#并查集

```
#include<cstdio>
#define MAX 200010
using namespace std;
int a[MAX];
int c,d;
int n,m,i,j,z;
int search(int x){
        return a[x]?a[x]=search(a[x]):x;
}
int main()
{
        scanf("%d%d",&n,&m);
        z=n;
        while(m--){
                scanf("%d%d",&c,&d);
                c=search(c);
                while(c<=d){
                        a[c]=c+1;
                        z--;
                        c=search(c);
                }
                printf("%d\n",z);
        }
        return 0;
}
```

#前缀和

```
#include<stdio.h>
int n,k;
int i;
int a[1000010];
long long int b[1000010];
long long int max;
int main()
{
        scanf("%d%d",&n,&k);
        for(i=0;i<n;i++){
                scanf("%d",&a[i]);
                if(i==0){
                        b[i]=a[i];
                }
                else{
                        b[i]=b[i-1]+a[i];
                }
        }
        for(i=0;i<=n-k;i++){
                if(i==0){
```

```
                        max=b[k-1];
                }
                else{
                        if(max<b[k+i-1]-b[i-1]){
                                max=b[k+i-1]-b[i-1];
                        }
                }
        }
        /*for(i=0;i<n;i++){
                printf("%lld ",b[i]);
        }*/
        printf("%lld",max);
}
```

#差分

```
#include<stdio.h>
int n,k,i;
int p,q;
long long int b[1000010];
long long max,sum;
int main()
{
        scanf("%d%d",&n,&k);
        for(i=0;i<k;i++){
                scanf("%d%d",&p,&q);
                b[p]=b[p]+1;
                b[p+q]=b[p+q]-1;
        }
        for(i=1;i<=n;i++){
                if(i==1){
                        max=b[1];
                        sum=b[1];
                }
                else{
                        if(max<sum+b[i]){
                                max=sum+b[i];
                        }
                        sum=sum+b[i];
                }
        }
        printf("%d",max);
}
```

#逆序对分治,即冒泡排序交换次数

```
#include<stdio.h>
int n,i;
long long int sum;
```

```
int w[100010],x[100010];

void merge(int w[],int x[],int left,int center,int right){
        int i=left,j=center+1,k=left;
        while(i<=center&&j<=right){
                if(w[i]<=w[j]){
                        x[k++]=w[i++];
                }
                else{
                        sum+=center-i+1;
                        x[k++]=w[j++];
                }
        }
        while(i<=center){
                x[k++]=w[i++];
        }
        while(j<=right){
                x[k++]=w[j++];
        }
        for(i=left;i<=right;i++){
                w[i]=x[i];
        }
}

void ms(int w[],int l,int r){
        int c;
        if(l<r){
                c=(l+r)/2;
                ms(w,l,c);
                ms(w,c+1,r);
                merge(w,x,l,c,r);
        }
}
int main()
{
        scanf("%d",&n);
        for(i=0;i<n;i++){
                scanf("%d",&w[i]);
        }
        ms(w,0,n-1);
        printf("%lld",sum);
}
```

#一种递归

```
#include<cstdio>
#include<vector>
using namespace std;
int n;
vector<int> a;
void pr(int w)
{
```

```
        if(w<=n){
                a.push_back(w);
                for(int i=0;i<a.size();i++){
                        printf("%d ",a[i]);
                }
                printf("\n");
                pr(w+1);
                a.pop_back();
                pr(w+1);
        }
}

int main()
{
        scanf("%d",&n);
        pr(1);
        return 0;
}
```

#过河 Zexal打算借助河中间的石砖过到河对岸去。Zexal从第一块石砖出发，接下来他可以走到第二块石砖或第三块石砖，有时候走的很不爽，甚至可以直接跨过两个石砖，到达第四块石砖，但是不能连续两次这种操作，因为这样消耗体能比较大。现在假设河中含n块石砖，且这些石砖呈直线分布，请你计算出Zexal从第一块石砖出发有多少种安全的过河方法。

```
#include<stdio.h>
int n,i;
long long int a[2][55];
int main()
{
        a[0][1]=1;
        a[0][2]=2;
        a[0][3]=3;
        a[1][3]=1;
        for(i=4;i<=50;i++)
        {
                a[1][i]=a[0][i-3];
                a[0][i]=a[0][i-1]+a[0][i-2]+a[1][i-1]+a[1][i-2];
        }
        while((scanf("%d",&n))!=EOF)
        {
                printf("%lld\n",a[1][n]+a[0][n]);
        }
        return 0;
}
```

#流水线问题

```
#include<stdio.h>
int n,i;
int p1[510];
int p2[510];
int t1[510];
int t2[510];
int q1[510];
int q2[510];
int main()
{
        while(~scanf("%d",&n)){
                for(i=1;i<=n;i++){
                        scanf("%d",&p1[i]);
                }
                for(i=1;i<=n;i++){
                        scanf("%d",&p2[i]);
                }
                for(i=1;i<n;i++){
                        scanf("%d",&t1[i]);
                }
                for(i=1;i<n;i++){
                        scanf("%d",&t2[i]);
                }
                q1[1]=p1[1];
                q2[1]=p2[1];
                for(i=2;i<=n;i++){
                        //printf("%d %d\n",q1[i-1]+t1[i-1]+p2[i],q2[i-1]+p2[i]);
                        if(q1[i-1]+t1[i-1]+p2[i]<q2[i-1]+p2[i]){
                                q2[i]=q1[i-1]+t1[i-1]+p2[i];
                        }
                        else{
                                q2[i]=q2[i-1]+p2[i];
                        }
                        if(q2[i-1]+t2[i-1]+p1[i]<q1[i-1]+p1[i]){
                                q1[i]=q2[i-1]+t2[i-1]+p1[i];
                        }
                        else{
                                q1[i]=q1[i-1]+p1[i];
                        }
                }
                if(q1[n]>q2[n]){
                        printf("%d\n",q2[n]);
                }
                else{
                        printf("%d\n",q1[n]);
                }
        }
}
```

#钢管切割

```
#include<stdio.h>
int n,i,j;
int a[1010];
int b[1010];
int main()
{
        while(~scanf("%d",&n)){
                for(i=1;i<=n;i++){
                        scanf("%d",&a[i]);
                        b[i]=0x3f3f3f3f;
                }
                for(i=1;i<=n;i++){
                        for(j=1;j<=i;j++){
                                if(b[i]>a[j]+b[i-j]){
                                        b[i]=a[j]+b[i-j];
                                }
                        }
                }
                printf("%d\n",b[n]);
        }
}
```

#排座位，男生互不相邻 w:最后一个是女生 m:最后一个是男生

```
#include<stdio.h>
int n,i;
int m[35];
int w[35];
int main()
{
        scanf("%d",&n);
        m[1]=1;
        w[1]=1;
        for(i=2;i<30;i++){
                w[i]=w[i-1]+m[i-1];
                m[i]=w[i-1];
        }
        printf("%d",w[n]+m[n]);
}
```

#点灯 有n个灯，编号0~n−1，一开始都是关闭状态 每次操作会拨动一个区间[L,R]灯的开关，也就是说，对于灯i，L≤i≤R，如果i是关闭状态，则操作会使灯亮，反之会使灯灭。 请问k次操作后有多少灯亮着。

多组输入数据 每组数据第一行两个数n,k（1≤n≤109,1≤k≤105） 接下来k行，每行两个数l,r（0≤l≤r≤n−1）

```
#include<cstdio>
#include<iostream>
#include<cstdlib>
```

```
using namespace std;
int l[100005];
int r[100005];
int t[200005][2];
int turn;
int cmp(const void *a,const void *b){return *(int*)a - *(int*)b;}
int main()
{
        int n,k;
        while(cin>>n>>k)
        {
                int i;
                int p=0;
                for(i=1;i<=k;i++)
                {
                        scanf("%d%d",&l[i],&r[i]);
                        t[++p][0]=l[i];
                        t[p][1]=1;
                        t[++p][0]=r[i]+1;
                        t[p][1]=2;
                }
                qsort(t+1,k*2,8,cmp);
                /*for(i=1;i<=2*k;i++)
                {
                        printf("%d %d\n",t[i][0],t[i][1]);
                }*/
                int ans=0;
                for(i=1;i<=k*2;i++){
                        if(t[i][1]==1){
                                turn++;
                        }
                        else{
                                turn--;
                        }
                        if(turn%2==0)
                        {
                                ans+=t[i][0]-t[i-1][0];
                        }
                }
                cout<<ans<<endl;
        }
        return 0;
}
```

#多路流水线调度 城市里有n个电脑城，并且每个电脑城都有所有的配件卖，除了价格不同外完全一样。一台电脑一共有m个配件，按照安装顺序编号为1−m。假设第i个电脑城的编号为j的配件售价为p[i][j]，从第i个电脑城到第j个电脑城的交通费用为f[i][j]。那么SkyLee组装好整台电脑最少需要多少钱呢？（配件费用+交通费用） 多组数据输入,第一行两个整数n和m，分别为电脑城数量和配件数量（2 < n,m≤500）接下来n行，每行m个整数，表示配件的价格p[i][j]（0≤p[i][j]≤500）接下来n行，每行n个整数，表示交通费用f[i][j]（0≤f[i][j]≤500）

```
#include<cstdio>
#include<algorithm>
using namespace std;
int n,m,i,j,k;
int p[510][510];
int mm[510][510];
int y[510][510];
int pay;
int main()
{
        while(~scanf("%d%d",&n,&m)){
                for(i=1;i<=n;i++){
                        for(j=1;j<=m;j++){
                                scanf("%d",&p[i][j]);
                        }
                }
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                scanf("%d",&mm[i][j]);
                        }
                }
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                if(i==1){
                                        y[j][1]=p[j][1];
                                }
                                else{
                                        int t=y[1][i-1]+mm[1][j]+p[j][i];
                                        for(k=2;k<=n;k++){
                                                t=min(t,y[k][i-1]+mm[k][j]+p[j]
[i]);
                                        }
                                        y[j][i]=t;
                                }
                        }
                }
                for(i=1;i<=n;i++)
                {
                        if(i==1){
                                pay=y[i][m];
                        }
                        else{
                                pay=min(pay,y[i][m]);
                        }
                }
                printf("%d\n",pay);
        }
}
```

#矩阵链乘 多组数据输入 第一行一个整数 n，表示矩阵链的长度（1<=n<=300） 接下来一行n+1个数表示这些矩阵的行数和列数 别问我为什么只有n+1个数，每相邻的两个数表示一个矩阵的大小 (次数相同优先计算左边

的)

```
#include<cstdio>
#define infinity 1000000000
using namespace std;
int n,i,j,k,l;
int a[310];
int m[310][310];
int w[310][310];
int s,q;

void print(int a,int b){

        if(a!=b){
                printf("(");
                print(a,w[a][b]);
                print(w[a][b]+1,b);
                printf(")");
        }
        else if(a==b){
                printf("A%d",a);
        }
}

int main()
{
        while(~scanf("%d",&n)){
                for(i=0;i<=n;i++){
                        scanf("%d",&a[i]);
                }
                for(i=1;i<=n;i++){
                        m[i][i]=0;
                }
                for(l=2;l<=n;l++){
                        for(i=1;i<=n-l+1;i++){
                                j=i+l-1;
                                m[i][j]=infinity;
                                for(k=j-1;k>=i;k--){
                                        q=m[i][k]+m[k+1][j]+a[i-1]*a[k]*a[j];
                                        if(q<m[i][j]){
                                                m[i][j]=q;
                                                w[i][j]=k;
                                        }
                                }
                        }
                }
                printf("%d\n",m[1][n]);
                print(1,n);
                printf("\n");
        }
}
```

#最小乘法 如果在一个正整数中间插入一些乘号，会得到一个更小的数字。 一个数字有不同的插入方法，会得到不同的数字，我们想知道最大的那个数字是多少？ 输入包括多组数据。每组数据第一行只含一个正整数n（0≤n≤10），代表添加乘号的个数。第二行是一个数字串（0＜长度≤18，保证长度大于n）。

```
#include<iostream>
#include<cstdio>
#include<string>
#include<algorithm>
#include<cmath>
using namespace std;
int n,i,j,k;
string s;
long long num[20];
long long a[20][15];
long long c[20][20];
int main()
{
        while(~scanf("%d",&n)){
                cin>>s;
                //printf("%d\n",s.length());
                //printf("%c\n",s[0]);
                for(i=0;i<20;i++){
                        num[i]=0;
                }
                for(i=0;i<20;i++){
                        for(j=0;j<15;j++){
                                a[i][j]=0;
                        }
                }
                for(i=0;i<20;i++){
                        for(j=0;j<20;j++){
                                c[i][j]=0;
                        }
                }
                for(i=0;i<s.length();i++){
                        num[i]=s[i]-'0';
                        c[i][i]=num[i];
                }
                for(i=1;i<s.length();i++){
                        for(j=s.length()-1;j>=i;j--){
                                c[j-i][j]=c[j-i][j-1]*10+num[j];
                        }
                }
                /*for(i=0;i<=s.length()-1;i++)
                {
                        for(j=0;j<=s.length()-1;j++)
                        {
                                printf("%lld ",c[i][j]);
                        }
                        printf("\n");
                }*/
                for(i=0;i<s.length();i++){
```

```
                        a[i][0]=c[0][i];
                }
                for(i=1;i<=n;i++){
                        for(j=i;j<s.length();j++){
                                for(k=j-1;k>=i-1;k--){
                                        //printf("%lld %lld %lld\n",a[k][i-
1],c[k+1][s.length()-1],a[j][i]);
                                        a[j][i]=max(a[k][i-1]*c[k+1][j],a[j][i]);

                                }
                        }
                }
                printf("%lld\n",a[s.length()-1][n]);
        }
}
```

#二叉搜索树的最小代价 （见《算法导论》）

```
#include<cstdio>
#include<algorithm>
#define sup 1000000000
using namespace std;
int n,i,j,l,r,t;
int p[510],q[510];
int e[510][510],w[510][510];
int root[510][510];
int main()
{
        scanf("%d",&n);
        for(i=1;i<=n;i++){
                scanf("%d",&p[i]);
        }
        for(i=0;i<=n;i++)
        {
                scanf("%d",&q[i]);
        }
        for(i=1;i<=n+1;i++){
                e[i][i-1]=q[i-1];
                w[i][i-1]=q[i-1];
        }
        for(l=1;l<=n;l++){
                for(i=1;i<=n-l+1;i++){
                        j=i+l-1;
                        e[i][j]=sup;
                        w[i][j]=w[i][j-1]+p[j]+q[j];
                        for(r=i;r<=j;r++){
                                t=e[i][r-1]+e[r+1][j]+w[i][j];
                                if(t<e[i][j]){
                                        e[i][j]=t;
                                        root[i][j]=r;
                                }
                        }
```

```
                }
        }
        printf("%d",e[1][n]);
}
```

#股票问题：最多购买2次

```
#include<stdio.h>
int n;
int a[100001];
int max(int x, int y);
int main()
{
        while(scanf("%d", &n) != EOF){
                int i = 0;
            long long profit = 0;
            for(i; i < n; i++){
                    scanf("%d", &a[i]);
            }
            int dp[100001][3][2];
            dp[0][1][0] = 0;
            dp[0][1][1] = - a[0];
            dp[0][2][0] = 0;
            dp[0][2][1] = - a[0];
            for(i = 1; i < n; i++){
                dp[i][2][0] = max(dp[i - 1][2][0], dp[i - 1][2][1] + a[i]);
                dp[i][2][1] = max(dp[i - 1][2][1], dp[i - 1][1][0] - a[i]);
                dp[i][1][0] = max(dp[i - 1][1][0], dp[i - 1][1][1] + a[i]);
                dp[i][1][1] = max(dp[i - 1][1][1], -1 * a[i]);
                }
                printf("%d\n", dp[n-1][2][0]);
        }
}
int max(int a, int b){
        return a > b ? a : b;
}
```

```
#include<stdio.h>
#include<limits.h>
#include<string.h>
#define max(a,b) ((a)>(b)?(a):(b))
#define min(a,b) ((a)<(b)?(a):(b))
int buy[100005], sell[100005], a[100005];
long long max_buy[100005];
int n;
long long ans;

int main()
{
```

```
        while(scanf("%d",&n)!=EOF){
                ans=0;
                memset(max_buy,0,sizeof(max_buy));
                for(int i=1;i<=n;i++){
                        scanf("%d",a+i);
                }
                int Min=INT_MAX,Max=0;
                for(int i=1;i<=n;i++){
                        Min=min(Min,a[i]);
                        sell[i]=a[i]-Min;
                }
                for(int i=n;i>=1;i--){
                        Max=max(Max,a[i]);
                        buy[i]=Max-a[i];
                        max_buy[i]=max(buy[i],max_buy[i+1]);
                }
                for(int i=1;i<=n;i++)
                {
                        if(sell[i]+max_buy[i]>ans) ans=sell[i]+max_buy[i];
                }
                printf("%lld\n",ans);
        }
        return 0;
}
```

#股票问题，最多k次

```
#include<cstdio>
#include<algorithm>
using namespace std;
long long int n,k,i,j,dif;
long long int a[1010];
long long int m[1010];
long long int t[1010];
int main()
{
        while(~scanf("%lld%lld",&n,&k)){
                for(i=1;i<=n;i++){
                        scanf("%lld",&a[i]);
                }
                for(i=0;i<1010;i++){
                        m[i]=0;
                        t[i]=0;
                }
                for(i=1;i<n;i++){
                        dif=a[i+1]-a[i];
                        for(j=k;j>=1;j--){
                                t[j]=max(m[j-1]+max(dif,(long long)0),t[j]+dif);
                                m[j]=max(m[j],t[j]);
                                //printf("%d\n",dif);
                        }
                }
```

```
                printf("%lld\n",m[k]);
        }
}
```

#股票问题，尽可能多交易

```
#include<stdio.h>
int n,i;
int a[100010];
long long int s,sum;
int main()
{
        while(~scanf("%d",&n)){
                for(i=1;i<=n;i++){
                        scanf("%d",&a[i]);
                }
                s=0;
                sum=0;
                for(i=2;i<=n;i++){
                        if(a[i]>a[i-1]){
                                sum+=a[i]-a[i-1];
                        }
                }
                printf("%lld\n",sum);
        }
}
```

#组合背包

```
#include<stdio.h>
#define max(a,b) ((a)>(b)?(a):(b))

int n,v,i;
int c[510],w[510],u[510];
int ca[30010];
void zopack(int a,int b){
        for(int i=v;i>=a;i--){
                ca[i]=max(ca[i],ca[i-a]+b);
        }
}

void mpack(int a,int b){
        for(int i=a;i<=v;i++){
                ca[i]=max(ca[i],ca[i-a]+b);
        }
}

void cpack(int a,int b,int c){
        if(a*c>=v){
```

```
                mpack(a,b);
                return;
        }

        int k=1;
        while(k<c){
                zopack(k*a,k*b);
                c-=k;
                k<<=1;
        }
        zopack(c*a,c*b);
}

int main()
{
        while((scanf("%d%d",&n,&v))!=EOF){
                for(i=1;i<=v;i++){
                        ca[i]=0;
                }
                for(i=1;i<=n;i++){
                        scanf("%d%d%d",&c[i],&w[i],&u[i]);
                        if(u[i]==1){
                                zopack(c[i],w[i]);
                        }
                        else if(u[i]==233){
                                mpack(c[i],w[i]);
                        }
                        else{
                                cpack(c[i],w[i],u[i]);
                        }
                }
                printf("%d\n",ca[v]);
        }
        return 0;
}
```