():
    for i in a : print(i,end=' ')
def initwsolve():
    n, k = map(int, input().split())
    a = [int(i) for i in input().split()[:k]] # nhập trên 1 dòng
    a.insert(0,0)
    i = k;
    while i > 0 and a[i] == n - k + i : i-=1
    if i==0:
        for i in range(0,k+1 ): a[i] = i+1
    else :
        a[i]+=1
        for j in range(i+1,k+1 ): a[j]=a[j-1]+1
    for i in range(1,k+1): print(a[i], end=' ')
    print()
t=int(input())
while(t>0):
    initwsolve();
    t-=1

```

TẬP CON LIỀN KÈ PHÍA TRƯỚC

Cho hai số N, K và một tập con K phần tử X[] =(X₁, X₂..., X_K) của 1, 2, .., N. Nhiệm vụ của bạn là hãy đưa ra tập con K phần tử trước đó của X[]. Ví dụ N=5, K=3, X[] ={2, 3, 5} thì tập con trước đó của X[] là {2, 3, 4}. Chú ý nếu tập con trong input là đầu tiên thì trước đó là tập con cuối cùng.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là hai số N và K; dòng tiếp theo đưa vào K phần tử của X[] là một tập con K phần tử của 1, 2, .., N.
- T, K, N, X[] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K \leq N \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	2 3 4
5 3	3 4 5
2 3 5	
5 3	
1 2 3	

123 124 125 135 145 234 235 245 345

Giải :

Cách 1 : C++

```
//1 ý tưởng lấy ngược từ bài sinh tổ hợp kế tiếp  
//2: nếu là cấu hình đầu tiên (i==0 tức là cấu hình 0,1,2,3 . a[0]=0) thì in cuối cùng  
//3 : còn không thì trừ tai vị trí đó 1 dv , các số sau là n-k+i  
#include<bits/stdc++.h>  
using namespace std;  
void initwsolve(){  
    int n,k;  
    cin>>n>>k;  
    int a[k+1]={0};  
    for(int i=1;i<=k;i++) cin>>a[i];  
    int i=k;  
    while(a[i]==a[i-1]+1) i--;//1  
    if(i==0){//2  
        for(int j=k;j>=1;j--) a[j]=n-k+j;  
    }  
    else{//3  
        a[i]--;  
        for(int j=i+1;j<=k;j++) a[j]=n-k+j;  
    }  
    for(int i=1;i<=k;i++) cout<<a[i]<<' ';cout<<endl;  
}  
int main(){  
    int t;cin>>t;  
    while(t--) initwsolve();  
}
```

TẬP QUÂN SỰ

Nội dung bài tập

Tại Chương Mỹ Resort, vào nửa đêm, cả trung đội nhận lệnh tập trung ở sân. Mỗi chiến sỹ được đánh số từ 1 đến N ($1 < N < 40$). Giám thị yêu cầu chọn ra một dãy K chiến sỹ để tập đội ngũ và cứ lần lượt duyệt hết tất cả các khả năng chọn K người như vậy từ nhỏ đến lớn (theo số thứ tự). Bài toán đặt ra là cho một nhóm K chiến sỹ hiện đang phải tập đội ngũ, hãy tính xem trong lượt chọn K người tiếp theo thì mấy người trong nhóm cũ sẽ được tạm nghỉ. Nếu đã là nhóm cuối cùng thì tất cả đều sẽ được nghỉ.

Dữ liệu vào: Dòng đầu ghi số bộ test, không quá 20. Mỗi bộ test viết trên hai dòng

- Dòng 1: hai số nguyên dương N và K ($K < N$)
- Dòng 2 ghi K số thứ tự của các chiến sỹ đang phải tập đội ngũ (viết từ nhỏ đến lớn)

Kết quả: Với mỗi bộ dữ liệu in ra số lượng chiến sỹ được tạm nghỉ.

Ví dụ:

INPUT	OUTPUT
3	
5 3	
1 3 5	1
5 3	2
1 4 5	4
6 4	
3 4 5 6	

//1 : thực hiện sinh tổ hợp kết tiếp nếu là tổ hợp cuối cùng rồi thì in ra k;

//2 : không thì đếm xem giữa cấu hình cũ , và mới có bao nhiêu số trùng (người ko dk nghỉ)=> kq là k-count;

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k; cin>>n>>k;
    int a[k+1],b[k+1],i=k;
    for(int i=1;i<=k;i++) cin>>a[i],b[i]=a[i];
    while(a[i]==n-k+i&&i>0) i--;
    /*1*/if(i==0) cout<<k<<endl;
    else {
        a[i]++;
        for(int j=i+1;j<=k;j++) a[j]=a[j-1]+1;
```

```

int count=0;
/*2*/for(int i=1;i<=k;i++){
    for(int j=1;j<=k;j++){
        if(a[i]==b[j]) {count++;break;}
    }
}
cout<<k-count<<endl;
}
int main() {
    int t;/**/cin>>t;
    while(t--) initwsolve();
return 0;
}

```

ĐẶT TÊN

Ôn lại

Vương quốc PTIT sử dụng bảng chữ cái gồm N chữ cái Latinh viết hoa. Quy tắc đặt tên của gia đình Hoàng gia PTIT là chọn ra K chữ cái (không trùng nhau) và sắp xếp lại theo thứ tự từ điển.

Hãy liệt kê tất cả các cái tên có thể có của gia đình Hoàng gia PTIT

Input

- Dòng đầu ghi số bộ test T (không quá 10)
- Mỗi bộ test ghi 2 số N và K ($3 < K < N < 16$)

Output

- Với mỗi bộ test, ghi ra tất cả các cái tên có thể được tạo ra, mỗi kết quả viết trên một dòng.

Ví dụ

Input	Output
1	AB
4 2	AC
	AD
	BC
	BD
	CD

Giải :

Cách 1:

```

#include <bits/stdc++.h>
using namespace std;
int a[16],n,k;
void out(){
    for(int i=1;i<=k;i++) cout<<char(a[i] +'@');
    cout<<endl;
}
void Try(int i){
    for(int j=a[i-1]+1;j<=n-k+i;j++){
        a[i]=j;
        if(i==k) out();
        else Try(i+1);
    }
}
int main() {
    int t;cin>>t;
    while(t--){
        cin>>n>>k;
        Try(1);
        cout<<endl;
    }
    return 0;
}

```

Cách 2:

//bản chất là sinh tổ hợp 2 của 4 phần tử nhưng mà số sau luôn phải lớn hơn số trước

```

#include <bits/stdc++.h>
using namespace std;
int n,k,a[100];
bool check() {
    for(int i=2; i<=k; i++) {
        if(a[i]<a[i-1]) return 0;
    }
    return 1;
}
void initwsolve() {
    cin>>n>>k;
    for(int i=1; i<=k; i++) a[i]=i;

```

```

while(1) {
    if(check()) {
        for(int i=1; i<=k; i++)
            cout<<char (a[i] +'A'-1);
        cout<<endl;
    }
    int i=k;
    while(i>0 && a[i]==n-k+i) i--;
    if(i==0) return;
    a[i]=a[i]+1;
    for(int j=i+1; j<=k; j++) {
        a[j]=a[i]-i+j;
    }
}
main() {
    int t; cin>>t;
    while(t--) initwsolve();
}

```

SINH HOÁN VỊ

Nội dung bài tập

Cho số nguyên dương N. Nhiệm vụ của bạn là hãy liệt kê tất cả các hoán vị của 1, 2, .., N. Ví dụ với N = 3 ta có kết quả: 123, 132, 213, 231, 312, 321.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng.
- T, n thỏa mãn ràng buộc: $1 \leq T, N \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	12 21
2	123 132 213 231 312 321
3	

Giải:

Cách 1:

```

//1: khởi tạo và in ra câu hình đầu tiên
//2:ktra xem phải câu hình cuối chưa thì break; vòng lặp
//3: không thì in ra câu hình tiếp theo
#include <bits/stdc++.h>
using namespace std;
int n,a[11];
void out(){
    for(int i=1;i<=n;i++) cout<<a[i]; cout<<' ';
}
void initwsolve(){
    cin>>n;
    for(int i=1;i<=n;i++) a[i]=i; out();//1
    while(1){
        int i=n,j=n;
        while(a[i]<a[i-1]&&i>0) i--; i--;
        if(i==0) break;//2
        /*3*/ else {
            while(a[j]<a[i]) j--;
            swap(a[i],a[j]);
            sort(a+i+1,a+n+1);
            out();
        }
    }
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

Cách 2: quay lui :

```

#include<bits/stdc++.h>
using namespace std;
int n,a[11],check[11]={ };
void out(){
    for(int i=1;i<=n;i++)cout<<a[i]; cout<<" ";
}
void Try(int i){

```

```

        for(int j=1;j<=n;j++){
            if(check[j]==0){
                check[j]=1;
                a[i]=j;
                if(i==n) out(); //n not =n+1
                else Try(i+1);
                check[j]=0;
            }
        }
    }
int main(){
    int t;cin>>t;
    while(t--){
        cin>>n;
        Try(1);
        cout<<endl;
    }
    return 0;
}

```

HOÁN VỊ XÂU KÝ TỰ

Nội dung bài tập

Cho xâu ký tự S bao gồm các ký tự in hoa khác nhau. Hãy đưa ra tất cả các hoán vị của xâu ký tự S. Ví dụ S="ABC" ta có kết quả
{ABC ACB BAC BCA CAB CBA}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là một xâu ký tự S được viết trên 1 dòng.
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq \text{length}(S) \leq 10$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	
AB	AB BA
ABC	ABC ACB BAC BCA CAB CBA

```

//note : coi mỗi ký tự của chuỗi như các phần tử của 1 mảng => tạo ra 1 chuỗi khác
(x) để lưu trữ kết quả
//thực hiện giống như duyệt hoán vị n phần tử của một mảng a[n].
#include<bits/stdc++.h>
using namespace std;
int check[11];
string s,x[11];
void out(){
    for(int i=0;i<s.length();i++) cout<<x[i];cout<<' ';
}
void Try(int i){//2
    for(int j=0;j<s.length();j++){
        if(check[j]==0){
            x[i]=s[j];
            check[j]=1;
            if(i==s.length()-1) out();
            else Try(i+1);
            check[j]=0;
        }
    }
}
int main(){
    int t;cin>>t;
    while(t--) {
        cin>>s;
        Try(0);
        cout<<endl;
    }
    return 0;
}

```

HOÁN VỊ KẾ TIẾP

Ôn lại

Nội dung bài tập

Cho số tự nhiên N và một hoán vị X[] của 1, 2, .., N. Nhiệm vụ của bạn là đưa ra hoán vị tiếp theo của X[]. Ví dụ N=5, X[] = {1, 2, 3, 4, 5} thì hoán vị tiếp theo của X[] là {1, 2, 3, 5, 4}.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số N; dòng tiếp theo đưa vào hoán vị X[] của 1, 2, .., N.
- T, N, X[] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	
5	
1 2 3 4 5	1 2 3 5 4
5	1 2 3 4 5
5 4 3 2 1	

Giải:

Cách 1 : rất ngắn:

```
//1 : duyệt ngược tìm vị trí thứ 1 cần tráo đổi
//2: duyệt ngược tìm vị trí thứ 2 cần tráo đổi
//3: sắp xếp mảng tăng dần các số sau vị trí thứ 1
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n; cin>>n;
    int a[n+1],i=n,j=n,check=0;
    for(int i=1;i<=n;i++) cin>>a[i];
/*1*/while(a[i]<a[i-1]&&i>0) i--; i--;
    if(i<0) for(int i=1;i<=n;i++) cout<<i<<' ';
    else {
        while(a[j]<a[i]) j--;//2
        swap(a[i],a[j]);
        sort(a+i+1,a+n+1);//3
        for(int i=1;i<=n;i++) cout<<a[i]<<' ';
    }
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
```

```

return 0;
}

Cách 2:
//1 : duyệt ngược tìm vị trí thứ 1 cần tráo đổi
//2: duyệt ngược tìm vị trí thứ 2 cần tráo đổi
//3: sắp xếp mảng tăng dần các số sau vị trí thứ 1
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n; cin>>n;
    int a[n+1],mark=0,check=0;
    for(int i=1;i<=n;i++) cin>>a[i];
/*1*/for(int i=n;i>1;i--){
    if(a[i-1]<a[i]){
        mark=i-1;check=1; break;
    }
}
/*2*/for(int i=n;i>mark;i--){
    if(a[i]>a[mark]){
        swap(a[i],a[mark]); break;
    }
}
sort(a+mark+1,a+n+1);//3
if(check==1) for(int i=1;i<=n;i++) cout<<a[i]<<' ';
else for(int i=1;i<=n;i++) cout<<i<<' ';
cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
return 0;
}

```

HOÁN VỊ TIẾP THEO CỦA CHUỖI SỐ

Nội dung bài tập

Hãy viết chương trình nhận vào một chuỗi (có thể khá dài) các ký tự số và đưa ra màn hình hoán vị kế tiếp của các ký tự số đó (với ý nghĩa là hoán vị có giá trị lớn hơn tiếp theo nếu ta coi chuỗi đó là một giá trị số nguyên). Chú ý: Các ký tự số trong dãy có thể trùng nhau.

Ví dụ: 123 -> 132

279134399742 -> 279134423799

Cũng có trường hợp sẽ không thể có hoán vị kế tiếp. Ví dụ như khi đầu vào là chuỗi 987.

Dữ liệu vào: Dòng đầu tiên ghi số nguyên t là số bộ test ($1 \leq t \leq 1000$). Mỗi bộ test có một dòng, đầu tiên là số thứ tự bộ test, một dấu cách, sau đó là chuỗi các ký tự số, tối đa 80 phần tử.

Kết quả: Với mỗi bộ test hãy đưa ra một dòng gồm thứ tự bộ test, một dấu cách, tiếp theo đó là hoán vị kế tiếp hoặc chuỗi “BIGGEST” nếu không có hoán vị kế tiếp.

Ví dụ:

INPUT	OUTPUT
3	1 132
1 123	2 279134423799
2 279134399742	3 BIGGEST
3 987	

Giải :

//1: tìm vị trí thứ nhất cần tráo đổi

//2: nếu vị trí tìm được =-1 (cấu hình max) thì in biggest không thì in hoán vị tiếp theo

/3 : tìm vị trí thứ nhất cần tráo đổi (số nhỏ nhất lớn hơn a[i])

//4 : sx chuỗi tăng dần từ i+1 đến cuối chuỗi.

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    string s;cin>>s;
    int i=s.length()-1,j=s.length()-1;
    while(s[i]<=s[i-1]&&i>0) i--;//1
    /*2*/if(i<0) {
        cout<<n<<" BIGGEST\n"; return;
    }
    else{
        while(s[j]<=s[i]) j--;//3
        swap(s[i],s[j]);
        sort(s.begin()+i+1,s.end());//4
        cout<<n<<' ';
```

```

        for(int i=0;i<s.length();i++) cout<<s[i]; cout<<endl;
    }
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

HOÁN VỊ NGƯỢC

Cho số nguyên dương N. Nhiệm vụ của bạn là hãy liệt kê tất cả các hoán vị của 1, 2, ..., N theo thứ tự ngược. Ví dụ với N = 3 ta có kết quả: 321, 312, 231, 213, 132, 123.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng.
- T, n thỏa mãn ràng buộc: $1 \leq T, N \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	21 12
2	321 312 231 213 132 123
3	

Giải :

Cách 1 : quay lui :

// ý tưởng y hệt liệt kê hoán vị thông thường chỉ khác mỗi 1 vòng for khởi tạo số lựa chọn cho phần tử đầu tiên ($n \geq 1$)

```

#include<bits/stdc++.h>
using namespace std;
int n,a[11],check[11];
void out(){
    for(int i=1;i<=n;i++) cout<<a[i]; cout<<' ';
}
void Try(int i){
    for(int j=n;j>=1;j--){
        if(check[j]==0){

```

```

        check[j]=1;
        a[i]=j;
        if(i==n) out();
        else Try(i+1);
        check[j]=0;
    }
}
int main(){
    int t;cin>>t;
    while(t--) {
        cin>>n;
        Try(1);
        cout<<endl;
    }
    return 0;
}

```

Cách 2 : phương pháp sinh:

```

#include<bits/stdc++.h>
using namespace std;
int n,a[11], ok = 0;
void out(){
    for(int i = 1 ; i <= n ; i++)cout << a[i]; cout << ' ';
}
void gennext(){
    int i = n - 1;
    while(a[i] < a[i+1] && i >= 1) i--;
    if( i == 0) ok = 1;
    else{
        int j=n;
        while(a[i]<a[j]) j--;
        swap(a[i],a[j]);
        int l = i + 1 , r = n;
        while( l <= r ){
            swap(a[l],a[r]);
            l++,r--;
        }
    }
}

```

```

}

void initwsolve(){
    cin >> n ; ok = 0;
    for(int i = 1 ; i <= n ; i++)a[i] = n - i + 1;
    while(ok == 0){
        out(); gennext();
    }
    cout << endl;
}
int main(){
    int t=1; cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

PHÂN TÍCH SỐ

Ôn lại

Cho số nguyên dương N. Nhiệm vụ của bạn là hãy liệt kê tất cả các cách phân tích số tự nhiên N thành tổng các số tự nhiên nhỏ hơn hoặc bằng N. Phép hoán vị vừa một cách được xem là giống nhau. Ví dụ với N = 5 ta có kết quả là: (5), (4, 1), (3, 2), (3, 1, 1), (2, 2, 1), (2, 1, 1, 1), (1, 1, 1, 1, 1).

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng.
- T, n thỏa mãn ràng buộc: $1 \leq T, N \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	(4) (3 1)
4	(2 2) (2 1 1)
5	(1 1 1 1)

Giải:

Cách 1: Quay lui

5	
4	1
3	2

	1	1	
2	2	1	
	1	1	1
1	1	1	1

//1: (1,0,n) phần tử đầu tiên của dãy , tổng hiện tại, số cách chọn phần tử đầu tiên
//2: gán kq vào phần tử đầu tiên và tổng hiện tại
//3: nếu tổng hiện tại =n thì in dãy kq không thì sẽ tiếp tục tìm phần tử thứ 2 nhỏ hơn phần tử trước đó của dãy

```
#include<bits/stdc++.h>
using namespace std;
int n,a[11];
void out(int n){
    cout<<"(";
    for(int i=1;i<n;i++) cout<<a[i]<<' ';
    cout<<a[n]<<") ";
}
void solve(int i,int sumcur,int k){
    for(int j=k;j>=1;j--){
        if(sumcur+j<=n){
            a[i]=j;//2
            sumcur+=j;
            if(sumcur==n) out(i);//3
            else solve(i+1,sumcur,j);//j : in (4 1), không in (1 4)
            sumcur-=j;
        }
    }
}
int main(){
    int t;cin>>t;
    while(t--){
        cin>>n;
        solve(1,0,n);//1
        cout<<endl;
    }
    return 0;
}
```

Hàm khác vẫn đúng:

```
void Try(int i, int k, int cursum){  
    for(int j=k;j>=1;j--){  
        if(cursum+j<=n){  
            a[i]=j;  
            if(cursum+j==n) out(i);  
            else Try(i+1,j,cursum+j);  
        }  
    }  
}
```

Cách 2 : sinh :

```
#include<bits/stdc++.h>  
using namespace std;  
int n,k,a[15];  
bool OK;  
void out(){  
    cout<<"(";  
    for(int i=1;i<k;i++)cout<<a[i]<<" ";  
    cout<<a[k]<<") "  
}  
void newcon(){  
    int i=k,j,R,S,D;  
    while(i>0&&a[i]==1) i--;  
    if(i>0){  
        a[i]--;  
        D=k-i+1;  
        R=D/a[i];  
        S=D%a[i];  
        k=i;  
        if(R>0){  
            for(j=i+1;j<=i+R;j++)a[j]=a[i];  
            k=k+R;  
        }  
        if(S>0){  
            k=k+1;a[k]=S;  
        }  
    }  
}
```

```

    else OK=0;
}
int main(){
    int t;cin>>t;
    while(t--){
        OK=1;cin>>n;k=1;a[k]=n;
        while(OK==1){
            out();
            newcon();
        }
        cout<<endl;
    }
    return 0;
}

```

PHÂN TÍCH SỐ 2

Cho số nguyên dương N. Nhiệm vụ của bạn là hãy liệt kê tất cả các cách phân tích số tự nhiên N thành tổng các số tự nhiên nhỏ hơn hoặc bằng N. Phép hoán vị của một cách được xem là giống nhau. Ví dụ với N = 5 ta có kết quả là: (5), (4, 1), (3, 2), (3, 1, 1), (2, 2, 1), (2, 1, 1, 1), (1, 1, 1, 1, 1) .

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng.
- T, n thỏa mãn ràng buộc: $1 \leq T, N \leq 10$.

Output:

- Dòng đầu tiên là số lượng cách phân tích thỏa mãn. Dòng tiếp theo liệt kê đáp án theo mẫu ví dụ đã cho.

Ví dụ:

Input	Output
2	5
4	(4) (3 1) (2 2) (2 1 1) (1 1 1 1)
5	7 (5) (4 1) (3 2) (3 1 1) (2 2 1) (2 1 1 1) (1 1 1 1 1)

Giải :

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

```

int n,a[11];
vector<vector<int>> res;
void add(int n){
    vector<int>b;
    for(int i=1;i<=n;i++)b.push_back(a[i]);
    res.push_back(b);
}
void solve(int i,int sumcur,int k){
    for(int j=k;j>=1;j--){
        if(sumcur+j<=n){
            a[i]=j;
            sumcur+=j;
            if(sumcur==n) add(i);
            else solve(i+1,sumcur,j);
            sumcur-=j;
        }
    }
}
int main(){
    int t;cin>>t;
    while(t--){
        cin>>n;
        res.clear();
        solve(1,0,n);
        cout<<res.size()<<endl;
        for(int i=0;i<res.size();i++){
            cout<<"(";
            for(int j=0;j<res[i].size()-1;j++) cout<<res[i][j]<<' ';
            cout<<res[i][res[i].size()-1]<<") ";
        }
        cout<<endl;
    }
    return 0;
}

```

MÃ GRAY 1

Ôn lại

Số nhị phân được xem là cách mặc định biểu diễn các số. Tuy nhiên, trong nhiều ứng dụng của điện tử và truyền thông lại dùng một biến thể của mã nhị phân đó là mã Gray. Mã Gray độ dài n có mã đầu tiên là n số 0, mã kế tiếp của nó là một xâu nhị phân độ dài n khác biệt với xâu trước đó một bit. Ví dụ với $n=3$ ta có 2^3 mã Gray như sau: 000, 001, 011, 010, 110, 111, 101, 100. Hãy viết chương trình liệt kê các mã Gray có độ dài n.

Input:

- Dòng đầu tiên là số lượng test T.
- T dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test là một số tự nhiên n.
- T, n thỏa mãn ràng buộc: $1 \leq T, n \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input: Output:

2	000 001 011 010 110 111 101 100
3	0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010
4	1011 1001 1000

Giải :

Cách 1 : quay lui :

```
//1: sinh nhị phân bằng quay lui
//2: chuyển mã nhị phân thành mã gray
//note :^ :XOR :Bitwise exclusive OR
#include<bits/stdc++.h>
using namespace std;
int n,a[11];
void out(){//2
    cout<<a[1];
    for(int i=2;i<=n;i++){
        int res=a[i]^a[i-1]; cout<<res;
    }
    cout<<' ';
}
void Try(int i){//1
    for(int j=0;j<=1;j++){
        a[i]=j;
        if(i==n) out();
        else Try(i+1);
    }
}
```

```

    }
}

int main(){
    int t;cin>>t;
    while(t--){
        cin>>n;
        Try(1); cout<<endl;
    }
    return 0;
}

```

Cách 2 : dùng phương pháp sinh sinh ra từng cấu hình và tiếp tục chuyên nó thành mã gray. (tự làm nhé) .

MÃ GRAY 2 (NP-GRAY)

Nhi phân => gray

Nội dung bài tập

Số nhị phân được xem là cách mặc định biểu diễn các số. Tuy nhiên, trong nhiều ứng dụng của điện tử và truyền thông lại dùng một biến thể của mã nhị phân đó là mã Gray. Mã Gray độ dài n có mã đầu tiên là n số 0, mã kế tiếp của nó là một xâu nhị phân độ dài n khác biệt với xâu trước đó một bit. Ví dụ với n=3 ta có 2³ mã Gray như sau: 000, 001, 011, 010, 110, 111, 101, 100. Hãy viết chương trình chuyên đổi một xâu mã nhị phân X có độ dài n thành một xâu mã Gray.

Input:

- Dòng đầu tiên là số lượng test T.
- T dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test là một xâu nhị phân độ dài n.
- T, n thỏa mãn ràng buộc: $1 \leq T, n \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2	
01001	01101
01101	01011

//note :^ :XOR :Bitwise exclusive OR

//1: kq chữ số đầu tiên giữ nguyên , còn kq các chữ số sau thì bằng tuyên loại của chữ số đang xét và số liền trước

```

#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s; cin>>s;
    cout<<s[0];//1
    for(int i=1;i<s.length();i++){
        int res=(s[i]-'0')^(s[i-1]-'0'); cout<<res;
    }
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

MÃ GRAY 3 (GRAY- NP)

Gray => nhị phân

Nội dung bài tập

Số nhị phân được xem là cách mặc định biểu diễn các số. Tuy nhiên, trong nhiều ứng dụng của điện tử và truyền thông lại dùng một biến thể của mã nhị phân đó là mã Gray. Mã Gray độ dài n có mã đầu tiên là n số 0, mã kế tiếp của nó là một xâu nhị phân độ dài n khác biệt với xâu trước đó một bít. Ví dụ với n=3 ta có 2^3 mã Gray như sau: 000, 001, 011, 010, 110, 111, 101, 100. Hãy viết chương trình chuyển đổi một xâu mã Gray X có độ dài n thành một xâu mã nhị phân.

Input::

- Dòng đầu tiên là số lượng test T.
- T dòng kế tiếp ghi lại mỗi dòng một test. Mỗi test là một xâu mã Gray độ dài n.
- T, n thỏa mãn ràng buộc: $1 \leq T, n \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2	01001
01101	01101
01011	

Giải :

```

//note :^ :XOR :Bitwise exclusive OR
//note : ngược lại so với bài toán chuyển nhị phân sang gray
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s; cin>>s;
    string res=s;
    for(int i=1;i<s.length();i++){
        if((res[i-1]-'0')^0==(s[i]-'0')) res[i]='0';
        else res[i]='1';
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

CHỌN SỐ TỪ MA TRẬN VUÔNG CẤP N

Ôn lại

Nội dung bài tập

Cho ma trận vuông $C_{i,j}$ cấp N ($1 \leq i, j \leq N \leq 10$) gồm N^2 số tự nhiên và số tự nhiên K (các số trong ma trận không nhất thiết phải khác nhau và đều không quá 100 , K không quá 10^4). Hãy viết chương trình lấy mỗi hàng, mỗi cột duy nhất một phần tử sao cho tổng các phần tử này đúng bằng K .

Dữ liệu vào: Dòng 1 ghi hai số N và K . N dòng tiếp theo ghi ma trận C .

Kết quả: dòng đầu ghi số cách tìm được. Mỗi dòng tiếp theo ghi một cách theo vị trí của số đó trong lần lượt từng hàng của ma trận. Xem ví dụ để hiểu rõ hơn.

Ví dụ:

INPUT	OUTPUT
3 10	
2 4 3	2
1 3 6	1 3 2
4 2 4	3 2 1

```

//note :1 3 2 : số ở vị trí thứ 1 h1 (2)+ số ở vị trí thứ 3 h2 (6)+ số ở vị trí thứ 2 h3
(2)
// 1: thực hiện quay lui hoán vị để tìm tất cả các vị trí của mỗi hàng
//2 : nếu đạt được 1 cấu hình (vd 1 3 2)
//3 : thì kiểm tra tổng a.1.1+ a.2.3+ a.3.2 ==k không , nếu có thì đưa và vector v 2
chiều
// 4: sau đó in ra kết quả
#include<bits/stdc++.h>
using namespace std;
int n,k,a[11][11],check[11];
vector <int> b(11);
vector <vector<int> > v;
void /*3*/checksum(){
    int sum=0;
    for(int i=1;i<=n;i++) sum+=a[i][b[i]];
    if(sum==k) v.push_back(b);
}
void Try(int i){// 1
    for(int j=1;j<=n;j++){
        if(check[j]==0){
            check[j]=1;
            b[i]=j;
            if(i==n) checksum();
            else Try(i+1);
            check[j]=0;
        }
    }
}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];
    Try(1);
/*4*/ cout<<v.size()<<endl;
    for(int i=0;i<v.size();i++){
        for(int j=1;j<=n;j++){
            cout<<v[i][j]<<' ';
        }
    }
}

```

```

        } cout<<endl;
    }
    return 0;
}

```

TÌM BỘI SỐ (bội của N chỉ chứa hai chữ số 0 và 9.

Ôn lại

Nội dung bài tập

Cho số nguyên N. Nhiệm vụ của bạn cần tìm số nguyên X nhỏ nhất là bội của N, và X chỉ chứa hai chữ số 0 và 9.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 10000$). Mỗi bộ test chứa số nguyên N trên một dòng ($1 \leq N \leq 500$).

Output: Với mỗi test in ra đáp án tìm được trên một dòng.

Ví dụ:

Input	Output
3	90
2	90
5	99
11	

Giải :

Cách 1 : rất ngắn:

//1 : tạo 1 hàng đợi có số 9 đầu tiên.

//2: kiểm tra xem số đầu tiên hàng đợi (x) chia hết cho n không có thì in ra

//3: không thì xóa số đó (x) khỏi hàng đợi đồng thời thêm $x*10$ và $x*10+9$ vào cuối hàng đợi.

```

#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    queue <long long> q;
    q.push(9);//1
    while(1){
        long long x=q.front();
        if(x%n==0) {//2
            cout<<x<<endl; break;
        }
        q.pop();
        q.push(x*10);
        q.push(x*10+9);
    }
}

```

```

    }
    q.pop();//3
    q.push(x*10);
    q.push(x*10+9);
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

CÁch 2:

```

// 1:dùng quay lui để kiệt kê tất cả các số chứa 2 số 0 và 9 đưa vào 1 mảng động
//2: sau đó sắp xếp mảng theo thứ tự tăng dần
//3: tìm kiếm mảng đến số nhỏ nhất %n ==0
#include<bits/stdc++.h>
using namespace std;
int n;
vector <long long> res;
void /*1*/ Try( long long i){
    if(i>=1e17) return;
    res.push_back(i);
    Try(i*10);
    Try(i*10+9);
}
int main(){
    int t;cin>>t;
    Try(9);
    sort(res.begin(),res.end());//2
    while(t--){
        cin>>n;
        /*3*/for(int i=0;i<res.size();i++){
            if(res[i]%n==0){
                cout<<res[i]<<endl; break;
            }
        }
    }
    return 0;
}

```

}

MÁY ATM

Ôn lại

Một máy ATM hiện có n ($n \leq 30$) tờ tiền có giá trị $t[1], t[2], \dots, t[n]$. Hãy tìm cách trả ít tờ nhất với số tiền đúng bằng S (các tờ tiền có giá trị bất kỳ và có thể bằng nhau, mỗi tờ tiền chỉ được dùng một lần).

Input: Dòng đầu tiên ghi số bộ test T ($T < 10$). Mỗi bộ test gồm 2 số nguyên n và S ($S \leq 10^9$). Dòng thứ hai chứa n số nguyên $t[1], t[2], \dots, t[n]$ ($t[i] \leq 10^9$)

Output: Với mỗi bộ test ghi ra số tờ tiền ít nhất phải trả.

Nếu không thể tìm được kết quả, in ra -1.

Ví dụ

Input	Output
1 3 5 1 4 5	1

//1: thực hiện duyệt tất cả các tổ hợp từng tổng có figure phần tử ($1 \Rightarrow n$) nếu tại một cấu hình nào đó có tổng đúng bằng s ($ok=1$) thì dừng luôn (kq :figure chính là số lượng các tờ tiền ít nhất)

//quay lui 3: nếu đạt đến cấu hình thỏa mãn gồm figure phần tử mà tổng = s thì báo đúng không thì kết thúc việc tìm kiếm.

//4 : vd tết 3 6,1 4 5 thì xét tổng các cặp theo trình tự :1,4,5,1+4,1+5,4+5,1+4+5

// note: phải để 2 dòng if ở trên vì để ngay sau khi tìm phần tử tiếp theo ta cần so sánh luôn, nếu để ở dưới if{} thì sẽ ko thể in ra số 5 được

```
#include<bits/stdc++.h>
using namespace std;
int n,check[31],ok=0;
long long s,sum,a[31];
void Try(int i,int figure){
    if(sum==s&&i==figure){ ok=1;return;}//3
    if(i==figure&&sum!=s) return;//3
    for(int j=1;j<=n;j++){
        if(j>check[i-1]){
            sum+=a[j];
            check[i]=j;
            if(sum<=s) Try(i+1,figure);
        }
    }
}
```

```

        sum-=a[j];
        check[i]=0;
    }
}
void initwsolve(){
    cin>>n>>s;
    for(int i=1;i<=n;i++) cin>>a[i];
    ok=0;
    int figure;
/*1*/for(figure=1;figure<=n;figure++){
    sum=0;
    Try(0,figure);
    if(ok==1) break;
}
if(ok==1) cout<<figure<<endl;
else cout<<-1<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

HAHAHA

Khi chat chit vui vẻ, anh em chiến hữu thường hay gõ HAHA để thể hiện sự sảng khoái. Đôi khi cũng có thể gõ HAHAAAAA chẳng hạn cho thêm phần nhấn mạnh.

Giả sử một xâu ký tự được coi là HAHA nếu thỏa mãn các điều kiện:

- Ký tự đầu tiên phải là chữ H, ký tự cuối cùng phải là chữ A
- Không có hai chữ H nào liền nhau

Cho trước độ dài N, hãy liệt kê tất cả các xâu ký tự HAHA theo thứ tự từ điển.

Input

- Dòng đầu ghi số bộ test T (không quá 10)
- Mỗi bộ test ghi độ dài N ($2 \leq N < 16$)

Output

- Với mỗi bộ test, ghi ra tất cả các xâu HAHA tìm được theo thứ tự từ điển, mỗi xâu viết trên một dòng.

Ví dụ

Input	Output
2	HA
2	HAAA
4	HAHA

Giai:

```
#include<iostream>
using namespace std;
int a[20],n,OK;
bool check(){
    if(a[1]==0||a[n]==1) return 0;
    for(int i=1;i<n;i++){
        if(a[i]&&a[i+1]) return 0;
    }
    return 1;
}
void initwsolve(){
    OK=1;cin>>n;
    for(int i=1;i<=n;i++) a[i]=0;
    while(OK==1){
        if(check()){
            for(int i=1;i<=n;i++)
                if(a[i]==0) cout<<"A";
                else cout<<"H";
            cout<<endl;
        }
        int i=n;
        while (i>0 && a[i]!=0) { a[i] = 0; i--; }
        if (i >0) a[i]=1;
        else OK = 0;
    }
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}
```

Quay lui , nhánh cận

```
//Quy tắc khởi tạo chung làm những bài tập về quay lui
#include<bits/stdc++.h>
using namespace std;
int n ,ok,check[11];
vector <int> v(11);
vector <vector <int> > res;
void initwsolve(){
    cin>>n;
    // v.resize(n);//dùng khi v đã bị pop.back hết sạch sau 1 lần test
    ok=0;//bắt buộc
    res.clear();//bắt buộc.
    v.clear();//không bắt buộc vì ở trong Try (đã xóa sau khi thêm rồi)
    for(int i=0;i<n;i++) check[i]=0;//không bắt buộc vì ở trong Try (đã xóa sau khi
    thêm rồi)
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

DÃY SỐ 1

Ôn lại

Cho dãy số $A[]$ gồm n số nguyên dương. Tam giác đặc biệt của dãy số $A[]$ là tam giác được tạo ra bởi n hàng, trong đó hàng thứ 1 là dãy số $A[]$, hàng i là tổng hai phần tử liên tiếp của hàng $i-1$ ($2 \leq i \leq n$). Ví dụ $A[] = \{1, 2, 3, 4, 5\}$, khi đó tam giác được tạo nên như dưới đây:

[1, 2, 3, 4, 5]
[3, 5, 7, 9]
[8, 12, 16]
[20, 28]
[48]

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .

- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N số của mảng A[].
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10$;

Output:

- Đưa ra tam giác tổng của mỗi test theo từng dòng. Mỗi dòng của tam giác tổng được bao bởi ký tự [,].

Input	Output
1	[1 2 3 4 5]
5	[3 5 7 9]
1 2 3 4 5	[8 12 16]
	[20 28]
	[48]

Cách 1 : c++

//1: tạo một vector (mảng động) để lưu kết quả hiện tại
//2: thực hiện tiếp lệnh cho đến khi nào mảng chỉ còn 1 phần tử thì thôi
//3 : công việc lần lặp : giá trị của số hiện tại là tổng số đó và sau nó , sau khi kết thúc vòng lặp thì xóa phần tử cuối cùng của mảng và in mảng.

```
#include<bits/stdc++.h>
using namespace std;
int n;
vector<int> a(11);
void out(){
    cout<<'[';
    for(int i=0;i<a.size()-1;i++) cout<<a[i]<<' ';
    cout<<a[a.size()-1]<<']'<<endl;
}
void initwsolve(){
    cin>>n; a.resize(n);//1
    for(int i=0;i<a.size();i++) cin>>a[i]; out();
/*2*/ while(a.size()>1){
    for(int i=0;i<a.size()-1;i++) a[i]=a[i]+a[i+1];//3
    a.pop_back();//3
    out();
}
int main(){
```

```

int t;cin>>t;
while(t--) initwsolve();
return 0;
}

```

DÃY SỐ 2

Ôn lại

Cho dãy số A[] gồm n số nguyên dương. Tam giác đặc biệt của dãy số A[] là tam giác được tạo ra bởi n hàng, trong đó hàng thứ n là dãy số A[], hàng i là tổng hai phần tử liên tiếp của hàng i+1 ($1 \leq i \leq n-1$). Ví dụ $A[] = \{1, 2, 3, 4, 5\}$, khi đó tam giác được tạo nên như dưới đây:

[48]

[20, 28]

[8, 12, 16]

[3, 5, 7, 9]

[1, 2, 3, 4, 5]

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N số của mảng A[].
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Mỗi dòng của tam giác tổng được bao bởi ký tự [,].

Input	Output
1 5 1 2 3 4 5	[48] [20 28] [8 12 16] [3 5 7 9] [1 2 3 4 5]]

//1 : tạo một vector res (mảng động 2 chiều) lưu trữ tất cả các kết quả ,mỗi kết quả nằm trên 1 hàng

//2 : thực hiện công việc tính tổng và xóa số cuối mảng cho đến khi nào mảng chỉ còn 1 phần tử . sau mỗi kết quả thì đưa mảng 1 chiều vào vector res.

//3 : duyệt ngược lại in kết quả

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

```
using namespace std;
```

```
int n;
```

```

vector<int> a(11);
vector<vector<int>> res;//1
void initwsolve(){
    cin>>n; a.resize(n);res.clear();//note
    for(int i=0;i<a.size();i++) cin>>a[i]; res.push_back(a);
/*2*/ while(a.size()>1){
    for(int i=0;i<a.size()-1;i++) a[i]=a[i]+a[i+1];
    a.pop_back();
    res.push_back(a);
}
/*3*/ for(int i=res.size()-1;i>=0;i--){
    cout<<'[';
    for(int j=0;j<res[i].size()-1;j++) cout<<res[i][j]<<' ';
    cout<<res[i].back()<<']'<<' ';
}
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

ĐI CHUYỀN TRONG MÊ CUNG 1 (DOWN RIGHT)

Ôn lại

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân $A[N][N]$. Một con chuột đi từ ô đầu tiên góc trái ($A[0][0]$) đến ô cuối cùng góc phải ($A[N-1][N-1]$) theo nguyên tắc:

Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.

Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.

Hãy đưa ra một hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể đi đến đích.

*Input:

Dòng đầu tiên đưa vào số lượng bộ test T.

Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là kích cỡ của mê cung; dòng tiếp theo đưa vào ma trận nhị phân $A[N][N]$.

T, N, $A[i][j]$ thỏa mãn ràng buộc: $1 \leq T \leq 10$; $2 \leq N \leq 10$; $0 \leq A[i][j] \leq 1$.

Output:

Đưa ra tất cả đường đi của con chuột trong mê cung theo thứ tự từ điển. Đưa ra -1 nếu chuột không đi được đến đích.

Input	Output
2	
4	
1 0 0 0	
1 1 0 1	
0 1 0 0	DRDDRR
1 1 1 1	DDRDRRDR DDRDRRRD DRDRRDR
5	DRDDRRRD DRRRRDDD
1 0 0 0 0	
1 1 1 1 1	
1 1 0 0 1	
0 1 1 1 1	
0 0 0 1 1	

Giải :

//1 : xét mảng nếu phần tử đầu mảng =0 thì ko tìm được , ko thì quay lui (hàng ,cột)

// 2:nếu phần tử ở hàng dưới =1 thì sẽ tiếp tục tìm kiếm không thì sẽ ngừng.

// 3:nếu phần tử ở cột phải =1 thì sẽ tiếp tục tìm kiếm không thì sẽ ngừng.

// 4: nếu tìm kiếm đến vị trí cuối cùng của mảng (ok=1) thì in

```
#include<bits/stdc++.h>
using namespace std;
int n;
bool a[11][11],ok;
string s;
void Try(int x,int y){
/*4*/if(x==n&&y==n){
    cout<<s<<' '; ok=1;return;
}
/*2*/if(a[x+1][y]==1){
    s+='D';
    Try(x+1,y);
    s.erase(s.length()-1);
}
/*3*/if(a[x][y+1]==1){
    s+='R';
    Try(x,y+1);}
```

```

        s.erase(s.length()-1);
    }
}

int main(){
    int t;cin>>t;
    while(t--){
        cin>>n;ok=0;//s="";
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                cin>>a[i][j];
        if(a[1][1]==0){cout<<-1<<endl;continue;}//1
        Try(1,1);
        if(ok==0) cout<<-1;
        cout<<endl;
    }
    return 0;
}

```

ĐI CHUYỂN TRONG MÊ CUNG 2 (D-L-R-U)

Ôn lại

Cho một mê cung bao gồm các khối được biểu diễn như một ma trận nhị phân $A[N][N]$. Một con chuột di từ ô đầu tiên góc trái ($A[0][0]$) đến ô cuối cùng góc phải ($A[N-1][N-1]$) theo nguyên tắc:

- Down (D): Chuột được phép xuống dưới nếu ô dưới nó có giá trị 1.
- Right (R): Chuột được phép sang phải dưới nếu ô bên phải nó có giá trị 1.
- Left (L): Chuột được phép sang trái dưới nếu ô bên trái nó có giá trị 1.
- Up (U): Chuột được phép lên trên nếu ô trên nó có giá trị 1.

Hãy đưa ra tất cả các hành trình của con chuột trên mê cung. Đưa ra -1 nếu chuột không thể di đến đích.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là kích cỡ của mê cung; dòng tiếp theo đưa vào ma trận nhị phân $A[N][N]$.
- T, N, $A[i][j]$ thỏa mãn ràng buộc: $1 \leq T \leq 10$; $2 \leq N \leq 8$; $0 \leq A[i][j] \leq 1$.

Output:

- Đưa ra các xâu ký tự được sắp xếp, trong đó mỗi xâu là một đường đi của con chuột trong mê cung. In ra đáp án theo thứ tự từ điển. Đưa ra -1 nếu chuột không đi được đến đích.

Input	Output
3	
4	
1 0 0 0	
1 1 0 1	
0 1 0 0	
0 1 1 1	DRDDRR
4	DDRDRR DRDDRR
1 0 0 0	DDRRURRDDD DDRURRRDDD
1 1 0 1	DRDRURRDDD DRRRRDDD
1 1 0 0	
0 1 1 1	
5	
1 0 0 0 0	
1 1 1 1 1	
1 1 1 0 1	
0 0 0 0 1	
0 0 0 0 1	

Giải :

//1 : xét mảng nếu phần tử đầu mảng =0 thì ko tìm được , ko thì quay lui (hàng ,cột)

// 2,3,4,5 : tìm kiếm theo chiều sâu

//6 : nếu tìm kiếm chạy đến cuối dãy(ok=1) thì in kq

//note : phải đủ điều kiện ($x < n, y >= 1, y < n, x >= 1$) và việc cho $check[x][y]=1$ là để việc tìm kiếm không bị quay đi quay lại mãi tại $a[i][j]$

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

```
using namespace std;
```

```
int n;
```

```
bool a[11][11],check[11][11],ok;
```

```
string s;
```

```
void Try(int x,int y){
```

```
/*6*/if(x==n&&y==n){
```

```
    cout<<s<<' '; ok=1;return;
```

```
}
```

```
    check[x][y]=1;
```

```

/*2*/if(x<n&&a[x+1][y]==1&&check[x+1][y]==0){
    s+='D';
    Try(x+1,y);
    s.erase(s.length()-1);
}
/*3*/if(y>=1&&a[x][y-1]==1&&check[x][y-1]==0){
    s+='L';
    Try(x,y-1);
    s.erase(s.length()-1);
}
/*4*/if(y<n&&a[x][y+1]==1&&check[x][y+1]==0){
    s+='R';
    Try(x,y+1);
    s.erase(s.length()-1);
}
/*5*/if(x>=1&&a[x-1][y]==1&&check[x-1][y]==0){
    s+='U';
    Try(x-1,y);
    s.erase(s.length()-1);
}
check[x][y]=0;
}
int main(){
int t;cin>>t;
while(t--){
    cin>>n;ok=0;//s="";
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];
    if(a[1][1]==0){cout<<-1<<endl;continue;}//1
    Try(1,1);
    if(ok==0) cout<<-1;
    cout<<endl;
}
return 0;
}

```

DÃY CON TỔNG BẰNG K
Ôn lại

Cho dãy số $A[] = (a_1, a_2, \dots, a_n)$ và số tự nhiên K . Hãy đưa ra tất cả các dãy con của dãy số $A[]$ sao cho tổng các phần tử của dãy con đó đúng bằng K . Các phần tử của dãy số $A[]$ được giả thuyết là nguyên dương và không có các phần tử giống nhau. Ví dụ với dãy con $A[] = \{5, 10, 15, 20, 25\}$, $K = 50$ ta có 3 dãy con $\{5, 10, 15, 20\}$, $\{5, 20, 25\}$, $\{10, 15, 25\}$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là số lượng phần tử của dãy số $A[]$ và số K ; dòng tiếp theo đưa vào N phần tử của dãy số $A[]$.
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10$; $1 \leq K, A[i] \leq 100$.

Output:

- Đưa ra tất cả các dãy con của dãy số $A[]$ thỏa mãn yêu cầu bài toán theo thứ tự từ điển, trong đó mỗi dãy con được bao bởi các ký tự [,]. Nếu không có dãy con nào thỏa mãn yêu cầu bài toán, hãy đưa ra -1.

Input	Output
2 5 50 5 10 15 20 25 8 53 15 22 14 26 32 9 16 8	[5 10 15 20] [5 20 25] [10 15 25] [8 9 14 22] [8 14 15 16] [15 16 22]

Giải:

//1 : sắp xếp mảng thành tăng dần rồi quay lui

//2 : quay lui (phần tử thứ i, tổng hiện tại sum, vị trí cur hiện hại của số được cộng

// 3: lấy tổng của từng phần tử có trong mảng đến khi tổng = s thì in ra mảng

// 4: không thì tìm phần tử tiếp theo (j+1: nhưng phần tử này phải thỏa mãn nằm sau vị trí của phần tử phía trước có trong mảng)

```
#include<bits/stdc++.h>
using namespace std;
int n,s;
int a[11],b[11],ok,sum;
void out(int x){
    cout<<'[';
    for(int i=1;i<x-1;i++) cout<<b[i]<<' ';
    cout<<b[x-1]<<"] ";
}
void Try(int i,int sum,int cur){
```

```

if(sum==s){ out(i);ok=1;return;}//3
for(int j=cur;j<=n;j++){
    sum+=a[j];
    b[i]=a[j];
    if(sum<=s) Try(i+1,sum,j+1);//4:note j kkhông phải i
    sum-=a[j];
}
void initwsolve(){
    cin>>n>>s;ok=0;
    for(int i=1;i<=n;i++) cin>>a[i];
    sort(a+1,a+n+1);//1
    Try(1,0,1);//2
    if(ok==0) cout<<-1;
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

Cách 2 :

Note : trong hàm Try để có thể tiết kiệm dòng , không cần ghi nhận thêm kết quả vào sum và không trừ đi a[i] sau mỗi lần quay lui lại ta có thể viết:

```

void Try(int i,int sum,int cur){
    if(sum==s){ out(i);ok=1;return;}//3
    for(int j=cur;j<=n;j++){
        b[i]=a[j];
        if(sum+a[j]<=s) Try(i+1,sum+a[j],j+1);//4:note j kkhông phải i
    }
}

```

Tổng dãy con = K

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào trên bàn phím (n=5, K=50), 5 số dòng thứ 2 là các phần tử dãy A:

5	50			
5	10	15	20	25

Các dãy con thoả mãn điều kiện tìm được liệt kê trên màn hình:

- Mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một khoảng trắng.
- Dòng cuối cùng ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được.

10	15	25	
5	20	25	
5	10	15	20

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int n,k;
int a[1000],b[1000];
vector <vector<int> > res;
void out(int n){
    vector <int> temp;
    for(int i=1;i<n;i++) temp.push_back(b[i]);
    res.push_back(temp);
}
void Try(int i,int sum,int cur){
    if(sum==k) {out(i);return;}
    if(sum>k) return;
    for(int j=cur;j<=n;j++){
        sum+=a[j];
        b[i]=a[j];
        if(sum<=k)Try(i+1,sum,j+1);
        sum-=a[j];
    }
}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++) cin>>a[i];
    sort(a+1,a+n+1);
    Try(1,0,1);
    for(int i=res.size()-1;i>=0;i--){
        for(int j=0;j<res[i].size();j++) cout<<res[i][j]<<' ';
        cout<<endl;
    }
}
```

```

cout<<res.size()<<endl;
cout<<endl;

return 0;
}

```

Tổng dãy con = K 2

Cho dãy A[] gồm N số tự nhiên khác nhau và số tự nhiên K. Hãy viết chương trình liệt kê tất cả các dãy con của dãy số A[] sao cho tổng các phần tử trong dãy con đó đúng bằng K. Dữ liệu vào trên bàn phím ($n=5$, $K=50$), 5 số dòng thứ 2 là các phần tử dãy A:

5	50			
5	10	15	20	25

Các dãy con thỏa mãn điều kiện tìm được liệt kê trên màn hình:

- Mỗi dòng ghi lại một dãy con. Hai phần tử khác nhau của dãy con được viết cách nhau bởi một khoảng trắng.
- Dòng cuối cùng ghi lại số các dãy con có tổng các phần tử đúng bằng K tìm được.

10	15	25		
5	20	25		
5	10	15	20	
3				

Giai:

```

#include<bits/stdc++.h>
using namespace std;
int n,k;
int a[1000],b[1000];
vector <vector<int> > res;
void out(int n){
    vector <int> temp;
    for(int i=1;i<n;i++) temp.push_back(b[i]);
    res.push_back(temp);
}
void Try(int i,int sum,int cur){
    if(sum==k) { out(i);return; }
    if(sum>k) return;
    for(int j=cur;j<=n;j++){
        sum+=a[j];

```

```

        b[i]=a[j];
        if(sum<=k)Try(i+1,sum,j+1);
        sum-=a[j];
    }
}
int main(){
    cin>>n>>k;
    for(int i=1;i<=n;i++) cin>>a[i];
    sort(a+1,a+n+1);
    Try(1,0,1);
    for(int i=res.size()-1;i>=0;i--){
        for(int j=0;j<res[i].size();j++) cout<<res[i][j]<<' ';
        cout<<endl;
    }
    cout<<res.size()<<endl;
    cout<<endl;

    return 0;
}

```

TỔ HỢP SỐ CÓ TỔNG BẰNG X

Ôn lại

Cho mảng A[] gồm N số nguyên dương phân biệt và số X. Nhiệm vụ của bạn là tìm phép tổ hợp các số trong mảng A[] có tổng bằng X. Các số trong mảng A[] có thể được sử dụng nhiều lần. Mỗi tổ hợp các số của mảng A[] được in ra theo thứ tự không giảm các số. Ví dụ với A[] = {2, 4, 6, 8}, X = 8 ta có các tổ hợp các số như sau:

[2, 2, 2, 2], [2, 2, 4], [2, 6], [4, 4], [8].

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và X; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, X, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq X, A[i] \leq 100$. $N \leq 20$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Mỗi đường tổ hợp được bao bởi cặp ký tự [,]. Đưa ra -1 nếu không có tổ hợp nào thỏa mãn yêu cầu bài toán.

Input	Output
-------	--------

1	
4 8	[2 2 2] [2 2 4] [2 6] [4 4] [8]
2 4 6 8	

Giải :

//1 : sắp xếp mảng thành tăng dần rồi quay lui
 //2 : quay lui (phần tử thứ i, tổng hiện tại sum, vị trí cur hiện hại của số được cộng
 // 3: lấy tổng của từng phần tử có trong mảng đến khi tổng =s thì in ra mảng
 // 4: không thì tìm phần tử tiếp theo (j:cur: nhưng phần tử này phải thỏa mãn nằm từ vị trí của phần tử phía trước có trong mảng trở đi (có lặp lại))

```
#include<bits/stdc++.h>
using namespace std;
int n,s;
int a[30],b[30],ok,sum;//note test sai để 21 sẽ wrong
void out(int x){
    cout<<'[';
    for(int i=1;i<x-1;i++) cout<<b[i]<<';
    cout<<b[x-1]<<"]";//ko dấu cách
}
void Try(int i,int sum,int cur){
    if(sum==s){out(i);ok=1;return;}
    for(int j=cur;j<=n;j++){
        sum+=a[j];
        b[i]=a[j];
        if(sum<=s) Try(i+1,sum,j);//note j kkhông phải I . nếu để I nó sẽ quay
        lại //các số cũ : kiểu 2 2 4 sau lại 4 2 2....
        sum-=a[j];
    }
}
void initwsolve(){
    cin>>n>>s;ok=0;
    for(int i=1;i<=n;i++) cin>>a[i];
    sort(a+1,a+n+1);
    Try(1,0,1);
    if(ok==0) cout<<-1;
    cout<<endl;
}
int main(){
    int t;cin>>t;
```

```

while(t--) initwsolve();
return 0;
}

```

TỔ HỢP SỐ CÓ TỔNG BẰNG X

Cho mảng A[] gồm N số nguyên dương phân biệt và số X. Nhiệm vụ của bạn là tìm phép tổ hợp các số trong mảng A[] có tổng bằng X. Các số trong mảng A[] có thể được sử dụng nhiều lần. Mỗi tổ hợp các số của mảng A[] được in ra theo thứ tự không giảm các số. Ví dụ với A[] = {2, 4, 6, 8}, X = 8 ta có các tổ hợp các số như sau:

{2, 2, 2, 2}, {2, 2, 4}, {2, 6}, {4, 4}, {8}.

Input: Dòng đầu tiên đưa vào số lượng bộ test T. Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và X; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng. T, N, X, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq X, A[i] \leq 100$. $N \leq 20$.

Output: Đưa ra kết quả mỗi test theo từng dòng. Đầu tiên là số lượng tổ hợp thỏa mãn. Mỗi tổ hợp được bao bởi cặp ký tự {} và cách nhau một dấu cách. Đưa ra -1 nếu không có tổ hợp nào thỏa mãn yêu cầu bài toán.

Ví dụ:

Input	Output
2	5 {2 2 2 2} {2 2 4} {2 6} {4 4} {8}
4 8	-1
2 4 6 8	
2 9	
10 11	

```

#include<bits/stdc++.h>
using namespace std;
int n,s;
int a[30],b[30],ok,sum;//note test sai đê 21 sẽ wrong
vector <vector<int> > res;
void add(int n){
    vector<int>temp;
    for(int i=1;i<=n;i++)temp.push_back(b[i]);
    res.push_back(temp);
}
void Try(int i,int sum,int cur){
    if(sum==s){add(i-1);ok=1;return;}

```

```

for(int j=cur;j<=n;j++){
    sum+=a[j];
    b[i]=a[j];
    if(sum<=s) Try(i+1,sum,j);//note j kkhông phải I . nếu đế I nó sẽ quay
lại //các số cũ : kiểu 2 2 4 sau lại 4 2 2....
    sum-=a[j];
}
void initwsolve(){
    res.clear();
    cin>>n>>s;ok=0;
    for(int i=1;i<=n;i++) cin>>a[i];
    sort(a+1,a+n+1);
    Try(1,0,1);
    if(ok==0) cout<<-1<<endl;
    else cout<<res.size()<<" ";
    for(int i=0;i<res.size();i++){
        cout<<"{";
        for(int j=0;j<res[i].size()-1;j++) cout<<res[i][j]<<' ';
        cout<<res[i][res[i].size()-1]<<" } ";
    }
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

TẬP CON BẰNG NHAU

Ôn lại

Cho tập các số $A[] = (a_1, a_2, \dots, a_n)$. Hãy kiểm tra xem ta có thể chia tập $A[]$ thành hai tập con sao cho tổng các phần tử của hai tập con bằng nhau hay không. Dưa ra YES nếu có thể thực hiện được, ngược lại đưa ra NO.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N là số lượng phần tử của dãy số A[]; dòng tiếp theo đưa vào N phần tử của dãy số A[].
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 100$; $1 \leq A[i] \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
2	
4	
1 5 11 5	YES
3	NO
1 3 5	

Giải:

Cách 1: quay lui

// 1: tính tổng của mảng , nếu tổng là số lẻ \Rightarrow không chia được "NO" , nếu là chẵn thì bắt đầu tìm kiếm = quay lui

// 2: quay lui (phần tử thứ i, tổng hiện tại)

// 4: Nếu tổng hiện tại bằng nửa tổng mảng thì ok=1, nếu ok=1 (test đã đúng) thì kết thúc tìm kiếm

// 3: không thì tìm kiếm phần tử tiếp theo

```
#include<bits/stdc++.h>
using namespace std;
int n,a[105],ok,sum;
void Try(int i,int s){//2
    if(s==sum/2){ok=1;return;}
    if(ok==1)return;
/*4*/ for(int j=i;j<=n;j++){
        if(s+a[j]<=sum/2) Try(i+1,s+a[j]);
    }
}
void initwsolve(){
    cin>>n;ok=0;sum=0;
    for(int i=1;i<=n;i++) cin>>a[i],sum+=a[i];//1
    if(sum%2==1) {cout<<"NO"<<endl; return;}
    Try(1,0);
    if(ok==0) cout<<"NO"<<endl;
    else cout<<"YES"<<endl;
}
```

```

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

Cách 2 : Quy hoạch động:
// 1: tính tổng của mảng , nếu tổng là số lẻ => không chia được "NO" , nếu là chẵn
thì chia kết quả cho 2 để xét tổng các tập con tổng =sum/2
//dp[i] : có tồn tại tổng các số = i hay không , dp[i]=1 => có
//2 : duyệt từng giá trị của mảng
// 3: đánh dấu tất cả các tổng mà có sự xuất hiện của số đó và lớn hơn số đó(> để
tránh tràn mảng)
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,a[105],sum;
    cin>>n;sum=0;
    for(int i=0;i<n;i++) cin>>a[i],sum+=a[i];
/*1*/ if(sum%2==1) {cout<<"NO"<<endl; return;}
    else sum/=2;
    int dp[sum+1];
    dp[0]=1;
    for(int i=0;i<n;i++){//2
        for(int j=sum;j>=a[i];j--){//3
            if(dp[j-a[i]]==1) dp[j]=1;
        }
    }
    if(dp[sum]==1) cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

ĐỔI CHỖ CÁC CHỮ SỐ

Ôn lại

Nội dung bài tập

Cho số tự nhiên K và xâu ký tự các chữ số S. Nhiệm vụ của bạn là đưa ra số lớn nhất bằng cách thực hiện nhiều nhất K lần đổi chỗ các ký tự trong S. Ví dụ K=3 và S = “1234567” ta được “7654321”.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số K; dòng tiếp theo là xâu ký tự S.
- T, K, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K \leq 10$; $1 \leq \text{length}(S) \leq 7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input	Output
3	
4	
1234567	7654321
3	5543333
3435335	4301
2	
1034	

Giải :

// 1: duyệt từng phần tử trong mảng

//2 : sau đó tìm phần tử có giá trị lớn nhất sau vị trí đang xét

// 3: nếu tồn tại số lướn hơn thì swap

// 4: không thì k++ (vì vẫn chưa được swap)

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int k;cin>>k;
    string s;cin>>s;
    for(int i=0;i<k&&i<s.length();i++){//1
        int max=i,j=s.length();
        while (j-->i) {if (s[j]>s[max]) max=j;}
        /*3*/ if(max!=i) swap(s[i],s[max]);
        else k++;
    }
    cout<<s<<endl;
}
```

```
int main(){
```

```

int t;cin>>t;
while(t--) initwsolve();
return 0;
}

```

CHIA MẢNG (thành k tập con bằng nhau)

Ôn lại

Nội dung bài tập

Cho mảng các số nguyên $A[]$ gồm N phần tử. Hãy chia mảng số nguyên $A[]$ thành K tập con khác rỗng sao cho tổng các phần tử của mỗi tập con đều bằng nhau. Mỗi phần tử thuộc tập con xuất hiện duy nhất một lần trong tất cả các tập con. Ví dụ với $A[] = \{2, 1, 4, 5, 6\}$, $K=3$ ta có kết quả $\{2, 4\}, \{1, 5\}, \{6\}$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số N và K ; dòng tiếp theo đưa vào N số của mmảng $A[]$; các số được viết cách nhau một vài khoảng trắng.
- $T, N, K, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, K \leq 20$, $0 \leq A[i] \leq 100$.

Output:

- Đưa ra 1 nếu có thể chia tập con thành K tập thỏa mãn yêu cầu bài toán, ngược lại đưa ra 0.

Input	Output
2	
5 3	
2 1 4 5 6	1
5 3	0
2 1 5 5 6	

Giải :

```

// 1: tính tổng của mảng và xét nó có được chia làm k phần hay không ? nếu ko thì
in 0 nếu có thì chia làm k phần rồi tìm kiếm quay lui
// 2: xét tổng từng tập con bắt đầu lấy từ phần tử đầu tiên,
// 3:nếu đạt được 1 tập con tổng bằng sum thì tìm kiếm tập con tiếp theo
// 4: nếu tổng vẫn nhỏ hơn k thì tiếp tục tìm phần tử tiếp theo để cộng , còn nếu
>sum thì dừng
// 5 : thực hiện đến khi nào đạt được k tập con vừa ý thì ok=1,dừng . nếu ok=1 thì
dừng
#include<bits/stdc++.h>

```

```

using namespace std;
int n,k,a[21],ok,sum,check[21];
void Try(int s,int count){
    if(count==k){ok=1;return;}//5
    if(ok==1) return ;
    for(int j=1;j<=n;j++){//2
        if(check[j]==0){
            check[j]=1;
            if(s==sum) Try(0,count+1);//3
            else if(s<sum) Try(s+a[j],count); //4 ,
                else return;

        }
        check[j]=0;//note : để ngoài để đánh dấu các số đó đã được đưa vào tập con
        hay chưa
    }
}
void initwsolve(){
    cin>>n>>k;
    sum=0;ok=0;//note
    for(int i=1;i<=n;i++) cin>>a[i],sum+=a[i];
/*1*/if(sum%k!=0) {cout<<0<<endl; return;}
    else sum/=k;
    Try(0,0);
    if(ok==0) cout<<0<<endl;
    else cout<<1<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

DI CHUYỂN TRONG MA TRẬN

Nội dung bài tập

Cho ma trận $A[M][N]$. Nhiệm vụ của bạn là đưa ra tất cả các đường đi từ phần tử $A[0][0]$ đến phần tử $A[M-1][N-1]$. Bạn chỉ được phép dịch chuyển xuống dưới hoặc sang phải phần tử liền kề với vị trí hiện tại.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là hai số M, N tương ứng với số hàng và số cột của ma trận; dòng tiếp theo đưa vào các phần tử của ma trận A[][]; các số được viết cách nhau một vài khoảng trắng.
- T, M, N, A[i][j] thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq M, N, A[i][j] \leq 100$.

Output:

- Đưa ra số cách di chuyển của mỗi test theo từng dòng.
- Giải thích test 1: Có 3 cách di chuyển là [1 4 5 6], [1 2 5 6] và [1 2 3 6].

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

Giải :

// 1: trong các đường đi nếu chỉ đi thẳng sang phải , hoặc đi thẳng xuống dưới thì có 1 cách đi.
// 2: còn lại tại một vị trí bất kì khác trong ma trận thì số cách tại vị trí đó là tổng ô ở trên và bên trái
//3 : ô cuối cùng của mảng 2 chiều chính là số cách cần tìm

```
#include<bits/stdc++.h>
using namespace std;
int m,n,a[101][101];
void initwsolve(){
    cin>>m>>n;
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            cin>>a[i][j];
            if(i==0||j==0) a[i][j]=1;//1
            else a[i][j]=a[i-1][j]+a[i][j-1];//2
        }
    }
}
```

```

cout<<a[m-1][n-1]<<endl;//3
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

SỐ NGUYÊN TỐ

Ôn lại

Cho ba số N, P, S. Trong đó, P là một số nguyên tố. Nhiệm vụ của bạn là đưa ra tất cả N số nguyên tố tính từ P có tổng bằng S. Ví dụ với S = 28, P=7, N =2 ta có kết quả $11 + 17 = 28$. Với N = 3, P = 2, S = 23 ta có kết quả : {3, 7, 13}, {5, 7, 11}

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ ba số S, P, N được viết trên một dòng.
- S, P, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10$; $2 \leq S, P \leq 200$.

Output:

- Với mỗi test, dòng đầu tiên in ra số lượng đáp án tìm được. Mỗi dòng tiếp theo in ra kết quả tìm được theo thứ tự từ điển.

Input	Output
2	1
2 7 28	11 17
3 2 23	2
	3 7 13
	5 7 11

Giải :

// Nhận xét : do n,p,s biến thiên sau mỗi test nên mảng các snt >p cũng biến thiên , mảng kết quả từng test (v) và kết quả tổng để in ra (res) cũng thay đổi => tất cả dùng vector để dễ tính toán/

// 1: b1: tạo một vector bao gồm các số nguyên tố (//2) từ sau p đến nhỏ hơn s
// 3: thực hiện quay lui tìm kiếm (phần tử thứ i của mảng , tổng hiện tại, biến đếm số các phần tử hiện tại trong mảng)

// 4: duyệt từng phần tử trong mảng nguyên tố prime và tìm tổng dần dần với các phần tử sau nó (j+1)

// 5: nếu tìm được tổng =sum và số phần tử đúng bằng n thì đưa kết quả vào res , nếu > thì dừng , nếu < thì tiếp tục tìm kiếm

```

#include<bits/stdc++.h>
using namespace std;
int n,p,sum;
vector <int> prime,v;
vector <vector <int> > res;
int /*2*/ snt(int n){
    if(n<2) return 0;
    for(int i=2;i<=sqrt(n);i++)
        if(n%i==0) return 0; return 1;
}
void Try(int i,int s,int count){//3
/*5*/ if(s==sum&&count==n){
    res.push_back(v);return;
}
if(s>sum||count>n) return;
/*4*/ for(int j=i;j<prime.size();j++){
    v.push_back(prime[j]);
    Try(j+1,s+prime[j],count+1);//j+1
    v.pop_back();
}
}
void initwsolve(){
    cin>>n>>p>>sum;
    res.clear(),prime.clear();
/*1*/ for(int i=p+1;i<=sum;i++){
    if(snt(i)==1) prime.push_back(i);
}
/*3*/ Try(0,0,0);
cout<<res.size()<<endl;
/*in*/ for(int i=0;i<res.size();i++){
    for(int j=0;j<res[i].size();j++){
        cout<<res[i][j]<<' ';
    }
    cout<<endl;
}
}
int main(){
    int t;cin>>t;

```

```

        while(t--) initwsolve();
        return 0;
    }
}

```

TỪ ĐIỂN

Ôn lại

Cho tập từ ghi trong từ điển dic[] và một bảng hai chiều A[M][N] các ký tự. Hãy tạo nên tất cả các từ có mặt trong từ điển dic[] bằng cách nối các ký tự kề nhau trong mảng A[][]]. Chú ý, phép nối các ký tự kề nhau trong mảng A[][] được thực hiện theo 8 hướng nhưng không có phần tử A[i][j] nào được lặp lại. Ví dụ với từ điển dic[] = { “GEEKS”, “FOR”, “QUIZ”, “GO”} và mảng A[][] dưới đây sẽ cho ta kết quả: “GEEKS”, “QUIZ”

G	I	Z
U	E	K
Q	S	E

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào ba số K, M, N tương ứng với số từ của từ điển dic[], số hàng và số cột của ma trận ký tự A[M][N]; dòng tiếp theo đưa vào K từ của từ điển dic[]; dòng cuối cùng đưa vào các phần tử A[i][j].
- T, K, M, N thỏa mãn ràng buộc: $1 \leq T \leq 10$; $1 \leq K \leq 100$; $1 \leq M, N \leq 3$.

Output:

- Đưa ra theo thứ tự tăng dần các từ có mặt trong từ điển dic[] được tạo ra từ ma trận A[][]]. Đưa ra -1 nếu không thể tạo ra từ nào thuộc dic[] từ A[][].

Input	Output
1 4 3 3 GEEKS FOR QUIZ GO G I Z U E K Q S E	GEEKS QUIZ

Giải:

//1 : duyệt tất cả các chuỗi có thể tạo được và kiểm tra chuỗi đó có trong từ điển dic không , nếu có thì đưa vào kết quả

```

//2: Try(vị trí hàng, vị trí cột , chuỗi tạo được hiện tại)
// 3: tại vị trí đang có lựa chọn tiếp tất cả các ký tự thỏa mãn (3) .
//4 : tại từng vị trí đó xem đã được đánh dấu chưa , nếu rồi thì thôi , nếu chưa thì
tiếp tục tìm các ký tự tiếp theo của chuỗi.
#include <bits/stdc++.h>
using namespace std;
int k,m,n,check[3][3];
char a[3][3];
string dic[105],str;
vector <string> res;
bool isword(string str){//1
    for(int i=0;i<k;i++)
        if(str==dic[i]) return 1; return 0;
}
void Try (int i,int j,string str){//2
    check[i][j]=1;
    str=str+a[i][j];
    if(isword(str)==1) res.push_back(str);
/*3*/ for(int row=i-1;row<m&&row<=i+1 ;row++ ){
    for(int col=j-1;col<n&&col<=j+1 ;col++ ){
        if(check[row][col]==0&&col>=0&&row>=0)//4
            Try(row,col,str);
    }
}
str.erase(str.length()-1);
check[i][j]=0;
}
void initwsolve(){
    res.clear(),str.clear();
/*init*/cin>>k>>m>>n;
    for(int i=0;i<k;i++) cin>>dic[i];
    for(int i=0;i<m;i++)
        for(int j=0;j<n;j++)
            cin>>a[i][j];
/*1*/ for(int i=0;i<m;i++)
    for(int j=0;j<n;j++)
        Try(i,j,str);
/*out*/if(res.size()==0) cout<<-1<<endl;

```

```

        else{
            for(int i=0;i<res.size();i++) cout<<res[i]<<' ';
            cout<<endl;
        }
    }
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

BÀI 32. LOẠI BỎ DẤU NGOẶC chưa làm

Nội dung bài tập

Cho biểu thức P chỉ chứa các ký tự ‘(’, ‘)’ và các ký tự. Không có phép toán nào trong biểu thức P. Nhiệm vụ của bạn là thực hiện ít nhất các phép loại bỏ các ký tự ‘(’, ‘)’ để P trở thành biểu thức đúng. Chú ý: một biểu thức chỉ có 1 ký tự chữ (không có dấu ngoặc) hoặc một biểu thức rỗng thì không được xem là biểu thức đúng.

Nếu có nhiều hơn một biểu thức đúng với cùng số phép loại bỏ ít nhất hãy đưa ra tất cả các biểu thức đúng theo thứ tự từ điển.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một biểu thức P được viết trên một dòng.
- T, P thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(P) \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Nếu không có đáp án, in ra -1.

Input	Output
2 00)0 (u)())()	(0)0 000 (u0)0 (u)00

SẮP XẾP QUÂN HẬU 1

Ôn lại

Cho một bàn cờ vua có kích thước $n * n$, ta biết rằng quân hậu có thể di chuyển theo chiều ngang, dọc, chéo. Vấn đề đặt ra rằng, có n quân hậu, bạn cần đếm số

cách đặt n quân hậu này lên bàn cờ sao cho với 2 quân hậu bất kì, chúng không “ăn” nhau.

Input: Dòng đầu ghi số bộ test T ($T < 5$). Mỗi bộ test ghi một số nguyên dương n duy nhất (không quá 10)

Output: Ghi kết quả mỗi bộ test trên một dòng. Số cách đặt quân hậu.

Ví dụ:

Input	Output
1	
4	2

Giải:

// 1: cố định đặt được 1 con hậu ở đầu mảng, thực hiện quay lui (int i: số quân hậu đặt được hiện tại)
// 2: tại một vị trí có thể đặt được quân hậu ta duyệt đánh dấu toàn bộ hàng, cột, đường chéo chính, đường chéo phụ đi qua nó
// 3: cho đến khi tìm được n vị trí thì tăng biến điểm không tiếp tục tìm.

```
#include <bits/stdc++.h>
using namespace std;
int n,res,hang[21],cot[21],xuoi[21],nguoc[21];
void Try (int i){//1
    for(int j=1;j<=n;j++){
        /*2*/ if(!hang[j]&&!cot[j]&&!xuoi[i-j+n]&&!nguoc[i+j-1]){
            hang[j]=cot[j]=xuoi[i-j+n]=nguoc[i+j-1]=1;
            if(i==n) res++; //3
            else Try(i+1);
            hang[j]=cot[j]=xuoi[i-j+n]=nguoc[i+j-1]=0;
        }
    }
}
int main() {
    int t;cin>>t;
    while(t--){
        res=0;
        cin>>n;
        Try(1);//1
        cout<<res<<endl;
    }
}
```

```
    return 0;  
}
```

SẮP XẾP QUÂN HẬU 2

Ôn lại

Nội dung bài tập

Cho một bàn cờ 8×8 , mỗi ô có một giá trị $A[i][j]$ nhất định ($0 \leq A[i][j] \leq 100$), tương ứng với điểm số đạt được nếu như bạn đặt một quân cờ vào đó.

Nhiệm vụ của bạn là đặt 8 quân hậu lên bàn cờ, sao cho không có 2 quân nào ăn nhau, và số điểm đạt được là lớn nhất.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 8 dòng, mỗi dòng 8 số nguyên mô tả bàn cờ.

Output: Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 48 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64	260

Giải :

```
// 1: cố định đặt được 1 con hậu ở đầu mảng, thực hiện quay lui (int i,int sum: số  
quân hậu đặt được hiện tại, tổng hiện tại)  
// 2: tại một vị trí có thể đặt được quân hậu ta duyệt đánh dấu toàn bộ hàng , cột ,  
đường chéo chính , đường chéo phụ đi qua nó  
// 3: cho đến khi tìm được 8 vị trí thì lấy giá trị sum >>, chưa đến 8 thì không tiếp  
tục tìm.
```

```
#include <bits/stdc++.h>  
using namespace std;  
int n,res,sum,a[10][10],hang[21],cot[21],xuoi[21],nguoc[21];  
void Try (int i,int sum){//1
```

```

for(int j=1;j<=8;j++){
/*2*/ if(!hang[j]&&!cot[j]&&!xuoi[i-j+8]&&!nguoc[i+j-1]){
    sum+=a[i][j];
    hang[j]=cot[j]=xuoi[i-j+8]=nguoc[i+j-1]=1;
    if(i==8) res=max(res,sum);//3
    else Try(i+1,sum);
    sum-=a[i][j];
    hang[j]=cot[j]=xuoi[i-j+8]=nguoc[i+j-1]=0;
}
}
int main() {
    int t;cin>>t;
    while(t--){
        res=0;
        /*init*/for(int i=1;i<=8;i++)
            for(int j=1;j<=8;j++)
                cin>>a[i][j];
        Try(1,0);//1
        cout<<res<<endl;
    }
    return 0;
}

```

TẬP HỢP (đếm số tập hợp số lượng =k tổng =s)

Ôn lại

Xét tất cả các tập hợp các số nguyên dương có các phần tử khác nhau và không lớn hơn số n cho trước. Nhiệm vụ của bạn là hãy đếm xem có tất cả bao nhiêu tập hợp có số lượng phần tử bằng k và tổng của tất cả các phần tử trong tập hợp bằng s?

Các tập hợp là hoán vị của nhau chỉ được tính là một.

Ví dụ với $n = 9$, $k = 3$, $s = 23$, $\{6, 8, 9\}$ là tập hợp duy nhất thỏa mãn.

Input: Gồm nhiều bộ test (không quá 100 test).

Mỗi bộ test gồm 3 số nguyên n , k , s với $1 \leq n \leq 20$, $1 \leq k \leq 10$ và $1 \leq s \leq 155$.

Input kết thúc bởi 3 số 0.

Output: Với mỗi test in ra số lượng các tập hợp thỏa mãn điều kiện đề bài.

Ví dụ:

Input	Output
-------	--------

9 3 23	1
9 3 22	2
10 3 28	0
16 10 107	20
20 8 102	1542
20 10 105	5448
20 10 155	1
3 4 3	0
4 2 11	0
0 0 0	

Giải :

// 1 : ý tưởng : duyệt tổ hợp có k phần tử xét xem tổng có bằng s không
 // 2: note : khi i==k (đạt được tổ hợp k phần tử đã) thì mới xét tổng của nó có bằng s không => 2 if

```
#include <bits/stdc++.h>
using namespace std;
int n,k,s,sum,res,a[11],check[21];
void Try(int i,int sum){//1
    for(int j=a[i-1]+1;j<=n-k+i;j++){
        if(check[j]==0){
            check[j]=1;
            a[i]=j;
            sum+=a[i];
            if(i==k) {if(sum==s)res++;}//2
            else Try(i+1,sum);
            check[j]=0;
            sum-=a[i];
        }
    }
}
int main() {
    while(1){
        res=0;
        cin>>n>>k>>s;
        if(n==0&&s==0&&k==0){cout<<endl; break;}
        Try(1,0);
        cout<<res<<endl;
    }
}
```

```
    return 0;  
}
```

BIỂU THỨC TOÁN HỌC (=23)

Ôn lại

Nội dung bài tập

Cho 5 số nguyên dương A, B, C, D, E. Bạn có thể hoán vị các phần tử cho nhau, hãy đặt các dấu biểu thức +, -, * sao cho biểu thức sau đúng:

$$[[[A o(1) B] o(2) C] o(3) D] o(4) E = 23$$

Trong đó: o(1) ... o(4) là các phép toán +, -, *.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 5 số nguyên dương A, B, C, D, E có giá trị không vượt quá 100.

Output: Với mỗi test, in ra đáp án tìm được, mỗi xâu in ra trên một dòng.

Ví dụ:

Input	Output
3	
1 1 1 1 1	NO
1 2 3 4 5	YES
2 3 5 7 11	YES

Giải :

// 1: thực hiện duyệt hoán vị đánh số mảng 5 phần tử đến khi được thì
// 2: tiếp tục thực hiện duyệt hoán vị các phép tính +,-,* giữa 2 toán tử trong mảng
đó

// 3: kiểm tra xem một tổng của mảng gồm 5 phần tử đó có bằng 23 hay không

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

```
using namespace std;
```

```
int a[6],x[6],cal[6],res=0,check[6];
```

```
bool out() { //3
```

```
    long long temp;
```

```
    if(cal[1]==1) temp= a[x[1]]+ a[x[2]];
```

```
    if(cal[1]==2) temp= a[x[1]]- a[x[2]];
```

```
    if(cal[1]==3) temp= a[x[1]]* a[x[2]];
```

```
    for(int i=3;i<=5;i++){
```

```
        if(cal[i-1]==1) temp+= a[x[i]];
```

```
        if(cal[i-1]==2) temp-= a[x[i]];
```

```

        if(cal[i-1]==3) temp*= a[x[i]];
    }
    return temp==23;
}
void Try2(int i){
    for(int j=1;j<=3;j++){
        cal[i]=j;
        if(i==4) {if(out()==1) res=1;}
        else Try2(i+1);
    }
}
void Try(int i){// 1
    for(int j=1;j<=5;j++){
        if(!check[j]){
            check[j]=1;
            x[i]=j;
            if(i==5) Try2(1);
            else Try(i+1);
            check[j]=0;
        }
    }
}
int main() {
    int t;cin>>t;
    while(t--){
        res=0;
        for(int i=1;i<=5;i++) cin>>a[i];
        Try(1);
        if(res==0) cout<<"NO" << endl;
        else cout<<"YES" << endl;
    }
    return 0;
}

```

ĐƯỜNG ĐI DÀI NHẤT

Nội dung bài tập

Cho đồ thị vô hướng có N đỉnh và M cạnh. Bạn hãy tìm đường đi dài nhất trên đồ thị, sao cho mỗi cạnh chỉ được đi qua nhiều nhất 1 lần.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 10$). Mỗi test bắt đầu bằng số nguyên N và M ($1 \leq N, M \leq 20$). Các đỉnh đánh dấu từ 0, 1, ..., N-1. M dòng tiếp theo, mỗi dòng gồm 2 số u, v cho biết có cạnh nối giữa u và v.

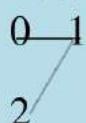
Output: Với mỗi test, in ra đáp án tìm được, mỗi xâu in ra trên một dòng.

Ví dụ

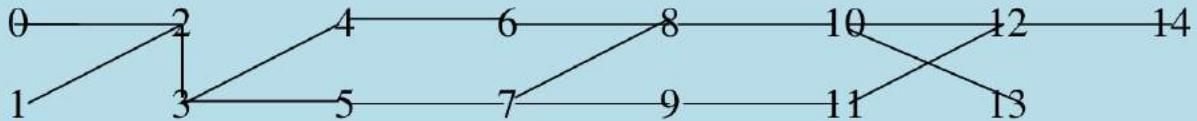
Input	Output
2	
3 2	
0 1	
1 2	
15 16	
0 2	
1 2	
2 3	
3 4	
3 5	
4 6	2
5 7	12
6 8	
7 8	
7 9	
8 10	
9 11	
10 12	
11 12	
10 13	
12 14	

Hình vẽ :

Test 1 : nối 1 nét sao cho có nhiều cạnh nhất có thể



Test 2:



Giải :

```

//note :Depth-first search - DFS là một thuật toán duyệt hoặc tìm kiếm trên một
cây hoặc một đồ thị.
//1:dfs() (vị trí của đỉnh u, độ dài chuỗi >> ô hiện tại)
//2:tại một đỉnh u duyệt các đỉnh v từ 0=>n
//3: nếu tại vị trí đỉnh u mà nối được với đỉnh v thì đánh dấu nó là ko nối được nữa
( đế max=1) và tiếp tục tìm đỉnh tiếp theo khi đó vị trí lại là đỉnh v và duyệt tiếp
tìm đỉnh thỏa mãn
#include<bits/stdc++.h>
using namespace std;
int m,n,a[21][21],res;
void dfs(int u,int temp){//1:
    res=max(res,temp);
    for(int v=0;v<n;v++){//2
        /*3*/if(a[u][v]==1){
            a[u][v]=a[v][u]=0;
            dfs(v,temp+1);
            a[u][v]=a[v][u]=1;
        }
    }
}
void initwsolve(){
    int u,v;res=-1;
    memset(a,0,sizeof(a));
    cin>>n>>m;
/*init*/for(int i=0;i<m;i++){
    cin>>u>>v;
    a[u][v]=a[v][u]=1;
}
for(int i=0;i<n;i++) dfs(i,0);
cout<<res<<endl;
}
int main(){

```

```

int t;cin>>t;
while(t--) initwsolve();
return 0;
}

```

SỐ NHỎ NHẤT CÓ N ƯỚC SỐ

Ôn lại

Cho số nguyên dương N. Nhiệm vụ của bạn là tìm số K nhỏ nhất, sao cho K có đúng N ước. Input đảm bảo rằng đáp án không vượt quá 10^{18} .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 1 số nguyên N ($1 \leq N \leq 1000$).

Output: Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
2	
4	6
6	12

Giải :

```

//1: :(phân tử thứ i của mảng, số ước hiện tại, số gồm aliquote(ước số) ước hiện
tại)
// note: x=a^x*b^y thi so ali cua x la (x+1)*(y+1)
#include<bits/stdc++.h>
using namespace std;
int n;long long res;
int p[]={2,3,5,7,11,13,17,19,23,29,31};
void Try(int i,long long ali,long long temp){
    if(ali>n) return;
    if(ali==n) res=min(temp,res);
    for(int j=1; ;j++){
        if(temp*p[i]>res) break;
        temp*=p[i];
        Try(i+1,ali*(j+1),temp);
    }
}
int main(){

```

```

int t;cin>>t;
while(t--){
    cin>>n;res=1e18;
    Try(0,1,1);//
    cout<<res<<endl;
}
return 0;
}

```

KÝ TỰ ĐẶC BIỆT

Ôn lại

Cho một xâu s. Xâu F(s) được xác định bằng cách ghép xâu xâu s ban đầu với xâu s sau khi đã được quay vòng sang bên phải 1 kí tự (kí tự cuối cùng của s được chuyển lên đầu).

Thực hiện liên tiếp các bước cộng xâu như trên với xâu mới thu được, ta có được xâu X.

Nhiệm vụ của bạn là hãy xác định kí tự thứ N trong xâu X là kí tự nào?

Input: Dòng đầu ghi số bộ test T ($T < 10$). Mỗi bộ test gồm một xâu s có độ dài không vượt quá 30 kí tự và số nguyên N ($1 \leq N \leq 10^{18}$).

Output: Với mỗi bộ test ghi ra trên một dòng kí tự tìm được.

Ví dụ:

Input	Output
1	C
COW 8	

Giải thích test: COW à COWWCO à COWWCOOCOWWC. Kí tự thứ 8 là ‘C’.

COW

COWWCO

COWWCOOCOWWC ($l=12, n=8$)

Giải :

//1 :nhân độ dài chuỗi (l) cho đến khi nào lớn hơn vị trí cần tìm (n)

//2: nếu $n <$ độ dài chuỗi ban đầu nhập vào thì trả về chỉ số $n-1$, dừng tìm kiếm

//3 :thực hiện tách mảng nếu vị trí tìm ở bên nửa phải của mảng thì trừ đi để đưa về y hệt nhưng nửa trái

// 4: nếu trừ đi được $n=0$ (tức vị trí cần tìm ở giữa mảng ban đầu) thì gán lại nó bằng L/2

```

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

```

```

long long n;
long long Try(long long l){
    if(n<=s.length()) return n-1;//2
    if(n>l/2) n= n-l/2-1; //3
    if(n==0) n=l/2;//4
    return Try(l/2);
}
int main(){
    int t; cin>>t;
    while(t--){
        cin>>s>>n;
        long long l=s.length();
        while (l<n) l*=2;//1
        cout<<s[Try(l)]<<endl;
    }
    return 0;
}

```

NGƯỜI DU LỊCH

Ôn lại

Cho n thành phố đánh số từ 1 đến n và các tuyến đường giao thông hai chiều giữa chúng, mạng lưới giao thông này được cho bởi mảng $C[1\dots n, 1\dots n]$ ở đây $C[i][j] = C[j][i]$ là chi phí di đoạn đường trực tiếp từ thành phố i đến thành phố j . Một người du lịch xuất phát từ thành phố 1, muốn đi thăm tất cả các thành phố còn lại mỗi thành phố đúng 1 lần và cuối cùng quay lại thành phố 1. Hãy chỉ ra chi phí ít nhất mà người đó phải bỏ ra.

Dữ liệu vào: Dòng đầu tiên là số nguyên n – số thành phố ($n \leq 15$); n dòng sau, mỗi dòng chứa n số nguyên thể hiện cho mảng 2 chiều C .

Kết quả: Chi phí mà người đó phải bỏ ra.

Ví dụ:

INPUT	OUTPUT
4 0 20 35 10 20 0 90 50 35 90 0 12 10 50 12 0	117

Giải :

```

// 1: điểm đặt chân ở thành phố đầu tiên ( phần tử mảng x vai trò như hàng , j vai
trò như cột)
// 2:nhánh cận (i: số thành phố đi qua hiện tại)
// 3 :nếu đi qua đủ n thành phố thì xét tổng hiện tại+chi phí về thành phố 1 có > res
không , nếu nhỏ hơn thì cập nhật
// 4: còn không nếu tổng hiện tại +(n-i)thành phố chưa đi qua + 1 (chi phí về
thành phố 1) mà < res thì sẽ tìm thành phố tiếp theo . nếu lớn hơn thì không vì cho
dù nếu đi qua n thành phố thì vẫn > res.
#include<bits/stdc++.h>
using namespace std;
int n,c[16][16],x[16],check[16],res=1e18,cmin=1e18,s;
void Try(int i){//2
    for(int j=2;j<=n;j++){
        if(!check[j]){
            check[j]=1;
            x[i]=j;
            s+=c[x[i-1]][j];
            if(i==n){//3
                if(s+c[j][1]<res) res=s+c[j][1];
            }
            else if(s+(n-i+1)*cmin<res) Try(i+1);//4
            s-=c[x[i-1]][j];
            check[j]=0;
        }
    }
}
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>c[i][j];
            if(c[i][j]<cmin&&c[i][j]!=0) cmin=c[i][j];
        }
    }
    x[1]=1;//1
    Try(2);
    cout<<res<<endl;
    return 0;
}

```

}

Giải thuật tham lam

ĐỔI TIỀN

Tại ngân hàng có các mệnh giá bằng 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000. Tổng số tiền cần đổi có giá trị bằng N. Hãy xác định xem có ít nhất bao nhiêu tờ tiền sau khi đổi tiền?

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 50$). Mỗi test gồm 1 số nguyên N ($1 \leq N \leq 100\,000$).

Output: Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input:	Output
2	
70	2
121	3

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int n;
int a[]={1, 2, 5, 10, 20, 50, 100, 200, 500, 1000};
void initwsolve(){
    cin>>n;
    int res=0;
    for(int i=9;i>=0;i--){
        if(n<=0) break;
        res+=n/a[i];
        n=n%a[i];
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

}

NHÀM CHỮ SỐ

Trong một buổi học toán, giáo viên viết 2 số nguyên, A và B, và yêu cầu Tèo thực hiện phép cộng. Tèo không bao giờ tính toán sai, nhưng thỉnh thoảng cậu ta chép các con số một cách không chính xác. Lỗi duy nhất của là ghi nhầm '5' thành '6' hoặc ngược lại. Cho hai số, A và B, tính tổng nhỏ nhất và lớn nhất mà Tèo có thể nhận được.

Input: Có một dòng chứa hai số nguyên dương A và B ($1 \leq A, B \leq 1\,000\,000$).

Output: In ra 2 số nguyên cách nhau một dấu cách, tổng nhỏ nhất và lớn nhất có thể nhận được.

Ví dụ:

Test 1	Test 2	Test 3
Input: 11 25	Input: 1430 4862	Input: 16796 58786
Ouput: 36 37	Ouput: 6282 6292	Ouput: 74580 85582

Giải :

Chú ý : Hàm chuyển từ một chuỗi sang một số : int convert(string s){

```
int res=0;
for(int i=0;i<s.length();i++){
    res=10*res+s[i]-48;
}
return res;
```

}

Giải

//1: chuyển hết chuỗi a,b thành chuỗi không có số 6 để sau đó tính tổng nhỏ nhất

//2: chuyển hết chuỗi a,b thành chuỗi không có số 5 để sau đó tính tổng lớn nhất

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

```
#include<stdlib.h>
```

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int convert(string s){
```

```
    int res=0;
```

```
    for(int i=0;i<s.length();i++){
```

```

        res=10*res+s[i]-48;
    }
    return res;
}
int main(){
    string a,b;
    cin>>a>>b;
    int min=0,max=0;
    for(int i=0;i<a.length();i++){//1
        if(a[i]=='6') a[i]='5';
    }
    for(int i=0;i<b.length();i++){//1
        if(b[i]=='6') b[i]='5';
    }
    cout<<convert(a)+convert(b) << " ";//1
    for(int i=0;i<a.length();i++){//2
        if(a[i]=='5') a[i]='6';
    }
    for(int i=0;i<b.length();i++){//2
        if(b[i]=='5') b[i]='6';
    }
    cout<<convert(a)+convert(b) << endl;//2
}

```

TÌM MAX

Cho mảng A[] gồm N phần tử. Nhiệm vụ của bạn là

tìm $\max = \sum_{i=0}^{n-1} A_i * i$ bằng cách sắp đặt lại các phần tử trong mảng. Chú ý, kết quả của bài toán có thể rất lớn vì vậy bạn hãy đưa ra kết quả lấy modulo với 10^9+7 .

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
5	
5 3 2 4 1	40
3	8
1 2 3	

Giải:

```
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
int n;
void initwsolve(){
    cin>>n;
    int a[n];
    long long res=0;
    for(int i=0;i<n;i++)cin>>a[i];
    sort(a,a+n);
    for(int i=0;i<n;i++){
        res= (res+a[i]*i)%mod;
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TỔNG NHỎ NHẤT

Cho mảng A[] gồm các số từ 0 đến 9. Nhiệm vụ của bạn là tìm tổng nhỏ nhất của hai số được tạo bởi các số trong mảng A[]. Chú ý, tất cả các số trong mảng A[] đều được sử dụng để tạo nên hai số.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương

ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trắng.

- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 20$; $0 \leq A[i] \leq 9$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
6	
6 8 4 5 2 3	604
5	82
5 3 0 7 4	

Giải:

//vòng for dùng để đây xem kẽ lần lượt các phần tử trong mảng tăng dần vào 2 số a và b

```
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
int n;
void initwsolve(){
    cin>>n;
    int a[n];
    long long x=0,y=0;
    for(int i=0;i<n;i++)cin>>a[i];
    sort(a,a+n);
    for(int i=0;i<n;i++){//1
        if(i%2==0) x=x*10+a[i];
        else y=y*10+a[i];
    }
    cout<<x+y<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

CHIA MẢNG THÀNH HAI MẢNG CON CÓ TỔNG LỚN NHẤT

Ôn lại

Cho mảng A[] gồm N số nguyên không âm và số K. Nhiệm vụ của bạn là hãy chia mảng A[] thành hai mảng con có kích cỡ K và N-K sao cho hiệu giữa tổng hai mảng con là lớn nhất. Ví dụ với mảng A[] = {8, 4, 5, 2, 10}, K=2 ta có kết quả là 17 vì mảng A[] được chia thành hai mảng {4, 2} và { 8, 5, 10} có hiệu của hai mảng con là $23 - 6 = 17$ là lớn nhất.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N và số K; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, K, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K < N \leq 50$; $0 \leq A[i] \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2 5 2 8 4 5 2 10 8 3 1 1 1 1 1 1 1 1	17 2

Giải :

//1 : chú ý mảng các số nhỏ luôn phải là mảng có ít phần tử hơn nên => đánh dấu ngăn cách mảng nhỏ và lớn thì mark=min(k,n-k)

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

```
using namespace std;
```

```
int n,k;
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    long long a[n];
```

```
    for(int i=0;i<n;i++)cin>>a[i];
```

```
    sort(a,a+n);
```

```
    int mark=min(k,n-k);//1
```

```
    for(int i=1;i<n;i++) a[i]+=a[i-1];
```

```
    cout<a\[n-1\]-2\*a\[mark-1\]<<endl;
```

```
}
```

```

int main(){
    int t; cin >> t;
    while(t--) initwsolve();
    return 0;
}

```

SẮP XẾP THAM LAM

Ôn lại

Cho mảng A[] gồm N số và thực hiện các thao tác theo nguyên tắc dưới đây:

- Ta chọn một mảng con sao cho phần tử ở giữa của mảng con cũng là phần tử ở giữa của mảng A[] (trong trường hợp N lẻ).
- Đảo ngược mảng con đã chọn trong mảng A[]. Ta được phép chọn mảng con và phép đảo ngược mảng con bao nhiêu lần tùy ý.

Ví dụ với mảng A[] = {1, 6, 3, 4, 5, 2, 7} ta có câu trả lời là Yes vì: ta chọn mảng con {3, 4, 5} và đảo ngược để nhận được mảng A[] = {1, 6, 5, 4, 3, 2, 7}, chọn tiếp mảng con {6, 5, 4, 3, 2} và đảo ngược ta nhận được mảng A[] = {1, 2, 3, 4, 5, 6, 7}. Hãy cho biết ta có thể sắp xếp được mảng A[] bằng cách thực hiện các thao tác kể trên hay không?

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số phần tử của mảng N; dòng tiếp theo đưa vào N số A[i] tương ứng với các phần tử của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 50$; $0 \leq A[i] \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2 7 1 6 3 4 5 2 7	Yes
7 1 6 3 4 5 7 2	No

Giải :

```

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

```

```

int n;
void initwsolve(){
    cin>>n;
    int a[n],b[n];
    for(int i=0;i<n;i++){cin>>a[i]; b[i]=a[i];}
    sort(b,b+n);
    for(int i=0;i<n;i++){
        if((a[i]!=b[i]&&a[n-i-1]!=b[i])){
            cout<<"No" << endl;
            return;
        }
    }
    cout<<"Yes" << endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

NỘI DÂY 1

Cho N sợi dây với độ dài khác nhau được lưu trong mảng $A[]$. Nhiệm vụ của bạn là nối N sợi dây thành một sợi sao cho tổng chi phí nối dây là nhỏ nhất. Biết chi phí nối sợi dây thứ i và sợi dây thứ j là tổng độ dài hai sợi dây $A[i]$ và $A[j]$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào số lượng sợi dây N ; dòng tiếp theo đưa vào N số $A[i]$ là độ dài của các sợi dây; các số được viết cách nhau một vài khoảng trắng.
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^6$; $0 \leq A[i] \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
4	
4 3 2 6	29
5	62
4 2 7 6 9	

Giải :

//1:do cần tổng chi phí nối dây là nhỏ nhất=> mỗi lần lấy 2 số nhỏ nhất tìm tổng đưa nó vào hàng đợi nhưng sau đó lại tiếp tục lấy 2 số nhỏ nhất => bản thân tổng sau khi đưa vào hàng đợi cũng phải được sắp xếp tăng dần => dùng hàng đợi ưu tiên priority_queue

// 2: lấy giá trị và loại bỏ 2 số nhỏ nhất của hàng đợi

// 3: lấy tổng 2 số đó và đưa vào hàng đợi, cập nhật sum1 chính là kết quả cần tìm
+=sum;

// 4: nếu như hàng đợi rỗng rồi thì kết thúc vòng lặp

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    int n;cin>>n;
```

```
    priority_queue<int,vector<int>,greater<int> > pq;//1
```

```
    int a[n];
```

```
    for(int i=0; i<n; i++){
```

```
        cin>>a[i];
```

```
        pq.push(a[i]);
```

```
}
```

```
    long long p, q, sum=0, sum1=0;
```

```
    while(pq.size()){

    }
```

```
        p = pq.top(); pq.pop();//2
```

```
        q = pq.top(); pq.pop();//2
```

```
        sum = p+q;
```

```
        sum1+=sum;
```

```
        if(pq.size()==0) break;//4
```

```
        pq.push(sum);//3
```

```
}
```

```
    cout<<sum1<<endl;
```

```
}
```

```
int main(){

    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

NỐI DÂY 2

Bài làm tốt nhất

Cho N sợi dây với độ dài khác nhau được lưu trong mảng $A[]$. Nhiệm vụ của bạn là nối N sợi dây thành một sợi sao cho tổng chi phí nối dây là nhỏ nhất. Biết chi phí nối sợi dây thứ i và sợi dây thứ j là tổng độ dài hai sợi dây $A[i]$ và $A[j]$.

Dữ liệu vào

Dòng đầu ghi số bộ test T ($T < 10$). Mỗi bộ test gồm 2 dòng. Dòng đầu tiên là số nguyên N ($N \leq 2 * 10^6$).

Dòng tiếp theo gồm N số nguyên dương $c[i]$ ($1 \leq A[i] \leq 10^9$).

Kết quả

In ra đáp án của bộ test trên từng dòng, theo modulo $10^9 + 7$.

Ví dụ:

Input:	Output
7 2 4 1 2 10 2 3	59

Giải:

//1: do cần tổng chi phí nối dây là nhỏ nhất => mỗi lần lấy 2 số nhỏ nhất tìm tổng đưa nó vào hàng đợi nhưng sau đó lại tiếp tục lấy 2 số nhỏ nhất => bản thân tổng sau khi đưa vào hàng đợi cũng phải được sắp xếp tăng dần => dùng hàng đợi ưu tiên priority_queue

// 2: lấy giá trị và loại bỏ 2 số nhỏ nhất của hàng đợi

// 3: lấy tổng 2 số đó và đưa vào hàng đợi, cập nhật sum1 chính là kết quả cần tìm
+=sum;

// 4: nếu như hàng đợi rỗng rồi thì kết thúc vòng lặp

```
#include <bits/stdc++.h>//
using namespace std;
const long long mod=1e9+7;
void initwsolve(){
    long long n;cin>>n;
    priority_queue<long long,vector<long long>,greater<long long> > pq;//1
    long long a[n];
    for(long long i=0; i<n; i++){
        cin>>a[i];
        pq.push(a[i]);
    }
    long long p, q, sum=0, sum1=0;
    while(pq.size()){
        p = pq.top(); pq.pop();//2
        q = pq.top(); pq.pop();//2
        sum = (p+q)%mod;
        sum1 = (sum1+sum)%mod;
    }
}
```

```

    sum1=(sum1+sum)%mod;
    if(pq.size()==0) break;//4
    pq.push(sum);//3
}
cout<<sum1<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

SẮP ĐẶT XÂU KÝ TỰ 1

Ôn lại

Cho xâu ký tự S bao gồm các ký tự in thường. Nhiệm vụ của bạn là kiểm tra xem ta có thể sắp đặt lại các ký tự trong S để hai ký tự giống nhau đều không kề nhau hay không? Đưa ra 1 nếu có thể sắp đặt lại các ký tự trong S thỏa mãn yêu cầu bài toán, ngược lại đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu ký tự S được viết trên một dòng.
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S) \leq 10000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	
geeksforgeeks	1
bbbabaaaacd	1
bbbbbb	-1

Giải :

//vd xét xâu ggggg(5 ký tự)=> nếu chỉ cần có ít nhất 4 ký tự khác g thì nghĩa là đã thỏa mãn bài toán

//=>1: lập mảng đếm tần số các chữ xuất hiện trong mảng

// 2: bây giờ chỉ cần so sánh chữ có số lần xuất hiện lớn nhất với tổng số lần xuất hiện của các số còn lại (3)

```

// nếu(sum+1>=f[0]) (vd test kia là 4+1>=5) thì thỏa mãn , còn lại thì sẽ là không
thỏa mãn
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s;cin>>s;
    int f[26],sum=0;
    memset(f,0,sizeof(f));
    for(int i=0;i<s.length();i++) f[s[i]-'a']++;//1
    sort(f,f+26,greater<int>());
    for(int i=1;i<26;i++)sum+=f[i];//3
    if(sum+1>=f[0])cout<<1<<endl;//2
    else cout<<-1<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while(t--)initwsolve();
}

```

SẮP ĐẶT XÂU KÝ TỰ 2

Ôn lại

Cho xâu ký tự S bao gồm các ký tự in thường và số D. Nhiệm vụ của bạn là kiểm tra xem ta có thể sắp đặt lại các ký tự trong S để tất cả các ký tự giống nhau đều có khoảng cách là D hay không? Đưa ra 1 nếu có thể sắp đặt lại các ký tự trong S thỏa mãn yêu cầu bài toán, ngược lại đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là số D; dòng tiếp theo là xâu S.
- T, S, D thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S) \leq 10000$; $1 \leq D \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	1
2	-1
ABB	

Giải :

```
//ta thấy nếu tại một vị trí 1 thì số sau đó phải là 1+2n...
//=> chuỗi có max các phân tử chỉ có thể là 1+(f[0]-1)*n
// với 2 chữ cái có tần số xuất hiện đều ở đầu và cuối .
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,f[26];
    string s;
    cin>>n>>s;
    memset(f,0,sizeof(f));
    for(int i=0;i<s.length();i++) f[s[i]-'A']++;//l
    sort(f,f+26,greater<int>());
    if((f[0]-1)*n+1<=s.length()) cout<<1<<endl;
    else cout<<-1<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

MUA LƯƠNG THỰC

Ôn lại

Giả sử bạn là một người nghèo trong địa phương của bạn. Địa phương của bạn có duy nhất một cửa hàng bán lương thực. Cửa hàng của bạn mở cửa tất cả các ngày trong tuần ngoại trừ chủ nhật. Cho bộ ba số N, S, M thỏa mãn ràng buộc sau:

- N : số đơn vị lương thực nhiều nhất bạn có thể mua trong ngày.
- S : số lượng ngày bạn cần được sử dụng lương thực để tồn tại.
- M : số đơn vị lương thực cần có mỗi ngày để bạn tồn tại.

Giả sử bạn đang ở ngày thứ 2 trong tuần và cần tồn tại trong S ngày tới. Hãy cho biết số lượng ngày ít nhất bạn cần phải mua lương thực từ cửa hàng để tồn tại hoặc bạn sẽ bị chết đói trong S ngày tới.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ 3 số N, S, M được viết trên một dòng.
- T, N, S, M thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, S, M \leq 30$.

Output:

- Đưa ra số ngày ít nhất bạn có thể mua lương thực để tồn tại hoặc đưa ra -1 nếu bạn bị chết đói.

Ví dụ:

Input	Output
2	2
16 10 2	-1
20 10 30	

Giai

//1:xét duyệt từ ngày 1 đến ngày s , bỏ qua ngày chủ nhật(vì CN không mua được lương thực)

// 2: mỗi lần lặp mới tăng số ngày mua lương thực =>nếu tại ngày đó số đơn vị lương thực \geq tối thiểu =>dùng lại và in luôn kết quả

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,s,m;
    cin>>n>>s>>m;
    int res=0,need=s*m;//need: lương thực tối thiểu
    for(int i=1;i<=s;i++){
        if(i%7==0)continue;//1
        else {
            res++;
            if(res*n>=need){//2
                cout<<res<<endl;return;
            }
        }
    }
    cout<<-1<<endl;
}
```

```
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
```

```
    return 0;  
}
```

SẮP XẾP CÔNG VIỆC 1

Ôn lại

Cho hệ gồm N hành động. Mỗi hành động được biểu diễn như một bộ đôi $\langle S_i, F_i \rangle$ tương ứng với thời gian bắt đầu và thời gian kết thúc của mỗi hành động. Hãy tìm phương án thực hiện nhiều nhất các hành động được thực hiện bởi một máy hoặc một người sao cho hệ không xảy ra mâu thuẫn.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào số lượng hành động N; dòng tiếp theo đưa vào N số S_i tương ứng với thời gian bắt đầu mỗi hành động; dòng cuối cùng đưa vào N số F_i tương ứng với thời gian kết thúc mỗi hành động; các số được viết cách nhau một vài khoảng trắng.
- T, N, S_i, F_i thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, F_i, S_i \leq 1000$.

Output:

- Đưa số lượng lớn nhất các hành động có thể được thực thi bởi một máy hoặc một người.

Ví dụ:

Input	Output
1	
6	
1 3 0 5 8 5	
2 4 6 7 9 9	4

Giải:

```
//1 :tạo 1 struct và sắp xếp theo thời gian kết thúc  
//2 :như vậy bây giờ chỉ cần tìm kiếm mỗi lần thời gian bắt đầu công việc sau mà  
>= tgián kết thúc cv trước thì tăng biến đếm
```

```
#include<bits/stdc++.h>  
using namespace std;  
long long n;  
struct work{  
    int s,f;  
}; work a[1005];  
bool cmp(work a,work b){  
    return a.f<b.f;
```

```

}

void initwsolve(){
    cin>>n;
    int res=1,temp=0;
    for(int i=0;i<n;i++)cin>>a[i].s;
    for(int i=0;i<n;i++)cin>>a[i].f;
    sort(a,a+n,cmp);//l
    for(int i=1;i<n;i++){
        if(a[i].s>=a[temp].f){
            res++;
            temp=i;
        }
    }
    cout<<res<<endl;
}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

GIÁ TRỊ NHỎ NHẤT CỦA XÂU

Cho xâu ký tự S. Ta gọi giá trị của xâu S là tổng bình phương số lần xuất hiện mỗi ký tự trong S. Hãy tìm giá trị nhỏ nhất của xâu S sau khi thực hiện K lần loại bỏ ký tự.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là số K; phần thứ hai là một xâu ký tự S được viết trên một dòng.
- T, S, K thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S) \leq 10000$; $1 \leq K \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	6
2	2

ABCCBC	
--------	--

| 2 | |
| AAAB | |

Giải:

// 1: lập bảng tần suất các chữ cái

// 2: sau đó đưa tần suất các số khác 0 đưa vào hàng đợi ưu tiên

// 3: với k lần bỏ đi 1 chữa cái \Leftrightarrow trừ đi 1 đơn vị ở số lớn nhất trong pq , \Rightarrow ý tưởng là lấy số lớn nhất trong pq và bỏ nó ra . sau đó lại đưa kết quả vào pq để sắp xếp

//4 : cuối cùng chỉ cần tính tổng bình phương các số trong pq

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    int k;string s; cin>>k>>s;
```

```
    if(k>s.length())return ;
```

```
    int f[26];memset(f,0,sizeof(f));
```

```
    for(int i=0;i<s.length();i++) f[s[i]-'A']++; //1
```

```
    priority_queue <int> pq;
```

```
    for(int i=0;i<26;i++) //2
```

```
        if(f[i]!=0) pq.push(f[i]);
```

```
    while(k>0){
```

```
        int temp=pq.top();
```

```
        pq.pop();
```

```
        temp--;
```

```
        pq.push(temp);
```

```
        k--;
```

```
}
```

```
long long res=0;
```

```
while(pq.size()){//4
```

```
    long long temp=pq.top();
```

```
    res+=temp*temp;
```

```
    pq.pop();
```

```
}
```

```
cout<<res<<endl;
```

```
}
```

```
int main(){
```

```

int t;cin>>t;
while(t--)initwsolve();
return 0;
}

```

SẮP XẾP CÔNG VIỆC 2

Ôn lại

Cho N công việc. Mỗi công việc được biểu diễn như một bộ 3 số nguyên dương $\langle \text{JobId}, \text{Deadline}, \text{Profit} \rangle$, trong đó JobId là mã của việc, Deadline là thời gian kết thúc của việc, Profit là lợi nhuận đem lại nếu hoàn thành việc đó đúng thời gian. Thời gian để hoàn toàn mỗi công việc là 1 đơn vị thời gian. Hãy cho biết lợi nhuận lớn nhất có thể thực hiện các việc với giả thiết mỗi việc được thực hiện đơn lẻ.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất là số lượng Job N; phần thứ hai đưa vào $3 \times N$ số tương ứng với N job.
- T, N, JobId, Deadline, Profit thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 1000$; $1 \leq \text{JobId} \leq 1000$; $1 \leq \text{Deadline} \leq 1000$; $1 \leq \text{Profit} \leq 1000$.

Output:

- Đưa số lượng công việc tương ứng và lợi nhuận lớn nhất có thể đạt được.

Ví dụ:

Input	Output
2	
4	
1 4 20	
2 1 10	
3 1 40	
4 1 30	2 60
5	2 127
1 2 100	
2 1 19	
3 2 27	
4 1 25	
5 1 15	

Vd test 2 :làm việc 1 trong 1 tiếng + việc 3 trong 1 tiếng =>127

Giải:

```

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

```

```

int n,check[1005],kq[1005];
struct data{
    int j,d,p;
}; data a[1005];
bool cmp(data a,data b){
    return a.p>b.p;
}
void initwsolve(){
    cin>>n;
    memset(check,0,sizeof(check));
    for(int i=0;i<n;i++)
        cin>>a[i].j>>a[i].d>>a[i].p;
    sort(a,a+n,cmp);// sắp xếp các data có lợi nhuận giảm dần
    for(int i=0;i<n;i++){
        for(int j=min(n,a[i].d)-1;j>=0;j--){//giá trị << của deadline
            if(check[j]==0){
                kq[j]=i;
                check[j]=1;break;
            }
        }
    }
    int res=0,count=0;
    for(int i=0;i<n;i++)
        if(check[i]==1)count++,res+=a[kq[i]].p;
    cout<<count<<' '<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

Chia để trị
LŨY THỪA
Ôn lại

Cho số nguyên dương N và K. Hãy tính N^K modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 1 số nguyên N và K ($1 \leq N \leq 1000$, $1 \leq K \leq 10^9$).

Output:

Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input:	Output
2	8
2 3	
4 2	16

Giải:

//vd : $2^9=2^4*2^4*2$, sau đó tính $2^4=2^2*2^2$; $2^2=2^1*2^1$;=>2^9

// => ý tưởng : để tính n^k ta tính $n^{(k/2)}$ nếu k chẵn thì kết quả chính à temp*temp
ngược lại nếu k lẻ thì kết quả chính à temp*temp*a

// => 3: nếu==0 thì kq là 1 , k=1 thì kết quả là n

```
#include<bits/stdc++.h>
const int mod=1e9+7;
using namespace std;
long long n,k;
long long luythua(long long n,long long k){
    if(k==0) return 1;//3
    if(k==1) return n%mod;//3
    long long temp=luythua(n,k/2);//1
    if(k%2==0) return temp*temp %mod;//2
    else return temp*temp%mod *n%mod;
}
void initwsolve(){
    cin>>n>>k;
    cout<<luythua(n,k)<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

GẤP ĐÔI DÃY SỐ

Ôn lại

Một dãy số tự nhiên bắt đầu bởi con số 1 và được thực hiện N-1 phép biến đổi “gấp đôi” dãy số như sau:

Với dãy số A hiện tại, dãy số mới có dạng A, x, A trong đó x là số tự nhiên bé nhất chưa xuất hiện trong A.

Ví dụ với 2 bước biến đổi, ta có [1] à [1 2 1] à [1 2 1 3 1 2 1].

Các bạn hãy xác định số thứ K trong dãy số cuối cùng là bao nhiêu?

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm số nguyên dương N và K ($1 \leq N \leq 50$, $1 \leq K \leq 2^N - 1$).

Output:

Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
2	2
3 2	4
4 8	

Giải thích test 1: Dãy số thu được là [1, 2, 1, 3, 1, 2, 1].

Giải thích test 2: Dãy số thu được là [1, 2, 1, 3, 1, 2, 1, 4, 1, 2, 1, 3, 1, 2, 1].

Giải :

//1: thực hiện n-1 phép nhân đôi .

//2: ta thấy nếu k=1 thì in 1;

// 3:nếu k ở chính giữa dãy thì in n+1;

//4:bây giờ xét k ở bên trái hay bên phải mảng . nếu k ở bên trái mảng thì chia mảng chỉ lấy bên trái gapdoi(n-1,k); nếu k ở bên phải mảng thì lấy k-pow(2,n) để đưa mảng bên phải về vị trí tương ứng với bên trái

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

```
const int mod=1e9+7;
```

```
using namespace std;
```

```
long long n,k;
```

```
long long gapdoi(long long n,long long k){
```

```
    //if(k==1) return 1;//2
```

```
    if(k==pow(2,n))return n+1;//3
```

```
/*4*/ if(k<=pow(2,n))return gapdoi(n-1,k);
```

```
    return gapdoi(n-1,k-pow(2,n));
```

```
}
```

```

void initwsolve(){
    cin>>n>>k;
    cout<<gapdoi(n-1,k)<<endl;//1
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

ĐÉM DÃY

Ôn lại

Cho số nguyên dương n. Hãy cho biết có bao nhiêu dãy số nguyên dương có tổng các phần tử trong dãy bằng n.

Dữ liệu vào: dòng đầu tiên chứa số nguyên T là số bộ dữ liệu, mỗi bộ dữ liệu ghi một số nguyên dương n duy nhất không quá 10^{18} .

Kết quả: Mỗi bộ dữ liệu ghi ra một số nguyên duy nhất là số dư của kết quả tìm được khi chia cho 123456789.

Ví dụ:

Input	Output
1	
3	4

Giải :

// //1 nếu n=1 thì chỉ có duy nhất 1 dãy tổng

//2 :còn lịa thì số cách chính là 2 mũ n-1

```

#include<bits/stdc++.h>
const int mod=123456789;
using namespace std;
long long n,k;
long long luythua(long long n,long long k){
    if(k==0) return 1;
    if(k==1) return n%mod;
    long long temp=luythua(n,k/2);
    if(k%2==0) return temp*temp %mod;
    else return temp*temp%mod *n%mod;
}
void initwsolve(){
    cin>>n;
}

```

```

if(n==1) cout<<1<<endl;
else cout<<luythua(2,n-1)<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

DÃY CON LIÊN TIẾP CÓ TỔNG LỚN NHẤT

Cho mảng A[] gồm N số có cả các số âm và số dương. Nhiệm vụ của bạn là tìm mảng con liên tục có tổng lớn nhất của mảng. Ví dụ với mảng A[]={-2, -5, 6, -2, -3, 1, 5, -6} ta có kết quả là 7 tương ứng với dãy con {6, -2, -3, 1, 5}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào hai số N tương ứng với số phần tử của mảng; dòng tiếp theo đưa vào N số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 100$; $-100 \leq A[i] \leq 100$.

Output:

- Đưa ra tổng con liên tục lớn nhất của mỗi test theo từng dòng.

Ví dụ:

Input	Output
1	
8	
-2 -5 6 -2 -3 1 5 -6	7

Giải :

// 1: duyệt từng số trong mảng để tìm dãy con tổng >> bắt đầu từ số đó
// 2: duyệt dãy tổng , cứ mỗi lần cập nhật một số mới thì cập nhật tổng lớn nhất luôn
// tại vị trí mà temp <0 thì ta sẽ dùng luôn (vì nếu cứ tìm tiếp ta luôn có 1 dãy con < dãy con hiện tại)

```

#include<bits/stdc++.h>
using namespace std;
int n,a[105];
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
}

```

```

int res=-1e9+7,temp=0;
for(int i=0;i<n;i++){//1
    temp=0;
    for(int j=i;j<n;j++){//2
        temp+=a[j];
        if(temp<0)break;//3
        else res=max(res,temp);
    }
}
cout<<res<<endl;
}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

PHẦN TỬ THÚ K

Cho hai mảng đã được sắp xếp A[], B[] gồm M, N phần tử theo thứ tự và số K. Nhiệm vụ của bạn là tìm phần tử ở vị trí số K sau khi trộn hai mảng để nhận được một mảng được sắp xếp.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào số M, N, K; dòng tiếp theo đưa vào M số của mảng A[]; dòng tiếp theo đưa vào N số của mảng B[]; các số được viết cách nhau một vài khoảng trắng.
- T, M,N, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i], B[i] \leq 10^6$; $1 \leq K \leq N+M$.

Output:

- Đưa ra giá trị phần tử thứ K của mỗi test theo từng dòng.

Ví dụ:

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

Giải :

```

// chỉ sắp xếp mảng và in ra a[k-1]
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,m,k;
    cin>>n>>m>>k;
    int a[m+n];
    for(int i=0;i<n+m;i++) cin>>a[i];
    sort(a,a+m+n);
    cout<<a[k-1]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

TÌM KIẾM NHỊ PHÂN

Cho dãy số A[] gồm có N phần tử đã được sắp xếp tăng dần và số K. Nhiệm vụ của bạn là kiểm tra xem số K có xuất hiện trong dãy số hay không. Nếu có hãy in ra vị trí trong dãy A[], nếu không in ra “NO”.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 10$).

Mỗi test bắt đầu bằng số nguyên N và K ($N \leq 100\ 000$, $0 \leq K \leq 10^6$).

Dòng tiếp theo gồm N số nguyên A[i] ($0 \leq A[i] \leq 10^6$), các phần tử là riêng biệt.

Output:

Với mỗi test in ra trên một dòng đáp án tìm được.

Ví dụ:

Input:	Output
2 5 3 1 2 3 4 5 6 5 0 1 2 3 9 10	3 NO

Giải :

```

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

```

```

void initwsolve(){
    int n,k;
    cin>>n>>k;
    int a[n+1];
    for(int i=1;i<=n;i++) cin>>a[i];
    int l=1,r=n,ok=0;
    while(l<=r){
        if(k==a[l]||k==a[r]){
            if(k==a[l])cout<<l<<endl;
            if(k==a[r])cout<<r<<endl;
            ok=1; break;
        } l++;r--;
    }
    if(ok==0)cout<<"NO"<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

TÍNH FLOOR(X)

Cho mảng đã được sắp xếp A[] gồm N phần tử không có hai phần tử giống nhau và số X. Nhiệm vụ của bạn là tìm floor(X). Trong đó, K=floor(X) là phần tử lớn nhất trong mảng A[] nhỏ hơn hoặc bằng X.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số N là số phần tử của mảng A[] và số X; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trống.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^7$; $1 \leq A[i] \leq 10^{18}$.

Output:

- Đưa ra vị trí của floor(X) trong mảng A[] hoặc -1 nếu không tồn tại floor(X) của mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	-1
7 0	2

1 2 8 10 11 12 19	4
7 5	
1 2 8 10 11 12 19	
7 10	
1 2 8 10 11 12 19	

Giải :

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,x;
    cin>>n>>x;
    int a[n+1];
    int res=-1;
    for(int i=1;i<=n;i++) {
        cin>>a[i];
        if(a[i]<=x)res=i;
    }
    if(res== -1)cout<<"-1" << endl;
    else cout<<res << endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

ĐẾM SỐ 0

Cho mảng A[] gồm N phần tử chỉ bao gồm các số 0 và 1. Các số 0 được đặt trước các số 1. Hãy đếm các số 0 với thời gian $\log(N)$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng thứ nhất đưa vào số N; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 1000$; $0 \leq A[i] \leq 1$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	
12	
0 0 0 0 0 0 0 0 1 1 1	9
5	5
0 0 0 0	0
6	
1 1 1 1 1 1	

Giải :

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;
    cin>>n;
    int a[n];
    int res=0;
    for(int i=0;i<n;i++) {
        cin>>a[i];
        if(a[i]==0) res++;
    }
    cout<<res<<endl;
}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

LŨY THÙA ĐẢO

Cho mảng số N. Ta gọi số đảo của N là R. Hãy tìm lũy thừa R của N. Đưa ra kết quả của bài toán dưới dạng modulo với $10^9 + 7$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm là số N được ghi trên một dòng.
- T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^{10}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	4
2	864354781
12	

Giải:

```
#include<bits/stdc++.h>
const int mod=1e9+7;
using namespace std;
long long n;
long long luythua(long long n,long long k){
    if(k==0) return 1;
    if(k==1) return n%mod;
    long long temp=luythua(n,k/2);
    if(k%2==0) return temp*temp %mod;
    else return temp*temp%mod *n%mod;
}
void initwsolve(){
    cin>>n;
    long long m=n,r=0;
    while(m){//tìm số đảo ngược
        r=r*10+m%10;
        m/=10;
    }
    cout<<luythua(n,r)<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

DÃY XÂU FIBONACI

Ôn lại

Một dãy xâu ký tự G chỉ bao gồm các chữ cái A và B được gọi là dãy xâu Fibonacci nếu thỏa mãn tính chất: $G(1) = A; G(2) = B; G(n) = G(n-2)+G(n-1)$. Với phép cộng (+) là phép nối hai xâu với nhau. Bài toán đặt ra là tìm ký tự ở vị trí thứ i (tính từ 1) của xâu Fibonacci thứ n.

Dữ liệu vào: Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng 2 số nguyên N và i ($1 < N < 93$). Số i đảm bảo trong phạm vi của xâu G(N) và không quá 18 chữ số. Kết quả: Ghi ra màn hình kết quả tương ứng với từng bộ test.

Input	Output
2	A
6 4	B
8 19	

Giải :

//1: sinh dãy fibonacci đến thứ 82
 //2 :g1='a' và g2='b' => có 1 ký tự
 //3:nếu i là nửa bên trái tức (i<=dp[n-2] thì ta chỉ tìm nó tại chuỗi xâu n-2 còn lại
 nếu bên phải thì lấy bên fibo i-1 , khi đó ta tìm kiếm tại vị trí i-dp[n-2]

```
#include<bits/stdc++.h>
using namespace std;
long long n,i,dp[93];
char fibo(long long n,long long i){
    if(i==1) return 'A';//2
    if(i==2) return 'B';
/*3*/ if(i<=dp[n-2]) return fibo(n-2,i);
    else return fibo(n-1,i-dp[n-2]);
}
```

```
void initwsolve(){
    cin>>n>>i;
    cout<<fibo(n,i)<<endl;
```

```
}
```

```
int main(){
    int t;cin>>t;
    dp[1]=1,dp[2]=1;//note;
/*1*/ for(int i=3;i<=92;i++)dp[i]=dp[i-1]+dp[i-2];
    while(t--)initwsolve();
    return 0;
}
```

HỆ CƠ SỐ K

Cho hai số A, B ở hệ cơ số K. Hãy tính tổng hai số đó ở hệ cơ số K.

Input: Dòng đầu ghi số bộ test T ($T < 10$). Mỗi bộ test ghi 3 số K,A,B.

($2 \leq K \leq 10$; A và B nếu biểu diễn trong hệ cơ số 10 đều nhỏ hơn 10^9)

Output: In ra tổng của A và B trong hệ cơ số K

Ví dụ:

Input	Output
1	11
2 1 10	

Giải:

```
//1 : sửa cho độ dài 2 chuỗi bằng nhau  
//2: thực hiện tính tổng 2 chuỗi trên cơ số k  
#include<bits/stdc++.h>  
using namespace std;  
void initwsolve(){  
    int k;string a,b;  
    cin>>k>>a>>b;  
    while(a.length()<b.length())a='0'+a;//1  
    while(b.length()<a.length())b='0'+b;  
    int carry=0,temp=0;  
    string res;  
/*2*/ for(int i=a.length()-1;i>=0;i--){  
    int temp=a[i]-'0'+b[i]-'0'+carry;  
    res= char(temp%k+'0')+res;  
    carry=temp/k;  
}  
if(carry>0)res='1'+res;  
cout<<res<<endl;
```

```
}
```

```
int main(){  
    int t;cin>>t;  
    while(t--)initwsolve();  
    return 0;  
}
```

SẮP XẾP KANGURU

Ôn lại

Có N con kanguru trong vườn thú, con thứ i có chiều cao bằng A[i]. Con kanguru có chiều cao X có thể chui được một con có chiều cao bằng Y trong túi của nó nếu như $X \geq 2 * Y$.

Một con đã chui một con kanguru rồi, thì không thể nhảy vào túi một con kanguru khác.

Bầy Kanguru rất thích chơi trốn tìm, vì vậy chúng thường xuyên nhảy vào túi của nhau. Các bạn hãy tính toán xem trong trường hợp tối ưu, số con kanguru nhìn thấy trong vườn thú ít nhất bằng bao nhiêu?

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm số nguyên N ($1 \leq N \leq 100\,000$).

Dòng tiếp theo gồm N số nguyên A[i] ($1 \leq A[i] \leq 100\,000$).

Output:

Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
2	
8	
2 5 7 6 9 8 4 2	5
8	5
9 1 6 2 6 5 8 3	

Giải thích test 1: Nhóm 2 – 5, 2 – 6, 4 – 8, 7, 9.

Giải :

//1 : chia làm 2 mảng 1 bên là số nhỏ , một bên là số lớn
//2: => tại một số là số nhỏ ta tìm kiếm 1 số lớn mà $\geq 2 * \text{số nhỏ}$. => khi đó res--
vì 1 con đã chui , còn mid++ tức là xét con lớn kahsc

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n; cin>>n;
    int a[n],check[n];
    memset(check,0,sizeof(check));
    for(int i=0;i<n;i++) cin>>a[i];
    sort(a,a+n);
    int res=n,mid=n/2;
    for(int i=0;i<n/2&&mid<n;i++){//1
        while(mid<n){
            if(a[i]*2<=a[mid]){//2
                check[a[mid]]++;
                mid++;
            }
        }
    }
}
```

```

        res-- ; mid++;
        break;
    }else mid++;
}
cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

TÍCH ĐA THỨC

Cho hai đa thức P và Q được biểu diễn như một mảng bao gồm các hệ số của đa thức. Ví dụ với $P(x) = 5 + 0x^1 + 10x^2 + 6x^3$ được biểu diễn như mảng $P[] = \{5, 0, 10, 6\}$. Hãy đưa ra đa thức $R = P \times Q$ theo các hệ số của R với cách biểu diễn như trên.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng thứ nhất đưa vào hai số M, N tương ứng với lũy thừa lớn nhất của đa thức P và Q; dòng tiếp theo đưa vào M số là hệ số của đa thức P; dòng cuối cùng đưa vào M số là hệ số của đa thức Q.
- $T, M, N, P[i], Q[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq M, N \leq 100$; $1 \leq P[i], Q[i] \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
4 3	
1 0 3 2	
2 0 4	2 0 10 4 12 8
5 4	4 36 14 39 79 23 34 35
1 9 3 4 7	
4 0 2 5	

Giải :

```

//1 : chú ý nếu một đa thức bậc m , còn lại bộn n thì max khi nhân là bậc m+n
// 2: thì ý tưởng là lấy từng phần tử của đa thức q nhân với đa thức q
//sau đó cộng hết các phần tử cùng bậc vào ta được đa thức r
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int m,n; cin>>m>>n;
    int p[m],q[n],r[m+n];//1
    memset(r,0,sizeof(r));
    for(int i=0;i<m;i++) cin>>p[i];
    for(int i=0;i<n;i++) cin>>q[i];
    for(int i=0;i<m;i++){//2
        for(int j=0;j<n;j++){
            r[i+j]+=p[i]*q[j];//3
        }
    }
    for(int i=0;i<m+n-1;i++)cout<<r[i]<<' '//1
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

TÍCH HAI SỐ NHỊ PHÂN

Cho hai xâu nhị phân biểu diễn hai số. Nhiệm vụ của bạn là đưa ra tích của hai số. Ví dụ với xâu S1="1100" và S2="1010" ta sẽ có kết quả là 120.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm 2 hai xâu nhị phân S1, S2 được viết trên một dòng.
- T, S1, S2 thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S1), \text{length}(S2) \leq 30$.

Output:

- Đưa ra tích của mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	12

1100 01	1
01 01	

Giải :

```
//1 :đề a và b ở dạng chuỗi nhị phân
//2 :sau đó chuyển sang 2 số dạng long long , nhân là ra kết quả
#include<bits/stdc++.h>
using namespace std;
long long convert(string a){//2
    long long res=0;
    for(int i=0;i<a.length();i++)
        res=res*2+a[i]-'0';
    return res;
}
void initwsolve(){
    string a,b;//1
    cin>>a>>b;
    cout<<convert(a)*convert(b)<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

DÃY XÂU NHỊ PHÂN

Người ta tạo ra một dãy xâu ký tự nhị phân $X[]$ trong đó:

$$X[1] = "0"$$

$$X[2] = "1"$$

$$X[n] = X[n-2] + X[n-1] \text{ với } n > 2$$

Với phép cộng (+) là phép nối hai xâu với nhau.

Cho hai số tự nhiên N và K ($1 < N < 93$; K đảm bảo trong phạm vi của xâu $X[N]$).

Hãy xác định ký tự thứ K trong xâu $X[N]$ là ký tự '0' hay ký tự '1'.

Input: Dòng 1 ghi số bộ test. Mỗi bộ test ghi trên một dòng 2 số nguyên N và K.

Output: Ghi ra màn hình kết quả tương ứng với từng bộ test.

Ví dụ:

Input	Output
2	0
3 1	1

Giải :

```
// tương tự y bài DÃY XÂU FIBONACI
#include<bits/stdc++.h>
using namespace std;
long long dp[93];
long long fibo(long long n,long long k){
    if(n==1) return 0;
    if(n==2) return 1;
    if(k<=dp[n-2])return fibo(n-2,k);
    else return fibo(n-1,k-dp[n-2]);
}
void initwsolve(){
    long long n,k;
    cin>>n>>k;
    cout<<fibo(n,k)<<endl;
}
int main(){
    int t;cin>>t;
    dp[1]=1,dp[2]=1;
    for(int i=3;i<=92;i++) dp[i]=dp[i-1]+dp[i-2];

    while(t--)initwsolve();
    return 0;
}
```

LÝ THUYẾT MA TRẬN 1

Ôn lại

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X .

Ví dụ:

Input:	Output
2	
2 5	8 5
1 1	5 3
1 0	597240088 35500972 473761863
3 1000000000	781257150 154135232 527013321
1 2 3	965274212 272769492 580264779
4 5 6	
7 8 9	

Giải :

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị
 // 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng
 để trả về

// 3: nhân 2 ma trận a và b

```
#include<bits/stdc++.h>
using namespace std;
const long long mod=1e9+7;
long long n,k;
struct matrix{long long pos[15][15] ;};//2
matrix multi(matrix a,matrix b){//3
    matrix temp;
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            temp.pos[i][j]=0;
            for(int k=0;k<n;k++)
                temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
        }
    }
    return temp;
}
matrix power(matrix a,long long k){//1
    if(k==1) return a;
    if(k%2==0) return power(multi(a,a),k/2 );
    else return multi(a,power(multi(a,a),k/2 ) );
}
void initwsolve(){
    cin>>n>>k;
    matrix a;
```

```

for(int i=0;i<n;i++)
    for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//l
for(int i=0;i<n;i++){
    for(int j=0;j<n;j++)
        cout<<b.pos[i][j]<<' ';
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

LŨY THỬA MA TRẬN 2

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. *Sau đó, tính tổng các phần tử trên đường chéo chính.* Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X.

Ví dụ:

Input	Output
2	11
2 5	331640092
1 1	
1 0	
3 1000000000	
1 2 3	
4 5 6	
7 8 9	

Giải thích:

$$A^5 = 85$$

53

Tổng các phần tử trên đường chéo chính bằng $8+3 = 11$.

597240088 35500972 473761863

$$B^5 = 1000000000 = 781257150 \ 154135232 \ 527013321$$

965274212 272769492 580264779

Tổng các phần tử trên đường chéo chính là:

$$(597240088 + 154135232 + 580264779) \% 1000000007 = 331640092$$

Giải:

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị

// 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng để trả về

// 3: nhân 2 ma trận a và b

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

```
using namespace std;
```

```
const long long mod=1e9+7;
```

```
long long n,k;
```

```
struct matrix{ long long pos[15][15] ; };//2
```

```
matrix multi(matrix a,matrix b){//3
```

```
    matrix temp;
```

```
    for(int i=0;i<n;i++){
```

```
        for(int j=0;j<n;j++){
```

```
            temp.pos[i][j]=0;
```

```
            for(int k=0;k<n;k++)
```

```
                temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
```

```
        }
```

```
    }
```

```
    return temp;
```

```
}
```

```
matrix power(matrix a,long long k){//1
```

```
    if(k==1) return a;
```

```
    if(k%2==0) return power(multi(a,a),k/2 );
```

```
    else return multi(a,power(multi(a,a),k/2 ) );
```

```
}
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    matrix a;
```

```
    for(int i=0;i<n;i++)
```

```

        for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//1
long long res=0;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        if(i==j) res= (res+b.pos[i][j])%mod;
cout<<res<<endl;

}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

LŨY THỬA MA TRẬN 3

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. Sau đó, tính tổng các phần tử trên đường chéo phụ. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X .

Ví dụ:

Input:	Output
2	10
2 5	593171300
1 1	
1 0	
3 1000000000	
1 2 3	
4 5 6	
7 8 9	

Giải thích:

$$A^5 = 85$$

53

Tổng các phần tử trên đường chéo phụ bằng $5+5 = 10$.

597240088 35500972 473761863

$$B^5 = 1000000000 = 781257150 \ 154135232 \ 527013321$$

965274212 272769492 580264779

Tổng các phần tử trên đường chéo phụ là:

$$(473761863 + 154135232 + 965274212) \% 1000000007 = 593171300$$

Giai:

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị

// 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng để trả về

// 3: nhân 2 ma trận a và b

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

```
using namespace std;
```

```
const long long mod=1e9+7;
```

```
long long n,k;
```

```
struct matrix{long long pos[15][15] ;};//2
```

```
matrix multi(matrix a,matrix b){//3
```

```
    matrix temp;
```

```
    for(int i=0;i<n;i++){
```

```
        for(int j=0;j<n;j++){
```

```
            temp.pos[i][j]=0;
```

```
            for(int k=0;k<n;k++)
```

```
                temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
```

```
        }
```

```
    }
```

```
    return temp;
```

```
}
```

```
matrix power(matrix a,long long k){//1
```

```
    if(k==1) return a;
```

```
    if(k%2==0) return power(multi(a,a),k/2 );
```

```
    else return multi(a,power(multi(a,a),k/2 ) );
```

```
}
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    matrix a;
```

```
    for(int i=0;i<n;i++)
```

```

        for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//1
long long res=0;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        if(i+j==n-1) res= (res+b.pos[i][j])%mod;
cout<<res<<endl;

}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

LŨY THỬA MA TRẬN 4

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. Sau đó, tính tổng các phần tử của hàng đầu tiên. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X .

Ví dụ:

Input:	Output
2	13
2 5	106502916
1 1	
1 0	
3 1000000000	
1 2 3	
4 5 6	
7 8 9	

Giải thích:

$$A^5 = 85$$

53

Tổng các phần tử trên hàng đầu tiên bằng $8+5 = 13$.

597240088 35500972 473761863

$B^1000000000 = 781257150 \ 154135232 \ 527013321$

965274212 272769492 580264779

Tổng các phần tử trên hàng đầu tiên là:

$(597240088+35500972+473761863) \% 1000000007 = 106502916$

Giai:

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị

// 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng để trả về

// 3: nhân 2 ma trận a và b

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

```
using namespace std;
```

```
const long long mod=1e9+7;
```

```
long long n,k;
```

```
struct matrix{ long long pos[15][15] ; };//2
```

```
matrix multi(matrix a,matrix b){//3
```

```
    matrix temp;
```

```
    for(int i=0;i<n;i++){
```

```
        for(int j=0;j<n;j++){
```

```
            temp.pos[i][j]=0;
```

```
            for(int k=0;k<n;k++)
```

```
temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
```

```
        }
```

```
    }
```

```
    return temp;
```

```
}
```

```
matrix power(matrix a,long long k){//1
```

```
if(k==1) return a;
```

```
if(k%2==0) return power(multi(a,a),k/2 );
```

```
else return multi(a,power(multi(a,a),k/2 ) );
```

```
}
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    matrix a;
```

```
    for(int i=0;i<n;i++)
```

```

        for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//1
long long res=0;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        if(i==0) res= (res+b.pos[i][j])%mod;
cout<<res<<endl;

}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

LŨY THỬA MA TRẬN 5

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. Sau đó, tính tổng các phần tử của hàng cuối cùng. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X .

Ví dụ:

Input	Output
2	8
2 5	818308476
1 1	
1 0	
3 1000000000	
1 2 3	
4 5 6	
7 8 9	

Giải thích:

$$A^5 = 85$$

53

Tổng các phần tử trên hàng cuối bằng $5+3=8$.

597240088 35500972 473761863

$$B^5 = 1000000000 = 781257150 \ 154135232 \ 527013321$$

965274212 272769492 580264779

Tổng các phần tử trên hàng cuối là:

$$(965274212+272769492+580264779) \% 1000000007 = 818308476$$

Giải:

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị

// 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng để trả về

// 3: nhân 2 ma trận a và b

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

```
using namespace std;
```

```
const long long mod=1e9+7;
```

```
long long n,k;
```

```
struct matrix{long long pos[15][15] ;};//2
```

```
matrix multi(matrix a,matrix b){//3
```

```
    matrix temp;
```

```
    for(int i=0;i<n;i++){
```

```
        for(int j=0;j<n;j++){
```

```
            temp.pos[i][j]=0;
```

```
            for(int k=0;k<n;k++)
```

```
                temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
```

```
        }
```

```
    }
```

```
    return temp;
```

```
}
```

```
matrix power(matrix a,long long k){//1
```

```
    if(k==1) return a;
```

```
    if(k%2==0) return power(multi(a,a),k/2 );
```

```
    else return multi(a,power(multi(a,a),k/2 ) );
```

```
}
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    matrix a;
```

```
    for(int i=0;i<n;i++)
```

```

        for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//1
long long res=0;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        if(i==n-1) res= (res+b.pos[i][j])%mod;
cout<<res<<endl;

}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

LỦY THƯA MA TRẬN 6

Cho ma trận vuông A kích thước $N \times N$. Nhiệm vụ của bạn là hãy tính ma trận $X = A^K$ với K là số nguyên cho trước. Sau đó, tính tổng các phần tử của cột đầu tiên. Đáp số có thể rất lớn, hãy in ra kết quả theo modulo 10^9+7 .

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test bao gồm một số nguyên N và K ($1 \leq N \leq 10$, $1 \leq K \leq 10^9$) là kích thước của ma trận và số mũ.

Output:

Với mỗi test, in ra kết quả của ma trận X.

Ví dụ:

Input:	Output
2	13
2 5	343771436
1 1	
1 0	
3 1000000000	
1 2 3	
4 5 6	
7 8 9	

Giải thích:

$$A^5 = 85$$

53

Tổng các phần tử trên cột đầu tiên bằng $8+5 = 13$.

597240088 35500972 473761863

$B^1000000000 = 781257150 154135232 527013321$

965274212 272769492 580264779

Tổng các phần tử trên cột đầu tiên là:

$(597240088+781257150+965274212) \% 100000007 = 343771436$

Giai :

// 1:tạo một mảng b là a mũ k và tính mũ theo chia để trị

// 2:(nhưng trong c++ ko có trả về 1 mảng)=> tạo một struct bên trong chứa mảng để trả về

// 3: nhân 2 ma trận a và b

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

```
using namespace std;
```

```
const long long mod=1e9+7;
```

```
long long n,k;
```

```
struct matrix{long long pos[15][15] ;};//2
```

```
matrix multi(matrix a,matrix b){//3
```

```
    matrix temp;
```

```
    for(int i=0;i<n;i++){
```

```
        for(int j=0;j<n;j++){
```

```
            temp.pos[i][j]=0;
```

```
            for(int k=0;k<n;k++)
```

```
temp.pos[i][j]=(temp.pos[i][j]+a.pos[i][k]*b.pos[k][j]%mod)%mod;
```

```
        }
```

```
    }
```

```
    return temp;
```

```
}
```

```
matrix power(matrix a,long long k){//1
```

```
if(k==1) return a;
```

```
if(k%2==0) return power(multi(a,a),k/2 );
```

```
else return multi(a,power(multi(a,a),k/2 ) );
```

```
}
```

```
void initwsolve(){
```

```
    cin>>n>>k;
```

```
    matrix a;
```

```
    for(int i=0;i<n;i++)
```

```

        for(int j=0;j<n;j++) cin>>a.pos[i][j];
matrix b=power(a,k);//1
long long res=0;
for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
        if(j==0) res= (res+b.pos[i][j])%mod;
cout<<res<<endl;

}

int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

ĐẾM SỐ BÍT 1

Ôn lại

Cho số nguyên dương N. Mỗi bước, bạn sẽ biến đổi N thành $[N/2]$, $N \bmod 2$, $[N/2]$. Sau khi thực hiện một cách triệt để, ta thu được một dãy số chỉ toàn số 0 và 1.

Nhiệm vụ của bạn là hãy đếm các số bằng 1 trong đoạn $[L, R]$ của dãy số cuối cùng.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 3 số nguyên N, L, R ($1 \leq N, L, R < 2^{50}$, $0 \leq R-L \leq 100\ 000$).

Output:

Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
2 7 2 5 10 3 10	4 5

Giải thích test 1: [7] à [3, 1, 3] à [1, 1, 1, 1, 3] à [1, 1, 1, 1, 1, 1, 1].

Giải thích test 2: Dãy số sau khi biến đổi là [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1].

Giải :

```
#include<bits/stdc++.h>
using namespace std;
long long N,L,R;
```

```

long long doDai(long long n){
    if(n==1||n==0) return 1;
    long long temp=doDai(n/2);
    return 2*temp+1;
}
long long dembit(long long n,long long l,long long r){
    if(l>R||r<L) return 0;
    if(l>=L&&r<=R) return n;
    long long mid=(l+r)/2;
    if(mid>=L&&mid<=R)
        return dembit(n/2,l,mid-1)+dembit(n/2,mid+1,r) +n%2;
    else return dembit(n/2,l,mid-1)+dembit(n/2,mid+1,r);
}
int main(){
    int t;cin>>t;
    while(t--){
        cin>>N>>L>>R;
        cout<<dembit(N,1,doDai(N))<<endl;
    }
    return 0;
}

```

Quy hoạch động

XÂU CON CHUNG DÀI NHẤT

Ôn tập 1

Cho 2 xâu S1 và S2. Hãy tìm xâu con chung dài nhất của 2 xâu này (*các phần tử không nhất thiết phải liên tiếp nhau*)..

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 20$). Mỗi test gồm hai dòng, mô tả xâu S1 và S2, mỗi xâu có độ dài không quá 1000 và chỉ gồm các chữ cái in hoa.

Output: Với mỗi test, in ra độ dài xâu con chung dài nhất trên một dòng.

Ví dụ:

Input	Output
2	
AGGTAB	4
GXTXAYB	0
AA	

Giải thích test 1: Dãy con chung là G, T, A, B.

AGGTAB :6

GXTXAYB: 7

	0	0	0	0	0	0	0	0
A	0	0	0	0	0	1.	0	0
G	0	1.	1	1	1	1	1	1
G	0	1.	1	1	1	1	1	1
T	0	1	1	2.	2	2	2	2
A	0	1	1	2	2	3.	3	3
B	0	1	1	2	2	3	3	4.

Giải :

//dp[i][j] : xâu con chung dài nhất tính đến phần tử thứ i của xâu s1 và phần tử thứ j của xâu s2

//2: tại trường hợp có kí tự chung thì kq chính là xâu con chung dài nhất tính đến phần tử thứ i-1 của xâu s1 và phần tử thứ j-1 của xâu s2 + thêm 1 (ký tự chung)=> dp[i-1][j-1]+1

//3: th nếu khác nhau thì độ dài xâu lớn nhất là độ dài lớn nhất tính đến ký tự i-1 của xâu s1 hoặc j-1 của xâu s2;

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

```
using namespace std;
```

```
string s1,s2;
```

```
int dp[1005][1005];//1
```

```
void initwsolve(){
```

```
    memset(dp,0,sizeof(dp));
```

```
    cin>>s1>>s2;
```

```
    int m=s1.length(),n=s2.length();
```

```
    s1=' '+s1,s2=' '+s2;
```

```
    for(int i=1;i<=m;i++){
```

```
        for(int j=1;j<=n;j++){
```

```
            if(s2[j]==s1[i]) dp[i][j]=dp[i-1][j-1]+1;//2
```

```
            else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);//3
```

```
}
```

```
}cout<<dp[m][n]<<endl;
```

```

}
int main (){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

DÃY CON TĂNG DÀI NHẤT

Ôn lại

Cho một dãy số nguyên gồm N phần tử A[1], A[2], ... A[N].

Biết rằng dãy con tăng là 1 dãy A[i₁],... A[i_k]

thỏa mãn i₁ < i₂ < ... < i_k và A[i₁] < A[i₂] < .. < A[i_k].

Hãy cho biết dãy con tăng dài nhất của dãy này có bao nhiêu phần tử?

Input: Dòng 1 gồm 1 số nguyên là số N ($1 \leq N \leq 1000$). Dòng thứ 2 ghi N số nguyên A[1], A[2], .. A[N] ($1 \leq A[i] \leq 1000$).

Output: Ghi ra độ dài của dãy con tăng dài nhất.

Ví dụ:

Input	Output
6 1 2 5 4 6 2	4

Giải:

//1: dp[i] : độ dài của dãy con tăng dài nhất tính đến phần tử thứ i của mảng

//2: tại một vị trí trong mảng , tìm xem tại các số trước nó nhỏ hơn nó và lấy dãy con nào tăng nhiều nhất trong các số đó lưu lại và +1 vào dp[i]

```

#include<bits/stdc++.h>
using namespace std;
int n;
int a[1005],dp[1005];
int main (){
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    dp[0]=1;dp[1]=1;//1
/*1*/for(int i=1;i<=n;i++){
    for(int j=1;j<i;j++){
        if(a[j]<a[i]) dp[i]=max(dp[i],dp[j]+1);
    }
}
sort(dp,dp+n+1,greater<int>());
cout<<dp[0];

```

}

DÃY CON CÓ TỔNG BẰNG S

Tham khảo từ bài 25 contest 2 (quay lui, nhánh cận)

Ôn lại

Cho N số nguyên dương tạo thành dãy $A=\{A_1, A_2, \dots, A_N\}$. Tìm ra một dãy con của dãy A (không nhất thiết là các phần tử liên tiếp trong dãy) có tổng bằng S cho trước.

Input: Dòng đầu ghi số bộ test T ($T < 10$). Mỗi bộ test có hai dòng, dòng đầu tiên ghi hai số nguyên dương N và S ($0 < N \leq 200$) và S ($0 < S \leq 40000$). Dòng tiếp theo lần lượt ghi N số hạng của dãy A là các số A_1, A_2, \dots, A_N ($0 < A_i \leq 200$).

Output: Với mỗi bộ test, nếu bài toán vô nghiệm thì in ra "NO", ngược lại in ra "YES"

Ví dụ:

Input	Output
2 5 6 1 2 4 3 5 10 15 2 2 2 2 2 2 2 2 2	YES NO

Giải :

//1: dp[i] là tổng các phần tử bất kì có trong mảng bằng i .==1:tạo được ,=0 : không tạo được

//2 : duyệt từng giá trị của mảng

// 3: đánh dấu tất cả các tổng mà có sự xuất hiện của số đó và lớn hơn số đó(> để tránh tràn mảng)

```
#include<bits/stdc++.h>
using namespace std;
int n,s;
int a[202],dp[40005];
void initwsolve(){
    memset(dp,0,sizeof(dp));
    cin>>n>>s;
    for(int i=0;i<n;i++) cin>>a[i];
    dp[0]=1;//1
    for(int i=0;i<n;i++){//2
```

```

        for(int j=s;j>=a[i];j--){
            if(dp[j-a[i]]==1) dp[j]=1;//3
        }
    cout<<(dp[s]==1?"YES":"NO")<<endl;
}
int main (){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

DÃY CON DÀI NHẤT CÓ TỔNG CHIA HẾT CHO K

Ôn lại

Nội dung bài tập

Cho một dãy gồm n ($n \leq 1000$) số nguyên dương A_1, A_2, \dots, A_n và số nguyên dương k ($k \leq 50$). Hãy tìm dãy con gồm nhiều phần tử nhất của dãy đã cho sao cho tổng các phần tử của dãy con này chia hết cho k.

Input: Dòng đầu ghi số bộ test T ($T < 10$). Mỗi bộ test gồm 2 dòng. Dòng đầu tiên chứa hai số n, k. Dòng tiếp theo ghi n số của dãy A. Các số đều không vượt quá 100.

Output: Gồm 1 dòng duy nhất ghi số lượng phần tử của dãy con dài nhất thỏa mãn. Dữ liệu vào luôn đảm bảo sẽ có ít nhất một dãy con có tổng chia hết cho k.

Ví dụ:

Input	Output
1 10 3 2 3 5 7 9 6 12 7 11 15	9

Giải :

```

// 1: //2 3 5 7 9 6 12 7 11 15 sau chia dư//2 0 2 1 0 0 0 1 2 0
//dp[i][j] : độ dài dãy con lớn nhất có tổng là j+k xét từ a1 đến a[i]
#include<bits/stdc++.h>
using namespace std;
int a[1005],n,k,dp[1005][1005];
void initwsolve(){
    cin>>n>>k;

```

```

for(int i=1;i<=n;i++) cin>>a[i],a[i]%=k;//1
for(int i=1;i<k;i++) dp[0][i]=-1e9;
dp[0][0]=0;
for(int i=1;i<=n;i++)
    for(int j=0;j<k;j++)
        dp[i][j]=max(dp[i-1][j] ,dp[i-1][(j-a[i]+k)%k] +1 );
cout<<dp[n][0]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

TỔ HỢP C(n, k)

Cho 2 số nguyên n, k. Bạn hãy tính C(n, k) modulo 10^9+7 .

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).
- Mỗi test gồm 2 số nguyên n, k ($1 \leq k \leq n \leq 1000$).

Output:

- Với mỗi test, in ra đáp án trên một dòng.

Ví dụ:

Input	Output
2	
5 2	10
10 3	120

Giải :

```

// dp[i][j] : dp[n][k] hay nCk.
//1 : tính chất của công thức paxcan : nCK=(n-1)C(k-1) +(n-1)Ck.
#include<bits/stdc++.h>
const int mod=1e9+7;
using namespace std;
int a[1005],n,k,dp[1005][1005];
void tinh(){//1
    for(int i=0;i<1005;i++){
        for(int j=0;j<=i;j++){

```

```

        if(j==0||j==i)dp[i][j]=1;
        else dp[i][j]=(dp[i-1][j-1]+dp[i-1][j])%mod;
    }
}
int main(){
    int t;cin>>t;
    tinh();
    while(t--) {
        cin>>n>>k;
        cout<<dp[n][k]<<endl;
    }
}

```

XÂU CON ĐỐI XỨNG DÀI NHẤT

Ôn lại

Cho xâu S chỉ bao gồm các ký tự viết thường và dài không quá 1000 ký tự.

Hãy tìm xâu con đối xứng dài nhất của S.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 10$).

Mỗi test gồm một xâu S có độ dài không vượt quá 1000, chỉ gồm các kí tự thường.

Output: Với mỗi test, in ra đáp án tìm được.

Ví dụ:

Input	Output
2	
abcbadd	5
aaaaa	5

Giải :

Abcbadd

1234567

1					
1	1				
1	1	1			

1	1	1	1			
1	1	1	1	1		
1	1	1	1	1	1	
1	1	1	1	1	1	1

K=1 => i(1=>6) => j(2=>7) => 12 23 34 45 56 67

K=2 => i(1=>5) => j(3=>7) => 13 24 35 46 57

K=3=> i(1=>4) => j(4=>7) => 14 25 36 47

..... k=6=>17

//1: Dp[i][j]: kiểm tra xâu con từ i đến j có phải xâu con đối xứng hay không . Nếu Dp[i][j] = 1 => đối xứng , độ dài là j-i+1; còn 0 thì ko đối xứng => ko quan tâm .

//2: Muốn Dp[i][j] đối xứng => đoạn từ i+1 đến j-1 là đối xứng và kèm theo điều kiện s[i]==s[j] ;

```
#include<bits/stdc++.h>
using namespace std;
string s;
bool dp[1005][1005];
void initwsolve(){
    memset(dp,0,sizeof(dp));
    cin>>s;
    s=' '+s;
    int n=s.length(),res=1;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=i;j++)
            dp[i][j]=1;//1
    for(int k=1;k<=n-1;k++){
        for(int i=1;i<=n-k;i++){
            int j=k+i;
            if(s[i]==s[j]) dp[i][j]=dp[i+1][j-1];//2
            if(dp[i][j]!=0) res=max(j-i+1,res);
        }
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
```

```
    return 0;  
}
```

BẬC THANG

Ôn lại

Một chiếc cầu thang có N bậc. Mỗi bước, bạn được phép bước lên trên tối đa K bước. Hỏi có tất cả bao nhiêu cách bước để đi hết cầu thang? (Tổng số bước dùng bằng N).

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 100$).

Mỗi test gồm hai số nguyên dương N và K ($1 \leq N \leq 100000, 1 \leq K \leq 100$).

Output:

Với mỗi test, in ra đáp án tìm được trên một dòng theo modulo 109+7.

Ví dụ:

Input	Output
2	
2 2	2
4 2	5

Giải thích test 1: Có 2 cách đó là (1, 1) và (2).

Giải thích test 2: 5 cách đó là: (1, 1, 1, 1), (1, 1, 2), (1, 2, 1), (2, 1, 1), (2, 2).

Giải :

Vd test 4,2

$dp[1] = dp[0] = 1 // i-k=1-2=-1$

$dp[2] = dp[1] + dp[0] = 2 // i-k=0$

$dp[3] = dp[2] + dp[1] = 3 // i-k=1$

$dp[4] = dp[3] + dp[2] = 5 // i-k=2 \Rightarrow output=5$

code c++:

```
//1: dp[i] : số các cách có thể tạo được tại bậc thứ i
```

```
//2:=> dp[0]=1; có 1 cách bước đến bậc đầu tiên
```

```
//3: vd:n=2,k=2 , tìm dp[3] : ta thấy các cách có thể bước đến bậc 3 có 2 th : ở bậc 2 rồi bước thêm 1 bậc .. hoặc ở bậc 1 (i-k) rồi bước thêm 2 bậc (bậc tối đa)
```

```
//=> dp[i]+=dp[j] trong đó j=i-1;j>=max(i-k,0);j-- . xét max với 0 vì một số lần lặp đầu có thể số âm.
```

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

```
const long long mod = 1e9+7;
```

```
using namespace std;
```

```

void initwsolve(){
    int n,k ; cin>>n>>k;
    int dp[n+1];//1
    memset(dp,0,sizeof(dp));
    dp[0]=1;//2
    for(int i=1;i<=n;i++){//3
        for(int j=i-1;j>=max(i-k,0);j--){
            dp[i]=(dp[i]+dp[j])%mod;//dp[i]+=dp[j]
        }
    }
    cout<<dp[n]<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

HÌNH VUÔNG LỚN NHẤT

Ôn lại

Cho một bảng số N hàng, M cột chỉ gồm 0 và 1. Bạn hãy tìm hình vuông có kích thước lớn nhất, sao cho các số trong hình vuông toàn là số 1.

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 10$).
- Mỗi test bắt đầu bởi 2 số nguyên N, M ($1 \leq N, M \leq 500$).
- N dòng tiếp theo, mỗi dòng gồm M số mô tả một hàng của bảng.

Output:

- Với mỗi test, in ra đáp án là kích thước của hình vuông lớn nhất tìm được trên một dòng.

Ví dụ:

Input:	Output
2 6 5 0 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1	3 0

0 0 0 0 0	
2 2	
0 0	
0 0	

Ví dụ mảng dp của test 6 5:

Duyệt từng hàng , từ trái sang phải

0	1	1	0	1
1	1	0	1	0
0	1	1	1	0
1	1	2	2	0
1	2	2	3	1
0	0	0	0	0

Giải:

// 1: đầu cột dp bằng đầu cột a và đầu hàng dp bằng đầu hàng a.

// 2: dp[i][j] : kích thước lớn nhất của hình vuông tính đến hàng i cột j .

// 3: nếu như tại vị trí đó a[i][j]==1 tức là có quyền xét thì ta thấy kích thước cần tìm chính là giá trị nhỏ nhất của 3 ô trái nó , trên nó , chéo lên của nó cuối cùng cộng thêm 1 , (th min 3 ô kia để ubawngf 0 tức là không có hình vuông to nào thì ô đang xét là hình vuông duy nhất =>0+1=1)

// 4: cuối cùng thì chỉ cần tìm số lớn nhất trong mảng dp đã xét.

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

```
using namespace std;
```

```
int n,m,res,a[505][505], dp[505][505];
```

```
void initwsolve(){
```

```
    memset(a,0,sizeof(a));
```

```
    memset(dp,0,sizeof(dp));
```

```
    cin>>n>>m;res=1;
```

```
    for(int i=0;i<n;i++)
```

```
        for(int j=0;j<m;j++)
```

```
            cin>>a[i][j];
```

```
        for(int i=0;i<n;i++) dp[i][0]=a[i][0];//1
```

```
        for(int i=0;i<m;i++) dp[0][i]=a[0][i];//1
```

```
/*2*/for(int i=1;i<n;i++)
```

```
        for(int j=1;j<m;j++)
```

```
            if(a[i][j]==1)//3
```

```

dp[i][j]=min(dp[i-1][j],min(dp[i][j-1],dp[i-1][j-1] ))+1;
/*4*/ for(int i=0;i<n;i++)
    for(int j=1;j<m;j++)
        res=max(res,dp[i][j]);
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while (t--) initwsolve();
    return 0;
}

```

SỐ CÓ TỔNG CHỮ SỐ BẰNG K

Ôn lại

Cho 2 số nguyên N và K. Bạn hãy đếm số lượng các số có N chữ số mà tổng các chữ số của nó bằng K. Lưu ý, chữ số 0 ở đầu không được chấp nhận.

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 50$).
- Mỗi test gồm 2 số nguyên N và K ($1 \leq N \leq 100$, $0 \leq K \leq 50000$).

Output:

- Với mỗi test, in ra đáp số tìm được theo modulo 10^9+7 trên một dòng.

Ví dụ:

Input:	Output
3	
2 2	2
2 5	5
3 6	21

Giải thích test 1: 11 và 20.

Giải thích test 2: 14, 23, 32, 41.

Giải :

// 1: dp[i][j] : số các số có i chữ số có tổng các chữ số là j => ta phải đánh dấu các số có 0cs thì =0 , các số có tổng bằng 0 là 0 và các số có 1 chữ số mà từ 1->9 thì có 1 số.

```

// 2: ta thấy số có i chữ số được tạo thành bằng cách thêm 1 chữ số s (từ 0->9) vào
// số có i-1 chữ số và khi đó nó sẽ có tổng mới hơn tổng cũ là s => dp[i][j]+=dp[i-
// 1][j-s]
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
int n,k,dp[101][50001];
void solve(){
    for(int i=0;i<101;i++) dp[i][0]=0;//1
    for(int j=1;j<50001;j++) dp[0][j]=0;//1
    for(int j=1;j<=9;j++) dp[1][j]=1;//1
/*2*/for(int i=1;i<101;i++){
    for(int s=0;s<=9;s++){
        for(int j=s;j<50001;j++){//j=s...
            dp[i][j]=(dp[i][j]+dp[i-1][j-s])%mod;
        }
    }
}
int main(){
    int t;cin>>t;
    solve();
    while (t--) {
        cin>>n>>k;
        cout<<dp[n][k]<<endl;
    }
    return 0;
}

```

ĐI NHỎ NHẤT

Ôn lại

Cho bảng A[] kích thước N x M (N hàng, M cột). Bạn được phép đi xuống dưới, đi sang phải và đi xuống ô chéo dưới. Khi đi qua ô (i, j), điểm nhận được bằng $A[i][j]$.

Hãy tìm đường đi từ ô (1, 1) tới ô (N, M) sao cho tổng điểm là nhỏ nhất.

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).
- Mỗi test gồm số nguyên dương N và M.
- N dòng tiếp theo, mỗi dòng gồm M số nguyên $A[i][j]$ ($0 \leq A[i] \leq 1000$).

Output:

- Với mỗi test, in ra độ dài dãy con tăng dài nhất trên một dòng.

Ví dụ:

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

Giải thích test: Đường đi (1, 1) à (1, 2) à (2, 3) à (3, 3).

Giải :

// 1: dp[i][j]: tổng điểm là nhỏ nhất tại hàng i , cột j => nếu chỉ đi thẳng theo cột đầu , hoặc theo hàng đầu thì chỉ có duy nhất 1 cách đi tổng điểm trong dp sẽ được cộng dồn

// 2: do được phép đi xuống dưới, đi sang phải và đi xuống ô chéo dưới => tổng điểm nhỏ nhất phụ thuộc vào min trong 3 ô bên trái , bên trên và chéo trên , lấy min và +a[i][j] sẽ ra kq

```
#include<bits/stdc++.h>
using namespace std;
const int mod=1e9+7;
int n,m,a[1001][1001],dp[1001][1001];
void initwsolve(){
    memset(dp,0,sizeof(dp));
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=m;j++)
            cin>>a[i][j];
    for(int i=1;i<=n;i++) dp[i][1]=dp[i-1][1]+a[i][1];//1
    for(int j=1;j<=m;j++) dp[1][j]=dp[1][j-1]+a[1][j];//1
/*2*/ for(int i=2;i<=n;i++){
    for(int j=2;j<=m;j++){
        dp[i][j]=min(dp[i-1][j],min(dp[i][j-1],dp[i-1][j-1]))+a[i][j];
    }
}
cout<dp\[n\]\[m\]<<endl;
}
```

```

int main(){
    int t; cin>>t;
    while (t--) initwsolve();
    return 0;
}

```

CATALAN NUMBER chưa làm

Ôn lại

Catalan Number là dãy số thỏa mãn biểu thức:

$$C_n = \begin{cases} 0 & \text{nếu } n = 0 \\ \sum_{i=0}^{n-1} C_i C_{n-i-1} & \text{nếu } n > 0 \end{cases}$$

Dưới đây là một số số Catalan với $n=0, 1, 2, \dots : 1, 1, 2, 5, 14, 42, 132, 429, \dots$. Cho số tự nhiên N. Nhiệm vụ của bạn là đưa ra số Catalan thứ N.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số nguyên n.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	
5	42
4	14
10	16796

Giải :

//1 và 2 : thực hiện tính tổng 2 chuỗi lớn , và tính 2 chuỗi lớn.

```

#include <bits/stdc++.h>
using namespace std;
string dp[102];
string sum(string a, string b){//1
    string res="";
    while(a.length() < b.length()) a="0"+a;
    while(b.length() < a.length()) b="0"+b;
    int carry=0;

```

```

for(int i=a.length()-1;i>=0;i--)
{
    int tmp=a[i]-48 + b[i]-48 + carry;
    carry=tmp/10;
    tmp=tmp%10;
    res=(char)(tmp+48)+res;
}
if(carry>0) res="1"+res;
return res;
}
string mul(string a, string b){//2
string res="";
int n=a.length(); int m=b.length();
int len=n+m-1;
int carry=0;
for(int i=len;i>=0;i--){
    int tmp=0;
    for(int j=n-1;j>=0;j--)
        if(i-j<=m && i-j>=1)
    {
        int a1=a[j]-48;
        int b1=b[i-j-1]-48;
        tmp+=a1*b1;
    }
    tmp+=carry;
    carry=tmp/10;
    res=(char)(tmp%10 + 48)+res;
}
while(res.length()>1 && res[0]=='0') res.erase(0,1);
return res;
}
void initwsolve(){
    int n;cin >> n ;
    cout << dp[n] << endl;
}
int main(){
    int t;cin>>t;
    dp[0]=dp[1]="1";

```

```

for(int i=2;i<=100;i++){
    dp[i]="0";
    for(int j=0;j<i;j++){
        dp[i]=sum(dp[i],mul(dp[j],dp[i-j-1]));
    }
}
while(t--) initwsolve();
return 0;
}

```

TÍNH P(N,K)

Ôn lại

$P(n, k)$ là số phép biểu diễn các tập con có thứ tự gồm k phần tử của tập gồm n phần tử. Số $P(n, k)$ được định nghĩa theo công thức sau:

$$P(n, k) = \begin{cases} 0 & \text{nếu } k > n \\ \frac{n!}{(n - k)!} = n \cdot (n - 1) \dots (n - k + 1) & \text{nếu } k \leq n \end{cases}$$

Cho số hai số n, k . Hãy tìm $P(n,k)$ theo modulo 10^9+7 .

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một cặp số n, k được viết trên một dòng.
- T, n, k thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, k \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
5 2	20
4 2	12

Cách 1 : tính trâu :

//nhân từ n đến $n-k+1$ và in kết quả ra màn hình

```
#include<bits/stdc++.h>
const int mod=1e9+7;
using namespace std;
```

```

int n,k;
void initwsolve(){
    cin>>n>>k;
    long long res=1;
    if(k>n){cout<<0<<endl; return;}
    for(int i=n;i>=n-k+1;i--){
        res= (res%mod*i%mod)%mod;
    }
    cout<<res<<endl;
}
int main(){
    int t; cin>>t;
    while(t--) initwsolve();
}

```

Cách 2 : quy hoạch động :

```

// 1:dp[i][j] : chính là iPj ; có iP0 luôn =1 => dp[i][0]=1
// 2:vd 7P3 :7.6.5 =6.5.4 (hay dp[i-1][j]) cộng với 6.5.(7-3) =>dp[i-1][j-1] *(n -(n-
k+1-1)) =j *dp[i-1][j-1])
#include<bits/stdc++.h>
const int mod=1e9+7;
using namespace std;
int n,k;
long long dp[1001][1001];
void solve(){
    for(int i=0;i<1001;i++) dp[i][0]=1;//1
    for(int i=1;i<1001;i++){
        for(int j=1;j<=i;j++){
            dp[i][j]=(dp[i-1][j]+j*dp[i-1][j-1])%mod;//2
        }
    }
}
int main(){
    int t; cin>>t;
    solve();
    while(t--) {
        cin>>n>>k;
        cout<<dp[n][k]<<endl;
    }
}

```

```
    }  
}
```

SỐ UGLY

Ôn lại

Số Ugly là các số chỉ có ước số là 2, 3, 5. Theo qui ước số 1 cũng là 1 số Ugly. Dưới đây là 11 số Ugly đầu tiên: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15. Cho số tự nhiên N, nhiệm vụ của bạn là tìm số Ugly thứ N.

Input: Dòng đầu tiên đưa vào số lượng bộ test T. Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số tự nhiên N được viết trên một dòng. T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^4$.

Output: Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ

Input	Output
2	12
10	4
4	

Giải :

Cách 1:

```
// i=1 :dp[1]=x=2,i2=1,x=4;  
// i=2 :dp[2]=y=3,i3=1,y=6;  
// i=3 :dp[3]=x=4,i2=2,x=6;  
// i=4 :dp[4]=z=5,i5=1,z=10;  
// i=5 :dp[5]=x=6,i2=3,x=8;      dp[5]=y=6,i3=2,y=9;....  
// 1:dp[i] : số ugly tại vị trí i+1;  
#include<bits/stdc++.h>  
using namespace std;  
int n;  
long long dp[1001];  
void solve(){  
    dp[0]=1;  
    int x=2,y=3,z=5,i2=0,i3=0,i5=0;  
    for(int i=1;i<1001;i++){  
        dp[i]=min(x,min(y,z));  
        if(dp[i]==x) i2++,x=dp[i2]*2;  
        if(dp[i]==y) i3++,y=dp[i3]*3;  
        if(dp[i]==z) i5++,z=dp[i5]*5;
```

```

    }
}

int main(){
    int t; cin>>t;
    solve();
    while(t--) {
        cin>>n;
        cout<<dp[n-1]<<endl;
    }
}

```

Cách 2:

```

#include<bits/stdc++.h>
using namespace std;
int n;
vector<long long> dp;
void solve(){
    for(int i=0;i<50;i++){
        for(int j=0;j<30;j++){
            for(int k=0;k<20;k++){
                long long temp= 1LL*pow(2,i)*pow(3,j)*pow(5,k);
                if(1<=temp&&temp<1e18)dp.push_back(temp);//note
1<=temp<=1e18 vi tra ve ngau nhien am duong
            }
        }
    }
    sort(dp.begin(),dp.end());
}

```

```

int main(){
    int t; cin>>t;
    solve();
    while(t--) {
        cin>>n;
        cout<<dp[n-1]<<endl;
    }
}

```

DÃY CON LẶP LẠI DÀI NHẤT

Ôn lại

Cho xâu ký tự S. Nhiệm vụ của bạn là tìm độ dài dãy con lặp lại dài nhất trong S. Dãy con có thể chứa các phần tử không liên tiếp nhau.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào độ dài xâu str; dòng tiếp theo đưa vào xâu S.
- T, str thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{size}(S) \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
3	
abc	0
5	2
axxxxy	

Giải thích test 5: axxxy(12345)

0	1	2	3	4	5
1	0	0	0	0	0
2	0	0	1	1	1
3	0	1	1	2	2
4	0	1	2	2	2
5	0	1	2	2	2

Giải :

Code c++:

```
//dp[i][j] : xâu con chung dài nhất tính đến phần tử thứ i của xâu s1 và phần tử thứ j của xâu s2 (trong bafi nayf s1=s2)
```

```
//2: tại trường hợp có kí tự chung thì kq chính là xâu con chung dài nhất tính đến phần tử thứ i-1 của xâu s1 và phần tử thứ j-1 của xâu s2 + thêm 1 (ký tự chung)=> dp[i-1][j-1]+1
```

```
//3: th nếu khác nhau thì độ dài xâu lớn nhất là độ dài lớn nhất tính đến ký tự i-1 của xâu s1 hoặc j-1 của xâu s2;
```

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

```
using namespace std;
```

```
string s;
```

```
int k,dp[105][105];//1
```

```

void initwsolve(){
    cin>>k>>s;
    int n=s.length();
    memset(dp,0,sizeof(dp));
    s=' '+s;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            if(s[i]==s[j]&&i!=j) dp[i][j]=dp[i-1][j-1]+1;
            else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
        }
    }
    cout<<dp[n][n]<<endl;
}
int main (){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

TỔNG LỚN NHẤT CỦA DÃY CON TĂNG DÀN

Cho dãy số $A[]$ gồm N số. Nhiệm vụ của bạn là tìm tổng lớn nhất của dãy con được sắp theo thứ tự tăng dần của dãy $A[]$. Ví dụ với dãy $A[] = \{1, 101, 2, 3, 100, 4, 5\}$ ta có kết quả là $106 = 1 + 2 + 3 + 100$. Với dãy $A[] = \{10, 7, 5\}$ ta có kết quả là 10. Với dãy $A[] = \{1, 2, 3, 5\}$ ta có kết quả là 11.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào N là số phần tử của dãy $A[]$; dòng tiếp theo đưa vào N số $A[i]$; các số được viết cách nhau một vài khoảng trắng.
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $0 \leq A[i] \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	106
7	10
1 101 2 3 100 4 5	11
3	

10 7 5	
4	
1 2 3 5	

//1: dp[i]: tổng dãy con lớn nhất tính đến chỉ số thứ i => dp[0]=a[0] tức số đầu
 //2: tại 1 vị trí bất kì trong mảng a, chỉ xét các số nhỏ hơn bằng trước nó . tìm tổng
 lớn nhất tăng dần đạt được từ 0 đến <i và gán giá trị bằng summax
 //=> tổng dãy con lớn nhất tính i bằng a[i]+maxsum

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    int a[n+1],dp[n+1];
    for(int i=0;i<n;i++)cin>>a[i];
    dp[0]=a[0];//1
    for(int i=1;i<n;i++){//2
        int summax=0;
        for(int j=0;j<i;j++)
            if(a[j]<a[i])summax=max(summax,dp[j]);
        dp[i]=a[i]+summax;
    }
    sort(dp,dp+n);
    cout<<dp[n-1]<<endl;
}
```

```
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}
```

Cách 2 :

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    int a[n+1],dp[n+1];
    for(int i=0;i<n;i++)cin>>a[i];
    memset(dp,0,sizeof(dp));
    dp[0]=a[0];
```

```

for(int i=1;i<n;i++){
    dp[i]=a[i];//không được để =0
    for(int j=0;j<i;j++)
        if(a[j]<=a[i])dp[i]=max(dp[i],dp[j]+a[i]);
}
sort(dp,dp+n);
cout<<dp[n-1]<<endl;
}

int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

SỐ BƯỚC ÍT NHẤT

Cho mảng A[] gồm N số nguyên. Nhiệm vụ của bạn là sắp xếp lại mảng số với số lượng bước là ít nhất. Tại mỗi bước, bạn chỉ được phép chèn phần tử bất kỳ của mảng vào vị trí bất kỳ trong mảng. Ví dụ A[] = {2, 3, 5, 1, 4, 7, 6 } sẽ cho ta số phép chèn ít nhất là 3 bằng cách lấy số 1 chèn trước số 2, lấy số 4 chèn trước số 5, lấy số 6 chèn trước số 7 ta nhận được mảng được sắp.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là một số N; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 1000$; $1 \leq A[i] \leq 1000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

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

Giải :

//1: dp[i]:dãy con tăng dần có độ dài lớn nhất từ vị trí 0 đến i (2 3 5 7) => số bước ít nhất là n- với dãy tăng >> của cả mảng

//2: chú ý là ta tìm dãy con tăng dần => phải xét if(a[j]<=a[i])

#include<bits/stdc++.h>

using namespace std;

```

void initwsolve(){
    int n; cin>>n;
    int a[n+1], dp[n+1];
    for(int i=0; i<n; i++) cin>>a[i];
    memset(dp, 0, sizeof(dp));
    dp[0]=1;//1
    for(int i=1; i<n; i++){//2
        dp[i]=1;
        for(int j=0; j<i; j++){
            if(a[j] <= a[i]) dp[i]=max(dp[i], dp[j]+1);
        }
    }
    int res=*max_element(dp, dp+n);
    cout<<n-res<<endl;
}
int main(){
    int t; cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

TỔNG LỚN NHẤT CỦA DÃY CON KHÔNG KỀ NHAU

Ôn lại

Cho dãy số $A[]$ gồm N phần tử. Hãy tìm tổng lớn nhất của dãy con của dãy số $A[]$ sao cho dãy con không có hai số cạnh nhau trong $A[]$. Ví dụ với $A[] = \{6, 7, 1, 3, 8, 2, 4\}$ ta có kết quả là 19 tương ứng với tổng của dãy con $\{6, 1, 6, 4\}$ không có hai phần tử nào kề nhau trong $A[]$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là một số N ; dòng tiếp theo đưa vào N số $A[i]$; các số được viết cách nhau một vài khoảng trắng.
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^6$; $1 \leq A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
6	
5 5 10 100 10 5	110
4	13
3 2 7 10	

Giải :

// 1:dp[i]: TỔNG LỚN NHẤT CỦA DÃY CON KHÔNG KÈ NHAU tính đến vị trí thứ i

// => khởi tạo dp[1]=a[1];dp[2]=a[2];dp[3]=a[1]+a[3];

// 2:tại vị trí i thì ta thấy chỉ cần xét tại vị trí i-2, và i-3 thôi , xem cái nào lớn hơn thì chọn và +a[i] thì ta được dp[i]

#include<bits/stdc++.h>

using namespace std;

void initwsolve(){

 int n;cin>>n;

 long long a[n+1],dp[n+1]={};

 for(int i=1;i<=n;i++) cin>>a[i],dp[i]=a[i];

 dp[1]=a[1];dp[2]=a[2];dp[3]=a[1]+a[3];//1

 for(int i=4;i<=n;i++){//2

 dp[i]=max(dp[i-2],dp[i-3])+a[i];

 }

 sort(dp,dp+n+1);

 cout<<dp[n]<<endl;

}

int main(){

 int t; cin>>t;

 while(t--) initwsolve();

 return 0;

}

XEM PHIM

ôn lại

John có một đàn bò. Một ngày đẹp trời, anh ta quyết định mua xe tải với khả năng chở được C kg ($1000 \leq C \leq 25000$) để đưa những con bò đi xem phim. Cho số con bò là N ($20 \leq N \leq 100$) và khối lượng w[i] của từng con (đều nhỏ hơn C), hãy cho biết khối lượng bò lớn nhất mà John có thể đưa đi xem phim là bao nhiêu.

Input:

- Dòng 1: 2 số nguyên C và N cách nhau bởi dấu cách
- Dòng 2..N+1: Ghi lần lượt các số nguyên: w[i]

Output:

- Một số nguyên là tổng khối lượng bò lớn nhất mà John có thể mang đi xem phim.

Ví dụ:

Input	Output
259 5	
81	
58	
42	
33	
61	
	242

Giải :

Cách 1:

// 1:dp[i][j] :khối lượng lớn nhất khi chọn đến con bò thứ I và khối lượng tối đa là j

// note : phải đưa biến toàn cục

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

```
using namespace std;
```

```
int c,n,a[101],dp[101][25001];
```

```
int main()
```

```
    cin>>c>>n;
```

```
    for(int i=1;i<=n;i++)cin>>a[i];
```

```
    memset(dp,0,sizeof(dp));
```

```
    for(int i=0;i<=c;i++)dp[0][i]=0;
```

```
    for(int i=1;i<=n;i++){
```

```
        for(int j=1;j<=c;j++){
```

```
            if(a[i]<=j)dp[i][j]=max(dp[i-1][j], dp[i-1][j-a[i]]+a[i]);
```

```
            else dp[i][j]=dp[i-1][j];
```

```
}
```

```
}
```

```
    cout<<dp[n][c]<<endl;
```

```
    return 0;
```

```
}
```

Cách 2:

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int c,n; cin>>c>>n;
    int a[n+1],dp[c+1];
    memset(dp,0,sizeof(dp));
    dp[0]=1;
    for(int i=0;i<n;i++) cin>>a[i];
    for(int i=0;i<n;i++){
        for(int j=c;j>=a[i];j--){
            dp[j]=dp[j]|dp[j-a[i]];
        }
    }
    while(c--) {
        if(dp[c]==1) {
            cout<<c<<endl;
            break;
        }
    }
    return 0;
}

```

CÁI TÚI

ôn lại

Một người có cái túi thể tích V ($V < 1000$). Anh ta có N đồ vật cần mang theo ($N \leq 1000$), mỗi đồ vật có thể tích là $A[i]$ ($A[i] \leq 100$) và giá trị là $C[i]$ ($C[i] \leq 100$). Hãy xác định tổng giá trị lớn nhất của các đồ vật mà người đó có thể mang theo, sao cho tổng thể tích không vượt quá V.

Input

- Dòng đầu ghi số bộ test T ($T < 10$)
- Mỗi bộ test gồm ba dòng. Dòng đầu ghi 2 số N và V. Dòng tiếp theo ghi N số của mảng A. Sau đó là một dòng ghi N số của mảng C.
- Dữ liệu vào luôn đảm bảo không có đồ vật nào có thể tích lớn hơn V.

Output

- Với mỗi bộ test, ghi trên một dòng giá trị lớn nhất có thể đạt được.

Ví dụ

Input	Output
1 15 10	15

5 2 1 3 5 2 5 8 9 6 3 1 4 7 8	
1 2 3 5 1 2 5 8 7 4 1 2 3 2 1	

Giải :

Tư duy ý tưởng :

Trường hợp mỗi vật được chọn 1 lần

-Ta nhận thấy rằng: Giá trị của cái túi phụ thuộc vào 2 yếu tố: Có bao nhiêu vật đang được xét và trọng lượng còn lại cái túi có thể chứa được, do vậy chúng ta có 2 đại lượng biến thiên. Cho nên hàm mục tiêu sẽ phụ thuộc vào hai đại lượng biến thiên. Do vậy bảng phương án của chúng ta sẽ là bảng 2 chiều.

-Gọi $dp[i,j]$ là tổng giá trị lớn nhất của cái túi khi xét từ vật 1 đến vật i và trọng của cái túi chưa vượt quá j . Với giới hạn j , việc chọn tối ưu trong số các vật $\{1,2,\dots,i-1,i\}$ để có giá trị lớn nhất sẽ có hai khả năng:

+, Nếu không chọn vật thứ i thì $dp[i,j]$ là giá trị lớn nhất có thể chọn trong số các vật $\{1,2,\dots,i-1\}$ với giới hạn trọng lượng là j , tức là: $dp[i,j]:=dp[i-1,j]$.

+, Nếu có chọn vật thứ i (phải thỏa điều kiện $a[i] \leq j$) thì $dp[i,j]$ bằng giá trị vật thứ i là $c[i]$ cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật $\{1,2,\dots,i-1\}$ với giới hạn trọng lượng $j-a[i]$ tức là về mặt giá trị thu được:

$$dp[i,j]:=c[i]+dp[i-1,j-a[i]]$$

Vậy chúng ta phải xem xét xem nếu chọn vật i hay không chọn vật i thì sẽ tốt hơn. Từ đó chúng ta có công thức truy hồi như sau.

- $dp[0,j] = 0$ (hiển nhiên) – Bài toán con nhỏ nhất.
- $dp[i,j] = \max(dp[i-1,j], c[i]+dp[i-1,j-a[i]])$

Giải :

Code c++:

```
//1 :dp[i,j] là tổng giá trị lớn nhất của cái túi khi xét từ vật 1 đến vật i và trọng của cái túi chưa vượt quá j
```

```
// => điều kiện đầu : dp[0][i]=0;
```

```
//2:đã trình bày ở phần ý tưởng
```

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

```
int n,v;
```

```
int a[1005],c[1005],dp[1005][1005];
```

```
using namespace std;
```

```
void initwsolve()
```

```
    cin>>n>>v;
```

```
    for(int i=1;i<=n;i++) cin>>a[i];
```

```
    for(int i=1;i<=n;i++) cin>>c[i];
```

```
    for(int i=0;i<=v;i++) dp[0][i]=0;//1
```

```
    for(int i=1;i<=n;i++){
```

```

        for(int j=1;j<=v;j++){//2
            if(a[i]<=j) dp[i][j]=max(dp[i-1][j],dp[i-1][j-a[i]]+c[i]);
            else dp[i][j]=dp[i-1][j];
        }
    cout<<dp[n][v]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

GIẢI MÃ

Ôn lại

Một bản tin M đã mã hóa bí mật thành các con số theo ánh xạ như sau: ‘A’->1, ‘B’->2, …, ‘Z’->26. Hãy cho biết có bao nhiêu cách khác nhau để giải mã bản tin M. Ví dụ với bản mã M=”123” nó có thể được giải mã thành ABC (1 2 3), LC (12 3), AW(1 23).

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu ký tự số M.
- T, M thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(M) \leq 40$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
123	3
2563	2

Giải :

Code c++:

```

//1: dp[i]: số cách mã hóa đến ký tự thứ i => dp[0]=1;dp[1]=1;
// ta thấy s[i] có thể hiểu giả mã bằng cách thêm 1 ký tự sau s[i-1] hoặc thêm 2 ký
tự vào sau s[i-2] (2 ký tự này thuộc từ 10 đến 26 )
//=>2: khi s[i]==0 ta không tính, s[i]>'0' thì dp[i]=dp[i-1]
//3:và ta phải xét thêm khi xét 2 ký tự trước nó thuộc từ 10 đến 26 thì :dp[i]+=dp[i-
2];(cộng thêm)
#include<bits/stdc++.h>

```

```

using namespace std;
void initwsolve(){
    string s;cin>>s;
    int dp[45],n=s.length();
    if(s[0]=='0'){cout<<0<<endl;return;}
    s=' '+s;
    memset(dp,0,sizeof(dp));
    dp[0]=1;dp[1]=1;//1
    for(int i=2;i<=n;i++){
        if(s[i]>'0')dp[i]=dp[i-1];//2
        if(s[i-1]=='1'&(s[i-1]=='2'&&s[i]<'7'))dp[i]+=dp[i-2];//3
    }
    cout<<dp[n]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

CATALAN NUMBER

Ôn lại

Catalan Number là dãy số thỏa mãn biểu thức:

Dưới đây là một số số Catalan với $n=0, 1, 2, \dots$: 1, 1, 2, 5, 14, 42, 132, 429, ... Cho số tự nhiên N. Nhiệm vụ của bạn là đưa ra số Catalan thứ N.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số nguyên n.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	42
5	14
4	16796
10	

Giải:

```
//1 tính tổng 2 chuỗi , //2 tính tích 2 chuỗi  
//3: xây dựng cách đ=tính theo công thức thông thường ;vd n=3 :  
=c0*c2+c1*c1+c2*c0  
#include <bits/stdc++.h>  
using namespace std;  
string dp[102];  
string sum(string a, string b){//1  
    string res="";  
    while(a.length() < b.length()) a="0"+a;  
    while(b.length() < a.length()) b="0"+b;  
    int carry=0;  
    for(int i=a.length()-1;i>=0;i--)  
    {  
        int tmp=a[i]-48 + b[i]-48 + carry;  
        carry=tmp/10;  
        tmp=tmp%10;  
        res=(char)(tmp+48)+res;  
    }  
    if(carry>0) res="1"+res;  
    return res;  
}  
string mul(string a, string b){//2  
    string res="";  
    int n=a.length(); int m=b.length();  
    int len=n+m-1;  
    int carry=0;  
    for(int i=len;i>=0;i--){  
        int tmp=0;  
        for(int j=n-1;j>=0;j--)  
            if(i-j<=m && i-j>=1)  
            {  
                int a1=a[j]-48;  
                int b1=b[i-j-1]-48;  
                tmp+=a1*b1;  
            }  
        tmp+=carry;  
        carry=tmp/10;
```

```

        res=(char)(tmp%10 + 48)+res;
    }
    while(res.length()>1 && res[0]=='0') res.erase(0,1);
    return res;
}
void initwsolve(){
    int n;cin >> n ;
    cout << dp[n] << endl;
}
int main(){
    int t;cin>>t;
    dp[0]=dp[1]="1";//3
    for(int i=2;i<=100;i++){
        dp[i]="0";
        for(int j=0;j<i;j++){
            dp[i]=sum(dp[i],mul(dp[j],dp[i-j-1]));
        }
    }
    while(t--) initwsolve();
    return 0;
}

```

DÃY CON CHUNG DÀI NHẤT CỦA BA XÂU

Bài làm tốt nhất

Cho ba xâu ký tự X, Y, Z. Nhiệm vụ của bạn là tìm độ dài dãy con chung dài nhất có mặt trong cả ba xâu.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào độ dài xâu X, Y, X; dòng tiếp theo đưa vào ba xâu X, Y, Z.
- T, X, Y, Z thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{size}(X), \text{size}(Y), \text{size}(Z) \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	5
5 8 13	3
geeks geeksfor geeksforgeeks	

7 6 5

abcd1e2 bc12ea bd1ea

Giải :

//1: dp[i][j][k]:dãy con chung dài nhất có mặt trong cả ba xâu tính đến vị trí i của xâu thứ nhất , j của xâu thứ 2 và k của xâu thứ 3
//2: nếu xét tại vị trí i của xâu thứ nhất , j của xâu thứ 2 và k của xâu thứ 3 mà 3 vị trí đó trùng nhau => độ dài dãy lớn nhất chính là xâu chung max trước đó +1 =>dp[i][j][k]=dp[i-1][j-1][k-1]+1;
//3 :còn không trùng nhau thì chắc chắn xâu dài nhất se là max trong 3 th : th1 tính đến vị trí i-1 của xâu thứ nhất,j,k của xâu thứ 2 và 3. Th2 là tính đến vị trí i của xâu 1, j-1 của xâu 2,k của xâu 3 . Và th3 là vị trí i,j của xâu 1,2 và k-1 của xâu thứ 3 => ta có ct qhđ nhu dưới

```
#include <bits/stdc++.h>
using namespace std;
int n1,n2,n3,dp[102][102][102];
string x,y,z;
void initwsolve(){
    cin>>n1>>n2>>n3;
    cin>>x>>y>>z;
    x=' '+x;y=' '+y,z=' '+z;
    for(int i=1;i<=n1;i++){
        for(int j=1;j<=n2;j++){
            for(int k=1;k<=n3;k++){
                if(x[i]==y[j]&&y[j]==z[k])dp[i][j][k]=dp[i-1][j-1][k-1]+1;//2
                else dp[i][j][k]=max(dp[i-1][j][k],max(dp[i][j-1][k],dp[i][j][k-1]));//3
            }
        }
    }
    cout<<dp[n1][n2][n3]<<endl;//1
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

DÃY SỐ BI-TONIC

Ôn lại

Một dãy số được gọi là Bi-tonic nếu nó được chia thành hai dãy đầu tăng dần và dãy tiếp theo giảm dần. Nhiệm vụ của bạn là tìm tổng lớn nhất dãy con Bi-tonic của dãy số A[]. Ví dụ với dãy $A[] = \{1, 15, 51, 45, 33, 100, 12, 18, 9\}$ ta có kết quả là 194 tương ứng với dãy Bi-tonic $\{1, 15, 51, 100, 18, 9\}$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào N là số phần tử của dãy A[]; dòng tiếp theo đưa vào N số A[i]; các số được viết cách nhau một vài khoảng trắng.
- $T, N, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 100$; $0 \leq A[i] \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
6	
80 60 30 40 20 10	210
9	194
1 15 51 45 33 100 12 18 9	

Giải:

// 1: dpl[i]=thực hiện tính tổng dãy con tăng dần tính đến vị trí i
// 2: dpr[i]=thực hiện tính tổng dãy con tăng dần ngược lại từ n đến vị trí i \Leftrightarrow dãy con giảm dần từ i đến n;
// 3=> công hai dpl[i]+dpr[i] và -a[i] ta được kết quả dãy bi tonic như mong muốn

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

```
int n,a[102],dpl[102],dpr[102];
string x,y,z;
```

```
void initwsolve()
```

```
    cin>>n;
```

```
    for(int i=1;i<=n;i++) cin>>a[i];
```

```
    memset(dpl,0,sizeof(dpl)); //bắt buộc phải memset vì nếu không
    memset(dpr,0,sizeof(dpr)); //các test sau sẽ bị sai khi so sánh dp[i]
```

```
    for(int i=1;i<=n;i++){//1
```

```
        for(int j=0;j<i;j++){
    
```

```
            if(a[j]<a[i])dpl[i]=max(dpl[i],dpl[j]+a[i]);
    
```

```
        }
    
```

```
}
```

```

for(int i=n+1;i>=1;i--){
    for(int j=n+1;j>i;j--){
        if(a[j]<a[i])dpr[i]=max(dpr[i],dpr[j]+a[i]);
    }
}
int res=0;
for(int i=1;i<=n;i++){
    res=max(res,dpl[i]+dpr[i]-a[i]);
}
cout<<res<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

KÝ TỰ GIỐNG NHAU

ôn lại

Giả sử bạn cần viết N ký tự giống nhau lên màn hình. Bạn chỉ được phép thực hiện ba thao tác dưới đây với chi phí thời gian khác nhau:

- Thao tác insert: chèn một ký tự với thời gian là X.
- Thao tác delete: loại bỏ ký tự cuối cùng với thời gian là Y.
- Thao tác copying: copy và paste tất cả các ký tự đã viết để số ký tự được nhân đôi với thời gian là Z.

Hãy tìm thời gian ít nhất để có thể đưa ra màn hình N ký tự giống nhau. Ví dụ với $N = 9$, $X = 1$, $Y = 2$, $Z = 1$ ta có kết quả là 5 bằng cách thực hiện: insert, insert, copying, copying, insert.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào N là số các ký tự giống nhau cần viết lên màn hình; dòng tiếp theo đưa vào bộ ba số X, Y, Z tương ứng với thời gian thực hiện ba thao tác; các số được viết cách nhau một vài khoảng trắng.
- T, N, X, Y, Z thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 100$; $1 \leq X, Y, Z \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
-------	--------

2	
9	
1 2 1	5
10	14
2 5 4	

Giải :

```
// 1:dp[i]: số thao tác ít nhất khi viết được i ký tự giống nhau => dp[1]=x
// 2: tại chuỗi đang xét phải xem lẻ hay chẵn : chuỗi chẵn thì có 2 cách tạo bằng
// cách thêm 1 ký tự vào chuỗi i-1 , hoặc nhân đôi chuỗi i/2; => dp[i]=min(dp[i-
// 1]+x,dp[i/2]+z);
// 3:nếu chuỗi lẻ có 2 cách : thêm 1 ký tự vào chuỗi i-1 , hoặc nhân đôi chuỗi
// (i+1)/2 rồi trừ đi 1 ký tự
#include <bits/stdc++.h>
using namespace std;
int n,x,y,z,dp[102];
void initwsolve(){
    cin>>n>>x>>y>>z;
    dp[1]=x;//1
    for(int i=2;i<=n;i++){
        if(i%2==0)dp[i]=min(dp[i-1]+x,dp[i/2]+z);//2
        else dp[i]=min(dp[i-1]+x,dp[(i+1)/2]+y+z);//3
    }
    cout<<dp[n]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TỔNG CÁC XÂU CON

Ôn lại

Cho số nguyên dương N được biểu diễn như một xâu ký tự số. Nhiệm vụ của bạn là tìm tổng của tất cả các số tạo bởi các xâu con của N. Ví dụ N="1234" ta có kết quả là $1670 = 1 + 2 + 3 + 4 + 12 + 23 + 34 + 123 + 234 + 1234$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một số N.

- T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^{12}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	1670
1234	491
421	

Giải :

// 1:dp[i]: tổng của tất cả các số tạo bởi các xâu con của N tính đến chỉ số i bao gồm s[i]; => dp[0]=s[0]-'0'; vd:N="1234" thì tính tổng của $1234+234+34+4$
// 2:=> cộng tất cả các dp[i] từ đầu đến cuối ta được res chính là tổng của tất cả các số tạo bởi các xâu con của N.

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s;cin>>s;
    long long n=s.length();
    long long dp[n+1];
    dp[0]=s[0]-'0';//1
    long long res=dp[0];
    for(int i=1;i<n;i++){
        dp[i]=10*dp[i-1]+(s[i]-'0')*(i+1);
        res+=dp[i];
    }
    cout<<res<<endl;
}
```

```
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TỔNG BẰNG K

Ôn lại

Cho một mảng A[] gồm N số nguyên và số K. Tính số cách lấy tổng các phần tử của A[] để bằng K. Phép lấy lặp các phần tử hoặc sắp đặt lại các phần tử được chấp thuận. Ví dụ với mảng A[] = {1, 5, 6}, K = 7 ta có 6 cách sau:

$7 = 1 + 1 + 1 + 1 + 1 + 1$ (lặp số 1 7 lần)

$7 = 1 + 1 + 5$ (lặp số 1)

$7 = 1 + 5 + 1$ (lặp và sắp đặt lại số 1)

$7 = 1 + 6$

$7 = 6 + 1$

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào N và K; dòng tiếp theo đưa vào N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- $T, N, K, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 1000$; $1 \leq A[i] \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng. Khi kết quả quá lớn đưa ra kết quả dưới dạng modulo với $10^9 + 7$.

Ví dụ:

Input	Output
2	
3 7	
1 5 6	6
4 14	150
12 3 1 9	

Giải :

Cách 1 c++:

// 1: dp[i] : số cách lấy tổng các phần tử của A[] để bằng i=>dp[0]=1;
// 2: vd ta thấy nếu mảng có số 2,các cách dp[3] thì các cách cho dp[5] sẽ bao gồm dp[3] ; bên cạnh đó ta thấy nếu mảng có số 1,các cách dp[4] thì các cách cho dp[5] cũng sẽ bao gồm dp[4] => dp[i] sẽ là tổng của các=dp[i-a[j]] (khi a[j]<i tức các số có trong mảng nhỏ hơn tổng mảng) => thực hiện vòng for là ra

```
#include <bits/stdc++.h>
using namespace std;
const long long mod=1e9+7;
int n,k,a[1001];
void initwsolve(){
    cin>>n>>k;
    long long dp[k+2];
    memset(dp,0,sizeof(dp));
    for(int i=1;i<=n;i++) cin>>a[i];
    dp[0]=1;
    for(int i=1;i<=k;i++){
        for(int j=1;j<=n;j++){
            if(j-a[i]>=0)
                dp[i]=(dp[i]+dp[j-a[i]])%mod;
        }
    }
}
```

```

        for(int j=1;j<=n;j++){
            if(a[j]<=i) dp[i]+=dp[i-a[j]]%mod;//2
        }
    }cout<<dp[k]%mod<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

BIẾN ĐỔI XÂU

Ôn lại

Cho hai xâu ký tự str1, str2 bao gồm các ký tự in thường và các thao tác dưới đây:

- Insert: chèn một ký tự bất kỳ vào str1.
- Delete: loại bỏ một ký tự bất kỳ trong str1.
- Replace: thay một ký tự bất kỳ trong str1.

Nhiệm vụ của bạn là đếm số các phép Insert, Delete, Replace ít nhất thực hiện trên str1 để trở thành str2.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ đôi hai xâu str1 và str2.
- T, str1, str2 thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length(str1)}, \text{length(str2)} \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
1	
geek gesek	1

Giải :

//1 :dp[i][j]: số các phép biến đổi ít nhất thực hiện trên str1 để trở thành str2 tính tại vị trí thứ i của s1 và j của s2 ;=> khi s1 bằng rỗng thì số phép biến đổi từ chuỗi s1 đến vị trí i của chuỗi s2 chính bằng i luôn . tương tự khi s2 rỗng

//2: ta thấy tại vị trí i chuỗi s1 và j chuỗi s2 , nếu 2 ký tự đó trùng nhau thì sẽ không cần phải chèn , xóa , thay thế nữa => $dp[i][j] = dp[i-1][j-1]$;

//3: còn nếu chúng khác nhau thì chắc chắn kq sẽ là 1 trong 3 th biến đổi kia => có hệ thứ như dưới

```

#include <bits/stdc++.h>
using namespace std;
string s1,s2;
int dp[105][105];
void initwsolve(){
    cin>>s1>>s2;
    int m=s1.length(),n=s2.length();
    s1=' '+s1;s2=' '+s2;
    for(int i=1;i<=n;i++) dp[0][i]=i;//1
    for(int i=1;i<=m;i++) dp[i][0]=i;//1
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(s1[i]==s2[j]) dp[i][j]=dp[i-1][j-1];//2
            else dp[i][j]=min(dp[i-1][j],min(dp[i][j-1],dp[i-1][j-1]))+1;//3
        }
    }
    cout<<dp[m][n]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

TỔNG BÌNH PHƯƠNG

Ôn lại

Mọi số nguyên dương N đều có thể phân tích thành tổng các bình phương của các số nhỏ hơn N. Ví dụ số $100 = 10^2$ hoặc $100 = 5^2 + 5^2 + 5^2 + 5^2$. Cho số nguyên dương N. Nhiệm vụ của bạn là tìm số lượng ít nhất các số nhỏ hơn N mà có tổng bình phương bằng N.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi test là một số tự nhiên N được viết trên 1 dòng.
- T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10000$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	1
100	3
6	1
25	

Giải :

```
// 1: dp[i]: số lượng ít nhất các số nhỏ hơn i mà có tổng bình phương bằng i.=>
dp[1]=1,dp[2]=2,dp[3]=3;
//2: ta thấy i = tổng các bình phương cộng lại <=> tổng các bình phương + x bình
nào đó => ý tưởng số cách nhỏ nhất của dp[i] chính là kết quả nhỏ nhất của 1
trong các dp[a] nào đó + x bình=> dp[i]=min(dp[i],dp[i-j*j]+1)
#include <bits/stdc++.h>
using namespace std;
int n,dp[10002];
void initwsolve(){
    cin>>n;
    cout<<dp[n]<<endl;
}
int main(){
    int t;cin>>t;
    dp[1]=1,dp[2]=2,dp[3]=3;//1
    for(int i=4;i<=10000;i++){
        dp[i]=i;
        for(int j=1;j<=sqrt(i);j++){//2
            dp[i]=min(dp[i],dp[i-j*j]+1);
        }
    }
    while(t--) initwsolve();
    return 0;
}
```

NHÀ KHÔNG KÈ NHAU

Trùng : tổng lớn nhất của dãy con không kề nhau

Có N ngôi nhà trên một dãy phố, mỗi ngôi nhà chứa đựng một số lượng tài sản khác nhau. Một tên trộm muốn cắp được nhiều nhất tài sản của dãy phố nhưng không muốn lấy tài sản của hai nhà kề nhau. Hãy cho biết, bằng cách đó tên trộm có thể đánh cắp được nhiều nhất bao nhiêu tài sản.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
 - Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất là một số N là số lượng ngôi nhà; dòng tiếp theo đưa vào N số là tài sản tương ứng trong mỗi ngôi nhà; các số được viết cách nhau một vài khoảng trắng.
 - T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^6$; $1 \leq A[i] \leq 10^7$.
- Output:
- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	110
6	13
5 5 10 100 10 5	
4	
3 2 7 10	

Giải :

Code c++:

```
// 1:dp[i]: TỔNG LỚN NHẤT CỦA DÃY CON KHÔNG KÈ NHAU tính đến vị trí thứ i
```

```
// => khởi tạo dp[1]=a[1];dp[2]=a[2];dp[3]=a[1]+a[3];
```

```
// 2:tại vị trí i thì ta thấy chỉ cần xét tại vị trí i-2, và i-3 thôi , xem cái nào lớn hơn thì chọn và +a[i] thì ta được dp[i]
```

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    int n;cin>>n;
```

```
    long long a[n+1],dp[n+1]={};
```

```
    for(int i=1;i<=n;i++) cin>>a[i],dp[i]=a[i];
```

```
    dp[1]=a[1];dp[2]=a[2];dp[3]=a[1]+a[3];//1
```

```
    for(int i=4;i<=n;i++){//2
```

```
        dp[i]=max(dp[i-2],dp[i-3])+a[i];
```

```
}
```

```
    sort(dp,dp+n+1);
```

```
    cout<<dp[n]<<endl;
```

```
}
```

```
int main(){
```

```
    int t; cin>>t;
```

```
    while(t--) initwsolve();
    return 0;
}
```

XÂU ĐỐI XỨNG 1 chưa

Ôn lại

Cho xâu ký tự str. Nhiệm vụ của bạn là tìm số phép chèn tối thiểu các ký tự vào str để str trở thành xâu đối xứng. Ví dụ: str = "ab" ta có số phép chèn tối thiểu là 1 để trở thành xâu đối xứng "aba" hoặc "bab". Với xâu str="aa" thì số phép chèn tối thiểu là 0. Với xâu str="abcd" có số phép chèn tối thiểu là 3 để trở thành xâu "dcbabcd"

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu ký tự được viết trên một dòng
- T, str thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length(str)} \leq 40$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	3
abcd	0
aba	3
geeks	

Giải:

XÂU ĐỐI XỨNG 2 chưa

Bài làm tốt nhất

Cho xâu ký tự S. Nhiệm vụ của bạn là tìm số phép loại bỏ ít nhất các ký tự trong S để S trở thành xâu đối xứng. Chú ý, phép loại bỏ phải bảo toàn tính trước sau của các ký tự trong S.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu ký tự được viết trên một dòng
- T, str thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S) \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	2
aebcbda	8
geeksforgeeks	

CẶP SỐ

Ôn lại

Cho N cặp số, trong đó số thứ nhất bao giờ cũng nhỏ hơn số thứ 2. Ta nói, cặp số $\langle c, d \rangle$ được gọi là sau cặp số $\langle a, b \rangle$ nếu $b < c$. Nhiệm vụ của bạn là tìm số lớn nhất chuỗi các cặp thỏa mãn ràng buộc trên. Ví dụ với các cặp $\{ \langle 5, 24 \rangle, \langle 39, 60 \rangle, \langle 15, 28 \rangle, \langle 27, 40 \rangle, \langle 50, 90 \rangle \}$ ta có kết quả là 3 tương ứng với chuỗi các cặp $\{ \langle 5, 24 \rangle, \langle 27, 40 \rangle, \langle 50, 90 \rangle \}$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào N là số cặp $\langle a, b \rangle$; dòng tiếp theo đưa vào $2*N$ số là N cặp số $\langle a, b \rangle$; các số được viết cách nhau một vài khoảng trắng.
- T, N, a, b thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, a, b \leq 100$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
5	
5 24 39 60 15 28 27 40 50 90	3
2	1
5 10 1 11	

Giải :

Code c++:

```
// 1: sắp xếp mảng arr theo chiều chiều tăng dần của số a trong từng cặp số.
//2: dp[i]: số lớn nhất chuỗi các cặp thỏa mãn ràng buộc bài toán tính đến vị trí thứ i của mảng
//3 : tại vị trí i , xét các cặp trước cặp i , nếu số đầu của cặp i > số thứ 2 của cặp j thì
    // cặp i sẽ thỏa mãn => cập nhật max dp[j] sau đó +1 (chính cặp thứ i) ta được dp[i]
#include<bits/stdc++.h>
```

```

using namespace std;
int n,dp[105];
struct p{
    int a,b;
};p arr[105];
bool cmp(p x,p y){return x.a<y.a;}
void initwsolve(){
    int n;cin>>n;
    for(int i=1;i<=n;i++) cin>>arr[i].a>>arr[i].b;
    sort(arr+1,arr+n+1,cmp);//1
    memset(dp,0,sizeof(dp));
    dp[1]=1;// không cần cho =1 vẫn đúng vì khi so sánh với dp[0] có thể cập nhật kết quả
    for(int i=1;i<=n;i++){
        for(int j=0;j<i;j++){//3
            if(arr[i].a>arr[j].b)dp[i]=max(dp[i],dp[j]+1);
        }
    }
    sort(dp,dp+n+1);
    cout<<dp[n]<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

DI CHUYỂN VỀ GÓC TỌA ĐỘ

Ôn lại

Giả sử bạn đang ở điểm có tọa độ nguyên dương (n,m) và cần dịch chuyển về tọa độ $(0,0)$. Mỗi bước dịch chuyển bạn chỉ được phép dịch chuyển đến tọa độ $(n-1, m)$ hoặc $(n, m-1)$; Từ ô $(0,m)$, hoặc $(n, 0)$ thì có thể di chuyển 1 bước để về gốc $(0,0)$.

Hãy đếm số cách bạn có thể dịch chuyển về tọa độ $(0,0)$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là bộ n, m được viết cách nhau một vài khoảng trắng.
- T, n, m thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, m \leq 25$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	10
3 2	84
3 6	1
3 0	

Giải :

Code c++:

```
//1 :dp[i][j]: số cách bạn có thể dịch chuyển từ tọa độ(i,j) về tọa độ (0,0)
// 2:mỗi số hàng 1 và cột 1 có 1 cách về (0,0) :=>dp[0][i]=1,dp[i][0]=1;
// 3:ta thấy có 2 cách để đi đến tọa độ (i,j):(i-1,j) đi thêm 1 bước , và(i,j-1) và tiến
1 bước => suy rộng cho các cách ta có dp[i][j]=dp[i-1][j]+dp[i][j-1];
#include<bits/stdc++.h>
using namespace std;
long long dp[26][26];//note ll
void initwsolve(){
    int n,m;cin>>n>>m;
    cout<<dp[n][m]<<endl;//1
}
int main(){
    int t;cin>>t;
    for(int i=1;i<=25;i++)dp[0][i]=1,dp[i][0]=1;//2
    for(int i=1;i<=25;i++){
        for(int j=1;j<=25;j++){
            dp[i][j]=dp[i-1][j]+dp[i][j-1];//3
        }
    }
    while (t--)initwsolve();
    return 0;
}
```

XÂU ĐỐI XỨNG 1

Ôn lại

Cho xâu ký tự str. Nhiệm vụ của bạn là tìm số phép chèn tối thiểu các ký tự vào str để str trở thành xâu đối xứng. Ví dụ: str = "ab" ta có số phép chèn tối thiểu là 1 để trở thành xâu đối xứng "aba" hoặc "bab". Với xâu str="aa" thì số phép chèn tối

thiểu là 0. Với xâu str="abcd" có số phép chèn tối thiểu là 3 để trở thành xâu "dcbabcd"

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test là một xâu ký tự được viết trên một dòng
- T, str thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length(str)} \leq 40$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	3
abcd	0
aba	3
geeks	

Giải :

//1:dp[i][j]: số cách chèn tối thiểu của 1 chuỗi tính từ ký tự i của chuỗi đến ký tự j của chuỗi

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string s; cin>>s;
    int n=s.length();
    int dp[45][45];
    memset(dp,0,sizeof(dp));
    for(int i=0;i<n;i++){
        for(int j=0,k=i;k<n;k++,j++){
            if(s[j]==s[k]) dp[j][k]=dp[j+1][k-1];
            else dp[j][k]=min(dp[j+1][k],dp[j][k-1])+1;
            //bên dưới hoặc bên trái
        }
    }
    cout<<dp[0][n-1]<<endl;//1
}
```

```

int main(){
    int t; cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

Sắp xếp - Tìm kiếm

SẮP XẾP ĐỔI CHỖ TRỰC TIẾP

Hãy thực hiện thuật toán sắp xếp đổi chỗ trực tiếp trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán. Dữ liệu vào: Dòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100). Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trắng.

Ví dụ:

Input	Output
4	Buoc 1: 2 7 5 3
5 7 3 2	Buoc 2: 2 3 7 5
	Buoc 3: 2 3 5 7

Giải :

Cách 1 : c++

```

#include<bits/stdc++.h>
using namespace std;
int n,a[101];
void out(){
    for(int i=0;i<n;i++) cout<<a[i]<<' ';
}
int main(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    for(int i=0;i<n-1;i++){
        for(int j=i+1;j<n;j++){
            if(a[j]<a[i]) swap(a[i],a[j]);
        }
        cout<<"Buoc "<<i+1<<": ";
        out();
        cout<<endl;
    }
}

```

```
    return 0;
```

```
}
```

Cách 2 : python 3

```
n=int(input())
a=[]
def out():
    for i in range(n) : print(a[i],end=' ')
a = [int(i) for i in input().split()]# nhập trên 1 dòng
for i in range(0,n-1):
    for j in range(i,n):
        if a[j] < a[i]:
            a[j],a[i] = a[i],a[j]
    print("Buoc "+str(i+1)+":",end=' ')
out()
print()
```

cách 3 : python 3 khác :

```
n = int(input())
a = [int(i) for i in input().split()]
def out():
    for i in a:
        print(i,end = " ")
    print()
for i in range(n-1):
    for j in range(i+1,n):
        if a[j] < a[i]:
            a[j],a[i] = a[i],a[j]
    print("Buoc "+str(i+1) + ":" ,end = " ")
out()
```

SẮP XẾP CHỌN

Hãy thực hiện thuật toán sắp xếp chọn trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán.

Dữ liệu vào: Dòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100).

Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trống.

Ví dụ:

Input	Output
4	Buoc 1: 2 7 3 5
5 7 3 2	Buoc 2: 2 3 7 5
	Buoc 3: 2 3 5 7

Giải :

Cách 1 : c++

```
#include<bits/stdc++.h>
using namespace std;
int n,a[101];
void out(){
    for(int i=0;i<n;i++) cout<<a[i]<<' ';
}
int main(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    for(int i=0;i<n-1;i++){
        int k=i,temp=a[i];
        for(int j=i+1;j<n;j++){
            if(a[j]<temp){
                k=j;temp=a[j];
            }
        }
        swap(a[i],a[k]);
        cout<<"Buoc "<<i+1<<": ";
        out();
        cout<<endl;
    }
    return 0;
}
```

Cách 2: python 3 :

```
n=int(input())
a = [int(i) for i in input().split()]# nhập trên 1 dòng
def out():
    for i in a : print(i,end=' ')
```

```

for i in range(n-1):
    k=i;temp=a[i]
    for j in range(i+1,n):
        if a[j] < temp:
            k=j;temp=a[j]
    a[i],a[k] = a[k],a[i]
    print("Buoc "+str(i+1)+":",end=' ')
    out()
    print()

```

SẮP XẾP CHÈN

Hãy thực hiện thuật toán sắp xếp chèn trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán.

Dữ liệu vào: Dòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100).

Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trống.

Ví dụ:

Input	Output
4 5 7 3 2	Buoc 0: 5 Buoc 1: 5 7 Buoc 2: 3 5 7 Buoc 3: 2 3 5 7

Giải :

Cách 1 : c++

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;cin>>n;
    int a[n+1];
    vector <int> res;
    for(int i=0;i<n;i++) cin>>a[i];
    for(int i=0;i<n;i++) {
        cout<<"Buoc "<<i<<" : ";
        res.push_back(a[i]);
        sort(res.begin(),res.end());
    }
}

```

```

        for(int i=0;i<res.size();i++) cout<<res[i]<<' ';
        cout<<endl;
    }
    return 0;
}

```

SẮP XẾP NỐI BỌT

Hãy thực hiện thuật toán sắp xếp nối bọt trên dãy N số nguyên. Ghi ra các bước thực hiện thuật toán.

Dữ liệu vào: Dòng 1 ghi số N (không quá 100). Dòng 2 ghi N số nguyên dương (không quá 100).

Kết quả: Ghi ra màn hình từng bước thực hiện thuật toán. Mỗi bước trên một dòng, các số trong dãy cách nhau đúng một khoảng trống.

Ví dụ:

Input	Output
4	Buoc 1: 3 2 5 7
5 3 2 7	Buoc 2: 2 3 5 7

Giải :

```

#include<bits/stdc++.h>
using namespace std;
void out(int *a,int n){
    for(int i=0;i<n;i++) cout<<a[i]<<' ';
    cout<<endl;
}
int main(){
    int n;cin>>n;
    int a[n+1];
    for(int i=0;i<n;i++) cin>>a[i];
    for(int i=0;i<n-1;i++) {
        int ok=0;
        for(int j=1;j<n;j++){
            if(a[j]<a[j-1]){
                ok=1;swap(a[j],a[j-1]);
            }
        }
        if(ok==1) cout<<"Buoc "<<i+1<<": ",out(a,n);
    }
}

```

```

        else break;
    }
    return 0;
}

```

SẮP XẾP XEN KẾ

Cho mảng A[] gồm n số nguyên khác nhau. Hãy đưa ra các phần tử của mảng theo khuôn dạng lớn nhất, nhỏ nhất, lớn thứ hai, nhỏ thứ 2, ... Ví dụ với A[] = {9, 7, 12, 8, 6, 5} ta đưa ra : 12, 5, 9, 6, 8, 7.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên là số phần tử của mảng n; dòng tiếp theo là n số A [i] của mảng A []; các số được viết cách nhau một vài khoảng trắng.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 7 7 1 2 3 4 5 6 8 1 6 9 4 3 7 8 2	7 1 6 2 5 3 4 9 1 8 2 7 3 6 4

Giải :

```

#include<bits/stdc++.h>
using namespace std;
int n,a[10000];
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++)cin>>a[i];
    sort(a,a+n,greater<int>());
    int l=0,r=n-1;
    while(l<r){
        cout<<a[l]<<' '<<a[r]<<' ';
        l++;r--;
    }
    if(l==r) cout<<a[l];
    cout<<endl;
}

```

```

}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

SẮP XẾP THEO GIÁ TRỊ TUYỆT ĐỐI

Ôn lại

Cho mảng A[] gồm n phần tử và số X. Hãy đưa sắp xếp các phần tử của mảng theo trị tuyệt đối của $|X - A[i]|$. Ví dụ với $A[] = \{10, 5, 3, 9, 2\}$ và $X = 7$ ta đưa ra mảng được sắp xếp theo nguyên tắc kể trên: $A[] = \{5, 9, 10, 3, 2\}$ vì $|7-10|=3$, $|7-5|=2$, $|7-3|=4$, $|7-9|=2$, $|7-2|=5$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên là số phần tử của mảng n và X; dòng tiếp theo là n số A [i] của mảng A []; các số được viết cách nhau một vài khoảng trắng.
- T, n, X thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, X, A[i] \leq 10^5$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 5 7 10 5 3 9 2 5 6 1 2 3 4 5	5 9 10 3 2 5 4 3 2 1

Giải :

```

// chú ý là sắp xếp theo nguyên tắc trên phải bảo toàn thứ tự
// vd test kia thì 5 và 9 đều có trị tuyệt đối hiệu với 7 = 2 , nhưng trong mảng a
// số 5 đứng trước 9 => kết quả thì 5 vẫn phải trước 9 => ý tưởng là cho n lần xét
// mỗi 1 lần xét thì lại xét 1 vòng lặp kiểm tra (n-1) cặp (a[j],a[j+1]);
#include<bits/stdc++.h>
using namespace std;
int n,x,a[100005];
void initwsolve(){
    cin>>n>>x;
    for(int i=0;i<n;i++)cin>>a[i];
}

```

```

for(int i=0;i<n;i++){
    for(int j=0;j<n-1;j++){
        if(abs(x-a[j+1])<abs(x-a[j]))swap(a[j],a[j+1]);
    }
}
for(int i=0;i<n;i++)cout<<a[i]<<' ';
cout<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

HỢP VÀ GIAO CỦA HAI DÃY SỐ 1

Cho mảng A[] gồm n phần tử, mảng B[] gồm m phần tử khác nhau. Các phần tử của mảng A[] và B[] đã được sắp xếp. Hãy tìm mảng hợp và giao được sắp giữa A[] và B[]. Ví dụ với A[] = {1, 3, 4, 5, 7}, B[] = {2, 3, 5, 6} ta có mảng hợp Union = {1, 2, 3, 4, 5, 6, 7}, mảng giao Intersection = {3, 5}. In ra đáp án theo giá trị phần tử từ nhỏ đến lớn.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m là số phần tử của mảng A[] và B[]; dòng tiếp theo là n số A[i] của mảng A[]; dòng tiếp theo là m số B[i] của mảng B[]; các số được viết cách nhau một vài khoảng trắng.
- T, n, m, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, m, A[i], B[i] \leq 10^5$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
1 5 3 1 2 3 4 5 1 2 3	1 2 3 4 5 1 2 3

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve()
```

```

int n,m;
cin>>n>>m;
int a[100005],b[100005],dp1[100005],dp2[100005];
for(int i=0;i<=100000;i++) {dp1[i]= 0;dp2[i]=0; }
for(int i=0;i<n;i++){
    cin>>a[i]; dp1[a[i]]=1;
}
for(int i=0;i<m;i++){
    cin>>b[i]; dp2[b[i]]=1;
}
for(int i=0;i<=100000;i++)
    if(dp1[i]==1||dp2[i]==1) cout<<i<<' ';
cout<<endl;
for(int i=0;i<=100000;i++)
    if(dp1[i]==1&&dp2[i]==1) cout<<i<<' ';
cout<<endl;
}
int main() {
    int t;
    cin>>t;
    while(t--) initwsolve();
return 0;
}

```

HỢP VÀ GIAO CỦA HAI DÃY SỐ 2

Cho mảng A[] gồm n phần tử, mảng B[] gồm m phần tử khác nhau. Các phần tử của mảng A[] và B[] chưa được sắp xếp. Hãy tìm mảng hợp và giao được sắp giữa A[] và B[]. Ví dụ với A[] = {7, 1, 5, 2, 3, 6}, B[] = {3, 8, 6, 20, 7} ta có mảng hợp Union = {1, 2, 3, 5, 6, 7, 8, 20}, mảng giao Intersection = {3, 6}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m là số phần tử của mảng A[] và B[]; dòng tiếp theo là n số A[i] của mảng A[]; dòng tiếp theo là m số B[i] của mảng B[]; các số được viết cách nhau một vài khoảng trắng.
- T, n, m, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, m, A[i], B[i] \leq 10^5$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:

Output:

1	
6 5	1 2 3 5 6 7 8 20
7 1 5 2 3 6	2 6
3 8 6 20 7	

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,m;
    cin>>n>>m;
    int a[100005],b[100005],dp1[100005],dp2[100005];
    for(int i=0;i<=100000;i++) {dp1[i]= 0;dp2[i]=0; }
    for(int i=0;i<n;i++){
        cin>>a[i]; dp1[a[i]]=1;
    }
    for(int i=0;i<m;i++){
        cin>>b[i]; dp2[b[i]]=1;
    }
    for(int i=0;i<=100000;i++)
        if(dp1[i]==1||dp2[i]==1) cout<<i<<' ';
    cout<<endl;
    for(int i=0;i<=100000;i++)
        if(dp1[i]==1&&dp2[i]==1) cout<<i<<' ';
    cout<<endl;
}
int main() {
    int t;
    cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

SẮP XẾP [0 1 2]

Bài làm tốt nhất

Cho mảng A[] gồm n phần tử. Các phần tử của mảng A[] chỉ bao gồm các số 0, 1, 2. Hãy sắp xếp mảng A[] theo thứ tự tăng dần. Ví dụ với A[] = {0, 2, 1, 2, 0} ta kết quả A[] = {0, 0, 1, 2, 2}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n là số phần tử của mảng A[]; dòng tiếp theo là n số A[i] của mảng A []các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $0 \leq A[i] \leq 2$; $1 \leq n \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2	
5	0 0 1 2 2
0 2 1 2 0	0 1
3	
0 1 0	

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;/**/cin>>n;
    int a[n];
    for(int i=0;i<n;i++) /**/cin>>a[i];
    sort(a,a+n);
    for(int i=0;i<n;i++) cout<<a[i]<<' ';
    cout<<endl;
}
int main() {
    int t;/**/cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

SẮP XẾP DÃY CON LIÊN TỤC

Cho mảng A[] gồm n phần tử. Hãy tìm dãy con liên tục của mảng A[R], .., A[L] sao cho khi sắp xếp lại dãy con ta nhận được một mảng được sắp xếp. Ví dụ với $A[] = \{10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60\}$ ta chỉ cần sắp xếp lại dãy con từ $A[4],.., A[9]: \{30, 25, 40, 32, 31, 35\}$ để có mảng được sắp.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n là số phần tử của mảng A[]; dòng tiếp theo là n số A [i] của mảng A []các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 10^6$; $0 \leq A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 11 10 12 20 30 25 40 32 31 35 50 60 9 0 1 15 25 6 7 30 40 50	4 9 3 6

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;
    cin>>n;
    int a[n],b[n];
    for(int i=0;i<n;i++){
        cin>>a[i]; b[i]=a[i];
    }
    sort(b,b+n);
    for(int i=0;i<n;i++)
        if(a[i]!=b[i]){
            cout<<i+1<<' ';break;
        }
    for(int i=n-1;i>=0;i--)
        if(a[i]!=b[i]){
            cout<<i+1<<' ';break;
        }
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

CẶP SỐ TỔNG BẰNG K

Cho mảng A[] gồm n phần tử và số k. Đếm tất cả các cặp phần tử của mảng có tổng bằng k. Ví dụ A[] = {1, 5, 3, 4, 2}, k = 7 ta có kết quả là 2 cặp (3, 4), (5, 2).

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n là số phần tử của mảng A[] và k; dòng tiếp theo là n số A[i] của mảng A[] các số được viết cách nhau một vài khoảng trắng.
- T, n, k, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 100$; $0 \leq k \leq 100$, $0 \leq A[i] \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 5 9 1 5 4 1 2 3 2 1 1 1	1 3

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k;/**/cin>>n>>k;
    int a[n];
    int count=0;
    for(int i=0;i<n;i++)/**/cin>>a[i];
    for(int i=0;i<n-1;i++)
        for(int j=i+1;j<n;j++)
            if (a[j]==k-a[i]) count++;
    cout<<count<<endl;
}
int main() {
    int t;/**/cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

SẮP XẾP CHỮ SỐ

Cho mảng A[] gồm n phần tử. Nhiệm vụ của bạn là đưa ra mảng đã được sắp xếp bao gồm các chữ số của mỗi phần tử trong A[]. Ví dụ A[] = {110, 111, 112, 113, 114 }ta có kết quả là {0, 1, 2, 3, 4}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n là số phần tử của mảng A[]; dòng tiếp theo là n số A[i] ; các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 10^7$; $0 \leq A[i] \leq 10^{16}$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 3 131 11 48 4 111 222 333 446	1 3 4 8 1 2 3 4 6

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    string s="",temp="";
    for(int i=0;i<n;i++){
        cin>>temp;s+=temp;
    }
    sort(s.begin(),s.end());
    for(int i=0;i<s.length();i++)
        if(s[i]!=s[i+1]) cout<<s[i]<<' ';
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TỔNG GẦN 0 NHẤT

Cho mảng A[] gồm n phần tử, hãy tìm cặp phần tử có tổng gần nhất so với 0.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào n là số phần tử của mảng A[]; dòng tiếp theo đưa vào n số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq N \leq 10^3$, $-10^6 \leq A[i] \leq 10^6$.

Output:

- Đưa ra tổng gần nhất với 0 của cặp phần tử.

Input:	Output:
2 3 -8 -66 -60 6 -21 -67 -37 -18 4 -65	-68 -14

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    int a[n];
    for(int i=0;i<n;i++)cin>>a[i];
    int res=1e9+7;
    for(int i=0;i<n-1;i++){
        for(int j=i+1;j<n;j++){
            if(abs(a[j]+a[i])<abs(res))res=a[j]+a[i];
        }
    }
    cout<<res<<endl;
}
```

```
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

PHẦN TỬ LỚN NHẤT

Cho mảng A[] gồm n phần tử, hãy tìm k phần tử lớn nhất của mảng. Các phần tử được đưa ra theo thứ tự giảm dần.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N và K; dòng tiếp theo đưa vào n số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, N, K, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq K < N \leq 10^3$, $1 \leq A[i] \leq 10^6$.

Output:

- Đưa ra k phần tử lớn nhất trên một dòng.

Input:	Output:
2 5 3 10 7 9 12 6 6 2 9 7 12 8 6 5	12 10 9 12 9

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k;/**/cin>>n>>k;
    int a[n];
    for(int i=0;i<n;i++)/**/ cin>>a[i];
        sort(a,a+n,greater<int>());
        for(int i=0;i<k;i++)
            cout<<a[i]<<' ';
        cout<<endl;
}
int main(){
    int t; /**/cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

SỐ LẦN XUẤT HIỆN

Cho mảng A[] gồm n phần tử đã được sắp xếp. Hãy tìm số lần xuất hiện số X trong mảng. Nếu số lần xuất hiện số x trong mảng là 0 hãy đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào N và X; dòng tiếp theo đưa vào n số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, N, X, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$, $1 \leq A[i], X \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 7 2 1 1 2 2 2 2 3 7 4 1 1 2 2 2 2 3	4 -1

Giải :

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,k; cin>>n>>k;
    int res=0,a[n];
    for(int i=0;i<n;i++)cin>>a[i];
    for(int i=0;i<n;i++){
        if(a[i]==k)res++;
    }
    if(res>0)cout<<res<<endl;
    else cout<<-1<<endl;
}
int main(){
    int t; cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

TỔNG CẶP SỐ NGUYÊN TỐ

Cho số tự nhiên N. Hãy tìm cặp số nguyên tố đầu tiên có tổng là N. Nếu không tồn tại cặp số nguyên tố có tổng bằng N, hãy đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm là một số N được ghi trên một dòng.
- T, N thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2	2 2
4	2 5
8	

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int const N=1e6+5;
int prime[N];
void initwsolve(){
    int n;cin>>n;
    for(int i=2;i<n;i++){
        if(prime[i]==0&&prime[n-i]==0){
            cout<<i<<' '<<n-i<<endl;return;
        }
    }
    cout<<-1<<endl;
}
int main(){
    int t; cin>>t;
    prime[0]=1;prime[1]=1;
    for(int i=2;i<=sqrt(N);i++){
        if(prime[i]==0)
            for(int j=i*i;j<N;j+=i)
                prime[j]=1;
    }
    while(t--)initwsolve();
    return 0;
}
```

MERGE SORT

Cho mảng A[] gồm N phần tử chưa được sắp xếp. Nhiệm vụ của bạn là sắp xếp các phần tử của mảng A[] theo thứ tự tăng dần bằng thuật toán Merge Sort.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào số N tương ứng với số phần tử của mảng A[]; phần thứ 2 là N số của mảng A[]; các số được viết cách nhau một vài khoảng trắng.
- T, N, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10^6$.

Output:

- Đưa ra kết quả các test theo từng dòng.

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

Giai:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    int a[n];
    for(int i=0;i<n;i++)cin>>a[i];
    sort(a,a+n);
    for(int i=0;i<n;i++)cout<<a[i]<<' ';
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TÍCH LỚN NHẤT - NHỎ NHẤT

Cho mảng A[] gồm n phần tử và mảng B[] gồm m phần tử. Nhiệm vụ của bạn là tìm tích giữa phần tử lớn nhất của mảng A[] và phần tử nhỏ nhất của mảng B[]. Ví

đụ A[] = {5, 7, 112, 9, 3, 6, 2 }, B[] = {1, 2, 6, -1, 0, 9} ta có kết quả là $-9 = 9*(-1)$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m tương ứng với số phần tử của mảng A[] và B[]; dòng tiếp theo là n số A[i] ; dòng cuối cùng là m số B[i]; các số được viết cách nhau một vài khoảng trắng.
- T, n, m, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, m \leq 10^6$; $-10^8 \leq A[i] \leq 10^8$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 6 6 5 7 9 3 6 2 1 2 6 -1 0 9 6 6 1 4 2 3 10 2 4 2 6 5 2 9	-9 20

Giải:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,m;cin>>n>>m;
    int a[n],b[m];
    for(int i=0;i<n;i++)cin>>a[i];
    for(int i=0;i<m;i++)cin>>b[i];
    sort(a,a+n);
    sort(b,b+m);
    cout<<(long long) a[n-1]*b[0]<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

TRỘN HAI DÂY

Cho mảng A[] gồm n phần tử và mảng B[] gồm m phần tử. Nhiệm vụ của bạn là hợp nhất hai mảng A[] và B[] để được một mảng mới đã được sắp xếp. Ví dụ A[] = {5, 7, 112, 9, 3, 6, 2}, B[] = {1, 2, 6, -1, 0, 9} ta có kết quả là C[] = {-1, 1, 0, 2, 3, 5, 6, 6, 7, .

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm ba dòng: dòng đầu tiên đưa vào n, m tương ứng với số phần tử của mảng A[] và B[]; dòng tiếp theo là n số A[i] ; dòng cuối cùng là m số B[i]; các số được viết cách nhau một vài khoảng trắng.
- T, n, m, A[i], B[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, m \leq 10^6$; $-10^8 \leq A[i] \leq 10^8$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
1 3 3 10 5 15 20 3 2	2 3 5 10 15 20

Giai:

```
#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,m;cin>>n>>m;
    int a[n+m];
    for(int i=0;i<n+m;i++)cin>>a[i];
    sort(a,a+n+m);
    for(int i=0;i<n+m;i++)
        cout<<a[i]<<' ';
    cout<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}
```

BỘ SƯU PHẦN TỬ

Cho mảng A[] gồm n số nguyên dương. Gọi L, R là max và min các phần tử của A[]. Nhiệm vụ của bạn là tìm số phần tử cần thiết cần thêm vào mảng để mảng có đầy đủ các số trong khoảng [L, R]. Ví dụ A[] = {5, 7, 9, 3, 6, 2} ta nhận được kết quả là 2 tương ứng với các số còn thiếu là 4, 8.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n, tương ứng với số phần tử của mảng A[]; dòng tiếp theo là n số A[i] ; các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, A[i] \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 5 4 5 3 8 6 3 2 1 3	1 0

Giai:

#note : count để đếm xem từ l-r có bao nhiêu số khác nhau

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    int n; cin >> n;
```

```
    int a[n], count = 1;
```

```
    for (int i = 0; i < n; i++) cin >> a[i];
```

```
    sort(a, a + n);
```

```
    for (int i = 1; i < n; i++)
```

```
        if (a[i] != a[i - 1]) count++;
```

```
    cout << a[n - 1] - a[0] - count + 1 << endl;
```

```
}
```

```
int main() {
```

```
    int t; cin >> t;
```

```
    while (t--) initwsolve();
```

```
    return 0;
```

```
}
```

SẮP XẾP THEO SỐ LẦN XUẤT HIỆN

Cho mảng A[] gồm n số nguyên. Nhiệm vụ của bạn là sắp xếp mảng theo số lần xuất hiện các phần tử của mảng. Số xuất hiện nhiều lần nhất đứng trước. Nếu hai phần tử có số lần xuất hiện như nhau, số nhỏ hơn đứng trước. Ví dụ A[] = {5, 5, 4, 6, 4}, ta nhận được kết quả là A[] = {4, 4, 5, 5, 6}.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên đưa vào n, tương ứng với số phần tử của mảng A[] và số k; dòng tiếp theo là n số A[i] ; các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 10^4$; $1 \leq k \leq 10^3$; $1 \leq A[i] \leq 10^5$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 5 5 5 4 6 4 5 9 9 9 2 5	4 4 5 5 6 9 9 9 2 5

Giai:

```
#include <bits/stdc++.h>
using namespace std;
int f[100005];
bool cmp(int a,int b){
    if(f[a]==f[b])return a<b;
    else return f[a]>f[b];
}
void initwsolve(){
    int n;cin>>n;
    int a[n];
    memset(f,0,sizeof(f));
    for(int i=0;i<n;i++){
        cin>>a[i];f[a[i]]++;
    }
    sort(a,a+n,cmp);
    for(int i=0;i<n;i++)cout<<a[i]<<' ';
    cout<<endl;
```

```

}
int main() {
    int t; cin >> t;
    while(t--) initwsolve();
    return 0;
}

```

TÌM KIÊM

Cho mảng A[] gồm n phần tử đã được sắp xếp. Hãy đưa ra 1 nếu X có mặt trong mảng A[], ngược lại đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào n, X là số các phần tử của mảng A[] và số X cần tìm; dòng tiếp theo đưa vào n số A[i] ($1 \leq i \leq n$) các số được viết cách nhau một vài khoảng trắng.
- T, n, A, X thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N$, $X, A[i] \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 5 16 2 4 7 9 16 7 98 1 22 37 47 54 88 96	1 -1

Giai:

```

#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,x; cin >> n >> x;
    int a[n];
    for(int i=0;i<n;i++) cin >> a[i];
    sort(a,a+n);
    for(int i=0;i<n;i++)
        if(a[i]==x){ cout << 1 << endl; return; }
    cout << -1 << endl;
}
int main() {

```

```

int t;cin>>t;
while(t--) initwsolve();
return 0;
}

```

TÌM KIẾM TRONG DÃY SẮP XẾP VÒNG

Một mảng được sắp dược chia thành hai đoạn tăng dần được gọi là mảng sắp xếp vòng. Ví dụ mảng $A[] = \{ 5, 6, 7, 8, 9, 10, 1, 2, 3, 4 \}$ là mảng sắp xếp vòng. Cho mảng $A[]$ gồm n phần tử, hãy tìm vị trí của phần tử x trong mảng $A[]$ với thời gian $\log(n)$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T .
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào n và x ; dòng tiếp theo đưa vào n số $A[i]$; các số được viết cách nhau một vài khoảng trắng.
- $T, n, A[i], x$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, x, A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

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

Giai:

```

#include <bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n,x;cin>>n>>x;
    int a[n];
    for(int i=0;i<n;i++)cin>>a[i];
    for(int i=0;i<n;i++)
        if(a[i]==x){cout<<i+1<<endl;return;}
    cout<<-1<<endl;
}
int main() {
    int t;cin>>t;
    while(t--) initwsolve();
}

```

```
return 0;  
}
```

SỐ NHỎ NHẤT VÀ NHỎ THỨ HAI

Cho mảng A[] gồm n phần tử, hãy đưa ra số nhỏ nhất và số nhỏ thứ hai của mảng. Nếu không có số nhỏ thứ hai, hãy đưa ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào n là số phần tử của mảng A[]; dòng tiếp theo đưa vào n số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N, A[i] \leq 10^7$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

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

Giai:

```
#include <bits/stdc++.h>  
using namespace std;  
void initwsolve(){  
    int n;cin>>n;  
    int a[n],res=-1;  
    for(int i=0;i<n;i++)cin>>a[i];  
    sort(a,a+n);  
    for(int i=1;i<n;i++){  
        if(a[i]!=a[i-1]) {res=a[i];break;}  
        if(res== -1)cout<<-1<<endl;  
        else cout<<a[0]<<' '<<res<<endl;  
    }  
    int main() {  
        int t;cin>>t;  
        while(t--) initwsolve();  
        return 0;  
    }
```

XÓA DỮ LIỆU TRONG DSLK ĐƠN

Cho danh sách liên kết đơn lưu giữ các số nguyên được quản lý bởi con trỏ First. Viết chương trình con xóa tất cả các phần tử có giá trị bằng x trong danh sách liên kết đơn; chương trình con trả về số lượng các phần tử đã xóa. Sau khi xóa xong, liệt kê các phần tử còn lại trong danh sách liên kết đơn First. Ví dụ: Ta có Input sau:

14 : là số lượng phần tử trong danh sách
1 1 1 4 5 1 1 1 1 7 1 8 1 9 : là 14 phần tử
1 : là số cần xóa
Output : 4 5 7 8 9

Giải:

```
#include<bits/stdc++.h>
using namespace std;
vector<string> res;
main(){
    int n;cin>>n;
    for(int i=1;i<=n;i++){
        string s;cin>>s;
        res.push_back(s);
    }
    string k;cin>>k;
    for(int i=0;i<res.size();i++)
        if(res[i]!=k) cout<<res[i]<<" ";
}
```

Cách 2:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> res;
main(){
    int n;cin>>n;
    for(int i=1;i<=n;i++){
        int s;cin>>s;
        res.push_back(s);
    }
    int k; cin>>k;
    for(int i=0;i<res.size();i++)
        if(res[i]!=k) cout<<res[i]<<" ";
```

}

LỌC DỮ LIỆU TRÙNG TRONG DSLK ĐƠN

Cho danh sách liên kết đơn lưu giữ các số nguyên được quản lý bởi con trỏ First. Viết chương trình con lọc tất cả các phần tử có giá trị trùng nhau trong danh sách liên kết đơn First, chỉ để lại 1 phần tử đại diện cho nhóm trùng. Sau khi lọc xong, liệt kê các phần tử trong danh sách liên kết đơn First.

Ví dụ: Ta có Input :

12 : là số lượng phần tử trong danh sách

1 1 1 4 5 1 4 7 7 8 1 9 : là số phần tử

Output : 1 4 5 7 8 9

Giải:

```
#include<bits/stdc++.h>
using namespace std;
vector<int> res;
main(){
    int n;cin>>n;
    int f[100005];
    memset(f,0,sizeof(f));
    for(int i=1;i<=n;i++){
        int n;cin>>n;
        res.push_back(n);
        f[n]++;
    }
    for(int i=0;i<res.size();i++)
        if(f[res[i]]>0) {
            cout<<res[i]<<" ";
            f[res[i]]=0;
        }
}
```

GIAO CỦA BA DÃY SỐ

Cho ba dãy số A[], B[], C[] gồm N1, N2, N3 phần tử đã được sắp xếp. Hãy đưa ra các phần tử có mặt trong cả ba dãy theo thứ tự tăng dần. Nếu không có đáp án, in ra -1.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm bốn dòng: dòng thứ nhất đưa vào N1, N2, N3 là số phần tử của mảng A[], B[], C[]; các dòng tiếp theo đưa vào 3 dãy A[], B[], C[].
 - Ràng buộc: $1 \leq T \leq 100$; $1 \leq N1, N2, N3 \leq 10^6$, $0 \leq A[i], B[j], C[k] \leq 10^{18}$.
 - Output:
 - Đưa ra kết quả mỗi test theo từng dòng.
- Ví dụ:

Input:	Output:
1 6 5 8 1 5 10 20 40 80 6 7 20 80 100 3 4 15 20 30 70 80 120	20 80

Giải :

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n1,n2,n3,i=0,j=0,k=0,check=0;
    cin>>n1>>n2>>n3;
    long long a[n1],b[n2],c[n3];
    for(int i=0;i<n1;i++) cin>>a[i];
    for(int i=0;i<n2;i++) cin>>b[i];
    for(int i=0;i<n3;i++) cin>>c[i];
    while(i<n1 &&j<n2 &&k<n3){
        if(a[i]==b[j]&&b[j]==c[k]){
            cout<<a[i]<<' '; check=1;
            i++;j++;k++;
        }
        else if(a[i]<b[j]) i++;
        else if(b[j]<c[k]) j++;
        else k++;
    }
    if(check==0) cout<<-1;
    cout<<endl;
}
int main (){
```

```

int t;/**/cin>>t;
while(t--) initwsolve ();
return 0;
}

```

SẮP XẾP CHẴN LẺ

Cho dãy số $a[]$ có n phần tử, đánh số từ 1 đến n . Hãy sắp xếp các phần tử ở vị trí lẻ theo thứ tự tăng dần, các phần tử ở vị trí chẵn theo thứ tự giảm dần.

Input

Dòng đầu tiên ghi số n , không quá 10^5

Dòng thứ 2 ghi n số của dãy $a[]$ ($a \leq a[i] \leq 10^9$)

Output

Ghi ra dãy số kết quả trên một dòng

Ví dụ

Input	Output
4	1 4 3 2
1 2 3 4	

Giai:

```

#include<bits/stdc++.h>
using namespace std;
int a[100005];
vector<int>in;
vector<int>de;
int main(){
    int n;cin>>n;
    for(int i=0;i<n;i++){
        cin>>a[i];
        if(i%2==0)in.push_back(a[i]);
        else de.push_back(a[i]);
    }
    sort(in.begin(),in.end());
    sort(de.begin(),de.end(),greater<int>());
    if(in.size()<de.size()){
        for(int i=0;i<in.size();i++)
            cout<<in[i]<<' '<<de[i]<<' ';
        cout<<de.size()-1;
    }
}

```

```

else if(in.size()>de.size()){
    for(int i=0;i<de.size();i++)
        cout<<in[i]<<' '<<de[i]<<' ';
    cout<<in[in.size()-1];
}
else {
    for(int i=0;i<de.size();i++)
        cout<<in[i]<<' '<<de[i]<<' ';
}
return 0;
}

```

ĐỔI CHỖ ÍT NHẤT

Ôn lại

Cho mảng A[] gồm n phần tử. Hãy tìm số phép đổi chỗ ít nhất giữa các phần tử của mảng để mảng A[] được sắp xếp. Ví dụ với A[] = {4, 3, 2, 1} ta cần thực hiện ít nhất 2 phép đổi chỗ: Swap(A[0], A[3]), Swap(A[1], A[2]).

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test gồm hai dòng: dòng đầu tiên là số phần tử của mảng n và X; dòng tiếp theo là n số A [i] của mảng A []; các số được viết cách nhau một vài khoảng trắng.
- T, n thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n \leq 10$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Input:	Output:
2 4 4 3 2 1 5 1 5 4 3 2	2

Giải:

```

#include<bits/stdc++.h>
#define MAX 10005
using namespace std;
int n;
int a[MAX], b[MAX], check[MAX];
void xuli(){
    cin>>n;
    for(int i=0;i<n;i++){

```

```

    cin>>a[i]; b[i]=a[i];
}
sort(b,b+n);
int res=0;
for(int i=0;i<n;i++)
    if(a[i]!=b[i])
        for(int j=i+1;j<n;j++)
            if(a[j]==b[i]){
                res++;
                swap(a[i],a[j]);
                break;
            }
    cout<<res<<endl;;
}
int main(){
    int t;cin>>t;
    while(t--) xuli();
}

```

Ngăn xếp

note

luôn đưa s.size() vào trong if(...) hoặc while(...)=> vì nếu sau if có s.pop sẽ có thể ko chạy được nếu s rỗng

2 hàm in stack hay sử dụng đến (chú ý :

1:stack thật được xếp từ phải sang trái. Đây in ngược cho đúng theo mảng, chuỗi nhập vào

2: đưa stack vào mới không ảnh hưởng toàn cục khi s.pop(). Chỉ có tác dụng in khi gọi hàm)

Mảng

//đưa stack vào mới không ảnh hưởng toàn cục khi s.pop()

void out(stack<int> s){

vector<int> a;

while(!s.empty()){

a.push_back(s.top());

s.pop();

}

for(int i=a.size()-1;i>=0;i--)cout<<a[i]<<' ';

```

    cout<<endl;
}
Chuỗi:
void out(stack<char> s){
    vector <char> a;
    while(!s.empty()){
        a.push_back(s.top());
        s.pop();
    }
    for(int i=a.size()-1;i>=0;i--)cout<<a[i]<<' ';
    cout<<endl;
}

```

NGĂN XẾP 1

Bài làm tốt nhất

Cho một ngăn xếp các số nguyên. Các thao tác gồm 3 lệnh: push, pop và show. Trong đó thao tác push kèm theo một giá trị cần thêm (không quá 1000). Hãy viết chương trình ghi ra kết quả của các lệnh show.

Input: Gồm nhiều dòng, mỗi dòng chứa một lệnh push, pop hoặc show. Input đảm bảo số lượng phần tử trong stack khi nhiều nhất cũng không vượt quá 200.

Output: Ghi ra màn hình các phần tử đang có trong stack theo thứ tự lưu trữ mỗi khi gặp lệnh show. Các số viết cách nhau đúng một khoảng trắng. Nếu trong stack không còn gì thì in ra dòng “empty”

Ví dụ:

Input	Output
push 3	
push 5	
show	3 5
push 7	3 5 7
show	3
pop	
pop	
show	

Giải:

Cách 1:

//tạo 1 stack để thực hiện 1 trong 3 thao tác push, pop và show.

//2: vì stack s đã khai báo toàn cục=> nếu không muốn đưa tham số vào hàm show để kết quả s không đổi thì phải tạo 1 stack khác (s1=s)

//3 : chú ý khi show thì lại in kết quả ngược lại stack => tạo 1 vector để lưu kết quả sau mỗi lần s.pop();

```
#include<bits/stdc++.h>
using namespace std;
stack <int> s;
string str;
int n;
void show(){
    stack <int> s1=s;//2
    vector <int> a;//3
    while(!s1.empty()){
        a.push_back(s1.top());
        s1.pop();
    }
    for(int i=a.size()-1;i>=0;i--)cout<<a[i]<<' '//3
    cout<<endl;
}
int main(){
    while (1){
        cin>>str;
        /*1*/if(str=="push"){cin>>n;s.push(n);}
        else if(str=="pop")s.pop();
        else if(str=="show") show();
        if(s.size()==0){cout<<"empty";break;}
    }
    return 0;
}
```

Cách 2 : không cần tạo stack khác => đưa s vào hàm:

```
#include<bits/stdc++.h>
using namespace std;
stack <int> s;
string str;
int n;
void show(stack<int> s){//đưa stack vào mới không ảnh hưởng toàn cục khi
    s.pop()
    vector <int> a;
```

```

while(!s.empty()){
    a.push_back(s.top());
    s.pop();
}
for(int i=a.size()-1;i>=0;i--)cout<<a[i]<<' ';
cout<<endl;
}
int main(){
    while (1){
        cin>>str;
        if(str=="push"){cin>>n;s.push(n);}
        else if(str=="pop")s.pop();
        else if(str=="show") show(s);
        if(s.size()==0){cout<<"empty";break;}
    }
    return 0;
}

```

NGĂN XẾP 2

Yêu cầu bạn xây dựng một stack với các truy vấn sau đây:

“PUSH x”: Thêm phần tử x vào stack ($0 \leq x \leq 1000$).

“PRINT”: In ra phần tử đầu tiên của stack. Nếu stack rỗng, in ra “NONE”.

“POP”: Xóa phần tử đầu tiên của stack. Nếu stack rỗng, không làm gì cả.

Input:

Dòng đầu tiên là số lượng truy vấn Q ($Q \leq 100000$).

Mỗi truy vấn có dạng như trên.

Output:

Với mỗi truy vấn “PRINT”, hãy in ra phần tử đầu tiên của stack. Nếu stack rỗng, in ra “NONE”.

Ví dụ:

Input:	Output
9	
PUSH 1	
PUSH 2	
POP	1
PRINT	3
PUSH 3	NONE
PRINT	

POP	
POP	
PRINT	

Giải:

//1: tạo 1 stack để thực hiện 1 trong 3 thao tác push, pop và print.
//3 : chú ý khi show chỉ xét stack rỗng hay không và in phần tử đầu => không phải tạo stack mới.

```
#include<bits/stdc++.h>
using namespace std;
stack <int> s;//1
void show(){
    if(s.size()==0)cout<<"NONE"<<endl;
    else cout<<s.top()<<endl;
}
int main(){
    int t;cin>>t;
    while (t--){
        string str;cin>>str;
        /*1*/if(str=="PUSH"){int n;cin>>n;s.push(n);}
        else if(str=="POP"){if(s.size()!=0)s.pop();}
        else show();
    }
    return 0;
}
```

ĐẢO TỪ

Cho xâu ký tự S. Nhiệm vụ của bạn là đảo ngược các từ trong S. Ví dụ S = “I like this program very much”, ta nhận được kết quả là “much very program this like I”.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test là một xâu ký tự S.
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{Length}(S) \leq 10^3$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input:

2

I like this program very much
much very program this like I

Giải :

```
#include <bits/stdc++.h>
using namespace std;
void show(stack<string> s){
    while(s.size()){
        cout<<s.top()<<' ';
        s.pop();
    } cout<<endl;
}
int main(){
    string str;
    int t;cin>>t;
    //cin.ignore();
    while(t--){
        stack <string> s;
        while(1){
            cin>>str;
            s.push(str);
            if(getchar()=='\n')break;
        }show(s);
    }
}
```

Output:

much very program this like I

I like this program very much

DÃY NGOẶC ĐÚNG DÀI NHẤT

Ôn lại

Cho một xâu chỉ gồm các kí tự ‘(’ và ‘)’. Một dãy ngoặc đúng được định nghĩa như sau:

- Xâu rỗng là 1 dãy ngoặc đúng.
- Nếu A là 1 dãy ngoặc đúng thì (A) là 1 dãy ngoặc đúng.
- Nếu A và B là 2 dãy ngoặc đúng thì AB là 1 dãy ngoặc đúng.

Cho một xâu S. Nhiệm vụ của bạn là hãy tìm dãy ngoặc đúng dài nhất xuất hiện trong xâu đã cho.

Input: Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm một xâu S có độ dài không vượt quá 10^5 kí tự.

Output: Với mỗi test in ra một số nguyên là độ dài dãy ngoặc đúng dài nhất tìm được.

Ví dụ:

Input:	Output
3	2
((0	4
)00)	6
0(0))))	

Giải thích test 2:)()() (0 1 2 3 4 5)

Chuỗi:) () ())
Chỉ số: 0 1 2 3 4 5

Ngăn xếp

0	1	3	5
0	0	0	0

 ^ ||

K quả Res=0 Res=2 Res=4 ||
//1:nếu là '(' thì thêm vào hàng đợi để lưu trữ số các dấu '(' và vị trí của nó:
//nếu là ')' thì ta phải xem 2 th :
//2: th1 là tồn tại '(' ở phía trước => ta s.pop() và lấy i-s.top() sẽ ra kết quả(vì s.pop()
chính là dấu ')'

//3: th còn lại : stack rỗng (\Leftrightarrow không có dấu '(' chưa được sử dụng ở phía trước
nữa)=> ta đưa ')' vào stack

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int t;cin>>t;
    while (t--){
        string str;cin>>str;
        stack <int> s;
        s.push(-1);
        int res=0;
        for(int i=0;i<str.length();i++){
            if(str[i]==')' s.push(i); //1
            else{
```

```

    s.pop();
    if(s.size()!=0)res=max(res,i-s.top());//2
    else s.push(i);//3
}
}cout<<res<<endl;
}
return 0;
}

```

KIỂM TRA BIẾU THỨC SỐ HỌC

Ôn lại

Cho biểu thức số học, hãy cho biết biểu thức số học có dư thừa các cặp kí hiệu ‘(,)’ hay không?

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 20$.

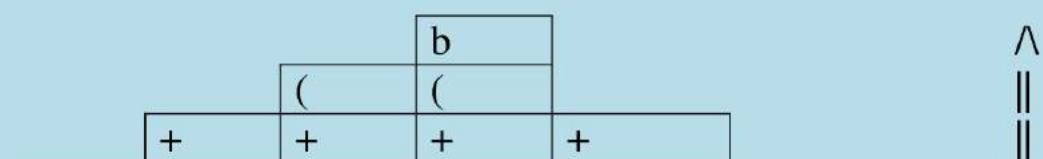
Ví dụ:

Input	Output
3	Yes
((a+b))	Yes
(a + (b)/c)	No
(a + b*(c-d))	

Giải thích :

(a + (b)/c)

Chuỗi: (a + (b) / c)
 Chỉ số 0 1 2 3 4 5 6 7 8
 :



||

	a	a	a	a	a					
stack	((((((
										Check=0

||

(a + b*(c-d))

Chuỗi: (0 a 1 + 2 b 3 * 4 (5 c 6 - 7 d 8) 9) 10

Chỉ số :
0 1 2 3 4 5 6 7 8 9 10

⋮

							d			
							-	-		
							c	c	c	
							(((
				*	*	*	*	*	*	*
				b	b	b	b	b	b	b
				+	+	+	+	+	+	+
				a	a	a	a	a	a	a
stack	((((((((((
										Check=1
										Check=1
										Check=1
										Rỗng

Giải :

//coi là thừa dấu khi không đủ cặp () hoặc đủ nhưng không có biểu thức giữa chúng . vd : (a)

// xét từ đầu chuỗi đến cuối xem tại một vị trí bất kì có lỗi nào không bằng cách :

//1 :nếu str[i] không phải là ')' thì đưa hết chúng vào stack

//2: khi là dấu '(' thì mới thực hiện kiểm tra:

//3:kiểm tra xóa một biểu thức có đầy đủ 1 cặp (và) . nếu có dấu +-*/\ thì biểu thức này đúng và xóa . nếu sai vd kiểu (a) thì thừa cặp => check =0 và in Yes luôn.

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    string str; cin.ignore();
```

```
    getline(cin,str);
```

```
    stack <char> s;
```

```
    for(int i=0;i<str.length();i++){
```

```
        if(str[i]!=')')s.push(str[i]);// 1
```

```
        else{//2
```

```
            bool check=0;
```

```

        while(s.size()!=0){//3
            char c=s.top(); s.pop();
            if(c=='(')break;
            if (c=='+'||c=='-'||c=='*'||c=='/') check=1;
        }
        if(check==0){cout<<"Yes\n"; return;}
    }
}
cout<<"No\n";
}

int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

ĐÉM SỐ DẤU NGOẶC ĐỔI CHIỀU

Ôn lại

Cho một xâu chỉ gồm các kí tự ‘(‘, ‘)’ và có độ dài chẵn. Hãy đếm số lượng dấu ngoặc cần phải đổi chiều ít nhất, sao cho xâu mới thu được là một dãy ngoặc đúng.

Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test gồm 1 xâu S có độ dài không vượt quá 100 000, chỉ gồm dấu (và).

Output:

Với mỗi test, in ra đáp án tìm được trên một dòng.

Ví dụ:

Input:	Output
4	
)())	2
((()	2
(((()	1
)()((3

Giải:

// 1: ý tưởng : cứ tại một vị trí gấp ')' ta sẽ xóa đi 1 cặp tương ứng ().=> nếu str[i]!=')' sẽ đưa vào chuỗi và => về lý thuyết cuối cùng lấy s.size()/2 là xong //2: nhưng trường hợp xâu đầu là)))...=> không s.pop() được sẽ âm => chỉ s.pop() khi s.size()!=0

```

//3=> khi s.size()==0 chuyển nó thành '('. khi đó đã chuyển ký tự thì count++; =>
kết quả cuối cùng :count+s.size()/2
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string str;
    cin>>str;
    stack <char> s;
    int count=0;
    for(int i=0;i<str.length();i++){
        if(str[i]!=')') s.push(str[i]);//1
        else {
            if(s.size()==0) { s.push('('); count++; } //3
            else s.pop(); //2
        }
    }
    cout<<count+s.size()/2<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

Ứng Dụng Stack - Biểu Thức Tiền Tố (Prefix)

Trong toán học thì các biểu thức thường được biểu diễn dưới dạng trung tố (các toán tử nằm giữa các toán hạng). Đối với máy tính thì khó sử dụng để tính toán, phải đưa biểu thức dạng trung tố về dạng tiền tố hoặc hậu tố.

Infix (trung tố)	Prefix(tiền tố)	Postfix(hậu tố)
$x+y$	$+xy$	$xy+$
$x+y-z$	$-+xyz$	$xy+z-$
$x+y^*z$	$+x^*yz$	xyz^*
$x+(y-z)$	$+x-yz$	$xyz-+$

Kinh nghiệm : tiền tố sang trung tố và hậu tố : duyệt ngược , đầy chữ và cứ gấp toán tử thì nhóm.

BIẾN ĐỔI TIỀN TỐ - TRUNG TỐ

Ôn lại

Hãy viết chương trình chuyển đổi biểu thức biểu diễn dưới dạng tiền tố về dạng trung tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10^6$.

Ví dụ:

Input	Output
2	$((A+B)*(C-D))$
$*+AB-CD$	$((A-(B/C))*((A/K)-L))$
$*-A/BC-/AKL$	

Giai thích :test 2:

$\begin{matrix} * & + & A & B & - & C & D \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{matrix}$

Note : mỗi một dấu cách là một chuỗi trong stack . s.top ở cuối mảng

6 D
5 DC
4 (C-D)
3 (C-D) B
2 (C-D) B A
1 (C-D) (A+B)
0 ((A+B)*(C-D))

Kết

qua: $((A+B)*(C-D))$

Giải:

//note: string(1,str[i]) : chuyển ký tự str[i] thành 1 chuỗi có 1 ký tự
//1: để in stack đúng biểu thức => duyệt ngược từ cuối về

```

//2:nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái
string đầu trong stack và cùng với toán tử để tạo thành biểu thức.
#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){//operator
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}
void initwsolve(){
    string str;cin>>str;
    stack <string> s;
    for(int i=str.length()-1;i>=0;i--){//1
        if(oprt(str[i])==0) s.push(string(1,str[i]));//2
        else{
            string str1=s.top();s.pop();
            string str2=s.top();s.pop();
            string temp="("+str1+ str[i] +str2+")";
            s.push(temp);
        }
    }
    cout<<s.top()<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

BIẾN ĐỔI TIỀN TỐ - HẬU TỐ

Ôn lại

Hãy viết chương trình chuyển đổi biểu thức biểu diễn dưới dạng tiền tố về dạng hậu tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10^6$.

Ví dụ:

Input	Output
2 *+AB-CD *-A/BC-/AKL	AB+CD-* ABC/-AK/L-*

Giải thích:

Test 2:

*	+	A	B	-	C	D
0	1	2	3	4	5	6

*+AB-CD
6 D
5 DC
4 CD-
3 CD- B
2 CD- BA
1 CD- AB+
0 AB+CD-*

kết

quả AB+CD-*

Giải:

```
//note: string(1,str[i]) : chuyển ký tự str[i] thành 1 chuỗi có 1 ký tự
//1: để in stack đúng biểu thức => duyệt ngược từ cuối về
//2: nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái
string đầu trong stack và cùng với toán tử để tạo thành biểu thức.
#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){//operator
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}
void initwsolve(){
    string str;cin>>str;
    stack <string> s;
    for(int i=str.length()-1;i>=0;i--){//1
```

```

        if(oprt(str[i])==0) s.push(string(1,str[i]));//2
    else{
        string str1=s.top();s.pop();
        string str2=s.top();s.pop();
        string temp=str1+str2+ str[i];
        s.push(temp);
    }
}
cout<<s.top()<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

BIẾN ĐỔI TRUNG TỐ - HẬU TỐ

Ôn lại

Hãy viết chương trình chuyển đổi biểu thức biểu diễn dưới dạng trung tố về dạng hậu tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10$.

Ví dụ:

Input	Output
2 (A+(B+C) ((A*B)+C)	ABC++ AB*C+

Giải thích test 1:

chỉ

số chuỗi stack res
0 ((

1	A	(A
2	+	(+	A
3	((+ (A
4	B	(+ (AB
		(+ (
5	+	+ (AB
		(+ (
6	C	+ (ABC
7)	(+	ABC+
kết			
quả			ABC++

Giải :

// 1:dùng 1 stack để lưu giữ các toán tử , và kết hợp hàm oprt để lưu trữ thứ tự xét toán tử và dùng 1 chuỗi res để lưu các chữ số A,B,C
 //2=> khi gặp chữ thì đưa vào res , khi gặp '(' thì đưa vào stack ,
 //3:khi gặp toán tử thì xét thứ tự ưu tiên và cuối cùng đưa toán tử vào một vị trí thích hợp
 //4:khi gặp ')' thì cập nhật toán tử từ stack sang res . Và cuối cùng xóa 1 dấu '(' trong stack.

```
#include<bits/stdc++.h>
using namespace std;
int oprt(char x){//operator//note int
    switch(x){
        case '^':return 0;
        case '%': case '*' :case '/' :return 1;
        case '+': case '-' :return 2;
        default:return 3;//dấu '('
    }
}
void initwsolve(){
    string str;cin>>str;
    string res="";
    stack <char> s;
    for(int i=0;i<str.size();i++){
        if('A'<=str[i]&&str[i]<='Z'||'a'<=str[i]&&str[i]<='z')res+=str[i];//1
        else {
            switch(str[i]){

```

```

/*2*/case '(:s.push(str[i]);break;
/*4*/case ')':
    while(s.top()!='&&s.size()){
        res+=s.top();s.pop();
    }s.pop();break;
/*3*/case '^':case '%': case '*' :case '/' :case '+': case '-':
    while(s.size()&&oprt(s.top())<=oprt(str[i])){
        res+=s.top();s.pop();
    }s.push(str[i]);break;
}
}
while(s.size()){
    if(s.top()!='(')res+=s.top();
    s.pop();
}cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

BIẾN ĐỔI HẬU TỐ - TIỀN TỐ

Hãy viết chương trình chuyển đổi biểu thức biểu diễn dưới dạng hậu tố về dạng tiền tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10^6$.

Ví dụ:

Input	Output
2	*+AB-CD

AB+CD-*	*-A/BC-/AKL
ABC/-AK/L-*	

GIẢI THÍCH TEST 2:

CHỈ SỐ	CHUỖI	STACK
0	A	A
1	B	A B
2	C	A B C
3	/	A /BC
4	-	-A/BC
5	A	-A/BC A
6	K	-A/BC A K
7	/	-A/BC /AK
8	L	-A/BC /AK L
9	-	-A/BC -/AKL
10	*	*-A/BC-/AKL
KQ		*-A/BC-/AKL

Giải:

//note: string(1,str[i]) : chuyển ký tự str[i] thành 1 chuỗi có 1 ký tự
 //1: để in stack đúng biểu thức (ta thấy có 2 chữ thì sau đó là 1 toán tử của 2 chữ đó
 => duyệt xuôi
 //2: nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái
 string đầu trong stack và cùng với toán tử để tạo thành biểu thức.
 //3: sau đó gộp biểu thức đó thành 1 string đưa vào stack . khi đó string lại coi như
 1 toán hạng lớn.

```
#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){//operator
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}
void initwsolve(){
    string str;cin>>str;
    stack <string> s;
    for(int i=0;i<str.length();i++){//1
        if(oprt(str[i])==0) s.push(string(1,str[i]));//2
        else{
            string str1=s.top();s.pop();
            s.push(str1+str[i]);
        }
    }
}
```

```

        string str2=s.top();s.pop();
        string temp=str[i]+str2+ str1;
        s.push(temp);//3
    }
}
cout<<s.top()<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}

```

BIẾN ĐỔI HẬU TỐ - TRUNG TỐ

Hãy viết chương trình chuyển đổi biểu thức biểu diễn dưới dạng hậu tố về dạng trung tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp.
- T, exp thỏa mảng ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10^6$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 10^6$.

Ví dụ:

Input	Output
2	
ABC++	(A+(B+C))
AB*C+	((A*B)+C)

Giải thích test 2:

CHỈ

SỐ CHUỖI STACK

0 A A

1 B A B

2 * (A*B)

3	C	$(A^*B)C$
4	+	$((A^*B)+C)$
KQ		$((A^*B)+C)$

Giải:

//note: string(1,str[i]) : chuyển ký tự str[i] thành 1 chuỗi có 1 ký tự
 //1: để in stack đúng biểu thức (ta thấy cứ 2 chữ thì sau đó là 1 toán tử của 2 chữ đó
 \Rightarrow duyệt xuôi

//2:nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái
 string đầu trong stack và cùng với toán tử để tạo thành biểu thức.

//3 :sau đó gộp biểu thức đó thành 1 string đưa vào stack . khi đó string lại coi như
 1 toán hạng lớn.

```
#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){//operator
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}
void initwsolve(){
    string str;cin>>str;
    stack <string> s;
    for(int i=0;i<str.length();i++){//1
        if(oprt(str[i])==0) s.push(string(1,str[i]));//2
        else{
            string str1=s.top();s.pop();
            string str2=s.top();s.pop();
            string temp='('+str2+ str[i]+str1+')';
            s.push(temp);//3
        }
    }
    cout<<s.top()<<endl;
}
int main(){
    int t;cin>>t;
    cin.ignore();
    while (t--)initwsolve();
    return 0;
}
```

TÍNH GIÁ TRỊ BIỂU THỨC HẬU TỐ

Hãy viết chương trình chuyển tính toán giá trị của biểu thức hậu tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức hậu tố exp. Các số xuất hiện trong biểu thức là các số đơn có 1 chữ số.

Output:

- Đưa ra kết quả mỗi test theo từng dòng, chỉ lấy giá trị phần nguyên.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 20$.

Ví dụ:

Input	Output
2	-4
231*+9-	34
875*+9-	

Giải:

//1: để tính đúng biểu thức => duyệt xuôi

//2: nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái số đầu trong stack và cùng với toán tử để tạo thành biểu thức.

//3 : sau đó đưa kết quả vừa có được là 1 số đưa vào stack . khi đó s.top() lại coi như 1 toán hạng lớn .

```
#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}
int cal(int a,int b,char x){
    if(x=='+')return a+b;  if(x=='-')return a-b;
    if(x=='*')return a*b;  if(x=='/')return a/b;
}
void initwsolve(){
    string str;cin>>str;
    stack<int> s;
    for(int i=0;i<str.length();i++){
        if(oprt(str[i])==0) s.push(str[i]-'0');//1
        else{
            int str1=s.top();s.pop();
            if(str1=='+') s.push(str1+cal(s.top(),s.top()));
            if(str1=='-') s.push(str1-cal(s.top(),s.top()));
            if(str1=='*') s.push(str1*s.top());
            if(str1=='/') s.push(str1/s.top());
        }
    }
}
```

```

        int str2=s.top();s.pop();
    /*2*/ int temp=cal(str2,str1,str[i]);//str2 sau str1 ới đúng thứ tự
        s.push(temp);//3
    }
    }cout<<s.top()<<endl;
}
int main(){
    int t; cin>>t;
    while(t--) initwsolve();
}

```

TÍNH GIÁ TRỊ BIỂU THỨC TIỀN TỐ

Hãy viết chương trình tính toán giá trị của biểu thức tiền tố.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức tiền tố exp. Các số xuất hiện trong biểu thức là các số đơn có 1 chữ số.

Output:

- Đưa ra kết quả mỗi test theo từng dòng, chỉ lấy giá trị phần nguyên.

Ràng buộc:

- T, exp thỏa mãn ràng buộc: $1 \leq T \leq 100$; $2 \leq \text{length(exp)} \leq 20$.

Ví dụ:

Input	Output
2	8
-+8/632	
-+7*45+20	25

Giải:

//1: để tính đúng biểu thức => duyệt ngược vì tiền tố là dấu ở trước

//2: nếu str[i] không phải toán tử ta s.push() còn nếu phải toán tử thì ta lấy 2 cái số đầu trong stack và cùng với toán tử để tạo thành biểu thức.

//3 : sau đó đưa kết quả vừa có được là 1 số đưa vào stack . khi đó s,top() lại coi như 1 toán hạng lớn .

```

#include<bits/stdc++.h>
using namespace std;
bool oprt(char x){
    if(x=='+'||x=='-'||x=='*'||x=='/')
        return 1; return 0;
}

```

```

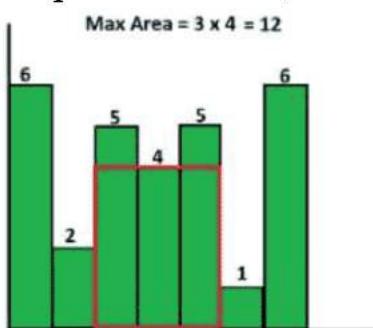
}
int cal(int a,int b,char x){
    if(x=='+')return a+b;  if(x=='-')return a-b;
    if(x=='*')return a*b;  if(x=='/')return a/b;
}
void initwsolve(){
    string str;cin>>str;
    stack<int> s;
    for(int i=str.length()-1;i>=0;i--){
        if(oprt(str[i])==0) s.push(str[i]-'0');//1
        else{
            int str1=s.top();s.pop();
            int str2=s.top();s.pop();
            /*2*/ int temp=cal(str1,str2,str[i]);//str2 sau str1 ới đúng thứ tự
            s.push(temp);//3
        }
    }cout<<s.top()<<endl;
}
int main(){
    int t; cin>>t;
    while(t--) initwsolve();
}

```

HÌNH CHỮ NHẬT LỚN NHẤT

Ôn lại

Cho N cột, mỗi cột có chiều cao bằng $H[i]$. Bạn hãy tìm hình chữ nhật lớn nhất bị che phủ bởi các cột?



Input:

Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).

Mỗi test bắt đầu bởi số nguyên N ($N \leq 100\,000$).

Dòng tiếp theo gồm N số nguyên $H[i]$ ($1 \leq H[i] \leq 10^9$).

Output:

Với mỗi test, in ra diện tích hình chữ nhật lớn nhất tìm được.

Ví dụ:

Input	Output
2	12
7	6
6 2 5 4 5 1 6	
3	
2 2 2	

Giải thích test 1

Vòng for thứ nhất

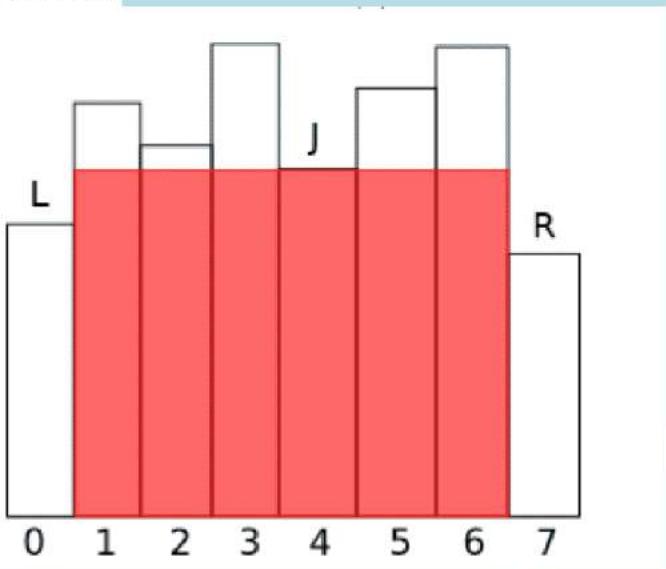
vị trí	mảng	stack	stack
		val	pos
0	6	6	0
1	2	2	1
2	5	2 5	1 2
3	4	2 4	1 3
4	5	2 4 5	1 3 4
5	1	1	5
6	6	1 6	5 6
mảng			
r	1 5 3 5 5 7 7		

Vòng for thứ 2:

vị trí	mảng	stack	stack
		val	pos
6	6	6	6
5	1	1	5
4	5	1 5	5 4
3	4	1 4	5 3
2	5	1 4 5	5 3 2
1	2	1 2	5 1
0	6	1 2 6	5 1 0
mảng		-1 -1 1 1 3 -1	
1	5		

$r-l-1 \Rightarrow$ khoảng cách >>

Với mỗi ô jj trên hàng ii, ta tìm $f(j)f(j)$ là số ô 1 liên tiếp trên cột jj, tính từ hàng ii trở lên. Sau đó, với mỗi cột jj, ta tiếp tục tìm ô gần nhất bên trái và ô gần nhất bên phải có ff nhỏ hơn $f(j)f(j)$, sau đó tính diện tích hình chữ nhật ở cột jj là $S=f(j) \times (r-l-1)S=f(j) \times (r-l-1)$ với l,rl,r là chỉ số 2 ô bên trái và bên phải nói trên.



Giải :

```
#include<bits/stdc++.h>
using namespace std;
long long n,a[1000005];
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    long long res=0;
    vector<long long> l(n),r(n);
    stack <long long> val, pos;
    //stack val và pos lưu giá trị và vị trí của số lớn hơn bê phai ngay sau đó
    for(int i=0;i<n;i++){
        while(val.size()&&a[i]<val.top()){
            r[pos.top()]=i;
            val.pop();pos.pop();
        } val.push(a[i]);pos.push(i);
    }
    while(val.size()){
        r[pos.top()]=n;
        val.pop();pos.pop();
    }
}
```

```

}
for(int i=n-1;i>=0;i--){
    while(val.size()&&a[i]<val.top()){
        l[pos.top()]=i;
        val.pop();pos.pop();
    }val.push(a[i]);pos.push(i);
}
while(val.size()){
    l[pos.top()]=-1;//note
    val.pop();pos.pop();
}
for(int i=0;i<n;i++){
    res=max(res,a[i]*(r[i]-l[i]-1));
}
cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

Cách 2:

```

#include<bits/stdc++.h>
using namespace std;
long long n,a[1000005];
void initwsolve(){
    cin>> n;
    for(int i=0;i<n;i++) cin>>a[i];
    long long res=0;
    vector<long long> l(n),r(n);
    for(int i=0;i<n;i++) l[i]=-1;
    for(int i=0;i<n;i++) r[i]=n;
    for(int i=0;i<n-1;i++){
        for(int j=i+1;j<n;j++){
            if(a[j]<a[i]) {r[i]=j;break;}
        }
    }
    for(int i=1;i<n;i++){

```

```

        for(int j=i-1;j>=0;j--){
            if(a[j]<a[i]) {l[i]=j;break;}
        }
    }
    for(int i=0;i<n;i++){
        res=max(res,a[i]*(r[i]-l[i]-1));
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

BIÊU THÚC TĂNG GIẢM

Ôn lại

Cho dãy ký tự S chỉ bao gồm các ký tự I hoặc D. Ký tự I được hiểu là tăng (Increasing) ký tự D được hiểu là giảm (Decreasing). Sử dụng các số từ 1 đến 9, hãy đưa ra số nhỏ nhất được đoán nhận từ S. Chú ý, các số không được phép lặp lại. Dưới đây là một số ví dụ mẫu:

- A[] = “I” : số tăng nhỏ nhất là 12.
- A[] = “D” : số giảm nhỏ nhất là 21
- A[] = “DD” : số giảm nhỏ nhất là 321
- A[] = “DDIDDDIID”: số thỏa mãn 321654798

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào T bộ test. Mỗi bộ test là một xâu S
- T, S thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(S) \leq 8$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input:	Output:
4	
I	12
D	21
DD	321
DDIDDDIID	321654798

Giải :

```
//1 : ta thấy chuỗi có n ký tự nhưng kq có n+1 ký tự =>for(int i=1;i<=str.length())
//2 : dùng 1 stack để lưu trữ dãy giảm => tìm được ký tự tăng ('I'hoặc '/') thì sẽ in
luôn , và in luôn cả stack
// 3: còn nếu phải ký tự giảm ('D') thì đưa nó vào ngăn xếp
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    string str;cin>>str;
    str=' '+str;
    stack<int> s;
    for(int i=1;i<=str.length();i++){//1
        if(str[i]=='D'){s.push(i); } //3
        else{//2
            cout<<i;
            while(!s.empty()){
                cout<<s.top(); s.pop();
            }
        }
    }
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}
```

BIỂU THỨC TƯƠNG ĐƯƠNG

Ôn lại

Cho biểu thức đúng P chỉ bao gồm các phép toán +, -, các toán hạng cùng với các ký tự '(', ')'. Hãy bỏ tất cả các ký tự '(', ')' trong P để nhận được biểu thức tương đương. Ví dụ với $P = a - (b + c)$ ta có kết quả $P = a - b - c$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test là một biểu thức P được viết trên một dòng.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, P thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(P) \leq 10^3$.

Ví dụ:

Input	Output
2	a-b-c
a-(b+c)	a-b+c+d+e-f
a-(b-c-(d+e))-f	

Giải thích test 2:

chỉ số	chuỗi	a-	lệnh
0	a		
1	-		
2	(-	if1
3	b	-	
4	+	-	if2
5	c	-	
6)		
kq		a-b-c	

Giải :

// xét trong một ngăn xếp lưu trữ dấu . có 3 th liên quan đến ngăn xếp:

//1: nếu gặp '(' thì ta đưa dấu + hoặc - trước '(' vào stack để xét xem các dấu trong biểu thức có bị đảo hay không

//2: ta thấy là các dấu sẽ bị đảo nếu dấu trước () là dấu '-' . => + thành - và - thành +

//3: khi gặp ')' thì ta s.pop() tức xóa dấu trước dấu ngoặc (ám chỉ là đã thực hiện

xét các dấu có trong biểu thức () rồi)

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    string str; cin>>str;
```

```
    int res=0;
```

```
    stack<char> s;
```

```
    for(int i=0;i<str.length();i++){
```

```
        if(str[i]=='('){//1
```

```
            if(str[i-1]=='+'||str[i-1]=='-') s.push(str[i-1]); // note : phải có dòng if
```

```
        }
```

```
        /*2*/if(s.size()&&s.top()=='-'){//note s.size trước s.top
```

```

        if(str[i]=='+') str[i]='-';
        else if(str[i]=='-') str[i]='+';
    }
    if(str[i]==')'&&s.size())s.pop();
}
for(int i=0;i<str.length();i++){
    if(str[i]!='(' && str[i]!=')') cout<<str[i];
}cout<<endl;
}

int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

PHẦN TỬ BÊN PHẢI NHỎ HƠN

Ôn lại

Cho mảng A[] gồm n phần tử. Hãy đưa ra các phần tử nhỏ hơn tiếp theo của phần tử lớn hơn đầu tiên phần tử hiện tại. Nếu phần tử hiện tại không có phần tử lớn hơn tiếp theo ta xem là -1. Nếu phần tử không có phần tử nhỏ hơn tiếp theo ta cũng xem là -1. Ví dụ với mảng A[] = {5, 1, 9, 2, 5, 1, 7} ta có kết quả là ans = {2, 2, -1, 1, -1, -1, -1} vì:

Next Greater	Right Smaller
5 → 9	9 → 2
1 → 9	9 → 2
9 → -1	-1 → -1
2 → 5	5 → 1
5 → 7	7 → -1
1 → 7	7 → -1
7 → -1	7 → -1

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào n là số phần tử của mảng A[], dòng tiếp theo đưa vào n số A[i].

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, n, A[i] thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq n, A[i] \leq 10^6$.

Ví dụ:

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

Giải thích test 1:

Vòng for 1

chi số	mảng	stack val	stack pos
0	5	5	0
1	1	5 1	0 1
2	9	9	2
3	2	9 2	2 3
4	5	9 5	2 4
5	1	9 5 1	2 4 5
6	7	9 7	2 6

mảng

b:

2 2 -1 4 6 6 -

1

Vòng for 2:

chi số	mảng	stack val	stack pos
0	5	5	0
1	1	1	1
2	9	1 9	1 2
3	2	1 2	1 3
4	5	1 2 5	1 3 4
5	1	1 1	1 5
6	7	1 1 7	1 5 6

mảng

c

1 -1 3 5 5 -1

-1

Giải :

//1:khi gặp 1 số lớn hơn s.top() thì ta xóa toàn bộ các số và vị trí mà số <a[i] có trong stack , đưa toàn bộ dữ liệu của pos vào mảng b, cập nhật số lớn hơn và vị trí của nó vào 2 stack

//mảng a lưu trữ các số gốc , mảng b lưu trữ vị trí các số > tiếp theo , mảng c lưu trữ vị trí các số < tiếp theo

//2: kết thúc vòng for đầu ta tìm được các phần tử ko thỏa mãn lớn hơn đầu tiên phần tử hiện tại => cho kết quả=-1 từ cách lấy các vị trí từ pos

//3:stack val sẽ lưu trữ tất cả các phần tử ko thỏa mãn nhỏ hơn tiếp theo của phần tử hiện tại và stack pos sẽ lưu trữ vị trí của chúng.

//4: kết thúc vòng for thứ hai ta tìm được các phần tử ko thỏa mãn nhỏ hơn tiếp theo phần tử hiện tại => cho kết quả =-1 từ việc lấy các vị trí từ pos

// 5:như vậy nếu như b[i]!=-1 (tức tìm được số lớn hơn đầu tiên p từ hiện tại) &&c[b[i]]!= -1(tìm được số nhỏ hơn tiếp theo của số lớn hơn đầu tiên p từ hiện tại) thì in ra số thỏa mãn , ngược lại in ra -1

```
#include<bits/stdc++.h>
using namespace std;
int n,a[1000005],b[1000005],c[1000005];//mảng a lưu trữ các số gốc , mảng b
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>a[i];
    stack<int> val, pos;
    /*1*/ for(int i=0;i<n;i++){
        while(val.size()&&a[i]>val.top()){
            b[pos.top()]=i;
            val.pop();pos.pop();
        }
        val.push(a[i]);pos.push(i);
    }
    /*2*/ while(val.size()){//cho những số >> << =-1
        b[pos.top()]=-1;
        val.pop();pos.pop();
    }
    /*3*/ for(int i=0;i<n;i++){
        while(val.size()&&a[i]<val.top()){
            c[pos.top()]=i;
            val.pop();pos.pop();
        }
        val.push(a[i]);pos.push(i);
    }
}
```

```

    }
/*4*/ while(val.size()){
    c[pos.top()]=-1;
    val.pop();pos.pop();
}
/*5*/ for(int i=0;i<n;i++){
    if(b[i]!=-1&&c[b[i]]!=-1) cout<<a[c[b[i]]]<<' ';
    else cout<<-1<<' ';
}
cout<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

SO SÁNH BIỂU THỨC

Gần y hệt BIỂU THỨC TƯƠNG ĐƯƠNG

Cho P1, P2 là hai biểu thức đúng chỉ bao gồm các ký tự mở ngoặc ‘(’ hoặc đóng ngoặc ‘)’ và các toán hạng in thường. Nhiệm vụ của bạn là định xem P1 và P2 có giống nhau hay không.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test gồm hai dòng: dòng thứ nhất đưa vào P1, dòng tiếp theo đưa vào P2.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- T, P thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq \text{length}(P) \leq 100$.

Ví dụ:

Input	Output
2 -(a+b+c) -a-b-c a-b-(c-d)	YES NO

a-b-c-d	
---------	--

Giai:

// xét trong một ngăn xếp lưu trữ dấu . có 3 th liên quan đến ngăn xếp:
//1: nếu gặp '(' thì ta đưa dấu + hoặc - trước '(' vào stack để xét xem các dấu trong
biểu thức có bị đảo hay không
//2: ta thấy là các dấu sẽ bị đảo nếu dấu trước () là dấu '-' . => + thành - và - thành +
//3: khi gặp ')' thì ta s.pop() tức xóa dấu trước dấu ngoặc (ám chỉ là đã thực hiện
xét các dấu có trong biểu thức () rồi)

```
#include<bits/stdc++.h>
using namespace std;
string repair(string str){
    stack<char> s;
    for(int i=0;i<str.length();i++){
        if(str[i]=='('){//1
            if(str[i-1]=='+'||str[i-1]=='-') s.push(str[i-1]);
        }
        /*2*/if(s.size()&&s.top()=='-'){//note s.size trước s.top
            if(str[i]=='+') str[i]='-';
            else if(str[i]=='-') str[i]='+';
        }
        if(str[i]==')'&&s.size())s.pop();
    }
    string res="";
    for(int i=0;i<str.length();i++){
        if(str[i]!='(' && str[i]!=')') res+=str[i];
    }
    return res;
}
void initwsolve(){
    string s1,s2;cin>>s1>>s2;
    if(repair(s1)==repair(s2)) cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}
```

PHẦN TỬ CÓ SỐ LẦN XUẤT HIỆN NHIỀU HƠN BÊN PHẢI

Ôn lại

Cho mảng $A[]$ gồm n phần tử. Nhiệm vụ của bạn là tìm phần tử gần nhất phía bên phải có số lần xuất hiện lớn hơn phần tử hiện tại. Nếu không có phần tử nào có số lần xuất hiện lớn hơn phần tử hiện tại hãy đưa ra -1.

Ví dụ với dãy $A[] = \{1, 1, 2, 3, 4, 2, 1\}$, ta nhận được kết quả $ans[] = \{-1, -1, 1, 2, 2, 1, -1\}$ vì số lần xuất hiện mỗi phần tử trong mảng là $F = \{3, 3, 2, 1, 1, 2, 3\}$ vì vậy phần tử $A[0] = 1$ có số lần xuất hiện là 3 và không có phần tử nào xuất hiện nhiều hơn 3 nên $ans[0] = -1$, tương tự như vậy với $A[2]=2$ tồn tại $A[6] = 1$ có số lần xuất hiện là 3 nên $ans[2] = 1$.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T ;
- Những dòng tiếp theo mỗi dòng đưa vào một bộ test. Mỗi bộ test gồm hai dòng, dòng đầu tiên đưa vào số n là số các phần tử của mảng $A[]$; dòng tiếp theo đưa vào n số của mảng $A[]$; các phần tử được viết cách nhau một vài khoảng trắng.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ràng buộc:

- $T, n, A[i]$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 0 \leq n, A[i] \leq 10^6$.

Ví dụ:

Input	Output
1 7 1 1 2 3 4 2 1	-1 -1 1 2 2 1 -1

chỉ số	mảng	stack val	stack pos
0	1	1	0
1	1	1 1	0 1
2	2	1 1 2	0 1 2
		1 1 2	0 1 2
3	3	3	3
4	4	1 1 2	0 1 2

		3 4	3 4
		1 1 2	0 1 2
5	2	2	5
6	1	1 1 1	0 1 6

mảng	-1 -1 6 5 5 6
b	-1
mảng	-1 -1 1 2 2 1
a	-1

Giải :

//mảng a lưu trữ các số gốc , mảng f lưu trữ tần suất , mảng b để lưu trữ vị trí của số có tần suất lớn hơn ở bên phải nó
 // stack val và pos dùng để lưu giá trị và vị trí của các số không thể tìm được số có tần suất lớn hơn ở bên phải nó
 // (cuối cùng tại các vị trí đó cho nó bằng = -1)
 // in ra như đầu bài.vì mảng b là mảng đánh dấu vị trí => in ra các a[b[i]]nếu b[i] !=1 , còn lại in -1

```
#include<bits/stdc++.h>
using namespace std;
int n,a[1000005],b[1000005],f[1000005];
void initwsolve(){
    cin>>n;
    memset(f,0,sizeof(f));
    for(int i=0;i<n;i++) {cin>>a[i];f[a[i]]++;}
    stack<int> val,pos;
    for(int i=0;i<n;i++){
        while(val.size()&&f[a[i]]>f[val.top()]){
            b[pos.top()]=i;
            val.pop();pos.pop();
        }
        val.push(a[i]);pos.push(i);
    }
    while(val.size()){
        b[pos.top()]=-1;
        val.pop();pos.pop();
    }
    for(int i=0;i<n;i++) {
        if(b[i]!=-1) cout<<a[b[i]]<<' ';
    }
}
```

```

        else cout<<-1<<' ';
    }cout<<endl;
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

Hàng đợi

Một số hàm sử dụng đến :

```

void out(priority_queue<long long> p){
    while(!p.empty()){
        cout<<p.top()<<' ';
        p.pop();
    }
    cout<<endl;
}

```

```

void out(queue<string> q){
    while(!q.empty()){
        cout<<q.front()<<' ';
        q.pop();
    }
    cout<<endl;
}

```

CẤU TRÚC DỮ LIỆU HÀNG ĐỢI 1

Ban đầu cho một queue rỗng. Bạn cần thực hiện các truy vấn sau:

1. Trả về kích thước của queue
2. Kiểm tra xem queue có rỗng không, nếu có in ra “YES”, nếu không in ra “NO”.
3. Cho một số nguyên và đẩy số nguyên này vào cuối queue.
4. Loại bỏ phần tử ở đầu queue nếu queue không rỗng, nếu rỗng không cần thực hiện.
5. Trả về phần tử ở đầu queue, nếu queue rỗng in ra -1.

6. Trả về phần tử ở cuối queue, nếu queue rỗng in ra -1.

Dữ liệu vào

Dòng đầu tiên chứa số nguyên T là số bộ dữ liệu, mỗi bộ dữ theo dạng sau.

Dòng đầu tiên chứa số nguyên n - lượng truy vấn ($1 \leq n \leq 1000$)

N dòng tiếp theo, mỗi dòng sẽ ghi loại truy vấn như trên, với truy vấn loại 3 sẽ có thêm một số nguyên, không quá 10^6 .

Kết quả: In ra kết quả của các truy vấn..

Ví dụ:

Input	Output
1	
14	
3 1	1
3 2	3
3 3	5
5	
6	
4	
4	5
4	2
4	
3 5	
3 6	
5	
1	

Giải : rất dễ

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    queue<int> q;
    while (n--){
        int ask;cin>>ask;
        switch(ask){
            case 1: cout<<q.size()<<endl;break;
            case 2: if(q.size()==0)cout<<"YES"<<endl;
                      else cout<<"NO"<<endl;break;
        }
    }
}
```

```

        case 3: int temp;cin>>temp;q.push(temp);break;
        case 4: if(q.size())q.pop();break;
        case 5:if(q.size()) cout<<q.front()<<endl;
                  else cout<<-1<<endl;break;
        case 6:if(q.size()) cout<<q.back()<<endl;
                  else cout<<-1<<endl;break;
    }
}
int main(){
    int t;cin>>t;
    while (t--)initwsolve();
    return 0;
}

```

CẤU TRÚC DỮ LIỆU HÀNG ĐỢI 2

Yêu cầu bạn xây dựng một queue với các truy vấn sau đây:

“PUSH x”: Thêm phần tử x vào cuối của queue ($0 \leq x \leq 1000$).

“PRINTFRONT”: In ra phần tử đầu tiên của queue. Nếu queue rỗng, in ra “NONE”.

“POP”: Xóa phần tử ở đầu của queue. Nếu queue rỗng, không làm gì cả.

Dữ liệu vào:

Dòng đầu tiên là số lượng truy vấn Q ($Q \leq 100000$).

Mỗi truy vấn có dạng như trên.

Kết quả:

Với mỗi truy vấn “PRINT”, hãy in ra phần tử đầu tiên của queue. Nếu queue rỗng, in ra “NONE”.

Ví dụ:

Input	Output
9	
PUSH 1	
PUSH 2	2
POP	2
PRINTFRONT	NONE
PUSH 3	
PRINTFRONT	

POP POP PRINTFRONT	
--------------------------	--

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    cin.ignore();
    queue<int> q;
    while(n--){
        string s;cin>>s;
        if(s=="PUSH"){ int temp;cin>>temp;q.push(temp);}
        else if(s=="PRINTFRONT"){
            if(q.size())cout<<q.front()<<endl;
            else cout<<"NONE"<<endl;
        }
        else{
            if(q.size()) q.pop();
        }
    }
}
int main(){
    initwsolve();
    return 0;
}
```

HÀNG ĐỢI HAI ĐẦU (DEQUEUE)

Ôn lại

Yêu cầu bạn xây dựng một hàng đợi hai đầu với các truy vấn sau đây:

- “PUSHFRONT x”: Thêm phần tử x vào đầu của dequeue ($0 \leq x \leq 1000$).
- “PRINTFRONT”: In ra phần tử đầu tiên của dequeue. Nếu dequeue rỗng, in ra “NONE”.
- “POPFRTONT”: Xóa phần tử đầu của dequeue. Nếu dequeue rỗng, không làm gì cả.
- “PUSHBACK x”: Thêm phần tử x vào cuối của dequeue ($0 \leq x \leq 1000$).
- “PRINTBACK”: In ra phần tử cuối của dequeue. Nếu dequeue rỗng, in ra “NONE”.
- “POPBCK”: Xóa phần tử cuối của dequeue. Nếu dequeue rỗng, không làm gì cả.

Dữ liệu vào:

Dòng đầu tiên là số lượng truy vấn Q ($Q \leq 100000$).

Mỗi truy vấn có dạng như trên.

Kết quả:

Với mỗi truy vấn “PRINTFRONT” và “PRINTBACK”, hãy in ra kết quả trên một dòng.

Ví dụ:

Input	Output
10	
PUSHBACK 1	
PUSHFRONT 2	2
PUSHBACK 3	1
PRINTFRONT	3
POPFRTONT	NONE
PRINTFRONT	
POPFRTONT	
PRINTBACK	
POPFRTONT	
PRINTBACK	

Giải :

```
//tham khảo deque trong : https://www.geeksforgeeks.org/deque-cpp-stl/
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    cin.ignore();
    deque<int> d;
    while(n--){
        string s;cin>>s;
        if(s=="PUSHFRONT"){ int temp;cin>>temp;d.push_front(temp);}
        else if(s=="PUSHBACK"){ int temp;cin>>temp;d.push_back(temp);}
        else if(s=="PRINTFRONT"){
            if(d.size())cout<<d.front()<<endl; else cout<<"NONE"<<endl;
        }
        else if(s=="PRINTBACK"){
            if(d.size())cout<<d.back()<<endl; else cout<<"NONE"<<endl;
        }
        else if(s=="POPFRTONT" && d.size()) d.pop_front();
    }
}
```

```
        else if(s=="POPBACK"&&d.size()) d.pop_back();
    }
int main(){
    initwsolve();
    return 0;
}
```

GIÁ TRỊ NHỎ NHẤT CỦA XÂU

Cho xâu ký tự $S[]$ bao gồm các ký tự in hoa [A, B, ..., Z]. Ta định nghĩa giá trị của xâu $S[]$ là tổng bình phương số lần xuất hiện mỗi ký tự trong xâu. Ví dụ với xâu $S[] = \text{"AAABBCD"}$ ta có $F(S) = 3^2 + 2^2 + 1^2 + 1^2 = 15$. Hãy tìm giá trị nhỏ nhất của xâu $S[]$ sau khi loại bỏ K ký tự trong xâu.

Input:

- Dòng đầu tiên đưa vào số lượng test T ($T \leq 100$).
 - Mỗi test được tổ chức thành 2 dòng. Dòng thứ nhất ghi lại số K. Dòng thứ 2 ghi lại xâu ký tự S[] có độ dài không vượt quá 10^6 .

Output:

- Đưa ra giá trị nhỏ nhất của mỗi test theo từng dòng.

Input	Output
2	
0	
ABCC	6
1	3
ABCC	

Vd.

//AAAABBCD

//queue ban đầu:3 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

11/10

Giai:

//1: đánh dấu các chữ cái xuất hiện trong mảng và sắp xếp giảm dần

//2: đưa toàn bộ mảng f vào quêu (có thể bbor các số 0 cũng AC)

//3: lấy phần tử đầu queue xem đã xóa hết ký tự chưa. nếu chưa thì trừ thay $p.top() = p.top() - 1$;

//4: còn rồi thì res+= các p.top()*p.top()

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

```
using namespace std;
```

```

void initwsolve(){
    int n;cin>>n;
    cin.ignore();
    string s;cin>>s;
    int f[26]; memset(f,0,sizeof(f));
    for(int i=0;i<s.length();i++) f[s[i]-'A']++;
    sort(f,f+26,greater<int>());//1
    priority_queue<long long> p;
    long long res=0;
    for(int i=0;i<26;i++) p.push(f[i]);//2
    while(p.size()){
        long long temp1=p.top();
        if(n){//3
            p.pop();
            p.push(temp1-1);n--;
        }
        else{//4
            res+=temp1*temp1;
            p.pop();
        }
    }
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

SỐ 0 VÀ SỐ 9

Cho số tự nhiên N. Hãy tìm số nguyên dương X nhỏ nhất được tạo bởi số 9 và số 0 chia hết cho N. Ví dụ với N = 5 ta sẽ tìm ra X = 90.

Input:

- Dòng đầu tiên ghi lại số lượng test T ($T \leq 100$).
- Những dòng kế tiếp mỗi dòng ghi lại một test. Mỗi test là một số tự nhiên N được ghi trên một dòng ($N \leq 100$).

Output:

- Đưa ra theo từng dòng số X nhỏ nhất chia hết cho N tìm được .

Ví dụ:

Input	Output
2	90
5	9009
7	

Giải thích: queue từ đầu while đến khi while kết thúc:

2
7
90 99
99 900 909
900 909 990 999
909 990 999 9000 9009
990 999 9000 9009 9090 9099
999 9000 9009 9090 9099 9900 9909
9000 9009 9090 9099 9900 9909 9990 9999
9009 9090 9099 9900 9909 9990 9999 90000 90009

Kq : 9009

Giải :

//1: ban đầu đưa 9 vào trong queue
//2: thực hiện thêm vào queue các số sau (90, 99,...) bằng cách
q.push(q.front()*10); q.push(q.front()*10+9);
//3: tại đó xóa số cũ đi . thêm vào cho đến khi nào tìm được q.front() chia hết cho n
thì kết thúc vòng lặp

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    long long n; cin >> n;
    queue<long long> q;
    q.push(9); //1
    while(1){
        /*3*/ if(q.front()%n==0){ cout << q.front() << endl; break; }
        else{ //2
            q.push(q.front()*10);
            q.push(q.front()*10+9);
            q.pop();
        }
    }
}
```

```

int main(){
    int t; cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

SỐ NHỊ PHÂN TỪ 1 ĐẾN N

Ôn lại

Cho số tự nhiên n. Hãy in ra tất cả các số nhị phân từ 1 đến n.

Input:

- Dòng đầu tiên ghi lại số lượng test T ($T \leq 100$).
- Mỗi test là một số tự nhiên n được ghi trên một dòng ($n \leq 10000$).

Output:

- Đưa ra kết quả mỗi test trên một dòng.

Ví dụ:

Input	Output
2	1 10
2	1 10 11 100 101
5	

Giải thích : trạng thái queue khi n=5:

10 11
 11 100 101
 100 101 110 111
 101 110 111 1000 1001
 110 111 1000 1001 1010 1011

Giải:

```

// note: inn từ 1 đến n => tư duy phải chạy vòng for 1-n;
// chạy từ 1 => đây 1 và queue , sau đó cứ mỗi lần lấy số đầu của queue thì đây
vào đó 2 số temp+'0' và temp+'1'

```

```

#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n; cin>>n;
    queue<string> q;
    q.push("1");
    for(int i=1;i<=n;i++){//1
        string temp=q.front();
        cout<<temp<<' ';
        q.pop();
        q.push(temp+"0");
        q.push(temp+"1");
    }
}

```

```

        q.push(temp+'0');
        q.push(temp+'1');
    }cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

Cách 2 : dùng long long
// note: inn từ 1 đến n => từ duy phải chạy vòng for 1-n;
// chạy từ 1 => đầy 1 và queue , sau đó cứ mỗi lần lấy số đầu của queue thì đầy
vào đó 2 số temp+'0' và temp+'1'
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    queue<long long> q;
    q.push(1);
    for(int i=1;i<=n;i++){//1
        long long temp=q.front();
        cout<<temp<<' ';
        q.pop();
        q.push(temp*10);
        q.push(temp*10+1);
    }cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

SỐ BDN 1 (nhị phân)

Ôn lại

Ta gọi số nguyên dương K là một số BDN nếu các chữ số trong K chỉ bao gồm các 0 hoặc 1 có nghĩa. Ví dụ số K = 1, 10, 101. Cho số tự nhiên N ($N < 2^{63}$). Hãy

cho biết có bao nhiêu số BDN nhỏ hơn N. Ví dụ N=100 ta có 4 số BDN bao gồm các số: 1, 10, 11, 100.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
- T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một số tự nhiên N.
Output:
- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	2
10	4
100	7
200	

Giải:

//1:đếm đến khi nào $\leq n$ thì thôi \Rightarrow dùng while

//2:bước đầu phải đẩy 1 và queue , sau đó cứ mỗi lần lấy số đầu của queue thì đẩy vào đó 2 số temp+'0' và temp+'1'

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    long long n;cin>>n;//note :ll
    queue<long long> q;
    q.push(1);//2
    int res=0;
    while(1){//1
        long long temp=q.front();
        if (temp<=n)res++; else break;
        q.pop();
        q.push(temp*10);//2
        q.push(temp*10+1);
    }cout<<res<<endl;
}
```

```
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

SỐ BDN 2 (chia hết cho N)

Ta gọi số nguyên dương K là một số BDN nếu các chữ số trong K chỉ bao gồm các 0 hoặc 1 có nghĩa. Ví dụ số K = 101 là số BDN, k=102 không phải là số BDN.

Số BDN của N là số P = M*N sao cho P là số BDN. Cho số tự nhiên N ($N < 1000$), hãy tìm số BDN nhỏ nhất của N.

Ví dụ. Với $N=2$, ta tìm được số BDN của N là $P = 5^2 = 10$. $N = 17$ ta tìm được số BDN của 17 là $P = 653 \cdot 17 = 11101$.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
- T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một số tự nhiên N.

Output:

Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	10
2	11100
12	11101
17	

Giải:

// ý tưởng dùng queue đầy các số chứa các cs 1 và 0 đến khi nào tại q.front() mà chia hết cho n thì in ra kết quả và dừng tìm kiếm

```
#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    queue<long long> q;
    q.push(1);
    while(1){
        long long temp=q.front();
        if(temp%n==0){cout<<temp<<endl;break;}
        q.pop();
        q.push(temp*10);
        q.push(temp*10+1);
    }
}
int main(){
    int t;cin>>t;
```

```

while(t--)initwsolve();
return 0;
}

```

SỐ LỘC PHÁT 1 (gồm 6 và 8)

Một số được gọi là lộc phát nếu chỉ có 2 chữ số 6 và 8. Cho số tự nhiên N. Hãy liệt kê các số lộc phát có không quá N chữ số.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng bộ test ($T < 10$);
- T dòng kế tiếp mỗi dòng ghi số N ($1 < N < 15$).

Output:

- In ra đáp án theo thứ tự giảm dần.

Ví dụ:

Input	Output
2	88 86 68 66 8 6
2	888 886 868 866 688 686 668 666 88 86 86 68 66 8 6
3	

Giải :

//1: ý tưởng dùng queue đầy các số chứa các cs 6 và 8 đến khi nào độ dài q.front() == n thì thôi

//2 như vậy mỗi 1 lần xóa q.front() thì phải đưa chúng vào 1 vector .

//3 cuối cùng các kết quả dư thừa trong queue cung phải đưa vào vector nốt => 2 lần

```

#include<bits/stdc++.h>
using namespace std;
void initwsolve(){
    int n;cin>>n;
    queue<string> q;
    q.push("6");q.push("8");
    vector<string> res;
    while(1){
        string temp=q.front();
        res.push_back(temp);//2
        q.pop();
        /*1*/ if(temp.length()<=n-1){
            q.push(temp+'6');
            q.push(temp+'8');
        }
    }
}

```

```

        else break;
    }
/*3*/while(q.size()){
    res.push_back(q.front());
    q.pop();
}
for(int i=res.size()-1;i>=0;i--) cout<<res[i]<<' ';
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

SỐ LỘC PHÁT 2

Một số được gọi là lộc phát nếu chỉ có 2 chữ số 6 và 8. Cho số tự nhiên N. Hãy liệt kê các số lộc phát có không quá N chữ số.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng bộ test ($T < 10$);
- T dòng kế tiếp mỗi dòng ghi số N ($1 < N < 15$).

Output:

- Dòng đầu tiên là số lượng số lộc phát tìm được. Dòng thứ hai in đáp án theo thứ tự tăng dần.

Ví dụ:

Input	Output
2	6
2	6 8 66 68 86 88
3	14 6 8 66 68 86 88 666 668 686 688 866 868 886 888

Giải :

//1: ý tưởng dùng queue đầy các số chứa các cs 6 và 8 đến khi nào độ dài q.front() > n thì thôi

//2 như vậy mỗi 1 lần xóa q.front() thì phải đưa chúng vào 1 vector .

#include <bits/stdc++.h>

using namespace std;

```

void initwsolve(){
    int n;cin>>n;
    queue<string> q;
    q.push("6"); q.push("8");//1
    vector <string> res;
    while( q.front().size()<=n ){//1
        string temp=q.front();
        res.push_back(temp);//2
        q.pop();//2
        q.push(temp+"6");
        q.push(temp+"8");
    }
    cout<<res.size()<<endl;//in biến đếm
    for(int i=0;i<res.size();i++)
        cout<<res[i]<<' ';
    cout<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

SỐ LỘC PHÁT 3

Một số được gọi là lộc phát nếu chỉ có 2 chữ số 6 và 8. Cho số tự nhiên N. Hãy liệt kê các số lộc phát có không quá N chữ số.

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng bộ test ($T < 10$);
- T dòng kế tiếp mỗi dòng ghi số N ($1 < N < 15$).

Output:

- Dòng đầu tiên là số lượng số lộc phát tìm được. Dòng thứ hai in ra đáp án theo thứ tự giảm dần.

Ví dụ:

Input	Output
2	6
2	88 86 68 66 8 6

3

14

888 886 868 866 688 686 668 666 88 86 86 68 66 8 6

Giải:

//1: ý tưởng dùng queue đẩy các số chưa có cs 6 và 8 đến khi nào độ dài q.front() > n thì thôi

//2 như vậy mỗi 1 lần xóa q.front() thì phải đưa chúng vào 1 vector.

//3 : giống y lộc phát 2 nhưng chỉ là in kết quả theo thứ tự ngược lại

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

```
using namespace std;
```

```
void initwsolve(){
```

```
    int n; cin >> n;
```

```
    queue<string> q;
```

```
    q.push("6"); q.push("8");//1
```

```
    vector<string> res;
```

```
    while( q.front().size() <= n ){//1
```

```
        string temp = q.front();
```

```
        res.push_back(temp);//2
```

```
        q.pop();//2
```

```
        q.push(temp + "6");
```

```
        q.push(temp + "8");
```

```
}
```

```
cout << res.size() << endl;//in biến đếm
```

```
for( int i = res.size() - 1; i >= 0; i-- )//3
```

```
    cout << res[i] << ' ';
```

```
cout << endl;
```

```
}
```

```
int main(){
```

```
    int t; cin >> t;
```

```
    while( t-- ) initwsolve();
```

```
    return 0;
```

```
}
```

BIẾN ĐỔI S – T (thao tác : nhân , trừ)

Ôn lại

Cho hai số nguyên dương S và T ($S, T < 10000$) và hai thao tác (a), (b) dưới đây:

Thao tác (a): Trừ S đi 1 ($S = S - 1$);

Thao tác (b): Nhân S với 2 ($S = S * 2$);

Hãy dịch chuyển S thành T sao cho số lần thực hiện các thao tác (a), (b) là ít nhất. Ví dụ với $S = 2$, $T = 5$ thì số các bước ít nhất để dịch chuyển S thành T thông qua 4 thao tác sau:

Thao tác (a): $2 * 2 = 4$;

Thao tác (b): $4 - 1 = 3$;

Thao tác (a): $3 * 2 = 6$;

Thao tác (b): $6 - 1 = 5$;

Input:

- Dòng đầu tiên ghi lại số tự nhiên T là số lượng Test;
- T dòng kế tiếp mỗi dòng ghi lại một bộ Test. Mỗi test là một bộ đôi S và T .
Output: Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
3	4
2 5	4
3 7	3
7 4	

Giải :

//1: chú ý mảng c là dùng để đếm , giá trị của c[i] là số bước tạo thành số i từ số s
`#include <bits/stdc++.h>`

`using namespace std;`

`void initwsolve(){`

`int s,t; cin>>s>>t;`

`if(s>=t){`

`cout<<s-t<<endl; return;`

`}`

`int c[20005];//1`

`for(int i=0;i<20005;i++) c[i]=1e9;`

`c[s]=0;`

`queue<int> q;`

`q.push(s);`

`while(c[t]==1e9){`

`int temp=q.front();q.pop();`

`if(temp-1>0&&c[temp-1]==1e9)`

`q.push(temp-1), c[temp-1]=c[temp]+1;`

`if(temp*2<=20000&&c[temp*2]==1e9)`

```

        q.push(temp*2), c[temp*2]=c[temp]+1;
    }
    cout<<c[t]<<endl;
}

int main(){
    int test;cin>>test;
    while(test--) initwsolve();
    return 0;
}

```

TÌM SỐ K THỎA MÃN ĐIỀU KIỆN (cs khác nhau và <=5)

Ôn lại

Cho hai số nguyên dương L, R. Hãy đưa ra số các số K trong khoảng [L, R] thỏa mãn điều kiện:

- Tất cả các chữ số của K đều khác nhau.
- Tất cả các chữ số của K đều nhỏ hơn hoặc bằng 5.

Ví dụ với L = 4, R = 13 ta có 5 số thỏa mãn yêu cầu là 4, 5, 10, 12, 13,

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Dòng tiếp theo đưa vào các bộ test. Mỗi bộ test được là một cặp L, R được viết trên một dòng.
- T, L, R thỏa mãn ràng buộc: $1 \leq T \leq 100$; $0 \leq L \leq R \leq 10^5$.

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
4 13	5
100 1000	100

Giải :

// 1:y tưởng kiểm tra từng số từ 1 đến r

// 2:tại từng số nó ta tách các chữ số ra đưa vào hàng đợi ưu tiên để sắp xếp giảm dần

// 3:sau đó cứ lấy từng chữ số : nếu nó >5 hoặc giống với chữ sốp trước thì loại

// 4:cho đến cuối cùng nếu các c số đều thỏa mãn thì mới return 1

#include<bits/stdc++.h>

using namespace std;

```

int check(int n){
    priority_queue <int>pq;
    while(n){//2
        pq.push(n%10);
        n/=10;
    }
    int temp=0;
/*3*/ if(pq.top()>5) return 0;
    else {
        temp=pq.top();
        pq.pop();}
    while(pq.size()){
        if(pq.top()>5) return 0;
        if(pq.top()==temp) return 0;
        temp=pq.top();pq.pop();}
    return 1;//4
}
void initwsolve(){
    int l,r; cin>>l>>r;
    int res=0;
    for(int i=l;i<=r;i++)//1
        if(check(i)==1) res++;
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}

```

Cách 2:

```

#include<bits/stdc++.h>
using namespace std;
int check(int n){
    vector<int > a;
    while(n){
        a.push_back(n%10);
        n/=10;
    }

```

```

    }
    a.push_back(-1);
    sort(a.begin(),a.end(),greater<int>());
    for(int i=0;i<a.size()-1;i++)
        if(a[i]>5&&a[i]==a[i+1]) return 0;
    return 1;
}
void initwsolve(){
    int l,r; cin>>l>>r;
    int res=0;
    for(int i=l;i<=r;i++)
        if(check(i)==1) res++;
    cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}

```

Đồ thị và đồ thị nâng cao:

CHUYỂN DANH SÁCH CẠNH SANG DANH SÁCH KÈ.

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy viết chương trình thực hiện chuyển đổi biểu diễn đồ thị dưới dạng danh sách kè.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm $|E| + 1$ dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh của đồ thị; $|E|$ dòng tiếp theo đưa vào các bộ đôi $u \tilde{V}, v \tilde{V}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 200; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2;$

Output:

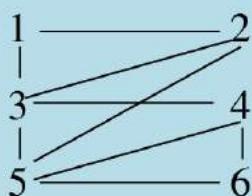
- Đưa ra danh sách kè của các đỉnh tương ứng theo khuôn dạng của ví dụ dưới đây. Các đỉnh trong danh sách in ra theo thứ tự tăng dần.

Ví dụ:

Input:	Output:
1	1: 2 3

6 9	2: 1 3 5
1 2	3: 1 2 4 5
1 3	4: 3 5 6
2 3	5: 2 3 4 6
2 5	6: 4 5
3 4	
3 5	
4 5	
4 6	
5 6	

Giải thích:



Giải :

//note :V(vertice :tập đỉnh trong toán học), E(edge :tập cạnh của đồ họa/bìa lè
sách)

```

#include<bits/stdc++.h>
using namespace std;
int a[1001][1001],V,E;
void initwsolve(){
    memset(a,0,sizeof(a));
    cin>>V>>E;
    for(int i=0;i<E;i++){
        int x,y;cin>>x>>y;
        a[x][y]=a[y][x]=1;// đánh dấu cạnh tồn tại được nối
    }
    for(int i=1;i<=V;i++){
        cout<<i<<" ";
        for(int j=1;j<=V;j++)
            if(a[i][j]==1) cout<<j<<' ';
        cout<<endl;
    }
}
int main()
  
```

```

int t;cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

CHUYÊN TỪ DANH SÁCH KÈ SANG DANH SÁCH CẠNH

Ôn lại

Nội dung bài tập

Cho đơn đồ thị G vô hướng liên thông được mô tả bởi danh sách kè. Hãy in ra danh sách cạnh tương ứng của G.

Input

- Dòng đầu tiên ghi số N là số đỉnh (1
- N dòng tiếp theo mỗi dòng ghi 1 danh sách kè lần lượt theo thứ tự từ đỉnh 1 đến đỉnh N

Output: Ghi ra lần lượt từng cạnh của đồ thị theo thứ tự tăng dần.

Ví dụ

Input	Output
3	
2 3	1 2
1 3	1 3
1 2	2 3

Test danh sách kè sẽ là :

Đỉnh 1 kè với: 2 3

Đỉnh 2 kè vs: 1 3

Đỉnh 3 kè vs: 1 2

=> danh sách cạnh

Giải :

```

#include<bits/stdc++.h>
using namespace std;
int main(){
    int V;cin>>V;
    cin.ignore();
    int a[51][51];memset(a,0,sizeof(a));
    for(int i=1;i<=V;i++){
        string s;
        getline(cin,s);
        s+= ' ';
    }
}

```

```

int j=0;
for(int k=0;k<s.length();k++){
    if(s[k]==' '){
        a[i][j]=a[j][i]=1;
        j=0;
    }
    else j=j*10+s[k]-'0';
}
for(int i=1;i<=V;i++){
    for(int j=i+1;j<=V;j++){
        if(a[i][j]==1)cout<<i<<' '<<j<<endl;
    }
}
return 0;
}

```

CHUYỂN DANH SÁCH KÈ SANG MA TRẬN KÈ

Cho đồ thị vô hướng có n đỉnh dưới dạng danh sách kè.

Hãy biểu diễn đồ thị bằng ma trận kè.

Input: Dòng đầu tiên chứa số nguyên n – số đỉnh của đồ thị ($1 \leq n \leq 1000$). n dòng tiếp theo, dòng thứ i chứa các số nguyên là các đỉnh kè với đỉnh i.

Output: Ma trận kè của đồ thị.

Ví dụ:

Input	Output
3	0 1 1
2 3	1 0 1
1 3	1 1 0
1 2	

Giải:

Cách 1:

```

#include<bits/stdc++.h>
using namespace std;
int a[1005][1005];
int main(){
    int V;cin>>V;
    cin.ignore();

```

```

for(int i=1;i<=V;i++){
    string s;
    getline(cin,s);
    s+=' ';
    int j=0;
    for(int k=0;k<s.length();k++){
        if(s[k]==' '){
            a[i][j]=a[j][i]=1;
            j=0;
        }
        else j=j*10+s[k]-'0';
    }
}
for(int i=1;i<=V;i++){
    for(int j=1;j<=V;j++){
        cout<<a[i][j]<<' ';
    }cout<<endl;
}
return 0;
}

```

Cách 2 : hay nhung sai :

```

#include<bits/stdc++.h>
using namespace std;
int a[1005][1005];
int main(){
    int V;cin>>V;
    for(int i=1;i<=V;i++){
        while(1){
            int temp;cin>>temp;
            a[i][temp]=1;a[temp][i]=1;
            if(getchar()=='\n') break;;
        }
    }
    for(int i=1;i<=V;i++){
        for(int j=1;j<=V;j++)
            cout<<a[i][j]<<' ';
        cout<<endl;
    }
}

```

```
    return 0;  
}
```

BIỂU DIỄN ĐỒ THỊ CÓ HƯỚNG.

Cho đồ thị có hướng G được biểu diễn dưới dạng danh sách cạnh. Hãy viết chương trình thực hiện chuyển đổi biểu diễn đồ thị dưới dạng danh sách kề.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm $|E| + 1$ dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh của đồ thị; $|E|$ dòng tiếp theo đưa vào các bộ đôi $u \rightarrow v$, $v \rightarrow u$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 200$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra danh sách kề của các đỉnh tương ứng theo khuôn dạng của ví dụ dưới đây. Các đỉnh trong danh sách in ra theo thứ tự tăng dần.

Ví dụ:

Input:	Output:
1	
6 9	
1 2	1: 2
2 5	2: 5
3 1	3: 1 2 5
3 2	4: 4
3 5	5: 4 6
4 3	6: 4
5 4	
5 6	
6 4	

Giải :

```
#include<bits/stdc++.h>  
using namespace std;  
int a[1001][1001],v,e;  
void initwsolve(){  
    cin>>v>>e;  
    memset(a,0,sizeof(a));  
    for(int i=1;i<=e;i++){
```

```

int x,y;cin>>x>>y;
a[x][y]=1;
}
for(int i=1;i<=v;i++){
    cout<<i<<" ";
    for(int j=1;j<=v;j++){
        if(a[i][j]==1) cout<<j<<' ';
    }
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

CHUYỂN MA TRẬN KÈ SANG DANH SÁCH KÈ

Nội dung bài tập

Ma trận kè A của một đồ thị vô hướng là một ma trận chỉ có các số 0 hoặc 1 trong đó $A[i][j] = 1$ có ý nghĩa là đỉnh i kè với đỉnh j (chỉ số tính từ 1).

Danh sách kè thì liệt kê các đỉnh kè với đỉnh đó theo thứ tự tăng dần.

Hãy chuyển biểu diễn đồ thị từ dạng ma trận kè sang dạng danh sách kè.

Input: Dòng đầu tiên chứa số nguyên n – số đỉnh của đồ thị ($1 < n \leq 1000$). n dòng tiếp theo, mỗi dòng có n số nguyên có giá trị 0 và 1 mô tả ma trận kè của đồ thị.

Output: Gồm n dòng, dòng thứ i chứa các số nguyên là đỉnh có nối với đỉnh i và được sắp xếp tăng dần. Dữ liệu đảm bảo mỗi đỉnh có kết nối với ít nhất 1 đỉnh khác.

Ví dụ:

Input	Output
3	2 3
0 1 1	1 3
1 0 1	1 2
1 1 0	

Giải :

Cách 1:

#include<bits/stdc++.h>

```

using namespace std;
int a[1001][1001],n;
int main(){
    int n;cin>>n;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];

    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            if( a[i][j]==1) cout<<j<<' ';
            }cout<<endl;
    }
    return 0;
}

```

Cách 2: dùng vector

```

#include<bits/stdc++.h>
using namespace std;
int a[1001][1001],n;
vector <vector<int> > res;
int main(){
    int n;cin>>n;
    res.resize(n+1);
    for(int i=1;i<=n;i++){
        for(int j=1;j<=n;j++){
            cin>>a[i][j];
            if( a[i][j]==1) res[i].push_back(j);
        }
    }
    for(int i=1;i<=n;i++){
        for(int j=0;j<res[i].size();j++){
            cout<<res[i][j]<<' ';
        }cout<<endl;
    }
    return 0;
}

```

DFS TRÊN ĐỒ THỊ VÔ HƯỚNG

Ôn lại

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều sâu bắt đầu tại đỉnh $u \in V$ ($DFS(u) = ?$)

Input:

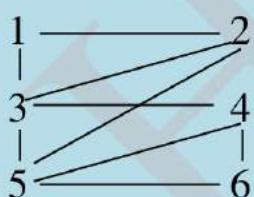
- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm $|E| + 1$ dòng: dòng đầu tiên đưa vào ba số $|V|, |E|$ tương ứng với số đỉnh và số cạnh của đồ thị, và u là đỉnh xuất phát; $|E|$ dòng tiếp theo đưa vào các bộ đôi $u \hat{v}, v \hat{v}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 200; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2;$

Output:

- Đưa ra danh sách các đỉnh được duyệt theo thuật toán $DFS(u)$ của mỗi test theo khuôn dạng của ví dụ dưới đây.

Ví dụ:

Input:	Output:
1 6 9 5 1 2 1 3 2 3 2 4 3 4 3 5 4 5 4 6 5 6	5 3 1 2 4 6



Giải :

Cách 1 :quay lui

// Depth-first search - DFS

// note : bài này không nên dùng mảng $a[1001][1001]$ sau...cho $a[x][y]=1..$ mà phải dùng vector để push_back từng đỉnh một vì để tối ưu hóa dfs , ta chỉ đưa

những số cần đưa vào , nếu dùng mảng thì có quá nhiều số \Rightarrow cách khai báo vector 2 chiều cũng khác những bài thông thường

// 1: duyệt số cạnh của đồ thị và đánh dấu từng đỉnh xem nó được nối với các đỉnh nào

// 2:dfs(tìm kiếm sau bắt đầu từ đỉnh u)

// 3: duyệt các đỉnh kề được nối với đỉnh u (v) nếu đỉnh kề đó chưa được tính (check[v]==0) thì chấp nhận và tiếp tục tìm đỉnh kề tiếp theo.

```
#include<bits/stdc++.h>
using namespace std;
int V,E,u,check[1005];
vector <int> a[1005];
void dfs(int u){//2
    cout<<u<<' ';
    check[u]=1;
    /*3*/ for(int i=0;i<a[u].size();i++){
        int v=a[u][i];
        if(check[v]==0) dfs(v);
    }
}
void initwsolve(){
    cin>>V>>E>>u;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
dfs(u);
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}
```

Cách 2: stack:

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

```

using namespace std;
int V,E,u,check[1005];
vector <int> a[1005];
void dfs(int u){//2
    stack<int> nx;
    nx.push(u);
    cout<<u<<' ';
    check[u]=1;
    while(nx.size()){
        int s=nx.top();nx.pop();
        for(int i=0;i<a[s].size();i++){
            int t=a[s][i];
            if(check[t]==0){
                cout<<t<<' ';
                check[t]=1;
                nx.push(s);
                nx.push(t);
                break;
            }
        }
    }
}
void initwsolve(){
    cin>>V>>E>>u;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
dfs(u);
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();

```

```
}
```

Cách 3 : quay lui dùng ma trận kè vẫn đúng : dùng cho TRR2

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

```
using namespace std;
```

```
int V,E,u,check[1005];
```

```
int a[1005][1005];
```

```
void dfs(int u){
```

```
    cout<<u<<' ';
```

```
    check[u]= 1;
```

```
    for(int v=1;v<=V;v++)
```

```
        if(a[u][v]==1&&check[v]==0) dfs(v);
```

```
}
```

```
void initwsolve(){
```

```
    cin>>V>>E>>u;
```

```
    memset(a,0,sizeof(a));
```

```
    memset(check,0,sizeof(check));
```

```
/*1*/ for(int i=1;i<=E;i++){
```

```
    int x,y;cin>>x>>y;
```

```
    a[x][y]=1;
```

```
    a[y][x]=1;
```

```
}
```

```
dfs(u);
```

```
cout<<endl;
```

```
}
```

```
int main(){
```

```
    int t;cin>>t;
```

```
    while(t--) initwsolve();
```

```
}
```

Cách 4 : stack dùng ma trận kè vẫn đúng dùng cho TRR2

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

```
using namespace std;
```

```
int V,E,u,check[1005];
```

```
int a[1005][1005];
```

```
void dfs(int u){
```

```
    stack<int> nganxep;
```

```
    cout<<u<<' ';
```

```

check[u]=1;
nganxep.push(u);
while(!nganxep.empty()){
    int s=nganxep.top();
    nganxep.pop();
    for(int t=1;t<=V;t++){
        if(a[s][t]==1&&check[t]==0){
            cout<<t<<' ';
            check[t]=1;
            nganxep.push(s);
            nganxep.push(t);
            break;
        }
    }
}
void initwsolve(){
    cin>>V>>E>>u;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
dfs(u);
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

BFS TRÊN ĐỒ THỊ VÔ HƯỚNG

Ôn lại

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy viết thuật toán duyệt theo chiều rộng bắt đầu tại đỉnh $u \in V$ ($BFS(u) = ?$)

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào ba số $|V|$, $|E|$, $u \hat{V}$ tương ứng với số đỉnh, số cạnh và đỉnh bắt đầu duyệt; Dòng tiếp theo đưa vào các bộ đôi $u \hat{V}$, $v \hat{V}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 200$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra danh sách các đỉnh được duyệt theo thuật toán BFS(u) của mỗi test theo khuôn dạng của ví dụ dưới đây.

Ví dụ:

Input:	Output:
1 6 9 1 1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	1 2 3 5 4 6

Giải :

```

// Breadth First Search : thuật toán duyệt theo chiều rộng
// note : bài này không nên dùng mảng a[1001][1001] sau...cho a[x][y]=1.. mà
phải dùng vector để push_back từng đỉnh một vì để tối ưu hóa dfs , ta chỉ đưa
những số cần đưa vào , nếu dùng mảng thì có quá nhiều số 0 => cách khai báo
vector 2 chiều cũng khác những bài thông thường
// 1: duyệt số cạnh của đồ thị và đánh dấu từng đỉnh xem nó được nối với các đỉnh
nào
// 2:bfs(tìm kiếm sau bắt đầu từ đỉnh u)
// 3 : giải thích cách tìm:
//lần 1: s=1 , t thỏa mãn=2,3 , check[2]=1;check[3]=1;
//lần 2: s=2 , t thỏa mãn =5 , check[5]=1;
//lần 3: s=3 , t thỏa mãn =4 , check[4]=1;
//lần 4 :s=5 ,t thỏa mãn =6 ,check[6]=1;
//lần 5 :s=4 in 4 . lần 6 :s=6 in 6
#include<bits/stdc++.h>
using namespace std;
int V,E,u,check[1005];
vector <int> a[1005];
void bfs(int u){//2
    queue <int> q;
    q.push(u);check[u]=1;
    while(q.empty() == 0){//3

```

```

int s=q.front();q.pop();
cout<<s<<' ';
for(int i=0;i<a[s].size();i++){
    int t=a[s][i];
    if(check[t]==0){
        check[t]=1;
        q.push(t);
    }
}
void initwsolve(){
    cin>>V>>E>>u;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
bfs(u);
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

TÌM ĐƯỜNG ĐI THEO DFS VỚI ĐỒ THỊ VÔ HƯỚNG

Ôn lại

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh $s \rightarrow V$ đến đỉnh $t \rightarrow V$ trên đồ thị bằng thuật toán DFS.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số $|V|, |E|, s \rightarrow V, t \rightarrow V$ tương ứng với số đỉnh, số cạnh, đỉnh u , đỉnh v ; Dòng tiếp theo đưa vào các bộ đôi $u \rightarrow V, v \rightarrow V$ tương ứng với một cạnh của đồ thị.

- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Dưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán DFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

Ví dụ:

Input:	Output:
1 6 9 1 6 1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	1 2 3 4 5 6

Ví dụ test khác :

1
6 9 1 6
1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 2
=>1 2 3 4 6 (vì 5 không nối với 6)

Giải :

// 1: duyệt số cạnh của đồ thị và đánh dấu từng đỉnh xem nó được nối với các đỉnh nào

// 2: dfs(tìm kiếm sau bắt đầu từ đỉnh u)

// 3 : duyệt dfs như bài đồ thị vô hướng , khác chỗ đánh dấu giá trị phần tử có chỉ số là đỉnh sau trong mảng chính = đỉnh trước

// 4 : in ra mảng res thích hợp theo thứ tự tăng dần (lấy dữ liệu từ mảng b)

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

```
using namespace std;
```

```
int V,E,s,t,check[1005],b[1005];//b: truoc s=> = b[t];
```

```
vector <int> a[1005];
```

```
void dfs(int u){//2
```

```
    check[u]=1;
```

```
/*3*/ for(int i=0;i<a[u].size();i++){
```

```
    int v=a[u][i];
```

```
    if(check[v]==0) {
```

```
        b[v]=u;
```

```
        dfs(v);
```

```
}
```

```
}
```

```
}
```

```
void initwsolve(){
```

```
    cin>>V>>E>>s>>t;
```

```

memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
dfs(s);
if(check[t]==0) cout<<-1<<endl;
/*4*/ else {
    vector <int> res;
    int i=t;cout<<s<<' ';
    while(i!=s){res.push_back(i); i=b[i];}
    for(int i=res.size()-1;i>=0;i--) cout<<res[i]<<' ';
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cach 2 : ma trận kề

```

#include<bits/stdc++.h>
using namespace std;
int V,E,s,t,check[1005],b[1005];//b: truoc s=> = b[t];
int a[1005][1005];
void dfs(int u){//2
    check[u]=1;
    /*3*/ for(int v=1;v<=V;v++){
        if(a[u][v]==1&&check[v]==0) {
            b[v]=u;
            dfs(v);
        }
    }
}
void initwsolve(){
    cin>>V>>E>>s>>t;

```

```

memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
dfs(s);
if(check[t]==0) cout<<-1<<endl;
/*4*/ else {
    vector <int> res;
    int i=t;cout<<s<<' ';
    while(i!=s){res.push_back(i); i=b[i];}
    for(int i=res.size()-1;i>=0;i--) cout<<res[i]<<' ';
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

ĐƯỜNG ĐI THEO BFS TRÊN ĐỒ THỊ VÔ HƯỚNG

Ôn tập

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy tìm đường đi từ đỉnh $s \in V$ đến đỉnh $t \in V$ trên đồ thị bằng thuật toán BFS.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào bốn số $|V|$, $|E|$, $s \in V$, $t \in V$ tương ứng với số đỉnh, số cạnh, đỉnh u , đỉnh v ; Dòng tiếp theo đưa vào các bộ đôi $u \in V$, $v \in V$ tương ứng với một cạnh của đồ thị.
- T , $|V|$, $|E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra đường đi từ đỉnh s đến đỉnh t của mỗi test theo thuật toán BFS của mỗi test theo khuôn dạng của ví dụ dưới đây. Nếu không có đáp án, in ra -1.

Ví dụ:

Input:	Output:
---------------	----------------

1	
6 9 1 6	
1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	

1 2 5 6

Giải :

```

// Breadth First Search : thuật toán duyệt theo chiều rộng
// note : bài này không nên dùng mảng a[1001][1001] sau...cho a[x][y]=1.. mà
phải dùng vector để push_back từng đỉnh một vì để tối ưu hóa bfs , ta chỉ đưa
những số cần đưa vào , nếu dùng mảng thì có quá nhiều số 0 => cách khai báo
vector 2 chiều cũng khác những bài thông thường
// 1: duyệt số cạnh của đồ thị và đánh dấu từng đỉnh xem nó được nối với các đỉnh
nào
// 2:bfs(tìm kiếm sau bắt đầu từ đỉnh s)
// 3 : giải thích cách tìm:
//lần 1: s=1 , t thỏa mãn=2,3 , check[2]=1;check[3]=1;
//lần 2: s=2 , t thỏa mãn =5 , check[5]=1;
//lần 3: s=3 , t thỏa mãn =4 , check[4]=1;
//lần 4 :s=5 ,t thỏa mãn =6 ,check[6]=1;
//lần 5 :s=4 . lần 6 :s=6 không có ý nghĩa
//4 :in ra mảng res thích hợp theo thứ tự tăng dần ( lấy dữ liệu từ mảng b)
#include<bits/stdc++.h>
using namespace std;
int V,E,s,t,check[1005],b[1005];
vector <int> a[1005];
void bfs(int u){//2
    queue <int> q;
    q.push(u);check[u]=1;
    while(q.empty() == 0){//3
        int s=q.front();q.pop();
        for(int i=0;i<a[s].size();i++){
            int t=a[s][i];
            if(check[t]==0){
                check[t]=1;
                q.push(t);
                b[t]=s;//b[dinh sau]=dinh truoc
            }
        }
    }
}

```

```

void initwsolve(){
    cin>>V>>E>>s>>t;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
bfs(s);
if(check[t]==0) cout<<-1<<endl;
/*4*/ else {
    vector <int> res;
    int i=t;cout<<s<<' ';
    while(i!=s){res.push_back(i); i=b[i];}
    for(int i=res.size()-1;i>=0;i--) cout<<res[i]<<' ';
    cout<<endl;
}
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA ĐƯỜNG ĐI

Ôn tập

Nội dung bài tập

Cho đồ thị vô hướng có N đỉnh và M cạnh. Có Q truy vấn, mỗi truy vấn yêu cầu trả lời câu hỏi giữa 2 đỉnh x và y có tồn tại đường đi tới nhau hay không?

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).
- Mỗi test gồm 2 số nguyên N, M ($1 \leq N, M \leq 1000$).
- M dòng tiếp theo, mỗi dòng gồm 2 số nguyên u, v cho biết có cạnh nối giữa đỉnh u và v.
- Dòng tiếp là số lượng truy vấn Q ($1 \leq Q \leq 1000$).
- Q dòng tiếp theo, mỗi dòng gồm 2 số nguyên x và y.

Output: Với mỗi truy vấn, in ra “YES” nếu có đường đi từ x tới y, in ra “NO” nếu ngược lại.

Ví dụ:

Input:	Output
1 5 5 1 2 2 3 3 4 1 4 5 6 2 1 5 2 4	NO YES

Giải

//1:tại từng bộ q ta tìm đường đi theo dfs với đồ thị vô hướng xuất phát từ đỉnh thứ nhất

//2:xét đường đi đó nếu đi qua đỉnh thứ 2 thì ok Q đúng in YES không thì in NO

```
#include<bits/stdc++.h>
using namespace std;
int v,e,u,U,check[1005];
vector <int> a[1005];
void dfs(int u){//xuất phát từ đỉnh u
    check[u]=1;
    for(int i=0;i<a[u].size();i++){
        int U=a[u][i];
        if(check[U]==0) {
            dfs(U);
        }
    }
}
void initwsolve(){
    cin>>v>>e;
    memset(a,0,sizeof(a));
/*init*/ for(int i=1;i<=e;i++){
    int u,v;cin>>u>>v;
```

```

        a[u].push_back(v);
        a[v].push_back(u);
    }
    int q; cin>>q;
/*1*/ for(int i=1;i<=q;i++){
    int u,v; cin>>u>>v;
    memset(check,0,sizeof(check));
    dfs(u); //1
/*2*/ if(check[v]==0) cout<<"NO" << endl;
    else cout<<"YES" << endl;
}
int main(){
    int t; cin>>t;
    while(t--) initwsolve();
}

```

ĐÉM SỐ THÀNH PHẦN LIÊN THÔNG (THEO DFS)

Nội dung bài tập

Cho đồ thị vô hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy tìm số thành phần liên thông của đồ thị..

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2;$

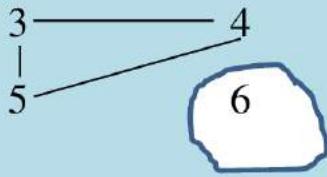
Output:

- Đưa ra số thành phần liên thông của đồ thị bằng thuật toán DFS.

Ví dụ:

Input:	Output:
1 6 6 1 2 1 3 2 3 3 4 3 5 4 5	2





Giải :

// note :Một đồ thị G được gọi là liên thông nếu có dây chuyền giữa mọi cặp đỉnh phân biệt đồ thị G.(tức là chúng nối được với nhau thành một đường)
 //2 đếm số thành phần liên thông bằng cách cho duyệt từng đỉnh , nếu như sau mỗi lần duyệt mà tại đỉnh đó chưa có trong các thành phần trước đó thì res++; tức là phải nằm ở thành phần liên thông khác.

//3: chỉ là tìm kiếm đường đi dfs

```
#include<bits/stdc++.h>
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];
vector<int> a[1005];
void dfs(int u){//3
    check[u]=1;
    for(int i=0;i<a[u].size();i++){
        int v=a[u][i];
        if(check[v]==0) dfs(v);
    }
}
void initwsolve(){
/*1init*/int res=0;
cin>>V>>E;
memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x].push_back(y);a[y].push_back(x);
}
/*2*/ for(int i=1;i<=V;i++){
    if(check[i]==0){
        dfs(i);res++;
    }
}
cout<<res<<endl;
```

```

}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

Cách 2 : ma trận kề
#include<bits/stdc++.h>
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];
int a[1005][1005];
void dfs(int u){
    check[u]= 1;
    for(int v=1;v<=V;v++)
        if(a[u][v]==1&&check[v]==0) dfs(v);
}
void initwsolve(){
/*1init*/int res=0;
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        a[x][y]=1;
        a[y][x]=1;
    }
/*2*/ for(int i=1;i<=V;i++){
    if(check[i]==0){
        dfs(i);res++;
    }
}
cout<<res<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

TÌM SỐ THÀNH PHẦN LIÊN THÔNG VỚI BFS

Nội dung bài tập

Cho đồ thị vô hướng $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Hãy tìm số thành phần liên thông của đồ thị bằng thuật toán BFS.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi u, v tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

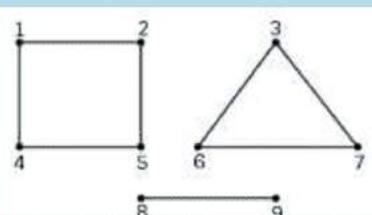
Output:

- Đưa ra số thành phần liên thông của đồ thị bằng thuật toán BFS.

Ví dụ:

Input:	Output:
1 6 6 1 2 1 3 2 3 3 4 3 5 4 5	2

Vd 1 test:



Số thành phần liên thông

Kết quả duyệt BFS

Giá trị trong mảng chuaxet[]

0	Chưa thực hiện	Chuaxet[] = {0,0,0,0,0,0,0,0,0}
1	BFS(1): 1,2,4,5	Chuaxet[] = {1,1,0,1,1,0,0,0,0}
2	BFS(3): 3,6,7	Chuaxet[] = {1,1,1,1,1,1,1,0,0}
3	BFS(8): 8,9	Chuaxet[] = {1,1,1,1,1,1,1,1,1}

Giải :

// note : Một đồ thị G được gọi là liên thông nếu có dây chuyền giữa mọi cặp đỉnh phân biệt đồ thị G . (tức là chúng nối được với nhau thành một đường)

```

//2 đếm số thành phần liên thông bằng cách cho duyệt từng đỉnh , nếu như sau mỗi
lần duyệt mà tại đỉnh đó chưa có trong các thành phần trước đó thì res++; tức là
phải nằm ở thành phần liên thông khác.
//3: chỉ là tìm kiếm đường đi bfs
#include<bits/stdc++.h>
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];
vector<int> a[1005];
void bfs(int u){//3
    queue <int> q;
    q.push(u);check[u]=1;
    while(q.empty()==0){
        int s=q.front();q.pop();
        for(int i=0;i<a[s].size();i++){
            int t=a[s][i];
            if(check[t]==0){
                check[t]=1;
                q.push(t);
            }
        }
    }
}
void initwsolve(){
/*1init*/int res=0;
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        a[x].push_back(y);a[y].push_back(x);
    }
/*2*/ for(int i=1;i<=V;i++){
    if(check[i]==0){//2
        bfs(i);res++;
    }
}
cout<<res<<endl;
}

```

```
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}
```

Cách 2: ma trận kề

```
#include<bits/stdc++.h>
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];
int a[1005][1005];
void bfs(int u){
    queue<int> q;
    check[u]=1;
    q.push(u);
    while(q.size()){
        int s=q.front();
        q.pop();
        for(int t=1;t<=V;t++){
            if(a[s][t]==1&&check[t]==0){
                check[t]=1;
                q.push(t);
            }
        }
    }
}
```

```
void initwsolve(){
/*1init*/int res=0;
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        a[x][y]=1;
        a[y][x]=1;
    }
}
```

```

/*2*/ for(int i=1;i<=V;i++){
    if(check[i]==0){
        bfs(i);res++;
    }
}
cout<<res<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA TÍNH LIÊN THÔNG MẠNH

Nội dung bài tập

Cho đồ thị có hướng $G =$ được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có liên thông mạnh hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi $u \hat{v}$, $v \hat{u}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

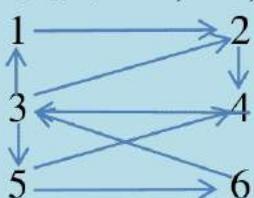
Output:

- Đưa ra “YES”, hoặc “NO” theo từng dòng tương ứng với test là liên thông mạnh hoặc không liên thông mạnh.

Ví dụ:

Input:	Output:
1 6 9 1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6 3	YES

Giải : 1 2 , 2 4 , 3 1, 3 2, 3 5 , 4 3 , 5 4 , 5 6 , 6 3.



```

Cách 1 : dama trận kè trr
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1002];
int a[1002][1002];
void dfs(int u){
    check[u]= 1;
    for(int v=1;v<=V;v++)
        if(a[u][v]==1&&check[v]==0) dfs(v);
}
void initwsolve(){
/*init*/cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        a[x][y]=1;
    }
/*2*/ for(int i=1;i<=V;i++){
    dfs(i);
    for(int j=1;j<=V;j++)
        if(check[j]==0){
            cout<<"NO"<<endl; return;
        }
    memset(check,0,sizeof(check));
} cout<<"YES"<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cách 2: ds kè

// note :Liên thông mạnh (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị. Xem thêm thành phần liên thông mạnh.

// 2: duyệt bắt đầu từ đỉnh một ,xem là sau khi tìm kiếm dfs thì tất cả các đỉnh có thuộc cùng 1 liên thông hay không , chỉ cần 1 đỉnh thuộc liên thông khác thì => liên thông yếu hoặc liên thông 1 phần

```

//3: chỉ là tìm kiếm đường đi dfs
#include<bits/stdc++.h>
using namespace std;
int v,e,u,res,check[1005];
vector <int> a[1005];
void dfs(int u){//3
    check[u]=1;
    for(int i=0;i<a[u].size();i++){
        int U=a[u][i];
        if(check[U]==0) dfs(U);
    }
}
void initwsolve(){
/*init*/cin>>v>>e;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=e;i++){
        int u,v;cin>>u>>v;
        a[u].push_back(v);
    }
/*2*/ for(int i=1;i<=v;i++){
    dfs(i);
    for(int j=1;j<=v;j++)
        if(check[j]==0){
            cout<<"NO"<<endl; return;
        }
    memset(check,0,sizeof(check));
} cout<<"YES"<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA TÍNH LIÊN THÔNG MẠNH VỚI BFS b buộc ds kề

Nội dung bài tập

Cho đồ thị có hướng $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán BFS, hãy kiểm tra xem đồ thị có liên thông mạnh hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|$, $|E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi $u \bar{v}$, $v \bar{v}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra “YES”, hoặc “NO” theo từng dòng tương ứng với test là liên thông mạnh hoặc không liên thông mạnh.

Ví dụ:

Input:	Output:
1 6 9 1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6 3	YES

Chú ý : nộp bài dfs vẫn đúng.

Giải :

// note :Liên thông mạnh (strongly connected): Đồ thị có hướng gọi là liên thông mạnh nếu có đường đi từ a tới b và từ b tới a với mọi cặp đỉnh a và b của đồ thị.
Xem thêm phần liên thông mạnh.

// 2: duyệt bắt đầu từ đỉnh một ,xem là sau khi tìm kiếm bfs thì tất cả các đỉnh có thuộc cùng 1 liên thông hay không , chỉ cần 1 đỉnh thuộc liên thông khác thì => liên thông yếu hoặc liên thông 1 phần

//3: chỉ là tìm kiếm đường đi bfs

```
#include<bits/stdc++.h>
using namespace std;
int v,e,u,res,check[1005];
vector <int> a[1005];
void bfs(int u){//3
    queue <int> q;
    q.push(u); check[u]=1;
    while(q.empty() == 0){
        int x=q.front(); q.pop();
        for(int i=0;i<a[x].size();i++){
            int U=a[x][i];
            if(check[U]==0){
                check[U]=1;
                q.push(U);
            }
        }
    }
}
```

```

        }
    }
}

void initwsolve(){
/*init*/cin>>v>>e;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=e;i++){
        int u,v;cin>>u>>v;
        a[u].push_back(v);
    }
/*2*/ for(int i=1;i<=v;i++){
    bfs(i);
    for(int j=1;j<=v;j++)
        if(check[j]==0){
            cout<<"NO"<<endl; return;
        }
    memset(check,0,sizeof(check));
} cout<<"YES"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

LIỆT KÊ ĐỈNH TRỤ (với dfs)

Nội dung bài tập

Cho đồ thị vô hướng liên thông G được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán DFS hoặc BFS, hãy đưa ra tất cả các đỉnh trụ của đồ thị?

Input:

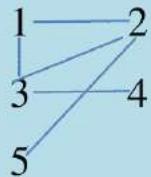
- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi $u \rightarrow v$, $v \rightarrow u$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra danh sách các đỉnh trụ của mỗi test theo từng dòng.

Ví dụ:

Input:	Output:
1 5 5 1 2 1 3 2 3 2 5 3 4	2 3



Giai :

// note: (Đỉnh trụ của đồ thị là đỉnh mà khi xóa nó đi thì số thành phần liên thông của đồ thị tăng lên)
 //2: Trước hết ta duyệt dfs 1 lần đồ thị để tìm ra số thành phần liên thông ban đầu .
 //3: Mỗi lần duyệt ta xóa 1 đỉnh của đồ thị. Dùng DFS duyệt tìm số thành phần liên thông của đồ thị. Nếu số thành phần liên thông của đồ thị tăng lên thì đỉnh bị xóa là đỉnh trụ.

//4: chỉ là tìm kiếm đường đi dfs

```

#include<bits/stdc++.h>
using namespace std;
int v,e,u,check[1005];
vector <int> a[1005];
void dfs(int u){//4
    check[u]=1;
    for(int i=0;i<a[u].size();i++){
        int U=a[u][i];
        if(check[U]==0) dfs(U);
    }
}
void initwsolve(){
/*init*/int inva=0,res=0;//invariable :cố định, bất biến
    cin>>v>>e;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=e;i++){
        int u,v;cin>>u>>v;
        a[u].push_back(v);
    }
}
    
```

```

        a[v].push_back(u);
    }
/*2*/ for(int i=1;i<=v;i++){
    if(check[i]==0){
        dfs(i);inva++;
    }
}
/*3*/ for(int i=1;i<=v;i++){
    memset(check,0,sizeof(check));
    check[i]=1;res=0;
    for(int j=1;j<=v;j++){
        if(check[j]==0){dfs(j);res++;}
    }
    if(res>inva) cout<<i<<' ';
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cách 2 : ma trận kề trr:

```

#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];//b: truoc s=> = b[t];
int a[1005][1005];
void dfs(int u){
    check[u]= 1;
    for(int v=1;v<=V;v++)
        if(a[u][v]==1&&check[v]==0) dfs(v);
}

```

```

void initwsolve(){
    cin>>V>>E;
    int inva=0,res=0;//invariable :cố định, bất biến
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){

```

```

int x,y;cin>>x>>y;
a[x][y]=1;
a[y][x]=1;
}
for(int i=1;i<=V;i++){
    if(check[i]==0){
        dfs(i);inva++;
    }
}
for(int i=1;i<=V;i++){
    memset(check,0,sizeof(check));
    check[i]=1;res=0;
    for(int j=1;j<=V;j++){
        if(check[j]==0){dfs(j);res++;}
    }
    if(res>inva) cout<<i<<' ';
}
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cách 3 : nhung chú ý bài toán coi như đồ thị này đã liên thông => chỉ cần so sánh >1 hay chưa vẫn đúng:

```

#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];//b: trước s=> = b[t];
int a[1005][1005];
void dfs(int u){
    check[u]= 1;
    for(int v=1;v<=V;v++)
        if(a[u][v]==1&&check[v]==0) dfs(v);
}

```

```

void initwsolve(){
    cin>>V>>E;
    int inva=0,res=0;//invariable :cố định, bắt biến

```

```

memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
for(int i=1;i<=V;i++){
    memset(check,0,sizeof(check));
    check[i]=1;res=0;
    for(int j=1;j<=V;j++){
        if(check[j]==0){dfs(j);res++;}
    }
    if(res>1) cout<<i<<' ';
}
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

LIỆT KÊ ĐỈNH TRỤ VỚI BFS

Nội dung bài tập

Cho đồ thị vô hướng liên thông $G=<V, E>$ được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán BFS, hãy đưa ra tất cả các đỉnh trụ của đồ thị?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi $u\bar{v}, v\bar{v}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra danh sách các đỉnh trụ của mỗi test theo từng dòng.

Ví dụ:

Input:	Output:
1 5 5	2 3

Giải :

Cách 1 : ma trận kề:

```
#include<bits/stdc++.h>
using namespace std;
int V,E,check[1005];//b: truoc s=> = b[t];
int a[1005][1005];
void bfs(int u){
    queue<int> q;
    check[u]=1;
    q.push(u);
    while(q.size()){
        int s=q.front();
        q.pop();
        for(int t=1;t<=V;t++){
            if(a[s][t]==1&&check[t]==0){
                check[t]=1;
                q.push(t);
            }
        }
    }
}
void initwsolve(){
    cin>>V>>E;
    int res=0;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
for(int i=1;i<=V;i++){
    memset(check,0,sizeof(check));
    check[i]=1;res=0;
    for(int j=1;j<=V;j++){
        if(a[i][j]==1&&check[j]==0)
            res++;
    }
}
cout<<res;
}
```

```

        if(check[j]==0){bfs(j);res++;}
    }
    if(res>1) cout<<i<<' ';
}
cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cách 2 : danh sách cạnh

// note: (Đỉnh trụ của đồ thị là đỉnh mà khi xóa nó đi thì số thành phần liên thông của đồ thị tăng lên)

//2: Trước hết ta duyệt dfs 1 lần đồ thị để tìm ra số thành phần liên thông ban đầu .
//3: Mỗi lần duyệt ta xóa 1 đỉnh của đồ thị. Dùng bFS duyệt tìm số thành phần liên thông của đồ thị. Nếu số thành phần liên thông của đồ thị tăng lên thì đỉnh bị xóa là đỉnh trụ.

//4: chỉ là đánh dấu tìm kiếm đường đi bfs

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

```
using namespace std;
```

```
int v,e,u,check[1005];
```

```
vector <int> a[1005];
```

```
void bfs(int u){//3
```

```
queue <int> q;
```

```
q.push(u); check[u]=1;
```

```
while(q.empty()==0){
```

```
    int x=q.front();q.pop();
```

```
    for(int i=0;i<a[x].size();i++){
```

```
        int U=a[x][i];
```

```
        if(check[U]==0){
```

```
            check[U]=1;
```

```
            q.push(U);
```

```
}
```

```
}
```

```
}
```

```
}
```

```
void initwsolve(){
```

```
/*init*/int inva=0,res=0;//invariable :cố định, bắt biến
```

```

cin>>v>>e;
memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
for(int i=1;i<=e;i++){
    int u,v;cin>>u>>v;
    a[u].push_back(v);
    a[v].push_back(u);
}
/*2*/ for(int i=1;i<=v;i++){
    if(check[i]==0){
        bfs(i);inva++;
    }
    memset(check,0,sizeof(check));
}
/*3*/ for(int i=1;i<=v;i++){
    check[i]=1;res=0;
    for(int j=1;j<=v;j++){
        if(check[j]==0){bfs(j);res++;}
    }
    if(res>inva) cout<<i<<' ';
    memset(check,0,sizeof(check));
}
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

LIỆT KÊ CẠNH CẦU

Ôn lại

Cho đồ thị vô hướng liên thông G được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán DFS hoặc BFS, hãy đưa ra tất cả các cạnh cầu của đồ thị?

Input:

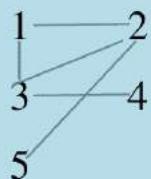
- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và số cạnh; Dòng tiếp theo đưa vào các bộ đôi uIV, vIV tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra danh sách các cách cầu của mỗi test theo từng dòng. In ra đáp án theo thứ tự từ điển, theo dạng “a b ...” với $a < b$.

Ví dụ:

Input:	Output:
1 5 5 1 2 1 3 2 3 2 5 3 4	2 5 3 4



Giải :

Cách 1 : ma trận kề:TLE

```

#include<bits/stdc++.h>
#include<bits/stdc++.h>
#define ii pair<int,int>
using namespace std;
int V,E,check[1001];
int a[1001][1001];
vector<pair<int,int> >canh;
void dfs(int u){
    check[u]= 1;
    for(int v=1;v<=V;v++)
        if(a[u][v]==1&&check[v]==0) dfs(v);
}
void initwsolve(){
    cin>>V>>E;
    canh.clear();
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        a[x][y]=1;
        a[y][x]=1;
    }
}
  
```

```

        canh.push_back(ii(x,y) );
    }
    for(int i=0;i<E;i++){
        a[canh[i].first][canh[i].second]=0;
        int res=0;
        memset(check,0,sizeof(check));
        for(int k=1;k<=V;k++){
            if(check[k]==0)res++;dfs(k);
        }
        if(res>1)
            cout<<canh[i].first<<' '<<canh[i].second<<' ';
        a[canh[i].first][canh[i].second]=1;
    }
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

Cách 2 :danh sách kè:

// (cạnh cầu của đồ thị là khi xóa cạnh đó đi thì số thành phần liên thông của đồ thị tăng lên)
// 1: Trước hết ta duyệt DFS 1 lần đồ thị để tìm ra số thành phần liên thông ban đầu.
// 2: Ta xóa lần lượt các cạnh của đồ thị, sau đó dùng DFS duyệt tìm số thành phần liên thông của đồ thị. Nếu số thành phần liên thông tăng chứng tỏ cạnh bị xóa là cạnh cầu.
// 3: chỉ là duyệt đường đi theo dfs.

// (cạnh cầu của đồ thị là khi xóa cạnh đó đi thì số thành phần liên thông của đồ thị tăng lên)

```

#include<bits/stdc++.h>
#include<bits/stdc++.h>
#define ii pair<int,int>
using namespace std;
int V,E,check[1001];
vector<int > a[1005];
vector<pair<int,int> >canh;

```

```

void init(int x){
    memset(check,0,sizeof(check));
    for(int i=1;i<=V;i++)a[i].clear();
    for(int i=0;i<canh.size();i++){
        if(i!=x){
            a[canh[i].first].push_back(canh[i].second);
            a[canh[i].second].push_back(canh[i].first);
        }
    }
}
void dfs(int u){
    check[u]=1;
    for(int i=0;i<a[u].size();i++){
        int v=a[u][i];
        if(check[v]==0) dfs(v);
    }
}
void initwsolve(){
    cin>>V>>E;
    canh.clear();
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    for(int i=1;i<=E;i++){
        int x,y;cin>>x>>y;
        canh.push_back(ii(x,y));
    }
    for(int i=0;i<E;i++){
        init(i);
        int res=0;
        memset(check,0,sizeof(check));
        for(int k=1;k<=V;k++){
            if(check[k]==0)res++;dfs(k);
        }
        if(res>1)
            cout<<canh[i].first<<' '<<canh[i].second<<' ';
    }
    cout<<endl;
}

```

```

}
int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA CHU TRÌNH

Ôn lại

Cho đồ thị vô hướng $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi $u \in V, v \in V$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2;$

Output:

- Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

Ví dụ:

Input:	Output:
1 6 9 1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	YES

Giải:

// theo như hình vẽ thì chu trình có 1 2 3 4 5 6.

// khi đó par là 5 nên nếu tìm được 1 đỉnh v nào đó khác 5 thì tức là được 1 chu trình

// => chu trình chính là 1 2 3 4 5 6 4.

// chuyển sang chu trình tìm từ đỉnh v nhưng par lúc này là đỉnh u trước đó

#include<bits/stdc++.h>

using namespace std;

int V,E,u,check[1005],cycle=0;

int a[1005][1005];

void dfs(int u,int par){

check[u]=1;

for(int v=1;v<=V;v++){

```

        if(a[u][v]==1){
            if(check[v]==0)(dfs(v,u));//
            else if(check[v]==1&&v!=par)cycle=1;
        }
    }
void initwsolve(){
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    cycle=0;
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
for(int i=1;i<=V;i++){
    if(check[i]==0) dfs(i,0);
}
if(cycle==1)cout<<"YES"<<endl;
else cout<<"NO"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA CHU TRÌNH VỚI BFS

Cho đồ thị vô hướng $G=<V, E>$ được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán BFS, hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

Input:

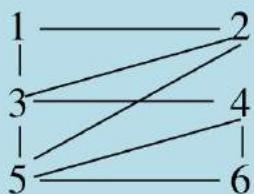
- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|$, $|E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi $u \hat{V}, v \hat{V}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

Ví dụ:

Input:	Output:
1 6 9 1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	YES



Giai:

// theo như hình vẽ thì chu trình có 1 2 3 4 5 6.

// khi đó par là 5 nên nếu tìm được 1 đỉnh v nào đó khác 5 thì tức là được 1 chu trình

// => chu trình chính là 1 2 3 4 5 6 4.

// chuyển sang chu trình tìm từ đỉnh v nhưng par lúc này là đỉnh u trước đó

#include<bits/stdc++.h>

using namespace std;

int V,E,u,check[1005],cycle=0;

int a[1005][1005];

void dfs(int u,int par){

 check[u]=1;

 for(int v=1;v<=V;v++){

 if(a[u][v]==1){

 if(check[v]==0)(dfs(v,u));//

 else if(check[v]==1&&v!=par)cycle=1;

 }

 }

}

void initwsolve(){

 cin>>V>>E;

 memset(a,0,sizeof(a));

 memset(check,0,sizeof(check));

 cycle=0;

 for(int i=1;i<=E;i++){

```

int x,y;cin>>x>>y;
a[x][y]=1;
a[y][x]=1;
}
for(int i=1;i<=V;i++){
    if(check[i]==0) dfs(i,0);
}
if(cycle==1)cout<<"YES"<<endl;
else cout<<"NO"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA CHU TRÌNH SỬ DỤNG DISJOIN SET

Cho đồ thị vô hướng $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Sử dụng Disjoin Set, hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uV, vV tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

Ví dụ:

Input:	Output:
1 6 9 1 2 1 3 2 3 2 5 3 4 3 5 4 5 4 6 5 6	YES

Giải : up code kiểm tra chu trình dfs cũng đúng:

```

#include<bits/stdc++.h>
using namespace std;
int V,E,u,check[1005],cycle=0;

```

```

int a[1005][1005];
void dfs(int u,int par){
    check[u]= 1;
    for(int v=1;v<=V;v++){
        if(a[u][v]==1){
            if(check[v]==0)(dfs(v,u));// 
            else if(check[v]==1&&v!=par)cycle=1;
        }
    }
}
void initwsolve(){
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    cycle=0;
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
for(int i=1;i<=V;i++){
    if(check[i]==0) dfs(i,0);
}
if(cycle==1)cout<<"YES"<<endl;
else cout<<"NO"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

KIỂM TRA CHU TRÌNH TRÊN ĐỒ THỊ CÓ HƯỚNG VỚI DFS

Ôn lại

Cho đồ thị có hướng $G=<V, E>$ được biểu diễn dưới dạng danh sách cạnh. Sử dụng thuật toán DFS, hãy kiểm tra xem đồ thị có tồn tại chu trình hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.

- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|$, $|E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi $u \hat{V}, v \hat{V}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra YES hoặc “NO” kết quả test theo từng dòng tương ứng với đồ thị tồn tại hoặc không tồn tại chu trình.

Ví dụ:

Input:	Output:
1 6 9 1 2 2 4 3 1 3 2 3 5 4 3 5 4 5 6 6 4	YES

Giải :

//1: xác nhận đỉnh u đã được thăm trong quá trình duyệt cây gốc u
//2: tại các đỉnh v kề với đỉnh u thì nếu v chưa xét thì tiếp tục tìm đường bắt đầu từ v

//3: nếu gặp lại 1 đỉnh v trong qua trình duyệt cây dfs gốc u rồi, thì đồ thị này có chu trình => cycle=1;

//4 : kết thúc quá trình duyệt cây , tất cả các đỉnh đều cập nhật check[i]=2

```
#include<bits/stdc++.h>
using namespace std;
int V,E,u,check[1005],cycle=0;
int a[1005][1005];
void dfs(int u){
    check[u]=1;//1
    for(int v=1;v<=V;v++){
        if(a[u][v]==1){
            if(check[v]==0) dfs(v);//2
            else if(check[v]==1) cycle=1;//3
        }
    }
    check[u]=2;
}
```

```
void initwsolve(){
    cin>>V>>E;
    memset(a,0,sizeof(a));
    memset(check,0,sizeof(check));
    cycle=0;
```

```

/* */ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
}
for(int i=1;i<=V;i++){
    if(check[i]==0)dfs(i);
}
if(cycle==1)cout<<"YES"<<endl;
else cout<<"NO"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}

```

ĐƯỜNG ĐI VÀ CHU TRÌNH EULER VỚI ĐỒ THỊ VÔ HƯỚNG

Ôn lại

Cho đồ thị vô hướng liên thông $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có đường đi Euler hay chu trình Euler hay không?

Đường đi Euler bắt đầu tại một đỉnh, và kết thúc tại một đỉnh khác.

Chu trình Euler bắt đầu tại một đỉnh, và kết thúc chu trình tại chính đỉnh đó.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi uV, vV tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100; 1 \leq |V| \leq 10^3; 1 \leq |E| \leq |V|(|V|-1)/2;$

Output:

- Đưa ra 1, 2, 0 kết quả mỗi test theo từng dòng tương ứng với đồ thị có đường đi Euler, chu trình Euler và trường hợp không tồn tại.

Ví dụ:

Input:	Output:
2 6 10 1 2 1 3 2 3 2 4 2 5 3 4 3 5 4 5 4 6 5 6 6 9	2 1

1 2 1 3 2 3 2 4 2 5 3 4 3 5 4 5 4 6	
-------------------------------------	--

Giai :

//chú ý : chu trình euler khi tất cả các bậc của các đỉnh là bậc chẵn (ko có bậc lẻ nào). Đường đi euler khi có 2 đỉnh bậc lẻ . còn lại thì ko có ctrinh hoặc đường đi euler

```
#include<bits/stdc++.h>
using namespace std;
vector< vector<int> > a;
int V,E,x,y;
void initwsolve(){
    a.clear();
    cin>>V>>E;
    a.resize(1005);
    for(int i=1;i<=E;i++){
        cin>>x>>y;
        a[x].push_back(y);
        a[y].push_back(x);
    }
    int odd=0;
    for(int i=1;i<=V;i++)
        if(a[i].size()%2!=0) odd++;
    if(odd==0) cout<<"2\n";
    else if(odd==2) cout<<"1\n";
    else cout<<"0\n";
}
int main(){
    int t; cin>>t;
    while(t--)initwsolve();
    return 0;
}
```

CHU TRÌNH EULER TRONG ĐỒ THỊ CÓ HƯỚNG

Cho đồ thị có hướng liên thông yếu $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh. Hãy kiểm tra xem đồ thị có chu trình Euler hay không?

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.

- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào hai số $|V|$, $|E|$ tương ứng với số đỉnh, số cạnh của đồ thị; Dòng tiếp theo đưa vào các bộ đôi $u \hat{V}, v \hat{V}$ tương ứng với một cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra 1, 0 kết quả mỗi test theo từng dòng tương ứng với đồ thị có chu trình Euler và trường hợp không tồn tại đáp án.

Ví dụ:

Input:	Output:
2 6 10 1 2 2 4 2 5 3 1 3 2 4 3 4 5 5 3 5 6 6 4 3 3 1 2 2 3 1 3	1 0

Giai:

```
#include<bits/stdc++.h>
using namespace std;
vector< vector<int> > a,b;
int V,E,x,y;
void initwsolve(){
    a.clear(),b.clear();
    cin>>V>>E;
    a.resize(1005);b.resize(1005);
    for(int i=1;i<=E;i++){
        cin>>x>>y;
        a[x].push_back(y);//số bán bắc ra đỉnh x
        b[y].push_back(x);//số bán bắc vào đỉnh y
    }
    for(int i=1;i<=V;i++)
        if(a[i].size()!=b[i].size()){
            cout<<"0\n";return;
        }
    cout<<1<<endl;
}

int main(){
    int t; cin>>t;
    while(t--)initwsolve();
```

```

        return 0;
    }
}

```

KIỂM TRA ĐỒ THỊ CÓ PHẢI LÀ CÂY HAY KHÔNG

Ôn lại

Một đồ thị N đỉnh là một cây, nếu như nó có đúng $N-1$ cạnh và giữa 2 đỉnh bất kỳ, chỉ tồn tại duy nhất 1 đường đi giữa chúng.

Cho một đồ thị N đỉnh và $N-1$ cạnh, hãy kiểm tra đồ thị đã cho có phải là một cây hay không?

Input:

- Dòng đầu tiên là số lượng bộ test T ($T \leq 20$).
- Mỗi test bắt đầu bởi số nguyên N ($1 \leq N \leq 1000$).
- $N-1$ dòng tiếp theo, mỗi dòng gồm 2 số nguyên u, v cho biết có cạnh nối giữa đỉnh u và v.

Output:

- Với mỗi test, in ra “YES” nếu đồ thị đã cho là một cây, in ra “NO” trong trường hợp ngược lại.

Ví dụ:

Input	Output
2	
4	
1 2	
1 3	
2 4	
4	
1 2	
1 3	
2 3	

Giải :

Cách 1:

```

#include<bits/stdc++.h>
using namespace std;
int V,E,u,check[1005],cycle=0;
int a[1005][1005];
void dfs(int u,int par){
    check[u]= 1;
    for(int v=1;v<=V;v++){
        if(a[u][v]==1){
            if(check[v]==0)(dfs(v,u));//
            else if(check[v]==1&&v!=par)cycle=1;
        }
    }
}
void initwsolve(){
    cin>>V;
    for(int i=1;i<=V;i++)
        check[i]=0;
    for(int i=1;i<=E;i++){
        int u,v;
        cin>>u>>v;
        a[u][v]=1;
        a[v][u]=1;
    }
}

```

```

int E=V-1;
memset(a,0,sizeof(a));
memset(check,0,sizeof(check));
cycle=0;
/*1*/ for(int i=1;i<=E;i++){
    int x,y;cin>>x>>y;
    a[x][y]=1;
    a[y][x]=1;
}
dfs(1,0);
for(int i=1;i<=V;i++)
    if(check[i]==0) cycle=1;
if(cycle==1)cout<<"NO"<<endl;
else cout<<"YES"<<endl;
}

int main(){
    int t;cin>>t;
    while(t--) initwsolve();
}


```

Cách 2:

```

#include<bits/stdc++.h>
using namespace std;
bool check[1005];
vector< vector<int> > a;
int V,E, x,y;
bool BFS(int u){
    vector<int> parent(1005, -1);
    queue<int> q;
    q.push(u);
    check[u]=true;
    while(!q.empty()){
        int s=q.front();
        q.pop();
        for(int i=0;i<a[s].size();i++){
            int t=a[s][i];
            if(parent[t]==-1){
                parent[t]=s;
                if(t==u) return true;
                check[t]=true;
                q.push(t);
            }
        }
    }
    return false;
}

```

```

        if(check[t]==0){
            check[t]=1;
            q.push(t);
            parent[t]=s;
        }
        else if(parent[s]!=t) return 1;
    }
    return 0;
}
bool initwsolve(){
    a.clear();
    cin>>V;
    a.resize(1005);
    memset(check,0,sizeof(check));
    for(int i=1;i<=V-1;i++){
        cin>>x>>y;
        a[x].push_back(y);
        a[y].push_back(x);
    }
    if(BFS(1)) return 0;
    for(int i=1;i<=V;i++)
    if(!check[i]) return 0;
    return 1;
}
int main(){
    int t; cin>>t;
    while(t--)
        if(initwsolve()) cout<<"YES"<<endl;
        else cout<<"NO"<<endl;
    return 0;
}

```

CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN DFS

Ôn lại

Cho đồ thị vô hướng $G=(V, E)$. Hãy xây dựng một cây khung của đồ thị G với đỉnh $u \in V$ là gốc của cây bằng thuật toán DFS.

Input

Dòng đầu tiên gồm một số nguyên T ($1 \leq T \leq 20$) là số lượng bộ test.

Tiếp theo là T bộ test, mỗi bộ test có dạng sau:

- Dòng đầu tiên gồm 3 số nguyên $N=|V|$, $M=|E|$, u ($1 \leq N \leq 10^3$, $1 \leq M \leq 10^5$, $1 \leq u \leq N$).
- M dòng tiếp theo, mỗi dòng gồm 2 số nguyên a , b ($1 \leq a, b \leq N$, $a \neq b$) tương ứng cạnh nối hai chiềutừ a tới b .
- Dữ liệu đảm bảo giữa hai đỉnh chỉ tồn tại nhiều nhất một cạnh nối.

Output

Với mỗi bộ test, nếu tồn tại cây khung thì in ra $N - 1$ cạnh của cây khung với gốc là đỉnh u trên $N - 1$ dòng theo thứ tự duyệt của thuật toán DFS. Ngược lại nếu không tồn tại cây khung thì in ra -1 .

Ví dụ

Input	Output
2	
4 3 2	
1 2	2 1
1 3	1 3
2 4	3 4
3 4	-1
4 2 2	
1 2	
3 4	

Giải :

```
#include <bits/stdc++.h>
#define p pair<int,int>
using namespace std;
vector <p> a[1005];
int v,e,s;
bool check[1005];
vector<p> st;

void DFS(int u){
    check[u] = 1;
    for(int i=0 ; i<a[u].size();i++){
        int v = a[u][i];
        if(!check[v]){
            st.push_back(p(u,v));
            DFS(v);
        }
    }
}
void initwsolve(){
    memset(check,0,sizeof(check));
    st.clear();
    cin>>v>>e>>s;
```

```

for(int i=0 ; i<1000 ; i++) a[i].clear();
for(int i=0 ; i<e ; i++){
    int x,y; cin>>x>>y;
    a[x].push_back(y);
    a[y].push_back(x);
}
DFS(s);
if(st.size() != v-1) cout<<"-1"<<endl;
else
    for(int i=0;i<st.size() ; i++)
        cout<<st[i].first<< " "<<st[i].second<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}

```

CÂY KHUNG CỦA ĐỒ THỊ THEO THUẬT TOÁN BFS

Ôn lại

Cho đồ thị vô hướng $G=(V, E)$. Hãy xây dựng một cây khung của đồ thị G với đỉnh $u \in V$ là gốc của cây bằng thuật toán BFS.

Input

Dòng đầu tiên gồm một số nguyên T ($1 \leq T \leq 20$) là số lượng bộ test.

Tiếp theo là T bộ test, mỗi bộ test có dạng sau:

- Dòng đầu tiên gồm 3 số nguyên $N=|V|$, $M=|E|$, u ($1 \leq N \leq 10^3$, $1 \leq M \leq 10^5$, $1 \leq u \leq N$).
- M dòng tiếp theo, mỗi dòng gồm 2 số nguyên a , b ($1 \leq a, b \leq N$, $a \neq b$) tương ứng cạnh nối hai chiêu từ a tới b .
- Dữ liệu đảm bảo giữa hai đỉnh chỉ tồn tại nhiều nhất một cạnh nối.

Output

Với mỗi bộ test, nếu tồn tại cây khung thì in ra $N - 1$ cạnh của cây khung với gốc là đỉnh u trên $N - 1$ dòng theo thứ tự duyệt của thuật toán BFS. Ngược lại nếu không tồn tại cây khung thì in ra -1 .

Ví dụ

Input	Output
2	
4 4 2	
1 2	2 1
1 3	2 4
2 4	1 3
3 4	-1
4 2 2	
1 2	

Giai :

```
#include <bits/stdc++.h>
#define p pair<int,int>
using namespace std;
vector <int> a[1005];
int v,e,s;
bool check[1005];
vector<p> st;

void BFS(int u){
    queue <int> q;
    q.push(u);
    check[u] = 1;
    while(!q.empty()){
        int s = q.front(); q.pop();
        for(int i=0 ; i<a[s].size();i++){
            int t = a[s][i];
            if(!check[t]){
                check[t] = 1;
                q.push(t);
                st.push_back(p(s,t));
            }
        }
    }
}

void initwsolve(){
    memset(check,0,sizeof(check));
    st.clear();
    cin>>v>>e>>s;
    for(int i=0 ; i<1000 ; i++) a[i].clear();
    for(int i=0 ; i<e ; i++){
        int x,y; cin>>x>>y;
        a[x].push_back(y);
        a[y].push_back(x);
    }
    BFS(s);
    if(st.size() != v-1) cout<<"-1"<<endl;
}
```

```

else
    for(int i=0;i<st.size() ; i++)
        cout<<st[i].first<<" "<<st[i].second<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}

}

```

ĐƯỜNG ĐI HAMILTON

Ôn lại

Đường đi đơn trên đồ thị có hướng hoặc vô hướng đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Cho đồ thị vô hướng $G = \langle V, E \rangle$, hãy kiểm tra xem đồ thị có đường đi Hamilton hay không?

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào hai số V, E tương ứng với số đỉnh, số cạnh của đồ thị; phần thứ hai đưa vào các cạnh của đồ thị.
- T, V, E thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq V \leq 10$; $1 \leq E \leq 15$.

Output:

- Đưa ra 1 hoặc 0 tương ứng với test có hoặc không có đường đi Hamilton theo từng dòng.

Ví dụ:

Input	Output
2 4 4 1 2 2 3 3 4 2 4 4 3 1 2 2 3 2 4	1 0

Giải :

```

#include<bits/stdc++.h>
using namespace std;
int n, m;
vector<int> a[1000];
bool check[100];
bool DFS(int i, int dem){
    if(dem == n) return 1;
    check[i] = 1;
    for(int j = 0; j < a[i].size(); j++){
        if(!check[a[i][j]] && DFS(a[i][j], dem + 1)) return 1;
    }
    check[i] = 0;
}

```

```

        return 0;
    }
void initwsolve(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++) a[i].clear();
    for(int i = 0; i < m; i++){
        int x, y;cin >> x >> y;
        a[x].push_back(y);a[y].push_back(x);
    }
    memset(check,false,sizeof(check));
    int stop = 0;
    for(int i = 1; i <= n; i++){
        if(DFS(i,1)){
            stop = 1;break;
        }
    }
    if(stop == 1)cout << "1\n";
    else cout << "0\n";
}
int main(){
    int t;
    cin >> t;
    while(t--)initwsolve();
}

```

DIJKSTRA

Ôn lại

Cho đồ thị có trọng số không âm $G = \langle V, E \rangle$ được biểu diễn dưới dạng danh sách cạnh trọng số. Hãy viết chương trình tìm đường đi ngắn nhất từ đỉnh $u \in V$ đến tất cả các đỉnh còn lại trên đồ thị.

Input:

- Dòng đầu tiên đưa vào T là số lượng bộ test.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm $|E|+1$ dòng: dòng đầu tiên đưa vào hai số $|V|, |E|$ tương ứng với số đỉnh và $u \in V$ là đỉnh bắt đầu; $|E|$ dòng tiếp theo mỗi dòng đưa vào bộ ba $u \in V, v \in V, w$ tương ứng với một cạnh cùng với trọng số cạnh của đồ thị.
- $T, |V|, |E|$ thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq |V| \leq 10^3$; $1 \leq |E| \leq |V|(|V|-1)/2$;

Output:

- Đưa ra kết quả của mỗi test theo từng dòng. Kết quả mỗi test là trọng số đường đi ngắn nhất từ đỉnh u đến các đỉnh còn lại của đồ thị theo thứ tự tăng dần các đỉnh.

Ví dụ:

Input:	Output:
1 9 12 1 1 2 4 1 8 8 2 3 8 2 8 11 3 4 7 3 6 4 3 9 2 4 5 9 4 6 14 5 6 10 6 7 2 6 9 6	0 4 12 19 21 11 9 8 14

Giải :

```
#include<bits/stdc++.h>
using namespace std;
#define p pair<int,int>
#define INF 1e9
vector< vector<p>> a;
int V,E,x;
void initwsolve(){
    a.clear();
    cin>>V>>E>>x;
    a.resize(1005);

    for(int i=1;i<=E;i++){
        int x,y,k;
        cin>>x>>y>>k;
        a[x].push_back(p(y,k));
        a[y].push_back(p(x,k));
    }
    priority_queue< p,vector<p>,greater<p>> pq;
    vector<int> dist(1005,INF);
    pq.push(p(0,x));
    dist[x]=0;
    while(!pq.empty()){
        int u=pq.top().second;
        pq.pop();
        for(int i=0;i<a[u].size();i++){
            int v=a[u][i].first;
            int w=a[u][i].second;
            if(dist[v]>dist[u]+w){
                dist[v]=dist[u]+w;
                pq.push(p(dist[v],v));
            }
        }
    }
}
```

```

        for(int i=0;i<a[u].size();i++){
            int v=a[u][i].first;
            int doDai=a[u][i].second;

            if(dist[v]>dist[u]+doDai){
                dist[v]=dist[u]+doDai;
                pq.push(p(dist[v],v));
            }
        }
    }

    for(int i=1;i<=V;i++) cout<<dist[i]<<" ";
    cout<<endl;
}

int main(){
    int t;
    cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

KRUSKAL

Ôn lại

Cho đồ thị vô hướng có trọng số $G = \langle V, E, W \rangle$. Nhiệm vụ của bạn là hãy xây dựng một cây khung nhỏ nhất của đồ thị bằng thuật toán Kruskal.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng kế tiếp đưa vào các bộ test. Mỗi bộ test gồm hai phần: phần thứ nhất đưa vào hai số V, E tương ứng với số đỉnh và số cạnh của đồ thị; phần thứ 2 đưa vào E cạnh của đồ thị, mỗi cạnh là một bộ 3: đỉnh đầu, đỉnh cuối và trọng số của cạnh.
- T, S, D thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq V \leq 100$; $1 \leq E, W \leq 10000$.

Output:

- Dưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2 3 3 1 2 5 2 3 3 1 3 1	4 5

2	1		
1	2	5	

Giải:

```
#include <bits/stdc++.h>
using namespace std;
struct edge {
    int fir, en, var;
};
int v,e;
vector<edge> vv;
int p[1005];
int find(int x){
    if(p[x] == -1) return x;
    return find(p[x]);
}
void union1(int x, int y){
    p[x] = y;
}
bool cycle(vector<edge> ve){
    memset(p,-1,sizeof(p));
    for(int i=0 ; i<ve.size() ;i++){
        int x = find(ve[i].fir);
        int y = find(ve[i].en);
        if(x == y){
            return 1;
        }
        union1(x, y);
    }
    return 0;
}
bool cmp(edge e, edge ee){
    return e.var < ee.var;
}
void initwsolve(){
    cin>>v>>e;
    vv.clear();
    for (int i=0; i<e; i++){
        edge e;
        cin>>e.fir>>e.en>>e.var;
        vv.push_back(e);
    }
}
```

```

        edge tt;
        cin>>tt.fir>>tt.en>>tt.var;
        vv.push_back(tt);
    }
vector <edge> vvv;
sort(vv.begin() , vv.end() ,cmp);
vvv.push_back(vv[0]);
int sum=vv[0].var;

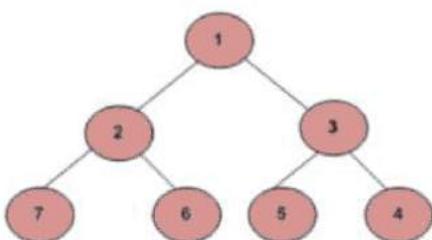
for(int i=1 ; i<vv.size() ; i++){
    vvv.push_back(vv[i]);
    if(cycle(vvv)){
        vvv.pop_back();
    }
    else {
        sum += vv[i].var;
    }
}
cout<<sum<<endl;
}
int main (){
    int t;cin >> t;
    while(t--)initwsolve();
}

```

Cây nhị phân

KIỂM TRA NODE LÁ

Cho cây nhị phân, nhiệm vụ của bạn là kiểm tra xem tất cả các node lá của cây có cùng một mức hay không? Ví dụ với cây dưới đây sẽ cho ta kết quả là Yes.



Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng cạnh của cây; dòng tiếp theo đưa vào N bộ ba (u, v, x), trong đó u là node cha, v là node con, x= R nếu v là con phải, x=L nếu v là con trái; u, v, x được viết cách nhau một vài khoảng trắng.
- T, N, u, v, thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq u, v \leq 10^4$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
2	
1 2 R 1 3 L	1
4	0
10 20 L 10 30 R 20 40 L 20 60 R	

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int n;
struct Tree{
    int value;
    Tree *left,*right;
};
Tree *newNode(int val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree* buildTree(){
    map<int,Tree*> mp;
    Tree *root = NULL;
    while(n--){
        int a,b;
        char x;
        cin>>a>>b>>x;
        Tree *cha = newNode(a);
        if(mp.find(a)==mp.end()){
            mp[a]=cha;
        }
        if(x=='L'){
            cha->left=&mp[b];
        }
        else{
            cha->right=&mp[b];
        }
    }
    return root;
}
```

```

        mp[a]=cha;
        if(root==NULL) root=cha;
    }
    else cha=mp[a];
    Tree *con=newNode(b);
    if(x=='L') cha->left=con;
    else if(x=='R')cha->right=con;
    mp[b]=con;
}
return root;
}
bool checkLevelLeaf(Tree* root){
if (root==NULL)
    return true;
queue<Tree*> q;
q.push(root);
int result = INT_MAX;
int level = 0;
while (!q.empty()) {
    int size = q.size();
    level += 1;
    while(size--){
        Tree* temp = q.front();
        q.pop();
        if (temp->left!=NULL) {
            q.push(temp->left);
            if(temp->left->right==NULL && temp->left->left==NULL){

                if (result == INT_MAX)
                    result = level;

                else if (result != level)
                    return false;
            }
        }
        if (temp->right!=NULL){
            q.push(temp->right);
            if (temp->right->left==NULL && temp->right->right==NULL) {

```

```

        if (result == INT_MAX)
            result = level;
        else if(result != level)
            return false;
    }
}
}
return true;
}
void initwsolve(){
    cin>>n;
    Tree *root=buildTree();
    if(checkLevelLeaf(root)) cout<<1<<endl;
    else cout<<0<<endl;
}
int main(){
    int t;
    cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

CÂY BIỂU THỨC 1

Cây biểu thức là một cây nhị phân trong đó mỗi node trung gian là một phép toán, mỗi node lá là một toán hạng. Ví dụ với biểu thức $P = 3 + ((5+9)*2)$ sẽ được biểu diễn như cây dưới đây. Đối với cây biểu thức, duyệt theo thứ tự trước ta sẽ được biểu thức trung tố, duyệt theo thứ tự sau ta sẽ được biểu thức hậu tố, duyệt theo thứ tự giữa ta được biểu thức tiền tố. Chú ý, cây biểu thức luôn là cây nhị phân đầy (mỗi node trung gian đều có hai node con).

Cho biểu thức hậu tố P, hãy sử dụng cây biểu thức để đưa ra biểu thức trung tố tương ứng với biểu thức P.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là một biểu thức hậu tố P.
- T, P thỏa mãn ràng buộc : $1 \leq T \leq 100$; $1 \leq \text{length}(P) \leq 100$.

Output:

- Đưa ra biểu thức trung tố tương ứng với P.

Ví dụ:

Input	Output
2 ab+ef*g*- wlrb+-*	a + b - e * f * g w * l - r + b

Giải:

```
#include<bits/stdc++.h>
using namespace std;
struct Tree{
    char value;
    Tree *left,*right;
};
bool isOp(char c){
    if(c=='+' || c=='-' || c=='*' || c=='/') return true;
    return false;
}
Tree *newNode(char val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
void LNR(Tree *t){
    if(t!=NULL){
        LNR(t->left);
        cout<<t->value;
        LNR(t->right);
    }
}
void initwsolve(){
    string s; cin>>s;
    stack<Tree *> st;
    Tree *t,*t1,*t2;
    for(int i=0;i<s.length();i++){
        if(!isOp(s[i])){
            t=newNode(s[i]);
            st.push(t);
        }
        else{
            t2=st.top();
            st.pop();
            t1=st.top();
            st.pop();
            t=newTree(t1,t2);
            st.push(t);
        }
    }
}
```

```

        t=newNode(s[i]);
        st.push(t);
    }
    else{
        t=newNode(s[i]);
        if(!st.empty()){
            t1=st.top();
            st.pop();
        }
        if(!st.empty()){
            t2=st.top();
            st.pop();
        }
        t->left=t2;
        t->right=t1;
        st.push(t);
    }
}
if(!st.empty()) t=st.top();
LNR(t);
cout<<endl;
}

int main(){
    int t;
    cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

CÂY BIỂU THỨC 2

Cho một cây biểu thức là một cây nhị phân đầy đủ bao gồm các phép toán +, -, *, / và một số toán hạng có giá trị nguyên. Nhiệm vụ của bạn là hãy tính toán giá trị biểu thức được biểu diễn trên cây nhị phân đầy đủ. Ví dụ với cây dưới đây là biểu diễn của biểu thức $P = ((5*4) + (100-20))$ sẽ cho ta giá trị là 100.

Input:

- Dòng đầu tiên đưa vào số lượng bộ test T.

- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test là gồm hai dòng: dòng thứ nhất đưa vào N là số lượng node của cây; dòng thứ hai đưa vào nội dung các node của cây; các node được viết cách nhau một vài khoảng trắng. Các số có giá trị nguyên không vượt quá 1000.
- T, N, P thỏa mãn ràng buộc : $1 \leq T \leq 100$; $1 \leq N$, $\text{length}(P) \leq 100$.

Output:

- Đưa ra giá trị của cây biểu thức.

Ví dụ:

Input	Output
2 7 + * - 5 4 100 20 3 - 4 7	100 -3

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int n;
string s[1005];
struct Tree{
    string value;
    Tree *left,*right;
};
int stringToNum(string a){
    int num=0;
    for(int i=0;i<a.length();i++) num=num*10+(a[i]-'0');
    return num;
}
Tree *newNode(string val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree *insertLevel(Tree* root,int i){
    if (i < n){
        Tree *temp = newNode(s[i]);
        root = temp;
    }
}
```

```

root->left = insertLevel(root->left,2*i+1);
root->right = insertLevel(root->right,2*i+2);
}
return root;
}
int cal(Tree *root){
    if(!root) return 0;
    if(!root->left && !root->right)
        return stringToNum(root->value);
    int l_val=cal(root->left);
    int r_val=cal(root->right);
    if(root->value=="+")
        return l_val+r_val;
    if (root->value=="-")
        return l_val-r_val;
    if (root->value=="*")
        return l_val*r_val;
    return l_val/r_val;
}
void initwsolve(){
    cin>>n;
    for(int i=0;i<n;i++) cin>>s[i];
    Tree *root=new Tree;
    root=insertLevel(root,0);
    cout<<cal(root)<<endl;
}
int main(){
    int t;
    cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

DUYỆT CÂY 1

Cho phép duyệt cây nhị phân Inorder và Preorder, hãy đưa ra kết quả phép duyệt Postorder của cây nhị phân. Ví dụ với cây nhị phân có các phép duyệt cây nhị phân của cây dưới đây:

Inorder : 4 2 5 1 3 6

Preorder: : 1 2 4 5 3 6

Postorder : 4 5 2 6 3 1

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng đầu tiên đưa vào số N là số lượng node; dòng tiếp theo đưa vào N số theo phép duyệt Inorder; dòng cuối cùng đưa vào N số là kết quả của phép duyệt Preorder; các số được viết cách nhau một vài khoảng trắng.
- T, N, node thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 1000$; $1 \leq \text{giá trị node} \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt Postorder theo từng dòng.

Ví dụ:

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

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int innor[1005], preor[1005], n;
struct Tree {
    int value;
    Tree* left, * right;
};
Tree* newNode(int val) {
    Tree* temp = new Tree;
    temp->left = temp->right = NULL;
    temp->value = val;
    return temp;
}
Tree* buildTree(int start, int end, int &preIndex, map<int, int>& mp) {
    if (start > end) return NULL;
    Tree* t = newNode(preor[preIndex++]);
    if (start == end) return t;
    int mid = mp[innor[start]];
    t->left = buildTree(start, mid - 1, preIndex, mp);
    t->right = buildTree(mid + 1, end, preIndex, mp);
    return t;
}
```

```

int inIndex = mp[t->value];
t->left = buildTree(start, inIndex - 1, preIndex, mp);
t->right = buildTree(inIndex + 1, end, preIndex, mp);
return t;
}
Tree* buildTreeWrap() {
    map<int, int> mp;
    for (int i = 0; i < n; i++)
        mp[innor[i]] = i;

    int preIndex=0;
    return buildTree(0, n - 1, preIndex, mp);
}
void printPos(Tree* t) {
    if (t != NULL) {
        printPos(t->left);
        printPos(t->right);
        cout << t->value << " ";
    }
}
void initwsolve() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> innor[i];
    for (int i = 0; i < n; i++) cin >> preor[i];
    Tree* root = buildTreeWrap();
    printPos(root);
    cout << endl;
}
int main() {
    int t;
    cin >> t;
    while (t--) initwsolve();
    return 0;
}

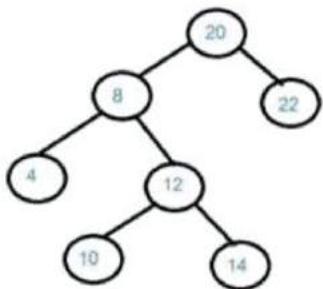
```

DUYỆT CÂY 2

Cho hai mảng là phép duyệt Inorder và Level-order, nhiệm vụ của bạn là xây dựng cây nhị phân và đưa ra kết quả phép duyệt Postorder. Level-order là phép duyệt theo từng mức của cây. Ví dụ như cây dưới đây ta có phép Inorder và Level-order như dưới đây:

Inorder : 4 8 10 12 14 20 22

Level order: 20 8 22 4 12 10 14



Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 3 dòng: dòng đầu tiên đưa vào số N là số lượng node; dòng tiếp theo đưa vào N số là phép duyệt Inorder; dòng cuối cùng đưa vào N số là phép duyệt Level-order; các số được viết cách nhau một vài khoảng trắng.
- T, N, node thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq A[i] \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt Postorder theo từng dòng.

Ví dụ:

Input	Output
2	
3	
1 0 2	
0 1 2	1 2 0
7	
3 1 4 0 5 2 6	3 4 1 5 6 2 0
0 1 2 3 4 5 6	

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int innor[1005], levelor[1005], n;
struct Tree {
    int value;
```

```

Tree* left, * right;
};

Tree* newNode(int val) {
    Tree* temp = new Tree;
    temp->left = temp->right = NULL;
    temp->value = val;
    return temp;
}

Tree* buildTree(int start, int end, map<int, int> mp) {
    if (start > end) return NULL;
    int index = start;
    for (int j = start + 1; j <= end; j++) {
        if (mp[innor[j]] < mp[innor[index]]) index = j;
    }
    Tree* t = newNode(innor[index]);
    t->left = buildTree(start, index - 1, mp);
    t->right = buildTree(index + 1, end, mp);
    return t;
}

Tree* buildTreeWrap() {
    map<int, int> mp;
    for (int i = 0; i < n; i++)
        mp[levelor[i]] = i;
    return buildTree(0, n - 1, mp);
}

void printPos(Tree* t) {
    if (t != NULL) {
        printPos(t->left);
        printPos(t->right);
        cout << t->value << " ";
    }
}

void initwsolve() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> innor[i];
    for (int i = 0; i < n; i++) cin >> levelor[i];
    Tree* root = buildTreeWrap();
    printPos(root);
}

```

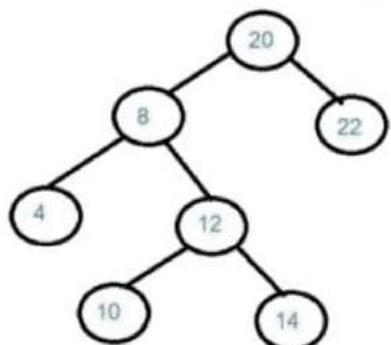
```

    cout << endl;
}
int main() {
    int t;
    cin >> t;
    while (t--) initwsolve();
    return 0;
}

```

DUYỆT CÂY THEO MỨC

Cho cây nhị phân, nhiệm vụ của bạn là duyệt cây theo Level-order. Phép duyệt level-order trên cây là phép thăm node theo từng mức của cây. Ví dụ với cây dưới đây sẽ cho ta kết quả của phép duyệt level-order: 20 8 22 4 12 10 14.



Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng cạnh của cây; dòng tiếp theo đưa vào N bộ ba (u, v, x), trong đó u là node cha, v là node con, x=R nếu v là con phải, x=L nếu v là con trái; u, v, x được viết cách nhau một vài khoảng trắng.
- T, N, u, v, thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq u, v \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt level-order theo từng dòng.

Ví dụ:

Input	Output
2	
2	
1 2 R 1 3 L	1 3 2 10 0 30 40 60
4	

Giai :

```
#include<bits/stdc++.h>
using namespace std;
int n;
struct Tree{
    int value;
    Tree *left,*right;
};
Tree *newNode(int val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree* buildTree(){
    map<int,Tree*> mp;
    Tree *root = NULL;
    while(n--){
        int a,b;
        char x;
        cin>>a>>b>>x;
        Tree *cha = newNode(a);
        if(mp.find(a)==mp.end()){
            mp[a]=cha;
            if(root==NULL) root=cha;
        }
        else cha=mp[a];
        Tree *con=newNode(b);
        if(x=='L') cha->left=con;
        else if(x=='R')cha->right=con;
        mp[b]=con;
    }
    return root;
}
void levelOrder(Tree *root){
    if(root==NULL) return;
```

```

queue<Tree*> q;
q.push(root);
while(!q.empty()){
    Tree* t=q.front();
    cout<<t->value<<" ";
    q.pop();
    if(t->left!=NULL) q.push(t->left);
    if(t->right!=NULL) q.push(t->right);
}
void initwsolve(){
    cin>>n;
    Tree *root=buildTree();
    levelOrder(root);
    cout<<endl;
}

int main(){
    int t; cin>>t;
    while(t--)initwsolve();
    return 0;
}

```

DUYỆT CÂY NHỊ PHÂN TÌM KIẾM 1

Cho mảng A[] gồm N node là biểu diễn phép duyệt theo thứ tự giữa (Preorder) của cây nhị phân tìm kiếm. Nhiệm vụ của bạn là đưa ra phép duyệt theo thứ tự sau của cây nhị phân tìm kiếm.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng node; dòng tiếp theo đưa vào N số A[i]; các số được viết cách nhau một vài khoảng trắng.
- T, N, node thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq A[i] \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt Postorder theo từng dòng.

Ví dụ:

Input	Output
-------	--------

2	35 30 100 80 40
5	35 32 30 120 100 90 80 40
40 30 35 80 100	
8	
40 30 32 35 80 90 100 120	

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int n, preor[1005];
struct Tree {
    int value;
    Tree* left, * right;
};
Tree* newNode(int val) {
    Tree* temp = new Tree;
    temp->left = temp->right = NULL;
    temp->value = val;
    return temp;
}
Tree* buildTree(int min, int max, int &preIndex) {
    if (preIndex >= n) return NULL;
    int val = preor[preIndex];
    if (val < min || val > max) return NULL;
    Tree* t = newNode(val);
    preIndex++;
    t->left = buildTree(min, val - 1, preIndex);
    t->right = buildTree(val + 1, max, preIndex);
    return t;
}
Tree* buildTreeWrap() {
    int preIndex = 0;
    return buildTree(INT_MIN, INT_MAX, preIndex);
}
void printPos(Tree* t) {
    if (t == NULL) return;
    if (t != NULL) {
        printPos(t->left);
        printPos(t->right);
    }
}
```

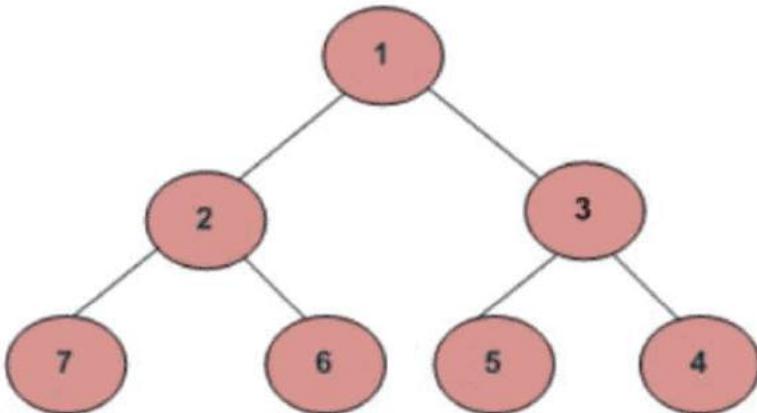
```

        printPos(t->right);
        cout << t->value << " ";
    }
}
void initwsolve() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> preor[i];
    Tree* root = buildTreeWrap();
    printPos(root);
    cout << endl;
}
int main() {
    int t;
    cin >> t;
    while (t--) initwsolve();
}

```

DUYỆT CÂY THEO MỨC ĐẢO NGƯỢC

Cho cây nhị phân, nhiệm vụ của bạn là duyệt cây theo mức đảo ngược (reverse-level-order). Với cây dưới đây sẽ cho ta kết quả của phép duyệt theo mức đảo ngược là : 7 6 5 4 3 2 1.



Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng cạnh của cây; dòng tiếp theo đưa vào N bộ ba (u, v, x), trong đó u là node cha, v là node con, x= R nếu v là con phải, x=L nếu v là con trái; u, v, x được viết cách nhau một vài khoảng trắng.

- T, N, u, v, thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq u, v \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt reverse-level-order theo từng dòng.

Ví dụ:

Input	Output
2	
2	
1 2 R 1 3 L	3 2 1
4	40 20 30 10
10 20 L 10 30 R 20 40 L 20 60 R	

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int n;
struct Tree{
    int value;
    Tree *left,*right;
};
Tree *newNode(int val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree* buildTree(){
    map<int,Tree*> mp;
    Tree *root = NULL;
    while(n--){
        int a,b;
        char x;
        cin>>a>>b>>x;
        Tree *cha = newNode(a);
        if(mp.find(a)==mp.end()){
            mp[a]=cha;
            if(root==NULL) root=cha;
        }
        else cha=mp[a];
    }
}
```

```

Tree *con=newNode(b);
if(x=='L') cha->left=con;
else if(x=='R')cha->right=con;
mp[b]=con;
}
return root;
}
void levelOrder(Tree *root){
    if(root==NULL) return;
    queue<Tree*> q;
    stack<int> st;
    q.push(root);
    while(!q.empty()){
        Tree* t=q.front();
        st.push(t->value);
        q.pop();
        if(t->right!=NULL) q.push(t->right);
        if(t->left!=NULL) q.push(t->left);
    }
    while(!st.empty()){
        cout<<st.top()<<" ";
        st.pop();
    }
}
void initwsolve(){
    cin>>n;
    Tree *root=buildTree();
    levelOrder(root);
    cout<<endl;
}
int main(){
    int t;cin>>t;
    while(t--)initwsolve();
}

```

DUYỆT CÂY KIỀU XOĂN ÔC (duyệt cây 7)

Cho cây nhị phân, nhiệm vụ của bạn là duyệt cây theo xoắn ốc (spiral-order).
 Phép. Ví dụ với cây dưới đây sẽ cho ta kết quả của phép duyệt spiral-order: 1 2
 3 4 5 6 7.

Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng cạnh của cây; dòng tiếp theo đưa vào N bộ ba (u, v, x), trong đó u là node cha, v là node con, x= R nếu v là con phải, x=L nếu v là con trái; u, v, x được viết cách nhau một vài khoảng trắng.
- T, N, u, v, thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq u, v \leq 10^4$;

Output:

- Đưa ra kết quả phép duyệt level-order theo từng dòng.

Ví dụ:

Input	Output
2	
2	
1 2 R 1 3 L	1 3 2
4	10 0 30 60 40
10 20 L 10 30 R 20 40 L 20 60 R	

Giải:

```
#include<bits/stdc++.h>
using namespace std;
int n;
struct Tree{
    int value;
    Tree *left,*right;
};
Tree *newNode(int val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree* buildTree(){
    map<int,Tree*> mp;
    Tree *root = NULL;
    while(n--){
        int a,b;
```

```

char x;
cin>>a>>b>>x;
Tree *cha = newNode(a);
if(mp.find(a)==mp.end()){
    mp[a]=cha;
    if(root==NULL) root=cha;
}
else cha=mp[a];

Tree *con=newNode(b);
if(x=='L') cha->left=con;
else if(x=='R')cha->right=con;
mp[b]=con;
}
return root;
}

void levelOrder(Tree *root){
if(root==NULL) return;
deque<Tree*> dq;
dq.push_front(root);
bool flag=false;
while(!dq.empty()){
    int dsize=dq.size();
    if(flag){
        while(dsize--){
            Tree* t=dq.front();
            dq.pop_front();
            cout<<t->value<<" ";
            if(t->left!=NULL) dq.push_back(t->left);
            if(t->right!=NULL) dq.push_back(t->right);
        }
    }
    else{
        while(dsize--){
            Tree* t=dq.back();
            dq.pop_back();
            cout<<t->value<<" ";
            if(t->right!=NULL) dq.push_front(t->right);
        }
    }
}
}

```

```

        if(t->left!=NULL) dq.push_front(t->left);
    }

}

flag=!flag;
}
}

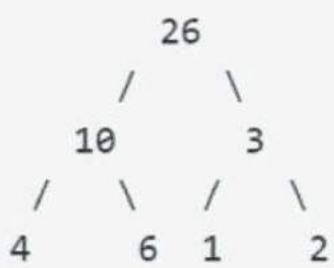
void initwsolve(){
    cin>>n;
    Tree *root=buildTree();
    levelOrder(root);
    cout<<endl;
}

int main(){
    int t; cin>>t;
    while(t--) initwsolve();
    return 0;
}

```

CÂY NHỊ PHÂN TỔNG

Cho cây nhị phân, nhiệm vụ của bạn là kiểm tra xem cây nhị phân có phải là một cây tổng hay không? Một cây nhị phân được gọi là cây tổng nếu tổng các node con của node trung gian bằng giá trị node cha. Node không có node con trái hoặc phải được hiểu là có giá trị 0. Ví dụ dưới đây là một cây tổng



Input:

- Dòng đầu tiên đưa vào số lượng test T.
- Những dòng tiếp theo đưa vào các bộ test. Mỗi bộ test gồm 2 dòng: dòng đầu tiên đưa vào số N là số lượng cạnh của cây; dòng tiếp theo đưa vào N bộ ba (u, v, x), trong đó u là node cha, v là node con, x=R nếu v là con phải, x=L nếu v là con trái; u, v, x được viết cách nhau một vài khoảng trắng.

- T, N, u, v , thỏa mãn ràng buộc: $1 \leq T \leq 100$; $1 \leq N \leq 10^3$; $1 \leq u, v \leq 10^4$;

Output:

- Đưa ra kết quả mỗi test theo từng dòng.

Ví dụ:

Input	Output
2	
2	
3 1 L 3 2 R	1
4	0
10 20 L 10 30 R 20 40 L 20 60 R	

Giải :

```
#include<bits/stdc++.h>
using namespace std;
int n;
struct Tree{
    int value;
    Tree *left,*right;
};
Tree *newNode(int val){
    Tree *temp=new Tree;
    temp->left=temp->right=NULL;
    temp->value=val;
    return temp;
}
Tree* buildTree(){
    map<int,Tree*> mp;
    Tree *root = NULL;
    while(n--){
        int a,b;
        char x;
        cin>>a>>b>>x;
        Tree *cha = newNode(a);
        if(mp.find(a)==mp.end()){
            mp[a]=cha;
            if(root==NULL) root=cha;
        }
        else cha=mp[a];
    }
}
```

```

Tree *con=newNode(b);
if(x=='L') cha->left=con;
else if(x=='R')cha->right=con;
mp[b]=con;
}
return root;
}
int sum(Tree* root){
    if(root==NULL) return 0;
    return sum(root->left) + root->value + sum(root->right);
}
bool checkSumTree(Tree* root){
    int leftData=0,rightData=0;
    if(root==NULL || (root->left==NULL && root->right==NULL)) return true;
    leftData=sum(root->left);
    rightData=sum(root->right);
    if((root->value==leftData+rightData) && checkSumTree(root->left)
    && checkSumTree(root->right)) return true;
    return false;
}
void initwsolve(){
    cin>>n;
    Tree *root=buildTree();
    if(checkSumTree(root)) cout<<1<<endl;
    else cout<<0<<endl;
}
int main(){
    int t; cin>>t;
    while(t--)initwsolve();
    return 0;
}

```