题目 1 逆序对

```
#include<stdio.h>
#include<string.h>
void mergesort(int[],int,int);
void merge(int[],int,int,int);
long long int sum=0;
int main()
{
	int a[200000];
	int i,j,n;
	scanf("%d",&n);
	for(i=0;i<n;i++) scanf("%d",&a[i]);
	mergesort(a,0,n-1);
	printf("%lld",sum);
	return 0;
}
void merge(int a[],int left,int mid,int right)
{
	int b[200000];
	int i=left,j=mid+1,k=0,x=0;
	while(i<=mid&&j<=right)
	{
		if(a[i]<=a[j])
		{
			b[k]=a[i];
			k++;i++;
		}
		else
		{
			x=x+mid-i+1;
			b[k]=a[j];
			k++;j++;
		}
	}
	if(j==right+1)
	{
		while(i<=mid)
		{
			b[k]=a[i];
			i++;k++;
		}
	}
	if(i==mid+1)
	{
```

```
        while(j<=right)
        {
            b[k]=a[j];
            j++;k++;
        }
    }
    for (j=0,i=left;j<k;i++,j++)
    {
        a[i]=b[j];
    }
    sum=sum+x;
}
void mergesort(int a[],int left,int right)
{
    if(left>=right) return ;
    int mid;
    mid=(left+right)/2;
    mergesort(a,left,mid);
    mergesort(a,mid+1,right);
    merge(a,left,mid,right);
}
```

题目二 兔子繁衍

```
#include<stdio.h>
long long int num[100],birth[100],old[100],young[100];
int main()
{
    long long int a,b,x,i,j;
    scanf("%lld%lld%lld",&a,&b,&x);
    birth[1]=2;
    num[1]=2;
    old[1]=0;
    young[1]=2;
    for(i=2;i<=a;i++)
    {
        num[i]=2;
        birth[i]=0;
        old[i]=0;
        young[i]=2;
    }
    for(i=a+1;i<=x;i++)
    {
        if(i>a+b) old[i]=birth[i-a-b]+old[i-1];
        else old[i]=0;
```

```
            birth[i]=(num[i-1]-old[i]-young[i-1]+birth[i-a])/2;
            young[i]=young[i-1]+birth[i]-birth[i-a];
            num[i]=num[i-1]+birth[i];
        }
        printf("%lld",num[x]);
}
```

题目三 投篮次数

```
#include<stdio.h>
void f(int,int,int);
int n,max,num=0;
int a[10000];
int xx=0;
int main()
{
        int sum=0,k=0,x=0;
        scanf("%d%d",&n,&max);
        f(sum,0,0);
        printf("%d",xx);
        return 0;
}
void f(int sum,int k,int x)
{
        int i,j;
        if(sum==n)
        {
                xx++;
        }
        else if(sum<n)
        {
                if(x<max)
                {
                        a[k]=2;
                        f(sum+2,k+1,x+1);
                        a[k]=0;

                }
                a[k]=3;
                f(sum+3,k+1,x);
                a[k]=0;
        }
}
```

题目四 查找

```
#include<stdio.h>
#include<stdlib.h>
int a[1000000][2];
int f(int,int,int);//二分查找
int rising(const int *p1,const int *p2)
{
	if(p1[0]>p2[0]||(p1[0]==p2[0]&&p1[1]>p2[1])) return 1;
	else return -1;
}
int main()
{
	int n,i,j,tmp,x,result;
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i][0]);
		a[i][1]=i+1;
	}
	qsort(a,n,sizeof(a[0]),rising);
	while(scanf("%d",&x)!=EOF)
	{
		result=f(x,0,n);
		if(result==-1) printf("NO\n");
		else printf("%d\n",result);
	}
	return 0;
}
int f(int key,int low,int high)
{
	int mid;
	while(low<=high)
	{
		mid=(low+high)/2;
		if(key==a[mid][0])
		{
			while(key==a[mid-1][0])
			{
				if(key==a[mid/2][0]) mid=mid/2;
				else mid--;
			}
			return a[mid][1];
		}
		else if(key>a[mid][0]) low=mid+1;
		else if(key<a[mid][0]) high=mid-1;
```

```
    }
    return -1;
}
```

题目五 贴瓷砖

```
#include<stdio.h>
int n;
long long int a[200000];
int main()
{
    scanf("%d",&n);
    int i,j;
    a[1]=1;a[0]=1;
    for(i=2;i<=n;i++)
    {
        a[i]=a[i-1]+2*a[i-2];
        a[i]=a[i]%998244353;
    }
    printf("%lld",a[n]);
    return 0;
}
```

题目六 下一个排列 输入 3　2 1 3 输出 2 3 1

```
#include<stdio.h>
int a[200000],b[200000];
int main()
{
    int n;
    scanf("%d",&n);
    int i,j,left,right,small,big,goal,tmp;
    for(i=0;i<n;i++) scanf("%d",&a[i]);
    for(i=1;i<n;i++)
    {
        if(a[i]>a[i-1])
        {
            small=a[i-1];
            big=a[i];
            left=i-1;
            right=i;
        }
    }
    for(i=n-1;i>=right;i--)
    {
```

```
            if(a[i]>a[left])
            {
                goal=i;
                break;
            }
        }
        tmp=a[goal];
        a[goal]=a[left];
        a[left]=tmp;
        for(i=right;i<n;i++) b[i]=a[i];
        for(i=right;i<n;i++) a[i]=b[n+right-i-1];
        for(i=0;i<n;i++) printf("%d ",a[i]);
        return 0;
}
```

题目七 数列中三角形个数

```
#include<stdio.h>
#include<stdlib.h>
int a[200000];
int n;
long long int sum=0;
int f(int,int,int);
int rising(const int *p1,const int *p2)
{
        return *p1-*p2;
}
int main()
{
        int i,j,left,right;
        scanf("%d",&n);
        for(i=0;i<n;i++) scanf("%d",&a[i]);
        qsort(a,n,sizeof(int),rising);
        for(i=n-1;i>=2;i--)
        {
            right=i-1;
            left=0;
            while(left<right)
            {
                if(a[right]+a[left]>a[i])
                {
                    sum=sum+right-left;
                    right--;
                }
                else left++;
```

```
            }
        }
        printf("%lld",sum);
        return 0;
}
```

题目八 非降序排列

```
#include<stdio.h>
typedef struct node
{
        int judge;
        long long int max;
        long long int min;
}LNode,*linklist;
LNode p[1000005];
int a[1000005];
int main()
{
        long long int i,j,n,k=0;
        scanf("%lld",&n);
        for(i=0;i<n;i++) scanf("%lld",&a[i]);
        for(i=0;i<n;i++)
        {
                if(i==n-1)
                {
                        p[k].max=a[i];
                        p[k].min=a[i];
                        k++;
                        break;
                }
                else if(a[i]<=a[i+1])
                {
                        p[k].max=a[i];
                        p[k].min=a[i];
                        k++;
                }
                else if(a[i]>a[i+1])
                {
                        p[k].max=a[i];
                        p[k].min=a[i+1];
                        for(j=i+1;j<n-1;j++)
                        {
                                if(a[j]>a[j+1]) p[k].min=a[j+1];
                                else break;
```

```
                }
                k++;
                i=j;
            }
        }
        long long int result=k;
        for(i=0;i<k-1;i++)
        {
            if(p[i].max>p[i+1].min&&p[i].judge==0&&p[i+1].judge==0)
            {
                p[i].judge=1;
                result--;
                if(p[i].max>=p[i+1].max) p[i+1].max=p[i].max;
                if(p[i].min<=p[i+1].min) p[i+1].min=p[i].min;
            }
        }
        for(i=k-1;i>0;i--)
        {
            if(p[i].judge!=0) continue;
            j=i-1;
            while(p[j].judge!=0)
            {
                j--;
                if(j<0) break;
            }
            if(j<0) continue;
            if(p[j].max>p[i].min)
            {
                p[i].judge=1;
                result--;
                if(p[i].max>=p[j].max) p[j].max=p[i].max;
                if(p[i].min<=p[j].min) p[j].min=p[i].min;
            }
        }
        printf("%lld",result);
        return 0;
}
```

题目九 数矩形

```
#include<stdio.h>
#include<stdlib.h>
int a[3000][3000];
int num[3000][3000];
long long int sum=0;
```

```
int main()
{
	int n,i,j,k,m,x,y;
	int max_x=0,max_y=0;
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		scanf("%d%d",&x,&y);
		a[x][y]=1;
		if(x>max_x) max_x=x;
		if(y>max_y) max_y=y;
	}
	for(i=0;i<=max_x;i++)
	{
		for(j=0;j<=max_y;j++)
		{
			if(a[i][j]==1)
			{
				for(k=j+1;k<=max_y;k++)
				{
					if(a[i][k]==1)
					{
						for(m=i+1;m<=max_x;m++)
						{
							if(a[m][j]==1&&a[m][k]==1) sum++;
						}
					}
				}
			}
		}
	}
	printf("%lld",sum);
	return 0;
}
```

题目十 杀恶龙

```
#include<stdio.h>
#include<stdlib.h>
long long int m,n,s,max=0,max_i=0,k=0;
long long int a[2000000],sum[2000000];
int rising(const long long int *p1,const long long int *p2)
{
	return *p2-*p1;
}
```

```
int main()
{
	long long int i,j;
	scanf("%lld%lld",&n,&m);
	for(i=0;i<n;i++) scanf("%d",&a[i]);
	qsort(a,n,sizeof(long long int),rising);
	for(i=0;i<n;i++)
	{
		if(i==0) sum[i]=a[i];
		else sum[i]=a[i]+sum[i-1];
	}
	for(i=0;i<m;i++)
	{
		scanf("%lld",&s);
		if(s>sum[n-1])
		{
			printf("Lose\n");
			continue;
		}
		if(s>=max) //这里可以整一个二分查找
		{
			max=s;
			k=max_i;
		}
		else
		{
			k=0;
		}
		for(j=k;j<n;j++)
		{
			if(sum[j]>=s)
			{
				printf("%lld\n",j+1);
				if(j>max_i&&s>=max)
				{
					max_i=j;
				}
				break;
			}
		}
	}
	return 0;
}
```

题目十一 乱序对变有序至少几次（只交换相邻两人）

```
#include<stdio.h>
#include<string.h>
void mergesort(int[],int,int);
void merge(int[],int,int,int);
long long int sum=0;
int main()
{
	int a[250000];
	int i,j,n;
	scanf("%d",&n);
	for(i=0;i<n;i++) scanf("%d",&a[i]);
	mergesort(a,0,n-1);
	printf("%lld",sum);
	return 0;
}
void merge(int a[],int left,int mid,int right)
{
	int b[200000];
	int i=left,j=mid+1,k=0;
	long long int x=0;
	while(i<=mid&&j<=right)
	{
		if(a[i]<=a[j])
		{
			b[k]=a[i];
			k++;i++;
		}
		else
		{
			x=x+mid-i+1;
			b[k]=a[j];
			k++;j++;
		}
	}
	if(j==right+1)
	{
		while(i<=mid)
		{
			b[k]=a[i];
			i++;k++;
		}
	}
	if(i==mid+1)
```

```
    {
        while(j<=right)
        {
            b[k]=a[j];
            j++;k++;
        }
    }
    for (j=0,i=left;j<k;i++,j++)
    {
        a[i]=b[j];
    }
    sum=sum+x;
}
void mergesort(int a[],int left,int right)
{
    if(left>=right) return ;
    int mid;
    mid=(left+right)/2;
    mergesort(a,left,mid);
    mergesort(a,mid+1,right);
    merge(a,left,mid,right);
}
```

题目十二 最大子数组

```
//sum=max{a[i],sum+a[i]}
#include<stdio.h>
#include<stdlib.h>
long long int a[2000000],sum[2000000];
long long int n,max=0;
int main()
{
    long long int i,j;
    scanf("%lld",&n);
    for(i=0;i<n;i++) scanf("%lld",&a[i]);
    sum[0]=a[0];
    max=sum[0];
    for(i=1;i<n;i++)
    {
        if(sum[i-1]+a[i]<a[i]) sum[i]=a[i];
        else sum[i]=a[i]+sum[i-1];
        if(sum[i]>max) max=sum[i];
    }
    printf("%lld",max);
    return 0;
```

```
}
```

题目十三 二分查找 第一次出现，最后一次出现

```
#include<stdio.h>
#include<stdlib.h>
long long int n,m;
long long int a[2000000];
int Bsearch(long long int left,long long int right,long long int key)
{
	long long int mid;
	while(left<=right)
	{
		mid=(left+right)/2;
		if(a[mid]==key) return mid;
		else if(a[mid]<key)
		{
			left=mid+1;
		}
		else
		{
			right=mid-1;
		}
	}
	return -1;
}
int main()
{
	long long int i,j,x,op,y;
	scanf("%lld%lld",&n,&m);
	for(i=0;i<n;i++) scanf("%lld",&a[i]);
	for(i=0;i<m;i++)
	{
		scanf("%lld%lld",&op,&x);
		y=Bsearch(0,n-1,x);
		if(y==-1) printf("Not Found\n");
		else
		{
			if(op==1)
			{
				while(a[y/2]==a[y]&&y>=3) y=y/2;
				while(a[y-1]==a[y]&&y-1>=0) y=y-1;
			}
			else if(op==2)
			{
```

```
                while(a[y*2]==a[y]&&y*2<n) y=y*2;
                while(a[y+1]==a[y]&&y+1<n) y=y+1;
            }
            printf("%lld\n",y+1);
        }
    }
    return 0;
}
```

题目十四 堆初见

```
#include<stdio.h>
int heap[3000000],heap_size=0;
void insert(int d)//加入堆，下标从 1 开始
{
    heap_size++;
    heap[heap_size]=d;
    int now=heap_size,next,tmp;
    while(now>1)
    {
        next=now/2;
        if(heap[now]>=heap[next]) return;
        tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
        now=now/2;
    }
    return;
}
int del()//删除并返回堆顶元素
{
    int res=heap[1];
    heap[1]=heap[heap_size];
    heap_size--;
    int now=1,next,tmp;
    while(now*2<=heap_size)
    {
        next=now*2;
        if(next<heap_size&&heap[next+1]<heap[next]) next++;
        if(heap[now]<=heap[next]) return res;
        tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
        now=next;
    }
    return res;
}
int main()
```

```
{
	int n,i,x,num;
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&x);
		if(x==1)
		{
			scanf("%d",&num);
			insert(num);
		}
		if(x==2)
		{
			del();
		}
		if(x==3) printf("%d\n",heap[1]);
	}
	int j=heap_size;
	for(i=1;i<=j;i++) printf("%d ",del());
	return 0;
}
```

题目十五 判断是小根堆还是大根堆

```
#include<stdio.h>
int heap[200000],x;
int judge1()
{
	int i;
	for(i=1;i<=x;i++)
	{
		if(i*2<=x)
		{
			if(heap[i]>heap[i*2]) return -1;
		}
		if(i*2+1<=x)
		{
			if(heap[i]>heap[i*2+1]) return -1;
		}
	}
	return 0;
}
int judge2()
{
	int i;
```

```
    for(i=1;i<=x;i++)
    {
        if(i*2<=x)
        {
            if(heap[i]<heap[i*2]) return -1;
        }
        if(i*2+1<=x)
        {
            if(heap[i]<heap[i*2+1]) return -1;
        }
    }
    return 0;
}
int main()
{
    int n,i,j;
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        scanf("%d",&x);
        for(j=1;j<=x;j++) scanf("%d",&heap[j]);
        if(judge1()==0) printf("Min heap\n");
        if(judge2()==0) printf("Max heap\n");
        if(judge1()==-1&&judge2()==-1) printf("Not a heap!\n");
    }
    return 0;
}
```

题目十六 滑动窗口（优先队列）

```
#include<stdio.h>
int a[2000000];
int max[2000000],min[2000000],max_i=0,min_i=0;
int stack[2000000];
int main()
{
    int n,k;
    scanf("%d%d",&n,&k);
    int i,max=0,min;
    for(i=0;i<n;i++) scanf("%d",&a[i]);
    int head=0,tail=-1;
    for(i=0;i<n;i++)
    {
        if(head<=tail&&stack[head]<i-k+1) head++;
```

```
        while(head<=tail&&a[i]<a[stack[tail]]) tail--;
        tail++;
        stack[tail]=i;
        if(i>=k-1) printf("%d ",a[stack[head]]);
    }
    printf("\n");
    head=0,tail=-1;
    for(i=0;i<n;i++)
    {
        if(head<=tail&&stack[head]<i-k+1) head++;
        while(head<=tail&&a[i]>=a[stack[tail]]) tail--;
        tail++;
        stack[tail]=i;
        if(i>=k-1) printf("%d ",a[stack[head]]);
    }
    return 0;
}
```

题目十七 堆操作

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
typedef struct node{
    int height;
    int num;
}LNode,*linklist;
LNode heap[3000000];
int heap_size=0;
void insert(int height,int num)//加入堆，下标从 1 开始
{
    heap_size++;
    heap[heap_size].height=height;
    heap[heap_size].num=num;
    int now=heap_size,next;
    LNode tmp;
    while(now>1)
    {
        next=now/2;
        if(heap[now].height>=heap[next].height) return;
        tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
        now=now/2;
    }
    return;
}
```

```
void del()//删除堆顶元素
{
	heap[1]=heap[heap_size];
	heap_size--;
	int now=1,next;
	LNode tmp;
	while(now*2<=heap_size)
	{
		next=now*2;
		if(next<heap_size&&heap[next+1].height<heap[next].height) next++;
		if(heap[now].height<=heap[next].height) return;
		tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
		now=next;
	}
	return ;
}
void delk(int k)//删除第 k 个种植的绿藤
{
	int i,goal;
	for(i=1;i<=heap_size;i++)
	{
		if(heap[i].num==k)
		{
			goal=i;
		}
	}
	heap[goal]=heap[heap_size];
	heap_size--;
	int now=goal,next;
	LNode tmp;
	while(now>1)//往上比较
	{
		next=now/2;
		if(heap[now].height>=heap[next].height) break;
		tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
		now=now/2;
	}
	while(now*2<=heap_size)//往下比较
	{
		next=now*2;
		if(next<heap_size&&heap[next+1].height<heap[next].height) next++;
		if(heap[now].height<=heap[next].height) return;
		tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
		now=next;
```

```
    }
    return ;
}
void fchange(int k,int x)//修改第 k 个种植的绿藤的高度，将其变为 x
{
    int i,goal;
    for(i=1;i<=heap_size;i++)
    {
        if(heap[i].num==k)
        {
            goal=i;
        }
    }
    heap[goal].height=x;
    int now=goal,next;
    LNode tmp;
    while(now>1)//往上比较
    {
        next=now/2;
        if(heap[now].height>=heap[next].height) break;
        tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
        now=now/2;
    }
    while(now*2<=heap_size)//往下比较
    {
        next=now*2;
        if(next<heap_size&&heap[next+1].height<heap[next].height) next++;
        if(heap[now].height<=heap[next].height) return;
        tmp=heap[now];heap[now]=heap[next];heap[next]=tmp;
        now=next;
    }
    return ;
}
int main()
{
    int n,i,num=1,height,k,x;
    char c[20];
    scanf("%d",&n);
    while(n!=0)
    {
        scanf("%s",c);
        if(strcmp(c,"I")==0)
        {
            scanf("%d",&height);
```

```
            insert(height,num);
            num++;
        }
        else if(strcmp(c,"PM")==0)
        {
            printf("%d\n",heap[1].height);
        }
        else if(strcmp(c,"DM")==0)
        {
            del();
        }
        else if(strcmp(c,"D")==0)
        {
            scanf("%d",&k);
            delk(k);
        }
        else if(strcmp(c,"C")==0)
        {
            scanf("%d%d",&k,&x);
            fchange(k,x);
        }
        n--;
    }
}
```

题目十八 第 k 小（两个堆）

```
#include<stdio.h>
#include<stdlib.h>
int heap1[3000000],heap_size1=0;
int heap2[3000000],heap_size2=0;
void insert1(int d)//加入堆，下标从 1 开始
{
    heap_size1++;
    heap1[heap_size1]=d;
    int now=heap_size1,next,tmp;
    while(now>1)
    {
        next=now/2;
        if(heap1[now]<=heap1[next]) return;
        tmp=heap1[now];heap1[now]=heap1[next];heap1[next]=tmp;
        now=now/2;
    }
    return;
}
```

```
int del1()//删除并返回堆顶元素
{
	int res=heap1[1];
	heap1[1]=heap1[heap_size1];
	heap_size1--;
	int now=1,next,tmp;
	while(now*2<=heap_size1)
	{
		next=now*2;
		if(next<heap_size1&&heap1[next+1]>heap1[next]) next++;
		if(heap1[now]>=heap1[next]) return res;
		tmp=heap1[now];heap1[now]=heap1[next];heap1[next]=tmp;
		now=next;
	}
	return res;
}
void insert2(int d)//加入堆，下标从 1 开始
{
	heap_size2++;
	heap2[heap_size2]=d;
	int now=heap_size2,next,tmp;
	while(now>1)
	{
		next=now/2;
		if(heap2[now]>=heap2[next]) return;
		tmp=heap2[now];heap2[now]=heap2[next];heap2[next]=tmp;
		now=now/2;
	}
	return;
}
int del2()//删除并返回堆顶元素
{
	int res=heap2[1];
	heap2[1]=heap2[heap_size2];
	heap_size2--;
	int now=1,next,tmp;
	while(now*2<=heap_size2)
	{
		next=now*2;
		if(next<heap_size2&&heap2[next+1]<heap2[next]) next++;
		if(heap2[now]<=heap2[next]) return res;
		tmp=heap2[now];heap2[now]=heap2[next];heap2[next]=tmp;
		now=next;
	}
```

```
    return res;
}
int main()
{
    int n,k,x,num;
    scanf("%d%d",&n,&k);
    while(n!=0)
    {
        scanf("%d",&x);
        if(x==1)
        {
            if(heap_size1==0) printf("-1\n");
            else printf("%d\n",heap1[1]);
        }
        if(x==2)
        {
            scanf("%d",&num);
            if(heap_size1<k) insert1(num);
            else if(heap_size1>=k&&num>=heap1[1]) insert2(num);
            else if(heap_size1>=k&&num<heap1[1])
            {
                insert2(heap1[1]);
                del1();
                insert1(num);
            }
        }
        if(x==3)
        {
            if(heap_size1==0) continue;
            else
            {
                del1();
                if(heap_size2>0)
                {
                    insert1(heap2[1]);
                    del2();
                }
            }
        }
        n--;
    }
}
```

题目十九 01 背包

```
#include<stdio.h>
int luck[2000],weight[2000];
long long int dp[200000];
long long int max(long long int a,long long int b)
{
	if(a>=b) return a;
	else return b;
}
//f(v)=max(f(v),f(v-weight[i])+luck[i])
int main()
{
	int n,k;
	scanf("%d%d",&n,&k);
	int i,j;
	for(i=0;i<n;i++) scanf("%d%d",&luck[i],&weight[i]);
	for(i=0;i<n;i++)
	{
		for(j=k;j>=weight[i];j--)
		{
			dp[j]=max(dp[j],dp[j-weight[i]]+luck[i]);
		}
	}
	printf("%lld",dp[k]);
	return 0;
}
```

题目二十 钢条切割

```
#include<stdio.h>
long long int a[200000],b[200000];

int main()
{
	int n,i,j;
	long long int max=0,x;
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		scanf("%d",&a[i]);
	}
	b[1]=a[1];
	for(i=2;i<=n;i++)
	{
		for(j=1;j<i;j++)
		{
```

```
                x=b[j]+b[i-j];
                if(x>max) max=x;
            }
            if(a[i]>max) b[i]=a[i];
            else b[i]=max;
            max=0;
        }
        printf("%lld",b[n]);
        return 0;
}
```

题目二十一 生命之水 不能相邻

```
#include<stdio.h>
long long int a[200000],b[200000];
int main()
{
        int t,i,n;
        scanf("%d",&t);
        while(t!=0)
        {
            scanf("%d",&n);
            for(i=0;i<n;i++) scanf("%d",&a[i]);
            if(n==1)
            {
                printf("%d\n",a[0]);
                continue;
            }
            b[0]=a[0];
            if(a[1]>a[0]) b[1]=a[1];
            else b[1]=a[0];
            for(i=2;i<n;i++)
            {
                if(b[i-2]+a[i]>b[i-1]) b[i]=b[i-2]+a[i];
                else b[i]=b[i-1];
            }
            printf("%lld\n",b[n-1]);
            t--;
        }
}
```

题目二十二 最大值的最小值（把数组分 k 段，求最大值的最小值）

```
#include<stdio.h>
int a[200000];
int judge(int mid,int divided_num,int n)
```

```
{
    int i,num=1,sum=0;
    for(i=0;i<n;i++)
    {
        sum=a[i]+sum;
        if(sum>mid)            //从左到右将数组元素之和与 mid 比较，如是大于则再起一
段，最后看段的大小
        {
            sum=a[i];
            num++;
        }
    }
    if(num>divided_num)                //若是段大于段数，则必然不和条件
        return 0;
    else
        return 1;

}
int merge(int left,int right,int divided_num,int n)//divided_num 为分段个数，n 为数组个数
{
    if(left>=right) return left;
    else
    {
        int mid=left+(right-left)/2;
        if(judge(mid,divided_num,n)==1) return merge(left,mid,divided_num,n);
        else    return merge(mid+1,right,divided_num,n);
    }
}
int main()
{
    int n,divided_num,i;
    int max=0,min=0;
    scanf("%d%d",&n,&divided_num);
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
        max=max+a[i];
        if(a[i]>min) min=a[i];
    }
    int ans=merge(min,max,divided_num,n);
    printf("%d",ans);
    return 0;
}
```

题目二十三 相邻元素不互质

```
#include<stdio.h>
int list[2000],ans[2000];
int judge(int a,int b)
{
	int tmp,c;
	if(a<b)
	{
		tmp=a;
		a=b;
		b=tmp;
	}
	while(b!=0)
	{
		c=a;
		a=b;
		b=c%b;
	}
	return a;//a 为最大公约数
}
int main()
{
	int n,i,j,max=0;;
	scanf("%d",&n);
	for(i=0;i<n;i++) scanf("%d",&list[i]);
	ans[0]=1;
	for(i=1;i<n;i++)
	{
		for(j=0;j<i;j++)
		{
			if(judge(list[i],list[j])!=1&&ans[j]>max) max=ans[j];
		}
		if(max!=0) ans[i]=max+1;
		else ans[i]=1;
		max=0;
	}
	for(i=0;i<n;i++)
	{
		if(ans[i]>max) max=ans[i];
	}
	printf("%d",max);
	return 0;
}
```

题目二十四

```
#include<stdio.h>
long long int a[2][200000],b[2][200000],ans[200000];
long long int max(long long int x,long long int y)
{
	if(x>=y) return x;
	else return y;
}
int main()
{
	long long int n,i,j;
	scanf("%lld",&n);
	for(i=0;i<2;i++)
	{
		for(j=0;j<n;j++)
		{
			scanf("%lld",&a[i][j]);
		}
	}
	b[0][n-1]=a[0][n-1];
	for(i=n-2;i>=0;i--)
	{
		b[0][i]=b[0][i+1]+a[0][i];
	}
	b[1][0]=a[1][0];
	for(i=1;i<n;i++)
	{
		b[1][i]=b[1][i-1]+a[1][i];
	}
	for(i=0;i<n;i++)
	{
		if(i==0) ans[i]=b[0][1];
		else if(i==n-1) ans[i]=b[1][i-1];
		else
		{
			ans[i]=max(b[0][i+1],b[1][i-1]);
		}
	}
	int ret=1e9;
	for(i=0;i<n;i++)
	{
		if(ans[i]<ret) ret=ans[i];
	}
	printf("%lld",ret);
```

```
	return 0;
}
```

题目二十五 最长公共子序列

```
#include<stdio.h>
#include<string.h>
//if(i==0||j==0) a[i][j]=0;
//else if(x==y) a[i][j]=a[i-1][j-1]+1;
//else if(x!=y) a[i][j]=max(a[i-1][j],a[i][j-1]);
int max(int aa,int bb)
{
	if(aa>=bb) return aa;
	else return bb;
}
int a[2000][2000];
int b[2000][2000];
char x[2000];
char y[2000];
char ans[2000];
int ans_i=0;
int main()
{
	int i,k,j,len1,len2,mmax=0;
	scanf("%d",&k);
	scanf("%s",x);
	scanf("%s",y);
	len1=strlen(x);len2=strlen(y);
	for(i=1;i<=len1;i++)
	{
		for(j=1;j<=len2;j++)
		{
			if(x[i-1]==y[j-1])
			{
				a[i][j]=1+a[i-1][j-1];
				b[i][j]=3;
			}
			else
			{
				a[i][j]=max(a[i-1][j],a[i][j-1]);
				if(a[i-1][j]>a[i][j-1]) b[i][j]=2;
				else b[i][j]=1;
			}
		}
	}
```

```
    if(k==0) printf("%d",a[len1][len2]);
    if(k==1)
    {
        i=len1;j=len2;
        while(b[i][j]!=0)
        {
            if(b[i][j]==2) i--;
            else if(b[i][j]==1) j--;
            else if(b[i][j]==3)
            {
                ans[ans_i]=x[i-1];
                ans_i++;
                i--;j--;
            }
        }
        for(i=ans_i-1;i>=0;i--) printf("%c",ans[i]);
    }
    return 0;
}
```

题目二十六 矩阵乘法最多次和最少次

```
#include<stdio.h>
#include<string.h>
// 如果 i=j，m[i,j]=0
// 如果 i<j，m[i,j]=min{m[i,k]+m[k+1,j]+p(i-1)p(k)p(j)}  i<=k<j
long long int line[2000],row[2000];
long long int dp_min[2000][2000],dp_max[2000][2000];
int main()
{
    long long int n,i,x,j,k,min=9e18,max=-1;
    scanf("%lld",&n);
    for(i=0;i<=n;i++)//读取矩阵 n 个
    {
        scanf("%lld",&x);
        if(i==0) line[i]=x;
        else if(i==n) row[i-1]=x;
        else row[i-1]=line[i]=x;
    }
    for(x=1;x<n;x++)//动态规划
    {
        for(i=0;i<n-x;i++)
        {
            j=x+i;
            for(k=i;k<j;k++)
```

```
                {
                    if(dp_min[i][k]+dp_min[k+1][j]+line[i]*row[j]*row[k]<min)
                    {
                        min=dp_min[i][k]+dp_min[k+1][j]+line[i]*row[j]*row[k];
                    }
                    if(dp_max[i][k]+dp_max[k+1][j]+line[i]*row[j]*row[k]>max)
                    {
                        max=dp_max[i][k]+dp_max[k+1][j]+line[i]*row[j]*row[k];
                    }
                }
                dp_min[i][j]=min;
                dp_max[i][j]=max;
                min=9e18;
                max=-1;
            }
        }
        printf("%lld\n",dp_min[0][n-1]);
        printf("%lld\n",dp_max[0][n-1]);
        return 0;
}
```

题目二十七 最优搜索二叉树

```
#include<stdio.h>
long long int a[1000],b[1000];
long long int dp[2000][2000],w[2000][2000];
int main()
{
        long long int n,i,j,l,r,sum;
        scanf("%lld",&n);
        for(i=1;i<=n;i++) scanf("%lld",&a[i]);
        for(i=0;i<=n;i++) scanf("%lld",&b[i]);
        for(i=1;i<=n+1;i++) dp[i][i-1]=w[i][i-1]=b[i-1];
        for(l=1;l<=n;l++)
        {
            for(i=1;i<=n-l+1;i++)
            {
                j=i+l-1;
                dp[i][j]=9e18;
                w[i][j]=w[i][j-1]+b[j]+a[j];
                for(r=i;r<=j;r++)
                {
                    sum=dp[i][r-1]+dp[r+1][j]+w[i][j];
                    if(sum<dp[i][j]) dp[i][j]=sum;
                }
```

```
            }
        }
        for(i=0;i<=n;i++) dp[1][n]=dp[1][n]-b[i];//这一句不是最优二叉搜索树，其余完全一致
        printf("%lld",dp[1][n]);
        return 0;
}
```

题目二十八 完全背包

```
#include<stdio.h>
//dp[i][j] = max(dp[i-1][j], dp[i][j-w[i]]+v[i]) ( j >= w[i] )
long long int v[200000],w[200000];
long long int dp[200000];
long long int max(long long int a,long long int b)
{
        if(a>=b) return a;
        else return b;
}
int main()
{
        long long int i,j,n,m;
        scanf("%lld%lld",&n,&m);
        for(i=0;i<n;i++) scanf("%lld%lld",&v[i],&w[i]);
        for(i=0;i<n;i++)
        {
                for(j=w[i];j<=m;j++)
                {
                        dp[j]=max(dp[j],dp[j-w[i]]+v[i]);//与 01 背包不同，此为正向
                }
        }
        printf("%lld\n",dp[m]);
        return 0;
}
```

题目二十九 最长波动子数组

```
#include<stdio.h>
int a[2000][2],b[2000];
int max(int x,int y)
{
        if(x>=y) return x;
        else return y;
}
int main()
{
        int n,i;
```

```
	scanf("%d",&n);
	a[0][1]=a[0][0]=1;
	//a[i][0]前大于后啊 a[i][1]后大于前
	for(i=0;i<n;i++) scanf("%d",&b[i]);
	for(i=1;i<n;i++)
	{
		if(b[i]>b[i-1])
		{
			a[i][1]=max(a[i-1][0]+1,a[i-1][1]);
			a[i][0]=a[i-1][0];
		}
		else if(b[i]<b[i-1])
		{
			a[i][0]=max(a[i-1][1]+1,a[i-1][0]);
			a[i][1]=a[i-1][1];
		}
		else
		{
			a[i][0]=a[i-1][0];
			a[i][1]=a[i-1][1];
		}
	}
	printf("%d",max(a[n-1][0],a[n-1][1]));
	return 0;
}
```

题目三十 更改字符串

```
#include<stdio.h>
char a[3000],b[3000];
int dp[3000][3000];
int min(int xx,int yy)
{
	if(xx<yy) return xx;
	else return yy;
}
int main()
{
	int i,j,len1,len2;
	scanf("%s",a);
	scanf("%s",b);
	len1=strlen(a);
	len2=strlen(b);
	for(i=0;i<len1;i++) dp[i][0]=i;
	for(j=0;j<len2;j++) dp[0][j]=j;
```

```
    for(i=1;i<=len1;i++)
    {
        for(j=1;j<=len2;j++)
        {
            if(a[i-1]==b[j-1]) dp[i][j]=dp[i-1][j-1];
            else dp[i][j]=min(dp[i-1][j]+1,dp[i][j-1]+1);
        }
    }
    printf("%d",dp[len1][len2]);
    return 0;
}
```

题目三十一 最多游玩个数

```
#include<stdio.h>
#include<stdlib.h>
int a[2000000][2];
int cmp(const int *p1,const int *p2)
{
    return p1[1]-p2[1];
}
int main()
{
    int n,i,sum=1,x;
    scanf("%d",&n);
    for(i=0;i<n;i++) scanf("%d%d",&a[i][0],&a[i][1]);
    qsort(a,n,sizeof(a[0]),cmp);
    x=a[0][1];
    for(i=0;i<n;i++)
    {
        if(a[i][0]>=x)
        {
            x=a[i][1];
            sum++;
        }
    }
    printf("%d",sum);
    return 0;

}
```

题目三十二 锯钢条的哈夫曼树

```
#include<iostream>
#include<cstdio>
#include<algorithm>
```

```
#include<queue>
using namespace std;
priority_queue<int, vector<int>, greater<int> > heap;
int main()
{
	long long int n,i,x;
	long long int result=0;
	cin>>n;
	for (i=0;i<n;i++)
	{
		cin>>x;
		heap.push(x);
	}
	while (heap.size()!=1)
	{
		long long int y1,y2;
		y1=heap.top();
		heap.pop();
		y2=heap.top();
		heap.pop();
		result=y2+y1+result;
		heap.push(y1+y2);
	}
	cout<<2*result<<endl;
	return 0;
}
```

题目三十三

```
#include<stdio.h>
#include<string.h>
char a[200000];
long long int sum[200000][2];
int main()
{
	scanf("%s",a);
	int i,len;
	len=strlen(a);
	sum[0][0]=1;sum[0][1]=0;
	for(i=1;i<len;i++)
	{
		if((a[i]=='u'&&a[i-1]=='u'&&a[i-2]!='u')||(a[i]=='n'&&a[i-1]=='n'&&a[i-2]!='n'))
sum[i][1]=sum[i][0]=sum[i-1][0]%1000000007;
		else	if((a[i]=='u'&&a[i-1]=='u'&&a[i-2]=='u')||(a[i]=='n'&&a[i-1]=='n'&&a[i-
2]=='n'))
```

```
            {
                sum[i][0]=(sum[i-1][0]+sum[i-1][1])%1000000007;
                sum[i][1]=sum[i-1][0];
            }
            else
            {
                sum[i][0]=(sum[i-1][0]+sum[i-1][1])%1000000007;
                sum[i][1]=0;
            }
        }
        printf("%lld",(sum[len-1][1]+sum[len-1][0])%1000000007);
        return 0;
}
```

题目三十四 DFS

```
#include<stdio.h>
char a[200][200];
int b[200][200];
typedef struct node
{
        int x;
        int y;
}LNode,*linklist;
LNode queue[20000];
int top=0;
int last=-1;
int n,m,sum=1;
void BFS(int x_i,int x_j)
{
        a[x_i][x_j]='x';
        if(x_i-1>=0&&a[x_i-1][x_j]=='b')
        {
                b[x_i-1][x_j]=1;
                queue[top].x=x_i-1;
                queue[top].y=x_j;
                top++;
                a[x_i-1][x_j]='x';
                sum++;
        }
        if(x_j-1>=0&&a[x_i][x_j-1]=='b')
        {
                b[x_i][x_j-1]=1;
                queue[top].x=x_i;
                queue[top].y=x_j-1;
```

```
            top++;
            a[x_i][x_j-1]='x';
            sum++;
        }
        if(x_i+1>=0&&a[x_i+1][x_j]=='b')
        {
            b[x_i+1][x_j]=1;
            queue[top].x=x_i+1;
            queue[top].y=x_j;
            top++;
            a[x_i+1][x_j]='x';
            sum++;
        }
        if(x_j+1>=0&&a[x_i][x_j+1]=='b')
        {
            b[x_i][x_j+1]=1;
            queue[top].x=x_i;
            queue[top].y=x_j+1;
            top++;
            a[x_i][x_j+1]='x';
            sum++;
        }
        last++;
        if(last<top) BFS(queue[last].x,queue[last].y);
        else return ;
}
void f(int x_i,int x_j)
{
        a[x_i][x_j]='y';
        if(x_i-1>=0&&a[x_i-1][x_j]=='w')
        {
            b[x_i-1][x_j]=1;
            queue[top].x=x_i-1;
            queue[top].y=x_j;
            top++;
            a[x_i-1][x_j]='y';
            sum++;
        }
        if(x_j-1>=0&&a[x_i][x_j-1]=='w')
        {
            b[x_i][x_j-1]=1;
            queue[top].x=x_i;
            queue[top].y=x_j-1;
            top++;
```

```
            a[x_i][x_j-1]='y';
            sum++;
        }
        if(x_i+1>=0&&a[x_i+1][x_j]=='w')
        {
            b[x_i+1][x_j]=1;
            queue[top].x=x_i+1;
            queue[top].y=x_j;
            top++;
            a[x_i+1][x_j]='y';
            sum++;
        }
        if(x_j+1>=0&&a[x_i][x_j+1]=='w')
        {
            b[x_i][x_j+1]=1;
            queue[top].x=x_i;
            queue[top].y=x_j+1;
            top++;
            a[x_i][x_j+1]='y';
            sum++;
        }
        last++;
        if(last<top) f(queue[last].x,queue[last].y);
        else return ;
}
int main()
{
        int i,j,ii,jj,max=0;
        scanf("%d%d",&n,&m);
        for(i=0;i<n;i++) scanf("%s",&a[i]);
        for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {
                if(a[i][j]=='b'&&b[i][j]==0)
                {
                    BFS(i,j);
                    if(sum>max) max=sum;
                    for(ii=0;ii<n;ii++)
                    {
                        for(jj=0;jj<m;jj++)
                        {
                            if(a[ii][jj]=='x') a[ii][jj]='b';
                        }
```

```
                }
                top=0;
                last=-1;
                sum=1;
            }
            else if(a[i][j]=='w'&&b[i][j]==0)
            {
                f(i,j);
                if(sum>max) max=sum;
                for(ii=0;ii<n;ii++)
                {
                    for(jj=0;jj<m;jj++)
                    {
                        if(a[ii][jj]=='y') a[ii][jj]='w';
                    }
                }
                top=0;
                last=-1;
                sum=1;
            }

        }
    }
    printf("%d",max);
}
```

题目三十五 克鲁斯卡尔

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
typedef struct node
{
    int x;
    int y;
    int value;
    struct node *next;
}LNode,*linklist;
int cmp(const struct node *p1,const struct node *p2)
{
    return (int)(p1->value-p2->value);
}
LNode edgelist[600000];
int ei=0;
int n,m;
```

```
long long int sum=0;
int father[500000];
void make()//初始化，每个节点父节点是他自己 ，每个父节点是一个集合
{
	int i;
	for(i=0;i<n;i++) father[i]=i;
}

int find_father(int x)//找父节点
{
	if(x!=father[x])
	{
		x=father[x];
		x=find_father(x);
	}
	return x;
}
void Union(int x, int y)//将 x,y 合并到同一个集合
{
	x=find_father(x);
	father[x]=find_father(y);
}
void Kruskal()
{
	int i;
	make();
	for(i=0;i<ei;i++)//将边的顺序按从小到大取出来
	{
		if(find_father(edgelist[i].x)!=find_father(edgelist[i].y))
		 //如果两个顶点不在一个集合中，就并到一个集合里，生成树的长度加上这条边
的长度
		{
			Union(edgelist[i].x,edgelist[i].y);  //合并两个顶点到一个集合
			sum=sum+edgelist[i].value;
		}
	}
	return;
}
int main()
{
	int i,j,x,y,z;
	scanf("%d%d",&n,&m);
	for(i=0;i<m;i++)//输入边
	{
```

```
        scanf("%d%d%d",&x,&y,&z);
        edgelist[ei].x=x;edgelist[ei].y=y;edgelist[ei].value=z;ei++;
    }
    qsort(edgelist,ei,sizeof(struct node),cmp);//按边的长度排序
    Kruskal();
    printf("%lld",sum);
    return 0;
}
```

题目三十六 拓扑排序 大的先出

```
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
priority_queue<int> q;
struct node//链式向前星
{
    int from;
    int to;
    int before;//同起点上一条边编号 ， -1 为无
    int value;
}edge[500000];
int head[500000];//以 i 为起点上一条
int in[500000];//入度
int n,m;
void tuopu()
{
    int i,j;
    for(i=n;i>=1;i--)
    {
        if(in[i]==0)
        {
            q.push(i);
            in[i]=-1;
        }
    }
    while(q.empty()!=1)
    {
        int x=q.top();
        q.pop();
        cout<<x<<' ';
        for(j=head[x];j!=-1;j=edge[j].before)
        {
            in[edge[j].to]--;
```

```
                if(in[edge[j].to]==0)
                {
                    in[edge[j].to]=-1;
                    q.push(edge[j].to);
                }
            }
        }
}
int main()
{
    int i,j,x,y;
    cin>>n>>m;
    for(i=0;i<=n;i++) head[i]=-1;
    for(i=0;i<m;i++)
    {
        cin>>x>>y;
        edge[i].from=x;edge[i].to=y;edge[i].before=head[x];head[x]=i;in[y]++;
    }
    tuopu();
    cout<<"\n";
    return 0;
}
```

题目三十七 弗洛伊德算法

```
#include<stdio.h>
long long int n,m,p;
long long int a[510][510];
int main()
{
    long long int i,j,x,y,z,k;
    for(i=0;i<510;i++)
    {
        for(j=0;j<510;j++)
        {
            if(i==j) a[i][j]=0;
            else a[i][j]=1e9;
        }
    }
    scanf("%lld%lld%lld",&n,&m,&p);
    for(i=0;i<m;i++)
    {
        scanf("%lld%lld%lld",&x,&y,&z);
        if(a[x][y]>z) a[x][y]=z;
    }
```

```
    for(k=1;k<=n;k++)
    {
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                if(a[i][j]>a[i][k]+a[k][j]) a[i][j]=a[i][k]+a[k][j];
            }
        }
    }
    for(i=0;i<p;i++)
    {
        scanf("%lld%lld",&x,&y);
        if(a[x][y]==1e9) printf("-1\n");
        else printf("%lld\n",a[x][y]);
    }
}
```

题目三十八 最大流问题

```
//网络流 Dinic 算法
#include<iostream>
#include<algorithm>
#include<queue>
#include<cstdio>
#include<cstring>
using namespace std;
struct node//链式向前星
{
    int to;int value;int before;
}edge[10020];
int head[10020],dep[10020];
int inque[10020];
int cnt=1;
int n,m,s,t;
int min(int a,int b)
{
    if(a<b) return a;
    else return b;
}
long long int dfs(int pos,long long int low)//当前位置和最小残量，dfs 用于寻找增广路
{
    long long int out=0;
    if(pos==t)//到达汇点
    {
```

```
        return low;//返回最小残量（当它为 0 时说明没有增广路了）
    }
    for(int i=head[pos];i!=0&&low!=0;i=edge[i].before)
    {
        int v=edge[i].to;
        if(edge[i].value!=0&&dep[v]==dep[pos]+1)//小优化：仅当 v 在当前位置的下一层
中才进行查找是否有增广路
        {
            long long int rlow=dfs(v,min(low,edge[i].value));
            low-=rlow;//该点使用的流量增加
            edge[i].value-=rlow;//过去的边是剩余可支配的量
            edge[i^1].value+=rlow;//反向边+流量
            out+=rlow;
        }
     }
     if(out==0) dep[pos]=0x3f3f3f3f;
     return out;//返回该点已使用流量
}
bool bfs()//给增广路上的点分层
{
    std::queue<int>q;
    memset(dep,0x3f3f3f3f,sizeof(dep));
    memset(inque,0,sizeof(inque));
    dep[s]=0;//源点深度当然为 0
    q.push(s);
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        inque[u]=0;//不在队伍中了
        for(int i=head[u];i!=0;i=edge[i].before)
        {
            int v=edge[i].to;//通向的点
            if(edge[i].value!=0&&dep[v]>dep[u]+1)//如果容量不为 0 且在 u 点之前还没有
被搜到
            {
                dep[v]=dep[u]+1;
                if(inque[v]==0)//如果点 v 不在当前队伍中
                {
                    q.push(v);
                    inque[v]=1;
                }
            }
        }
```

```
        }
        if (dep[t]!=0x3f3f3f3f)//只要汇点被搜到了，就还有增广路
            return 1;
        return 0;
}
int main()
{
        long long int maxflow=0;
        cin>>n>>m>>s>>t;
        int i,a,b,c;
        for(i=0;i<m;i++)
        {
            cin>>a>>b>>c;
            cnt++;edge[cnt].to=b;edge[cnt].value=c;edge[cnt].before=head[a];head[a]=cnt;//
正边为偶数，从 2 开始
            cnt++;edge[cnt].to=a;edge[cnt].value=0;edge[cnt].before=head[b];head[b]=cnt;//
反边为奇数，从 3 开始，初始流量为 0
        }
        while(bfs())//如果还有增广路就继续 dfs
            maxflow=maxflow+dfs(s, 1e9);
        cout<<maxflow<<endl;
        return 0;
}
```

题目三十九 公共祖先结点

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
struct node
{
        int to;
        int before;
}edge[200000];
int lg[200000];
int head[200000],depth[200000],fa[200000][22];//fa[u][i]表示 u 的第 2 的 i 次方个祖先（fa[u][0]
就是 u 的父亲）
int n,cnt=0;
void dfs(int,int);
int LCA(int,int);
int main()
{
        int fu,sum;
        int a,b,i,j,q;
```

```
    cin>>n;
    for(i=1;i<=n-1;i++)
    {
        cin>>a>>b;
        cnt++;edge[cnt].to=b;edge[cnt].before=head[a];head[a]=cnt;
        cnt++;edge[cnt].to=a;edge[cnt].before=head[b];head[b]=cnt;
    }
    for(i=1;i<=n;i++) lg[i]=lg[i-1]+(1<<lg[i-1]==i);
    dfs(1,0);
    cin>>q;
    for(i=0;i<q;i++)
    {
        cin>>a>>b;
        fu=LCA(a,b);
        sum=depth[a]+depth[b]-2*depth[fu];
        if(sum%2==0) cout<<"YE5"<<endl;
        else cout<<"N0"<<endl;
    }
    return 0;
}
void dfs(int now,int father)
{
    int i;
    fa[now][0]=father;depth[now]=depth[father]+1;
    for(i=1;i<=lg[depth[now]];i++)
        fa[now][i]=fa[fa[now][i-1]][i-1]; //这个转移可以说是算法的核心之一
                                          //意思是 now 的 2^i 祖先等于 now 的 2^(i-1)祖先
的 2^(i-1)祖先
                                                //2^i = 2^(i-1) + 2^(i-1)
    for(i=head[now];i;i=edge[i].before)
        if(edge[i].to!=father) dfs(edge[i].to,now);
}
int LCA(int x,int y)
{
    int k;
    if(depth[x]<depth[y])
    {
        int tmp=x;x=y;y=tmp;
    }
    while(depth[x]>depth[y])
        x=fa[x][lg[depth[x]-depth[y]]-1]; //先跳到同一深度
    if(x==y) //如果 x 是 y 的祖先，那他们的 LCA 肯定就是 x 了
    {
        return x;
```

```
    }
    for(k=lg[depth[x]]-1;k>=0;k--) //不断向上跳（lg 就是之前说的常数优化）
    {
        if(fa[x][k]!=fa[y][k])  //因为我们要跳到它们 LCA 的下面一层，所以它们肯定不相等，
如果不相等就跳过去。
        {
            x=fa[x][k];
            y=fa[y][k];
        }
    }
    return fa[x][0];  //返回父节点
}
```

题目四十 拓扑排序求环路

```
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
priority_queue<int> q;
struct node//链式向前星
{
    int from;
    int to;
    int before;//同起点上一条边编号 ， -1 为无
    int value;
}edge[500000];
int head[500000];//以 i 为起点上一条
int in[500000];//入度
int n,m,num=0;
int tuopu()
{
    int i,j;
    for(i=n;i>=1;i--)
    {
        if(in[i]==0)
        {
            q.push(i);
            in[i]=-1;
        }
    }
    while(q.empty()!=1)
    {
        int x=q.top();
        q.pop();
```

```
            num++;
            for(j=head[x];j!=-1;j=edge[j].before)//链式向前星访问套路，需要记住
            {
                in[edge[j].to]--;
                if(in[edge[j].to]==0)
                {
                    in[edge[j].to]=-1;
                    q.push(edge[j].to);
                }
            }
        }
        if(q.empty()==1&&num!=n) return -1;
        else return 0;
}
int main()
{
        int i,j,x,y;
        cin>>n>>m;
        for(i=0;i<=n;i++) head[i]=-1;
        for(i=0;i<m;i++)
        {
            cin>>x>>y;
            edge[i].from=x;edge[i].to=y;edge[i].before=head[x];head[x]=i;in[y]++;
        }
        if(tuopu()==-1) printf("YE5\n");
        else printf("N0\n");
        return 0;
}
```

题目四十一 两个城市通不通 并查集

```
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
int father[500000];
int tmp[500000];
int find_father(int x)//找父节点
{
        if(x!=father[x])
        {
            x=father[x];
            x=find_father(x);
        }
        return x;
```

```
}
int main()
{
	int i,j,x,y,z,cnt=0,n,m;
	cin>>n>>m;
	for(i=0;i<=n;i++) father[i]=i;
	for(i=0;i<m;i++)
	{
		int a,b;
		int flag=0;
		cin>>z>>x>>y;
		if(z==1)
		{
			a=find_father(x);
			b=find_father(y);
			father[x]=a;
			father[y]=b;
			if(a!=b)
			{
				father[b]=a;
			}
		}
		else if(z==2)
		{
			a=find_father(x);
			b=find_father(y);
			if(a!=b) printf("NO\n");
			else printf("YES\n");
		}
	}
	return 0;
}
```

题目四十二 迪杰斯特拉

```
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
struct node
{
	int to,value,before;
}edge[300000];
int head[300000],cnt=0,n,m,vis[300000],dis[300000];
#define P pair<long long,int>
```

```
priority_queue<P,vector<P>,greater<P> >q;
//把最小的元素放在队首的优先队列
void dijkstra(int);
void add(int u,int v,int w)
{
	cnt++;
	edge[cnt].to=v;
	edge[cnt].value=w;
	edge[cnt].before=head[u];
	head[u]=cnt;
}
int main()
{
	cin>>n>>m;
	int i,x,y,z,from,to1,to2;
	cin>>from>>to1>>to2;
	for(i=1;i<=m;i++)
	{
		cin>>x>>y>>z;
		add(x,y,z);
		add(y,x,z);
	}
	int flag=0;
	dijkstra(from);
	x=dis[to1];
	if(x==1e9)
	{
		flag=1;
		printf("-1\n");
	}
	for(i=0;i<300000;i++) vis[i]=dis[i]=0;
	dijkstra(to1);
	x=dis[to2]+x;
	if(dis[to2]==1e9)
	{
		flag=1;
		printf("-1\n");

	}
	for(i=0;i<300000;i++) vis[i]=dis[i]=0;
	dijkstra(from);
	y=dis[to2];
	if(y==1e9&&flag==1)
	{
```

```
            flag=1;
            printf("-1\n");
            return 0;
        }
        for(i=0;i<300000;i++) vis[i]=dis[i]=0;
        dijkstra(to2);
        y=dis[to1]+y;
        if(dis[to1]==1e9&&flag==1)
        {
            printf("-1\n");
            return 0;
        }
        if(x<=y) printf("%d",x);
        else printf("%d",y);
        return 0;
}
void dijkstra(int from)
{
        for(int i=1;i<=n;i++)
        {
            dis[i]=1e9;
        }
        dis[from]=0;
        q.push(make_pair(0,from));
        while(!q.empty())
        //堆为空即为所有点都更新
        {
            int x=q.top().second;
            q.pop();
            //记录堆顶并将其弹出
            if(!vis[x])
            //没有遍历过才需要遍历
            {
                vis[x]=1;
                for(int i=head[x];i;i=edge[i].before)
                //搜索堆顶所有连边
                {
                    int v=edge[i].to;
                    dis[v]=min(dis[v],dis[x]+edge[i].value);
                    //松弛操作
                    q.push(make_pair(dis[v],v));
                }
            }
        }
```

```
}
```

题目四十三 二分图

```
//经典二分图算法。题意：n 个间谍 n 个安全屋，都有一个坐标，间谍最多可以移动距离 d
//问最少有多少人躲不进安全屋
#include<iostream>
#include<cmath>
using namespace std;
int map[300][300],visit[300];
int n,d;
int man[300][2];//存间谍坐标
int house[300];//记录这个房子有没有人住进去了
bool dfs(int x)
{
	int i;
	for(i=1;i<=n;i++)
	{
		if(map[x][i]==1&&visit[i]==0)//有线且本次没被找过
		{
			visit[i]=1;
			if(house[i]==0||dfs(house[i]))//没人住进去或者可以换一个人连线
			{
				house[i]=x;
				return true;
			}
		}
	}
	return false;
}
int main()
{
	int i,j;
	cin>>n>>d;//n 指安全屋与间谍数量
	for(i=1;i<=n;i++) cin>>man[i][0]>>man[i][1];
	for(i=1;i<=n;i++)
	{
		int x,y;
		cin>>x>>y;
		for(j=1;j<=n;j++)
		{
			if(d*d>=(man[j][0]-x)*(man[j][0]-x)+(man[j][1]-y)*(man[j][1]-y))  map[i][j]=1;//
能不能进入
		}
	}
```

```
	int num=0;
	for(i=1;i<=n;i++)
	{
		for(j=1;j<=n;j++) visit[j]=0;
		if(dfs(i)) num++;
	}
	cout<<num<<endl;
	return 0;
}
```

题目四十四 两边迪杰斯特拉

```
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;
struct node
{
	int to,value,before;
}edge[1000010];
long long int head[600000],cnt=0,n,m,k,vis[600000],dis[600000];
long long int dis2[600000],record[600000];
#define P pair<long long int,int>
priority_queue<P,vector<P>,greater<P> >q;
void dijkstra(int);
void add(int u,int v,int w)
{
	cnt++;
	edge[cnt].to=v;
	edge[cnt].value=w;
	edge[cnt].before=head[u];
	head[u]=cnt;
}
int main()
{
	cin>>n>>m>>k;
	int i,x,y,z;
	for(i=1;i<=m;i++)//链式前向星存储
	{
		cin>>x>>y>>z;
		add(x,y,z);
		add(y,x,z);
	}
	for(i=0;i<k;i++)
	{
```

```
            cin>>record[i];
        }
        dijkstra(1);//从 1 出发
        for(i=0;i<=n;i++)//还原
        {
            dis2[i]=dis[i];
            vis[i]=0;
            dis[i]=0;
        }
        dijkstra(n);//从 n 出发
        long long int min=dis2[record[0]]+dis[record[0]];
        for(i=1;i<k;i++)//最短路径
        {
            if(dis2[record[i]]+dis[record[i]]<min) min=dis2[record[i]]+dis[record[i]];
        }
        cout<<min<<endl;
        return 0;
}
void dijkstra(int from)
{
        for(int i=1;i<=n;i++)
        {
            dis[i]=1e9;
        }
        dis[from]=0;
        q.push(make_pair(0,from));//pair 入队需要用 make_pair
        while(!q.empty())
        //堆为空即为所有点都更新
        {
            int x=q.top().second;
            q.pop();
            //记录堆顶并将其弹出
            if(!vis[x])
            //没有遍历过才需要遍历
            {
                vis[x]=1;
                for(int i=head[x];i;i=edge[i].before)
                //搜索堆顶所有连边
                {
                    int v=edge[i].to;
                    dis[v]=min(dis[v],dis[x]+edge[i].value);
                    //松弛操作
                    q.push(make_pair(dis[v],v));
                }
```

```
            }
        }
}
```

题目四十五 克鲁斯卡尔改编

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<math.h>
typedef struct node
{
    int x;
    int y;
    double value;
    struct node *next;
}LNode,*linklist;
int cmp(const struct node *p1,const struct node *p2)
{
    double ret=(double)(p1->value)-(double)(p2->value);
      if(ret > 0){
        return 1;
      }
      else if(ret < 0){
        return -1;
      }
}
LNode edgelist[600000];
int g[2000][2];
int ei=0;
int n,m;
double sum=0;
int father[500000];
void make()//初始化，每个节点父节点是他自己 ， 每个父节点是一个集合
{
    int i;
    for(i=0;i<n;i++) father[i]=i;
}

int find_father(int x)//找父节点
{
    if(x!=father[x])
    {
        x=father[x];
        x=find_father(x);
```

```
    }
    return x;
}
void Union(int x, int y)//将 x,y 合并到同一个集合
{
    x=find_father(x);
    father[x]=find_father(y);
}
void Kruskal()
{
    int i;
    make();
    for(i=0;i<ei;i++)//将边的顺序按从小到大取出来
    {
        if(find_father(edgelist[i].x)!=find_father(edgelist[i].y))
         //如果两个顶点不在一个集合中，就并到一个集合里，生成树的长度加上这条边
的长度
        {
            Union(edgelist[i].x,edgelist[i].y);  //合并两个顶点到一个集合
            sum=sum+edgelist[i].value;
        }
    }
    return;
}
int main()
{
    int i,j,x,y,z;
    scanf("%d",&n);
    for(i=1;i<=n;i++) scanf("%d%d",&g[i][0],&g[i][1]);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            double      value=sqrt((g[i][0]-g[j][0])*(g[i][0]-g[j][0])+(g[i][1]-g[j][1])*(g[i][1]-
g[j][1]));
            edgelist[ei].x=i;edgelist[ei].y=j;edgelist[ei].value=value;ei++;
        }
    }
    qsort(edgelist,ei,sizeof(struct node),cmp);//按边的长度排序
    Kruskal();
    printf("%.2lf",sum);
    return 0;
}
```

题目四十六 题目排序

```
#include<stdio.h>
#include<stdlib.h>
int a[200000],b[200000],c[200000];
int cmp(const int *p1,const int *p2)
{
	return *p1-*p2;
}
int main()
{
	int n,i,j,sum=0;
	scanf("%d",&n);
	for(i=0;i<n;i++) scanf("%d",&a[i]);
	for(i=0;i<n;i++) scanf("%d",&b[i]);
	for(i=0;i<n;i++) scanf("%d",&c[i]);
	qsort(a,n,sizeof(int),cmp);
	qsort(b,n,sizeof(int),cmp);
	qsort(c,n,sizeof(int),cmp);
	int i1=0,i2=0,i3=0;
	while(i1!=n&&i2!=n&&i3!=n)
	{
		while(b[i2]<=a[i1]&&i2<n) i2++;
		if(i2==n) break;
		while(c[i3]<=b[i2]&&i3<n) i3++;
		if(i3==n) break;
		sum++;
		i1++;
		i2++;
		i3++;
	}
	printf("%d",sum);
	return 0;
}
```

题目四十七 n 条线段两两相交个数

```
#include<iostream>
using namespace std;
struct node
{
	double x1,y1,x2,y2;
	int cnt;//第 cnt 条线段
}line[40000];
int cnt=0;
int n;
```

```
double direction(double x,double y,double x1,double y1,double x2,double y2)   //叉积
{
    double a1=x1-x;
    double b1=y1-y;
    double a2=x2-x;
    double b2=y2-y;
    return a1*b2-a2*b1;
}
int on_segment(double x1,double y1,double x2,double y2,double x,double y)   //判断共线
{
    if((min(x1,x2)<=x && x<=max(x1,x2)) && (min(y1,y2)<=y && y<=max(y1,y2)))   return
1;
    return 0;
}
int cross(int v,int t)//是否交叉，根据算法导论写的
{
    double d1,d2,d3,d4;
    d1=direction(line[t].x1,line[t].y1,line[t].x2,line[t].y2,line[v].x1,line[v].y1);      //算叉积
    d2=direction(line[t].x1,line[t].y1,line[t].x2,line[t].y2,line[v].x2,line[v].y2);
    d3=direction(line[v].x1,line[v].y1,line[v].x2,line[v].y2,line[t].x1,line[t].y1);
    d4=direction(line[v].x1,line[v].y1,line[v].x2,line[v].y2,line[t].x2,line[t].y2);
    if(d1*d2<0 && d3*d4<0) return 1;
    if(!d1 && on_segment(line[t].x1,line[t].y1,line[t].x2,line[t].y2,line[v].x1,line[v].y1)) return 1;
    if(!d2 && on_segment(line[t].x1,line[t].y1,line[t].x2,line[t].y2,line[v].x2,line[v].y2)) return 1;
    if(!d3 && on_segment(line[v].x1,line[v].y1,line[v].x2,line[v].y2,line[t].x1,line[t].y1)) return 1;
    if(!d4 && on_segment(line[v].x1,line[v].y1,line[v].x2,line[v].y2,line[t].x2,line[t].y2)) return 1;
    return 0;
}
int main()
{
    int i,j,sum=0;
    cin>>n;
    while(n--)
    {
        cin>>line[cnt].x1>>line[cnt].y1>>line[cnt].x2>>line[cnt].y2;
        for(i=0;i<cnt;i++)
        {
            if(cross(i,cnt)==1) sum++;
        }
        cnt++;
    }
    cout<<sum<<endl;
    return 0;
}
```

题目四十八 多项式乘积 FFT

```
#include<iostream>
#include<cstdio>
#include<cmath>
using namespace std;
const double Pi=acos(-1.0);
struct complex
{
    double x,y;
    complex (double xx=0,double yy=0){x=xx,y=yy;}//配合 complex w(1,0);指初始化为 1，0
}a[2000000],b[2000000];
complex operator + (complex a,complex b){ return complex(a.x+b.x,a.y+b.y);}
complex operator - (complex a,complex b){ return complex(a.x-b.x,a.y-b.y);}
complex operator * (complex a,complex b){ return complex(a.x*b.x-a.y*b.y , a.x*b.y+a.y*b.x);}//不懂的看复数的运算那部分
int l,r[2000000];
int limit=1;
void FFT(complex[],int);
int main()
{
    int i;
    int n,m;
    cin>>n>>m;
    for(i=0;i<=n;i++) cin>>a[i].x;
    for(i=0;i<=m;i++) cin>>b[i].x;
    while(limit<=n+m)
    {
        limit<<=1;
        l++;
    }
    for(i=0;i<limit;i++)
        r[i]=(r[i>>1]>>1)|((i&1)<<(l-1)) ;
    // 在原序列中 i 与 i/2 的关系是 ： i 可以看做是 i/2 的二进制上的每一位左移一位得来
    // 那么在反转后的数组中就需要右移一位，同时特殊处理一下复数
    FFT(a,1);
    FFT(b,1);
    for(i=0;i<=limit;i++) a[i]=a[i]*b[i];
    FFT(a,-1);
    for(i=0;i<=n+m;i++)
        cout<<(int)(a[i].x/limit+0.5)<<" ";
    cout<<"\n";
    return 0;
```

```
}
void FFT(complex *A,int type)
{
    int i,j,R,k,mid;
    for(i=0;i<limit;i++)
        if(i<r[i]) swap(A[i],A[r[i]]);//求出要迭代的序列
    for(mid=1;mid<limit;mid<<=1)//待合并区间的中点
    {
        complex Wn(cos(Pi/mid),type*sin(Pi/mid)); //单位根
        j=0;
        for(R=mid<<1;j<limit;j=j+R)//R 是区间的右端点，j 表示前已经到哪个位置了
        {
            complex w(1,0);//幂
            for(k=0;k<mid;k++,w=w*Wn)//枚举左半部分
            {
                complex x=A[j+k],y=w*A[j+mid+k];//蝴蝶效应
                A[j+k]=x+y;
                A[j+mid+k]=x-y;
            }
        }
    }
}
```

题目四十九 凸包周长

```
#include <iostream>
#include <cmath>
#include <cstdio>
#include <cstdlib>
using namespace std;
struct point
{
    double x;
    double y;
}p[300000],stack[300000];
int top=2;
double crossProd(point A,point B,point C) //极角
{
    return (B.x-A.x)*(C.y-A.y)-(B.y-A.y)*(C.x-A.x);
}
//以最左下的点为基准点，其他各点（逆时针方向）以极角从小到大的排序规则
double dis(point A,point B)
{
    return sqrt((A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y));
}
```

```
bool iszero(double x)
{
	return (x>0?x:-x)<1e-8;
}
int cmp(const void *a, const void *b)
{
	point *c=(point*)a;
	point *d=(point*)b;
	double k=crossProd(p[0],*c,*d);//极角大小转化为求叉乘
	if (k<1e-8||iszero(k)&&dis(p[0],*c)>dis(p[0],*d)) return 1;
	return -1;
}
void Graham(int n)
{
	int x,y,min=0,i;
	x=p[0].x;
	y=p[0].y;
	for(i=1;i<n;i++) //找到最左下的一个点
	{
		if(p[i].x<x||(p[i].x==x&&p[i].y<y))
		{
			x=p[i].x;
			y=p[i].y;
			min=i;
		}
	}
	point tmp=p[min];
	p[min]=p[0];
	p[0]=tmp;
	qsort(p+1,n-1,sizeof(point),cmp);
	p[n]=p[0];
	stack[0]=p[0];
	stack[1]=p[1];
	stack[2]=p[2];
	for(i=3;i<=n;i++) //加入一个点后,向右偏拐或共线，则上一个点不在凸包内，则--top，该过程直到不向右偏拐或没有三点共线的点
	{
		while(crossProd(stack[top-1],stack[top],p[i])<=1e-8&&top>=2) top--;
		stack[++top]=p[i];//在当前情况下符合凸包的点，入栈
	}
	return;
}

int main()
```

```
{
	int n,i;
	double len;
	cin>>n;
	for(i=0;i<n;i++) cin>>p[i].x>>p[i].y;
	Graham(n);
	for(i=0;i<top;i++)		len=len+sqrt((stack[i].x-stack[i+1].x)*(stack[i].x-
stack[i+1].x)+(stack[i].y-stack[i+1].y)*(stack[i].y-stack[i+1].y));
	printf("%.2lf\n",len);
	return 0;
}
```

题目五十 多边形面积

```
#include <iostream>
using namespace std;
int n,x[20005],y[20005],res;
int main()
{
    int i;
    cin>>n;
    for(i=0;i<n;i++) cin>>x[i]>>y[i];
    x[n]=x[0];
    y[n]=y[0];
    for(i=0;i<n;i++) res+=(x[i]-x[i+1])*(y[i]+y[i+1]);
    res=res/2;
    printf("%d",res);
    return 0;
}
```

题目五十一 大整数相乘

```
#include<iostream>
using namespace std;
int ione[200000],itwo[200000],result[400000];
string slargeone;
string slargetwo;
int main()
{
	int lengths1=0,lengths2=0,i,j=0;
	cin>>slargeone;
	cin>>slargetwo;
	lengths1=slargeone.length();
	lengths2=slargetwo.length();
	for(i=lengths1-1;i>=0;i--)
	{
```

```
            ione[j]=slargeone[i]-'0';
            j++;
        }
        j=0;
        for(i=lengths2-1;i>=0;i--)
        {
            itwo[j]=slargetwo[i]-'0';
            j++;
        }
        for(i=0;i<lengths1;i++)
        { //模拟乘法,将 ione 的每一位都与 itwo 的每一位相乘,然后结果存储于 i+j
            for(j=0;j<lengths2;j++)
            {
                result[i+j] += (ione[i] * itwo[j]);
            }
        }
        for(i=0;i<400000;i++)
        {
            if(result[i] >= 10)
            {
                result[i+1]+=result[i]/10;
                result[i] %= 10;
            }
        }
        int flag=0;
        for(i=399999;i>=0;i--)
        {
            if(flag==1) cout<<result[i];
            else if(result[i])
            {//如果不为零的话,那么就输出结果
                cout<<result[i];
                flag=1;
            }
        }
        return 0;
}
```

题目五十二 已知最小生成树求无向图最小

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int a;
    int b;
```

```
    int value;
}LNode;
LNode edge[200000];
int father[200000],num[200000];
int cmp(const struct node *p1,const struct node *p2)
{
    return p1->value-p2->value;
}
int find_father(x)
{
    if(x!=father[x]) father[x]=find_father(father[x]);
    return father[x];
}
int main()
{
    int t,n,i,j;
    scanf("%d",&t);
    while(t--)
    {
        long long int res=0;
        scanf("%d",&n);
        for(i=1;i<n;i++)
        {
            scanf("%d%d%d",&edge[i].a,&edge[i].b,&edge[i].value);
            res=res+edge[i].value;
        }
        qsort(edge+1,n-1,sizeof(struct node),cmp);
        for(i=1;i<=n;i++)
        {
            father[i]=i;
            num[i]=1;
        }
        for(i=1;i<n;i++)
        {
            int fx=find_father(edge[i].a);
            int fy=find_father(edge[i].b);
            if(fx!=fy)
            {
                father[fx]=fy;
                res=res+(long long int)(edge[i].value+1)*(num[fx]*num[fy]-1);
                num[fy]+=num[fx];
                num[fx]=0;
            }
        }
```

```
            printf("%lld\n",res);
        }
        return 0;
}
```