EH整理的の板子（有一半不是自己写的）

这是代码块。

A-基础算法

A1 高精度加法

```python
# P1601 高精度加法

a = int(input())
b = int(input())
print(a + b)
```

```cpp
#include <iostream>
using namespace std;

const int N=505;
int a[N],b[N],c[N];
int la,lb,lc;

void add(int a[],int b[],int c[]){ //a+b=c
    for(int i=1; i<=lc; i++){
        c[i]+=a[i]+b[i]; //求和
        c[i+1]+=c[i]/10; //进位
        c[i]%=10;        //存余
    }
    if(c[lc+1]) lc++;  //最高位
}
int main(){
    string sa,sb; cin>>sa>>sb;
    la=sa.size(),lb=sb.size(),lc=max(la,lb);
    for(int i=1; i<=la; i++) a[i]=sa[la-i]-'0';
    for(int i=1; i<=lb; i++) b[i]=sb[lb-i]-'0';
    add(a,b,c);
    for(int i=lc; i; i--) printf("%d",c[i]);
    return 0;
}
```

A2 高精度减法

```python
# P2142 高精度减法

a = int(input())
b = int(input())
print(a - b)
```

```cpp
#include <iostream>
using namespace std;

const int N=20000;
int a[N],b[N],c[N];
int la,lb,lc;

bool cmp(int a[],int b[]){
    if(la!=lb) return la<lb;
    for(int i=la; i; i--)
        if(a[i]!=b[i]) return a[i]<b[i];
    return false;  //相等时,避免-0
}
void sub(int a[],int b[],int c[]){ //a-b=c
    for(int i=1; i<=lc; i++){
        if(a[i]<b[i]) a[i+1]--,a[i]+=10;
        c[i]=a[i]-b[i];
    }
    while(c[lc]==0&&lc>1) lc--; //去0
}
int main(){
    string sa,sb; cin>>sa>>sb;
    la=sa.size(),lb=sb.size(),lc=max(la,lb);
    for(int i=1;i<=la;i++) a[i]=sa[la-i]-'0';
    for(int i=1;i<=lb;i++) b[i]=sb[lb-i]-'0';
    if(cmp(a,b)) swap(a,b),cout<<'-';
    sub(a,b,c);
    for(int i=lc;i;i--) printf("%d",c[i]);
    return 0;
}
```

A3 高精度乘法

```python
 # P1303 A*B Problem

a = int(input())
b = int(input())
print(a * b)
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=4001;
int a[N],b[N],c[N];
int la,lb,lc;

void mul(int a[],int b[],int c[]){ //a*b=c
```

```
    for(int i=1;i<=la;i++)
        for(int j=1;j<=lb;j++)
            c[i+j-1]+=a[i]*b[j]; //存乘积

    for(int i=1;i<lc;i++){
        c[i+1]+=c[i]/10;  //存进位
        c[i]%=10;         //存余数
    }
    while(c[lc]==0&&lc>1) lc--; //去0
}
int main(){
    char A[N],B[N];
    cin>>A>>B;
    la=strlen(A); lb=strlen(B); lc=la+lb;
    for(int i=1;i<=la;i++)a[i]=A[la-i]-'0';
    for(int i=1;i<=lb;i++)b[i]=B[lb-i]-'0';
    mul(a,b,c);
    for(int i=lc;i>=1;i--) cout<<c[i];
    return 0;
}
```

A4 高精度除法

```
# P1480 A/B Problem

a = int(input())
b = int(input())
print(a // b)
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=5005;
int a[N],b,c[N];
int len;

void div(int a[],int b,int c[]){ //a/b=c
    long long t=0;
    for(int i=len;i>=1;i--){
        t=t*10+a[i];  //被除数
        c[i]=t/b;     //存商
        t%=b;         //余数
    }
    while(c[len]==0&&len>1) len--; //去0
}
int main(){
    char A[N]; cin>>A>>b; len=strlen(A);
    for(int i=1;i<=len;i++) a[i]=A[len-i]-'0';
    div(a,b,c);
    for(int i=len;i;i--) cout<<c[i];
    return 0;
}
```

A5 二分查找

```python
# P2249 查找

import bisect
n, m = map(int, input().split())
lis = list(map(int, input().split()))
q_lis = list(map(int, input().split()))
for i in range(m):
    q = q_lis[i]
    id = bisect.bisect_left(lis, q)
    if id >= n or lis[id] != q:
        print(-1, end=" ")
    else:
        print(id + 1, end=" ")
```

```cpp
// 我喜欢的板子
#include<cstdio>
using namespace std;
int n,m,q,a[1000005];

int find(int q){
    int l=0,r=n+1;//开区间
    while(l+1<r){ //l+1=r时结束
        int mid=l+r>>1;
        if(a[mid]>=q) r=mid; //最小化
        else l=mid;
    }
    return a[r]==q ? r : -1;
}
int main(){
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++)scanf("%d",&a[i]);
    for(int i=1;i<=m;i++)
        scanf("%d",&q), printf("%d ",find(q));
    return 0;
}
```

A6 二分答案

```python
# P2440 木材加工

def check(x):
    y = 0
    for i in range(1, n + 1):
        y += a[i] // x
    return y >= k

def find():
    l = 0
    r = int(1e8 + 1)
    while l + 1 < r:
        mid = (l + r) // 2
        if check(mid):
            l = mid
```

```
        else:
            r = mid
    return l
n, k = map(int, input().split())
a = [int(input()) for _ in range(n)]
a.insert(0, 0)
print(find())
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
const int N = 100005;
int n, k, a[N];

bool check(int x){
  LL y=0; //段数
  for(int i=1;i<=n;i++) y+=a[i]/x;
  return y>=k; //x小,y大
}
int find(){
  int l=0, r=1e8+1;
  while(l+1<r){
    int mid=l+r>>1;
    if(check(mid)) l=mid; //最大化
    else r=mid;
  }
  return l;
}
int main(){
  scanf("%d%d",&n,&k);
  for(int i=1; i<=n; i++)scanf("%d",&a[i]);
  printf("%d\n",find());
  return 0;
}
```

A7 分数规划

```
# http://poj.org/problem?id=2976

def check(x):
    s = 0
    for i in range(1, n + 1):
        c[i] = a[i] - x * b[i]
    c[1:n+1] = sorted(c[1:n+1])
    for i in range(k + 1, n + 1):
        s += c[i]
    return s >= 0
def find():
    l = 0
    r = 1
    while r - l > 1e-4:
        mid = (l + r) / 2
        if check(mid):
```

```
            l = mid
        else:
            r = mid
    return l


while 1:
    n, k = map(int, input().split())
    if n == 0 and k == 0:
        break
    a = list(map(int, input().split()))
    b = list(map(int, input().split()))
    a.insert(0, 0)
    b.insert(0, 0)
    c = [0] * (n + 1)
    print("{0:.0f}".format(100 * find()))
```

```
//分数规划+二分+排序 复杂度：nlogn*log(1e4)
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=1010;
int n,k;
double a[N], b[N], c[N];

bool check(double x){
  double s=0;
  for(int i=1;i<=n;++i)c[i]=a[i]-x*b[i];
  sort(c+1, c+n+1);
  for(int i=k+1;i<=n;++i) s+=c[i];
  return s>=0;
}
double find(){
  double l=0, r=1;
  while(r-l>1e-4){
    double mid=(l+r)/2;
    if(check(mid)) l=mid;//最大化
    else r=mid;
  }
  return l;
}
int main(){
  while(scanf("%d%d",&n,&k),n){
    for(int i=1;i<=n;i++)scanf("%lf",&a[i]);
    for(int i=1;i<=n;i++)scanf("%lf",&b[i]);
    printf("%.0lf\n", 100*find());
  }
  return 0;
}
```

A8 前缀和

```python
# P8218 【深进1.例1】求区间和

n = int(input())
a = list(map(int, input().split()))
s = [0] * (n + 1)
for i in range(1, n + 1):
    s[i] = s[i - 1] + a[i - 1]
m = int(input())
for i in range(m):
    l, r = map(int, input().split())
    print(s[r] - s[l - 1])
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

int n,m;
int a[100005],s[100005];

int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
      scanf("%d",&a[i]);
        s[i]=s[i-1]+a[i];
    }

    scanf("%d",&m);
    for(int i=1;i<=m;i++){
        int l,r; scanf("%d%d",&l,&r);
        printf("%d\n",s[r]-s[l-1]);
    }
}
```

二维前缀和

```python
# P1387 最大正方形

a = [[0 for i in range(103)] for j in range(103)]
b = [[0 for i in range(103)] for j in range(103)]
n, m = map(int, input().split())
for i in range(1, n + 1):
    temp = list(map(int, input().split()))
    for j in range(1, m + 1):
        a[i][j] = temp[j - 1]
        b[i][j] = b[i][j - 1] + b[i - 1][j] - b[i - 1][j - 1] + a[i][j]
ans = 1
for l in range(2, min(m, n) + 1):
    for i in range(l, n + 1):
        for j in range(l, m + 1):
            if b[i][j]-b[i-l][j]-b[i][j-l]+b[i-l][j-l] == l * l:
                ans = l
print(ans)
```

```cpp
#include <algorithm>
#include <iostream>
using namespace std;

int a[103][103];
int b[103][103];

int main(){
  int n,m;
  cin>>n>>m;
  for(int i=1; i<=n; i++)
    for(int j=1; j<=m; j++){
      cin>>a[i][j];
      b[i][j]=b[i][j-1]+b[i-1][j]-b[i-1][j-1]+a[i][j];
    }

  int ans=1;
  for(int l=2;l<=min(n,m);l++)
    for(int i=l; i<=n; i++)
      for(int j=l; j<=m; j++)
        if(b[i][j]-b[i-l][j]-b[i][j-l]+b[i-l][j-l]==l*l)
          ans=l;
  cout<<ans;
}
```

A10 差分

```python
# P4552 [Poetize6] IncDec Sequence

n = int(input())
a = [0]
b = [0] * (n + 1)
for i in range(1, n + 1):
    a.append(int(input()))
    b[i] = a[i] - a[i - 1]
p, q = 0, 0
for i in range(2, n + 1):
    if b[i] > 0:
        p += b[i]
    else:
        q += abs(b[i])
print(max(p, q))
print(abs(p - q) + 1)
```

```cpp
#include <iostream>
#include <cstring>
using namespace std;

typedef long long LL;
const int N=100010;
int a[N], b[N];

int main(){
  int n;
  cin>>n;
```

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

  LL p=0, q=0;
  for(int i=2; i<=n; i++)
    if(b[i]>0) p+=b[i];
    else q+=abs(b[i]);

  cout<<max(p,q)<<'\n'<<abs(p-q)+1;
}
```

A12 ST表

```
# P3865 【模板】ST 表 && RMQ 问题

import math
n, m = map(int, input().split())
h = list(map(int, input().split()))

# 使用ST表处理最大值
max_log = 20
f = [[0] * (max_log + 1) for _ in range(n + 1)]
for i in range(1, n + 1):
    f[i][0] = h[i - 1]
for j in range(1, max_log + 1):
    i = 1
    while i + 2 ** j - 1 <= n:
        mid = i + 2 ** (j - 1)
        f[i][j] = max(f[i][j - 1], f[mid][j - 1])
        i += 1
def find(l, r):
    k = int(math.log2(r - l + 1))
    ma = max(f[l][k], f[r - 2 ** k + 1][k])
    return ma
for i in range(m):
    l, r = map(int, input().split())
    print(find(l, r))
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

int f[100005][22];

int main(){
  int n,m; scanf("%d%d",&n,&m);

  for(int i=1;i<=n;i++) scanf("%d",&f[i][0]);
  for(int j=1;j<=20;j++) //枚举区间长度
    for(int i=1;i+(1<<j)-1<=n;i++) //枚举起点
      f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);

  for(int i=1,l,r;i<=m;i++){
    scanf("%d%d",&l,&r);
```

```
    int k=log2(r-l+1); //区间长度的指数
    printf("%d\n",max(f[l][k],f[r-(1<<k)+1][k]));
  }
}
```

A13 快速排序、第k小的数

```
# P1923 【深基9.例4】求第 k 小的数

import sys
sys.setrecursionlimit(5000000)
def qnth_element(l, r):
    global a
    if l == r:
        return a[l]
    i = l - 1
    j = r + 1
    x = a[(l + r) // 2]
    while i < j:
        while 1:
            i += 1
            if not a[i] < x:
                break
        while 1:
            j -= 1
            if not a[j] > x:
                break
        if i < j:
            a[i], a[j] = a[j], a[i]
    if k <= j:
        return qnth_element(l, j)
    else:
        return qnth_element(j + 1, r)
n, k = map(int, input().split())
a = list(map(int, input().split()))
print(qnth_element(0, n - 1))
```

```
#include <iostream>
using namespace std;

int n,k,a[5000010];

int qnth_element(int l, int r){
  if(l==r) return a[l];
  int i=l-1, j=r+1, x=a[(l+r)/2];
  while(i<j){
    do i++; while(a[i]<x); //向右找>=x的数
    do j--; while(a[j]>x); //向左找<=x的数
    if(i<j) swap(a[i],a[j]);
  }
  if(k<=j) return qnth_element(l,j);
  else return qnth_element(j+1,r);
}

int main(){
  scanf("%d%d",&n,&k);
```

```
  for(int i=0;i<n;i++) scanf("%d",&a[i]);
  printf("%d\n",qnth_element(0,n-1));
}
```

A14 归并排序、逆序对

```
# P1908 逆序对

import sys
sys.setrecursionlimit(1000000)

def merge(l, r):
    global res, a, b
    if l >= r:
        return
    mid = (l + r) // 2
    merge(l, mid)
    merge(mid + 1, r)

    i = l; j = mid + 1; k = l
    while i <= mid and j <= r:
        if a[i] <= a[j]:
            b[k] = a[i]
            k += 1; i += 1
        else:
            b[k] = a[j]
            k += 1; j += 1
            res += mid - i + 1
    while i <= mid:
        b[k] = a[i]
        k += 1; i += 1
    while j <= r:
        b[k] = a[j]
        k += 1; j += 1
    for i in range(l, r + 1):
        a[i] = b[i]

n = int(input())
a = list(map(int, input().split()))
b = [0] * n
res = 0
merge(0, n - 1)
print(res)
```

```
#include <iostream>
using namespace std;

int n,a[500010],b[500010];
long long res;

void merge(int l,int r){
  if(l>=r) return;
  int mid=l+r>>1;
  merge(l,mid);
  merge(mid+1,r); //拆分
```

```
  int i=l,j=mid+1,k=l; //合并
  while(i<=mid && j<=r){
    if(a[i]<=a[j]) b[k++]=a[i++];
    else b[k++]=a[j++], res+=mid-i+1;
  }
  while(i<=mid) b[k++]=a[i++];
  while(j<=r) b[k++]=a[j++];
  for(i=l; i<=r; i++) a[i]=b[i];
}

int main(){
  cin>>n;
  for(int i=0;i<n;i++) scanf("%d",&a[i]);
  merge(0,n-1);
  printf("%lld\n",res);
}
```

A15 堆

```
# P3378 【模板】堆

import heapq

n = int(input())
q = []
while n:
    n -= 1
    op = list(map(int, input().split()))
    if op[0] == 1:
        heapq.heappush(q, op[1])
    elif op[0] == 2:
        print(q[0])
    else:
        heapq.heappop(q)
```

```
// STL代码
#include <iostream>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;

priority_queue<int,vector<int>,greater<int> > q;

int main(){
  int n; scanf("%d",&n); //操作次数
  while(n--){
    int op,x; scanf("%d",&op);
    if(op==1) scanf("%d",&x), q.push(x);
    else if(op==2) printf("%d\n",q.top());
    else q.pop();
  }
}
```

A16 对顶堆

```python
# 对应题目：https://www.luogu.com.cn/problem/P7072

import heapq
a = [] # 大根堆
b = [] # 小根堆
n, w = map(int, input().split())
lis = list(map(int, input().split()))
for i in range(n):
    x = lis[i]
    k = max(1, int((i + 1) * w / 100))
    if len(b) == 0 or x >= b[0]:
        heapq.heappush(b, x)
    else:
        heapq.heappush(a, -x)
    while len(b) > k:
        heapq.heappush(a, -heapq.heappop(b))
    while len(b) < k:
        heapq.heappush(b, -heapq.heappop(a))
    print(b[0], end=" ")
```

```cpp
#include<cstring>
#include<iostream>
#include<algorithm>
#include<queue>
using namespace std;

int main(){
  int n,w; scanf("%d%d",&n,&w); //选手总数,获奖率
  priority_queue<int> a; //大根堆
  priority_queue<int,vector<int>,greater<int> > b;

  for(int i=1; i<=n; i++){
    int x; scanf("%d",&x);
    if(b.empty()||x>=b.top()) b.push(x); //插入
    else a.push(x);

    int k=max(1,i*w/100); //第k大
    while(b.size()>k) a.push(b.top()), b.pop(); //调整
    while(b.size()<k) b.push(a.top()), a.pop();
    printf("%d ", b.top()); //取值
  }
}
```

A17 距离之和最小、中位数

```python
# CF1486B Eastern Exhibition

t = int(input())
for _ in range(t):
    n = int(input())
    a = [0] * n
```

```
    b = [0] * n
    for i in range(n):
        a[i], b[i] = map(int, input().split())
    a.sort()
    b.sort()
    x = a[n // 2] - a[(n - 1) // 2] + 1
    y = b[n // 2] - b[(n - 1) // 2] + 1
    print(x * y)
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=1010;
int n,a[N],b[N];

int main(){
  int t; cin>>t;
  while(t--){
    cin>>n;
    for(int i=0; i<n; i++) cin>>a[i]>>b[i];
    sort(a,a+n); sort(b,b+n);

    int x=a[n/2]-a[(n-1)/2]+1;
    int y=b[n/2]-b[(n-1)/2]+1;
    cout<<1LL*x*y<<'\n';
  }
  return 0;
}
```

A18 双指针（定量）

```
# P1147 连续自然数和

m = int(input())
i = 1
j = 1
sum = 1
while i <= m // 2:
    if sum < m:
        j += 1
        sum += j
    if sum >= m:
        if sum == m:
            print(i, j)
        sum -= i
        i += 1
```

```
#include<cstdio>

int main(){
  int m; scanf("%d",&m);
  int i=1,j=1,sum=1;
  while(i<=m/2){      //i<=j
```

```
    if(sum<m){
      j++;
      sum+=j;
    }
    if(sum>=m){
      if(sum==m) printf("%d %d\n",i,j);
      sum-=i;
      i++;
    }
  }
  return 0;
}
```

A19 双指针（定性）

```
# P1381 单词背诵
# 24年新星赛-小猪的纸花
from collections import defaultdict
n = int(input())
word = defaultdict(bool)
cnt = defaultdict(int)
len = 0
sum = 0
for i in range(n):
    s1 = input()
    word[s1] = True
m = int(input())
s = [0] * (m + 1)
i = 1
for j in range(1, m + 1):
    s[j] = input()
    if word[s[j]]:
        cnt[s[j]] += 1
    if cnt[s[j]] == 1:
        sum += 1
        len = j - i + 1
    while i <= j:
        if cnt[s[i]] == 1:
            break
        if cnt[s[i]] >= 2:
            cnt[s[i]] -= 1
            i += 1
        if not word[s[i]]:
            i += 1
    len = min(len, j - i + 1)
print(sum)
print(len)
```

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

int n,m;
string s[100005],s1;
map<string,bool> word;
map<string,int> cnt;
int sum,len;
```

```
//s[] 记录文章中的单词
//word[] 记录单词表中的单词
//cnt[] 记录文章当前区间内单词出现次数
//sum 记录文章中出现单词表的单词数
//len 记录包含表中单词最多的区间的最短长度

int main(){
  cin>>n;
  for(int i=1;i<=n;i++)cin>>s1,word[s1]=1;
  cin>>m;
  for(int j=1,i=1; j<=m; j++){  //i<=j
    cin>>s[j];
    if(word[s[j]]) cnt[s[j]]++;
    if(cnt[s[j]]==1) sum++, len=j-i+1;
    while(i<=j){
      if(cnt[s[i]]==1) break; //保持i指针位置不动
      if(cnt[s[i]]>=2) cnt[s[i]]--,i++; //去重,更优
      if(!word[s[i]]) i++; //去掉非目标单词,更优
    }
    len=min(len,j-i+1); //更新
  }
  cout<<sum<<endl<<len<<endl;
}
```

A20 双指针（异或释放）

```
# https://www.luogu.com.cn/problem/AT_arc098_b

n = int(input())
a = list(map(int, input().split()))
a.insert(0, 0)
s1 = 0
s2 = 0
ans = 0
i = 1
j = 0
while i <= n:
    while j + 1 <= n and s1 + a[j + 1] == (s2 ^ a[j + 1]):
        j += 1
        s1 += a[j]
        s2 ^= a[j]
    ans += j - i + 1
    s1 -= a[i]
    s2 -= a[i]
    i += 1
print(ans)
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
LL a[200005];
LL s1,s2,ans;
//s1:算术和, s2:异或和, ans:方案数
```

```cpp
int main(){
  int n; cin>>n;
  for(int i=1; i<=n; i++) cin>>a[i];

  for(int i=1,j=0; i<=n; ){  //i<=j
    while(j+1<=n&&s1+a[j+1]==(s2^a[j+1])){ //预判
      j++;
      s1+=a[j];
      s2^=a[j];
    }
    ans+=j-i+1;
    s1-=a[i];
    s2^=a[i];
    i++;
  }
  cout<<ans<<endl;
  return 0;
}
```

A21 双指针区间合并

```python
# 对应题目：https://www.luogu.com.cn/problem/P1496
# py代码大概率超时，这是洛谷的问题，不要在意，其他地方py给开额外时间的。

n = int(input())
lis = []
for i in range(n):
    a, b = map(int, input().split())
    lis.append([a, b])
lis.sort(key=lambda x: (x[0], x[1]))
l = -1e18
r = -1e18
ans = 0
for i in range(n):
    a, b = lis[i]
    if r < a:
        ans += r - l
        l = a
        r = b
    else:
        r = max(r, b)
ans += r - l
print(int(ans))
```

```cpp
#include<iostream>
#include<cstdio>
#include<algorithm>
using namespace std;

#define N 20005
struct line{    //线段
  int l,r;
  bool operator<(line &t){
    return l<t.l;
  }
```

```
}a[N];
int n,st,ed,sum;
//a[] 存储每条线段的起点,终点
//st  存储合并区间的起点
//ed  存储合并区间的终点
//sum 存储合并区间的长度

int main(){
  scanf("%d",&n);
  for(int i=1;i<=n;i++)
    cin>>a[i].l>>a[i].r;
  sort(a+1,a+n+1); //按起点排序

  st=a[1].l; ed=a[1].r;
  sum+=a[1].r-a[1].l;
  for(int i=2; i<=n; i++){
    if(a[i].l<=ed){
      if(a[i].r<ed) //覆盖
        continue;
      else {        //重叠
        st=ed;
        ed=a[i].r;
        sum+=ed-st;
      }
    }
    else{           //相离
      st=a[i].l;
      ed=a[i].r;
      sum+=ed-st;
    }
  }
  cout<<sum<<endl;
  return 0;
}
```

A22 堆序列合并

```
# P1631 序列合并

import heapq
n = int(input())
q = []
a = list(map(int, input().split()))
a.insert(0, 0)
b = list(map(int, input().split()))
b.insert(0, 0)
id = [0] * (n + 1)
for i in range(1, n + 1):
    id[i] = 1
    heapq.heappush(q, [a[1] + b[i], i])
for j in range(n):
    print(q[0][0], end=" ")
    i = q[0][1]
    heapq.heappop(q)
    id[i] += 1
    heapq.heappush(q, [a[id[i]] + b[i], i])
```

```cpp
#include <cstdio>
#include <queue>
using namespace std;

const int N=100005;
int a[N],b[N],id[N];
priority_queue<pair<int,int>,
       vector<pair<int,int>>,
       greater<pair<int,int>>>q;
//id[i]: 记录b[i]的搭档的下标
//q: 小根堆，存储<两数和,组的下标>

int main(){
  int n; scanf("%d",&n);
  for(int i=1; i<=n; i++) scanf("%d",&a[i]);
  for(int i=1; i<=n; i++){
    scanf("%d",&b[i]);
    id[i]=1;
    q.push({a[1]+b[i],i});
  }
  while(n--){
    printf("%d ",q.top().first);
    int i=q.top().second; q.pop();
    q.push({a[++id[i]]+b[i],i});
  }
  return 0;
}
```

A24 贪心

```python
# P1842

n = int(input())
a = []
for i in range(n):
    w, s = map(int, input().split())
    a.append((w, s, w + s))
a.sort(key=lambda x: x[2])
res = int(-2e9)
t = 0
for i in range(n):
    res = max(res, t - a[i][1])
    t += a[i][0]
print(res)
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=50005;
struct node{
  int w,s;
  bool operator<(node &t){
    return w+s<t.w+t.s;
```

```
  }
}a[N];

int main(){
  int n; cin>>n;
  for(int i=1; i<=n; i++)
    cin>>a[i].w>>a[i].s;
  sort(a+1,a+n+1);

  int res=-2e9, t=0;
  for(int i=1; i<=n; i++){
    res=max(res,t-a[i].s);
    t+=a[i].w;
  }
  cout<<res<<endl;
}
```

A29 线段覆盖

```
# P1803

L = []
n = int(input())
for i in range(n):
    l, r = map(int, input().split())
    L.append([l, r])
L.sort(key=lambda x: x[1])
last = 0
cnt = 0
for i in range(n):
    if last <= L[i][0]:
        last = L[i][1]
        cnt += 1
print(cnt)
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

struct line{
  int l,r; //线段的左,右端点
  bool operator<(line &b){
    return r<b.r;
  }
}L[1000005];
int n,last,cnt;

int main(){
  scanf("%d",&n);
  for(int i=1;i<=n;i++)
    scanf("%d%d",&L[i].l,&L[i].r);

  sort(L+1,L+n+1); //右端点排序
  for(int i=1;i<=n;i++){
    if(last<=L[i].l){
```

```
      last=L[i].r;
      cnt++;
    }
  }
  printf("%d\n",cnt);
  return 0;
}
```

B-搜索算法

B6 DFS深搜

```python
# P1219 [USACO1.5] 八皇后 Checker Challenge

N = 30
pos = [0] * N
c = [0] * N
p = [0] * N
q = [0] * N
ans = 0
def pr():
    if ans <= 3:
        for i in range(1, n + 1):
            print(pos[i], end=" ")
        print()
def dfs(i):
    global ans
    if i > n:
        ans += 1
        pr()
        return
    for j in range(1, n + 1):
        if c[j] or p[i + j] or q[i - j + n]:
            continue
        pos[i] = j
        c[j] = p[i + j] = q[i - j + n] = 1
        dfs(i + 1)
        c[j] = p[i + j] = q[i - j + n] = 0
n = int(input())
dfs(1)
print(ans)
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=30;
int n, ans;
int pos[N],c[N],p[N],q[N];

void print(){
  if(ans<=3){
    for(int i=1;i<=n;i++)
      printf("%d ",pos[i]);
```

```
    puts("");
  }
}
void dfs(int i){
  if(i>n){
    ans++; print(); return;
  }
  for(int j=1; j<=n; j++){
    if(c[j]||p[i+j]||q[i-j+n])continue;
    pos[i]=j; //记录第i行放在了第j列
    c[j]=p[i+j]=q[i-j+n]=1; //宣布占领
    dfs(i+1);
    c[j]=p[i+j]=q[i-j+n]=0; //恢复现场
  }
}
int main(){
  cin >> n;
  dfs(1);
  cout << ans;
  return 0;
}
```

B15 BFS广搜

```
# P1588 [USACO07OPEN] Catch That Cow S

def bfs():
    global x, y
    N = int(2e5 + 1)
    dis = [-1 for i in range(N)]
    dis[x] = 0
    l = 0
    q = []; q.append(x)
    while l < len(q):
        x = q[l]
        l += 1
        if x + 1 < N and dis[x + 1] == -1:
            dis[x + 1] = dis[x] + 1
            q.append(x + 1)
        if x - 1 > 0 and dis[x - 1] == -1:
            dis[x - 1] = dis[x] + 1
            q.append(x - 1)
        if 2 * x < N and dis[x * 2] == -1:
            dis[x * 2] = dis[x] + 1
            q.append(x * 2)
        if x == y:
            print(dis[y])
            return

t = int(input())
for i in range(t):
    x, y = map(int, input().split())
    bfs()
```

```
#include <cstring>
#include <iostream>
```

```cpp
#include <algorithm>
#include <queue>
using namespace std;

const int N=100005;
int x, y, dis[N];

void bfs(){
  memset(dis,-1,sizeof dis); dis[x]=0;
  queue<int> q; q.push(x);
  while(q.size()){
    int x=q.front(); q.pop();
    if(x+1<N && dis[x+1]==-1){
      dis[x+1]=dis[x]+1; //前进一步
      q.push(x+1);
    }
    if(x-1>0 && dis[x-1]==-1){
      dis[x-1]=dis[x]+1; //后退一步
      q.push(x-1);
    }
    if(2*x<N && dis[2*x]==-1){
      dis[2*x]=dis[x]+1; //走到2x位置
      q.push(2*x);
    }
    if(x==y){printf("%d\n",dis[y]);return;}
  }
}
int main(){
  int T; cin>>T;
  while(T--) cin>>x>>y, bfs();
}
```

C-数据结构

C1 并查集

```python
# https://www.luogu.com.cn/problem/P3367

n, m = map(int, input().split())
pa = [i for i in range(n + 1)]
def find(x):
    if pa[x] == x:
        return x
    pa[x] = find(pa[x])
    return pa[x]
def union(x, y):
    pa[find(x)] = find(y)

for _ in range(m):
    z, x, y = map(int, input().split())
    if z == 1:
        union(x, y)
    else:
        if find(x) == find(y):
            print("Y")
        else:
```

```
            print("N")
```

```
//并查集 路径压缩
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=10005;
int n,m,x,y,z;
int pa[N];

int find(int x){
  if(pa[x]==x) return x;
  return pa[x]=find(pa[x]);
}
void unset(int x,int y){
  pa[find(x)]=find(y);
}
int main(){
  cin>>n>>m;
  for(int i=1;i<=n;i++) pa[i]=i;
  while(m --){
    cin>>z>>x>>y;
    if(z==1) unset(x,y);
    else{
      if(find(x)==find(y)) puts("Y");
      else puts("N");
    }
  }
}
```

C2 线段树

```
# 对应题目：https://www.luogu.com.cn/problem/P3372
# py代码100%超时，这是洛谷的问题，不要在意，其他地方py给开额外时间的。

class Node:
    def __init__(self, l, r, he, add):
        self.l = l
        self.r = r
        self.he = he
        self.add = add
N = 100005
tr = [Node(0, 0, 0, 0) for i in range(N * 4)]
def pushdown(p):
    lc = 2 * p
    rc = 2 * p + 1
    if tr[p].add:
        tr[lc].he += tr[p].add * (tr[lc].r - tr[lc].l + 1)
        tr[rc].he += tr[p].add * (tr[rc].r - tr[rc].l + 1)
        tr[lc].add += tr[p].add
        tr[rc].add += tr[p].add
        tr[p].add = 0
def pushup(p):
    tr[p].he = tr[p * 2].he + tr[p * 2 + 1].he
```

```python
def build(p, l, r):
    tr[p] = Node(l, r, w[l], 0)
    if l == r:
        return
    m = (l + r) // 2
    build(p * 2, l, m)
    build(p * 2 + 1, m + 1, r)
    pushup(p)
def update(p, x, y, k):
    if x <= tr[p].l and tr[p].r <= y:
        tr[p].he += (tr[p].r - tr[p].l + 1) * k
        tr[p].add += k
        return
    m = (tr[p].l + tr[p].r) // 2
    pushdown(p)
    if x <= m:
        update(p * 2, x, y, k)
    if y > m:
        update(p * 2 + 1, x, y, k)
    pushup(p)
def query(p, x, y):
    if x <= tr[p].l and tr[p].r <= y:
        return tr[p].he
    m = (tr[p].l + tr[p].r) // 2
    pushdown(p)
    he = 0
    if x <= m:
        he += query(p * 2, x, y)
    if y > m:
        he += query(p * 2 + 1, x, y)
    return he
n, m = map(int, input().split())
w = list(map(int, input().split()))
w.insert(0, 0)
build(1, 1, n)
for i in range(m):
    ru = list(map(int, input().split()))
    if ru[0] == 1:
        update(1, ru[1], ru[2], ru[3])
    else:
        print(query(1, ru[1], ru[2]))
```

```cpp
// 结构体版
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;

#define N 100005
#define LL long long
#define lc u<<1
#define rc u<<1|1
LL w[N];
struct Tree{ //线段树
  LL l,r,sum,add;
}tr[N*4];
```

```
void pushup(LL u){ //上传
  tr[u].sum=tr[lc].sum+tr[rc].sum;
}
void pushdown(LL u){ //下传
  if(tr[u].add){
    tr[lc].sum+=tr[u].add*(tr[lc].r-tr[lc].l+1),
    tr[rc].sum+=tr[u].add*(tr[rc].r-tr[rc].l+1),
    tr[lc].add+=tr[u].add,
    tr[rc].add+=tr[u].add,
    tr[u].add=0;
  }
}
void build(LL u,LL l,LL r){ //建树
  tr[u]={l,r,w[l],0};
  if(l==r) return;
  LL m=l+r>>1;
  build(lc,l,m);
  build(rc,m+1,r);
  pushup(u);
}
void change(LL u,LL l,LL r,LL k){ //区修
  if(l<=tr[u].l&&tr[u].r<=r){
    tr[u].sum+=(tr[u].r-tr[u].l+1)*k;
    tr[u].add+=k;
    return;
  }
  LL m=tr[u].l+tr[u].r>>1;
  pushdown(u);
  if(l<=m) change(lc,l,r,k);
  if(r>m) change(rc,l,r,k);
  pushup(u);
}
LL query(LL u,LL l,LL r){ //区查
  if(l<=tr[u].l && tr[u].r<=r) return tr[u].sum;
  LL m=tr[u].l+tr[u].r>>1;
  pushdown(u);
  LL sum=0;
  if(l<=m) sum+=query(lc,l,r);
  if(r>m) sum+=query(rc,l,r);
  return sum;
}
int main(){
  LL n,m,op,x,y,k;
  cin>>n>>m;
  for(int i=1; i<=n; i ++) cin>>w[i];

  build(1,1,n);
  while(m--){
    cin>>op>>x>>y;
    if(op==2)cout<<query(1,x,y)<<endl;
    else cin>>k,change(1,x,y,k);
  }
  return 0;
}
```

C8 主席树

```
import bisect

class Node:
    def __init__(self, l=0, r=0, s=0):
        self.l = l
        self.r = r
        self.s = s


n, m = map(int, input().split())
a = list(map(int, input().split()))
a = [0] + a  # 转换为1-based索引

# 离散化处理
sorted_b = sorted(a[1:])
unique_b = []
prev = None
for num in sorted_b:
    if num != prev:
        unique_b.append(num)
        prev = num
bn = len(unique_b)

# 初始化主席树
tr = [Node(0, 0, 0)]  # 空节点，索引0
idx = 1
root = [0] * (n + 1)
root[0] = 0

def insert(x, l, r, pos):
    global idx
    y = Node(tr[x].l, tr[x].r, tr[x].s + 1)
    tr.append(y)
    y_idx = idx
    idx += 1
    if l == r:
        return y_idx
    mid = (l + r) // 2
    if pos <= mid:
        new_l = insert(tr[x].l, l, mid, pos)
        y.l = new_l
    else:
        new_r = insert(tr[x].r, mid + 1, r, pos)
        y.r = new_r
    return y_idx

# 构建主席树的每个版本
for i in range(1, n + 1):
    num = a[i]
    pos = bisect.bisect_left(unique_b, num) + 1  # 转换为1-based的id
    root[i] = insert(root[i-1], 1, bn, pos)

# 查询函数
def query(x, y, l, r, k):
    if l == r:
```

```
        return l
    mid = (l + r) // 2
    left_x = tr[x].l
    left_y = tr[y].l
    s = tr[left_y].s - tr[left_x].s
    if k <= s:
        return query(left_x, left_y, l, mid, k)
    else:
        return query(tr[x].r, tr[y].r, mid + 1, r, k - s)

# 处理每个查询并输出结果
output = []
for _ in range(m):
    l, r, k = map(int, input().split())
    id = query(root[l-1], root[r], 1, bn, k)
    output.append(str(unique_b[id-1]))  # id转换为0-based索引
print('\n'.join(output))
```

```
// 主席树 O(nlognlogn)
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

#define N 200005
#define lc(x) tr[x].l
#define rc(x) tr[x].r
struct node{
  int l,r,s; //s:节点值域中有多少个数
}tr[N*20];
int root[N],idx;
int n,m,a[N],b[N];

void insert(int x,int &y,int l,int r,int pos){
  y=++idx; //开点
  tr[y]=tr[x]; tr[y].s++;
  if(l==r) return;
  int m=l+r>>1;
  if(pos<=m) insert(lc(x),lc(y),l,m,pos);
  else insert(rc(x),rc(y),m+1,r,pos);
}
int query(int x,int y,int l,int r,int k){
  if(l==r) return l;
  int m=l+r>>1;
  int s=tr[lc(y)].s-tr[lc(x)].s;
  if(k<=s) return query(lc(x),lc(y),l,m,k);
  else return query(rc(x),rc(y),m+1,r,k-s);
}
int main(){
  scanf("%d%d",&n,&m);
  for(int i=1; i<=n; i++){
    scanf("%d",&a[i]); b[i]=a[i];
  }
  sort(b+1,b+n+1);
  int bn=unique(b+1,b+n+1)-b-1; //去重后的个数

  for(int i=1; i<=n; i++){
```

```
    int id=lower_bound(b+1,b+bn+1,a[i])-b;//下标
    insert(root[i-1],root[i],1,bn,id);
  }
  while(m--){
    int l,r,k; scanf("%d%d%d",&l,&r,&k);
    int id=query(root[l-1],root[r],1,bn,k);
    printf("%d\n",b[id]);
  }
}
```

C19 kd树

py代码还没有，还不怎么会。

```
// 交替建树 970ms
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <cmath>
#define lc t[p].l
#define rc t[p].r
using namespace std;

const int N=200010;
double ans=2e18;
int n,K,root,cur; //K维度,root根,cur当前节点
struct KD{     //KD树节点信息
  int l,r;     //左右孩子
  double v[2]; //点的坐标值
  double L[2],U[2]; //子树区域的坐标范围
  bool operator<(const KD &b)const{return v[K]<b.v[K];}
}t[N];

void pushup(int p){ //更新p子树区域的坐标范围
  for(int i=0;i<2;i++){
    t[p].L[i]=t[p].U[i]=t[p].v[i];
    if(lc)
      t[p].L[i]=min(t[p].L[i],t[lc].L[i]),
      t[p].U[i]=max(t[p].U[i],t[lc].U[i]);
    if(rc)
      t[p].L[i]=min(t[p].L[i],t[rc].L[i]),
      t[p].U[i]=max(t[p].U[i],t[rc].U[i]);
  }
}
int build(int l,int r,int k){ //交替建树
  if(l>r) return 0;
  int m=(l+r)>>1;
  K=k; nth_element(t+l,t+m,t+r+1); //中位数
  t[m].l=build(l,m-1,k^1);
  t[m].r=build(m+1,r,k^1);
  pushup(m);
  return m;
}
double sq(double x){return x*x;}
double dis(int p){ //当前点到p点的距离
  double s=0;
  for(int i=0;i<2;i++)
```

```
    s+=sq(t[cur].v[i]-t[p].v[i]);
  return s;
}
double dis2(int p){ //当前点到p子树区域的最小距离
  if(!p) return 2e18;
  double s=0;
  for(int i=0;i<2;++i)
    s+=sq(max(t[cur].v[i]-t[p].U[i],0.0))+
       sq(max(t[p].L[i]-t[cur].v[i],0.0));
  return s;
}
void query(int p){ //查询当前点的最小距离
  if(!p) return;
  if(p!=cur) ans=min(ans,dis(p));
  double dl=dis2(lc),dr=dis2(rc);
  if(dl<dr){
    if(dl<ans) query(lc);
    if(dr<ans) query(rc);
  }
  else{
    if(dr<ans) query(rc);
    if(dl<ans) query(lc);
  }
}
int main(){
  scanf("%d",&n);
  for(int i=1; i<=n; i++)
    scanf("%lf%lf",&t[i].v[0],&t[i].v[1]);
  root=build(1,n,0);
  for(cur=1; cur<=n; cur++) query(root);
  printf("%.4lf\n",sqrt(ans));
}
```

D-图论

D2 狄克斯特拉算法-单源最短路径

暴力$O(n^2)$:

```
class Edge():
    def __init__(self, v, w):
        self.v = v
        self.w = w

def dijkstra(s):
    for i in range(n + 1):
        d[i] = int(2 ** 31 - 1)
    d[s] = 0
    for i in range(1, n):
        u = 0
        for j in range(1, n + 1):
            if not vis[j] and d[j] < d[u]:
                u = j
```

```
        vis[u] = 1
        for ed in e[u]:
            v = ed.v
            w = ed.w
            if d[v] > d[u] + w:
                d[v] = d[u] + w

n, m, s = map(int, input().split())
e = [[] for i in range(n + 1)]
d = [0] * (n + 1)
vis = [0] * (n + 1)
for i in range(m):
    a, b, c = map(int, input().split())
    e[a].append(Edge(b, c))

dijkstra(s)
for i in range(1, n + 1):
    print(d[i], end=" ")
```

```
//Dijkstra
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
#define inf 2147483647
using namespace std;

int n,m,s,a,b,c;
const int N=100010;
struct edge{int v,w;};
vector<edge> e[N];
int d[N], vis[N];

void dijkstra(int s){
  for(int i=0;i<=n;i++)d[i]=inf;
  d[s]=0;
  for(int i=1;i<n;i++){//枚举次数
    int u=0;
    for(int j=1;j<=n;j++)//枚举点
      if(!vis[j]&&d[j]<d[u]) u=j;
    vis[u]=1; //标记u已出圈
    for(auto ed:e[u]){//枚举邻边
      int v=ed.v, w=ed.w;
      if(d[v]>d[u]+w){
        d[v]=d[u]+w;
      }
    }
  }
}
int main(){
  cin>>n>>m>>s;
  for(int i=0; i<m; i++){
    cin>>a>>b>>c;
    e[a].push_back({b,c});
  }
  dijkstra(s);
  for(int i=1;i<=n;i++)
```

```
    printf("%d ",d[i]);
  return 0;
}
```

堆优化$O(mlogm)$:

```
import heapq

class Edge():
    def __init__(self, v, w):
        self.v = v
        self.w = w

def dijkstra(s):
    for i in range(n + 1):
        d[i] = int(1e20)
    d[s] = 0
    q = []
    heapq.heappush(q, [0, s])
    while len(q) > 0:
        t = heapq.heappop(q)
        u = t[1]
        if vis[u]:
            continue
        vis[u] = 1
        for ed in e[u]:
            v = ed.v
            w = ed.w
            if d[v] > d[u] + w:
                d[v] = d[u] + w
                heapq.heappush(q, [d[v], v])

n, m, s = map(int, input().split())
e = [[] for i in range(n + 1)]
d = [0] * (n + 1)
vis = [0] * (n + 1)
for i in range(m):
    a, b, c = map(int, input().split())
    e[a].append(Edge(b, c))

dijkstra(s)
for i in range(1, n + 1):
    print(d[i], end=" ")
```

```
//堆优化Dijkstra
#include <cstring>
#include <iostream>
#include <algorithm>
#include <queue>
using namespace std;

const int N=100010;
int n,m,s,a,b,c;
struct edge{int v,w;};
vector<edge> e[N];
int d[N],vis[N];
```

```
void dijkstra(int s){
  memset(d,0x3f,sizeof d); d[s]=0;
  priority_queue<pair<int,int>> q;
  q.push({0,s});
  while(q.size()){
    auto t=q.top(); q.pop();
    int u=t.second;
    if(vis[u])continue; //再出队跳过
    vis[u]=1; //标记u已出队
    for(auto ed : e[u]){
      int v=ed.v, w=ed.w;
      if(d[v]>d[u]+w){
        d[v]=d[u]+w;
        q.push({-d[v],v}); //大根堆
      }
    }
  }
}
int main(){
  cin>>n>>m>>s;
  for(int i=0; i<m; i++)
    cin>>a>>b>>c, e[a].push_back({b,c});
  dijkstra(s);
  for(int i=1;i<=n;i++) printf("%d ",d[i]);
}
```

D4 Floyd算法

```
def floyd():
    for k in range(1, n + 1):
        for i in range(1, n + 1):
            for j in range(1, n + 1):
                d[i][j] = min(d[i][j], d[i][k] + d[k][j])


n, m = map(int, input().split())
d = [[int(1e20) for __ in range(n + 1)] for _ in range(n + 1)]
for i in range(1, n + 1):
    d[i][i] = 0
for i in range(m):
    a, b, c = map(int, input().split())
    d[a][b] = min(d[a][b], c)
    floyd()
for i in range(1, n + 1):
    print(*d[i][1:])
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;
const int N=210,M=20010;
int n,m,a,b,c;
int d[N][N];

void floyd(){
```

```
  for(int k=1; k<=n; k++)
    for(int i=1; i<=n; i++)
      for(int j=1; j<=n; j++)
        d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
}
int main(){
  cin>>n>>m;
  memset(d,0x3f,sizeof d);
  for(int i=1; i<=n; i++)d[i][i]=0;
  for(int i=0; i<m; i++){
    cin>>a>>b>>c;
    d[a][b]=min(d[a][b],c); //重边
  }
  floyd();
  for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++)
      printf("%d ",d[i][j]);
    puts("");
  }
  return 0;
}
```

D5 Johnson算法

```
import heapq

class Edge():
    def __init__(self, v, w):
        self.v = v
        self.w = w

def spfa():
    global h, vis, cnt
    q = []; l = 0
    h = [int(1e20) for i in range(n + 1)]
    vis = [False for i in range(n + 1)]
    h[0] = 0; vis[0] = True; q.append(0)
    while len(q) > l:
        u = q[l]; l += 1; vis[u] = False
        for ed in e[u]:
            v = ed.v; w = ed.w
            if h[v] > h[u] + w:
                h[v] = h[u] + w
                cnt[v] = cnt[u] + 1
                if cnt[v] > n:
                    print(-1);exit()
                if not vis[v]:
                    q.append(v)
                    vis[v] = True

def dijkstra(s):
    global h, vis, cnt
    for i in range(n + 1):
        d[i] = int(1e9)
    vis = [False for i in range(n + 1)]
    d[s] = 0
    q = []
```

```
    heapq.heappush(q, [0, s])
    while len(q) > 0:
        t = heapq.heappop(q)
        u = t[1]
        if vis[u]:
            continue
        vis[u] = 1
        for ed in e[u]:
            v = ed.v
            w = ed.w
            if d[v] > d[u] + w:
                d[v] = d[u] + w
                heapq.heappush(q, [d[v], v])



n, m = map(int, input().split())
e = [[] for i in range(n + 1)]
h = [0] * (n + 1)
d = [0] * (n + 1)
vis = [0] * (n + 1)
cnt = [0] * (n + 1)
for i in range(m):
    a, b, c = map(int, input().split())
    e[a].append(Edge(b, c))
for i in range(1, n + 1):
    e[0].append(Edge(i, 0))
spfa()
for u in range(1, n + 1):
    for ed in e[u]:
        ed.w += h[u] - h[ed.v]
for i in range(1, n + 1):
    dijkstra(i)
    ans = 0
    for j in range(1, n + 1):
        if d[j] == int(1e9):
            ans += j * int(1e9)
        else:
            ans += j * (d[j] + h[j] - h[i])
    print(ans)
```

```
#include<algorithm>
#include<cstring>
#include<iostream>
#include<queue>
#define N 30010
#define INF 1000000000
using namespace std;

int n,m,a,b,c;
struct edge{int v,w;};
vector<edge> e[N];
int vis[N],cnt[N];
long long h[N],d[N];

void spfa(){
    queue<int>q;
```

```
    memset(h,63,sizeof h);
    memset(vis,false,sizeof vis);
    h[0]=0,vis[0]=1;q.push(0);
    while(q.size()){
        int u=q.front(); q.pop();vis[u]=0;
        for(auto ed : e[u]){
            int v=ed.v,w=ed.w;
            if(h[v]>h[u]+w){
                h[v]=h[u]+w;
        cnt[v]=cnt[u]+1;
        if(cnt[v]>n){
          printf("-1\n");exit(0);
        }
                if(!vis[v])q.push(v),vis[v]=1;
            }
        }
    }
}
void dijkstra(int s){
    priority_queue<pair<long long,int>>q;
    for(int i=1;i<=n;i++)d[i]=INF;
    memset(vis,false,sizeof vis);
    d[s]=0; q.push({0,s});
    while(q.size()){
        int u=q.top().second;q.pop();
        if(vis[u])continue;
        vis[u]=1;
        for(auto ed : e[u]){
            int v=ed.v,w=ed.w;
            if(d[v]>d[u]+w){
                d[v]=d[u]+w;
                if(!vis[v]) q.push({-d[v],v});
            }
        }
    }
}
int main(){
  cin>>n>>m;
  for(int i=0;i<m;i++)
    cin>>a>>b>>c, e[a].push_back({b,c});
    for(int i=1;i<=n;i++)
      e[0].push_back({i,0});//加虚拟边

    spfa();
    for(int u=1;u<=n;u++)
      for(auto &ed:e[u])
        ed.w+=h[u]-h[ed.v];//构造新边
    for(int i=1;i<=n;i++){
        dijkstra(i);
        long long ans=0;
        for(int j=1;j<=n;j++){
            if(d[j]==INF) ans+=(long long)j*INF;
            else ans+=(long long)j*(d[j]+h[j]-h[i]);
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

D7 Prim算法

```
N = 5010
d = [0] * N
vis = [0] * N
class Edge():
    def __init__(self, v, w):
        self.v = v
        self.w = w
e = [[] for i in range(N)]

def prim(s):
    global ans, cnt
    for i in range(n + 1):
        d[i] = int(1e9)
    d[s] = 0
    for i in range(1, n + 1):
        u = 0
        for j in range(1, n + 1):
            if not vis[j] and d[j] < d[u]:
                u = j
        vis[u] = 1
        ans += d[u]
        if d[u] != 1e9:
            cnt += 1
        for ed in e[u]:
            v = ed.v
            w = ed.w
            if d[v] > w:
                d[v] = w
    return cnt == n
n, m = map(int, input().split())
ans, cnt = 0, 0
for i in range(m):
    a, b, c = map(int, input().split())
    e[a].append(Edge(b, c))
    e[b].append(Edge(a, c))
if not prim(1):
    print("orz")
else:
    print(ans)
```

```
// Luogu P3366 【模板】最小生成树
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
#define inf 1e9
using namespace std;

int n,m,a,b,c,ans,cnt;
const int N=5010;
struct edge{int v,w;};
vector<edge> e[N];
```

```
int d[N], vis[N];

bool prim(int s){
  for(int i=0;i<=n;i++)d[i]=inf;
  d[s]=0;
  for(int i=1;i<=n;i++){
    int u=0;
    for(int j=1;j<=n;j++)
      if(!vis[j]&&d[j]<d[u]) u=j;
    vis[u]=1; //标记u已出圈
    ans+=d[u];
    if(d[u]!=inf) cnt++;
    for(auto ed : e[u]){
      int v=ed.v, w=ed.w;
      if(d[v]>w) d[v]=w;
    }
  }
  return cnt==n;
}
int main(){
  cin>>n>>m;
  for(int i=0; i<m; i++){
    cin>>a>>b>>c;
    e[a].push_back({b,c});
    e[b].push_back({a,c});
  }
  if(!prim(1))puts("orz");
  else printf("%d\n",ans);
  return 0;
}
```

D8 Kruscal算法

```
class Edge():
    def __init__(self, u, v, w):
        self.u = u
        self.v = v
        self.w = w

def find(x):
    if fa[x] == x:
        return x
    fa[x] = find(fa[x])
    return fa[x]
def union(x, y):
    fa[find(x)] = find(y)
def kruskal():
    global ans, cnt
    e.sort(key=lambda k:k.w)
    for i in range(m):
        x = find(e[i].u)
        y = find(e[i].v)
        if x != y:
            union(x, y)
            ans += e[i].w
            cnt += 1
    return cnt == n - 1
```

```
n, m = map(int, input().split())
fa = [i for i in range(n + 1)]
ans, cnt = 0, 0
e = []
for i in range(m):
    u, v, w = map(int, input().split())
    e.append(Edge(u, v, w))
if not kruskal():
    print("orz")
else:
    print(ans)
```

```
// Luogu P3366 【模板】最小生成树
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=200006;
int n, m;
struct edge{
  int u,v,w;
  bool operator<(const edge &t)const
  {return w < t.w;}
}e[N];
int fa[N],ans,cnt;

int find(int x){
  if(fa[x]==x) return x;
  return fa[x]=find(fa[x]);
}
bool kruskal(){
  sort(e,e+m);
  for(int i=1;i<=n;i++)fa[i]=i;
  for(int i=0; i<m; i++){
    int x=find(e[i].u);
    int y=find(e[i].v);
    if(x!=y){
      fa[x]=y;
      ans+=e[i].w;
      cnt++;
    }
  }
  return cnt==n-1;
}
int main(){
  cin>>n>>m;
  for(int i=0; i<m; i++)
    cin>>e[i].u>>e[i].v>>e[i].w;
  if(!kruskal()) puts("orz");
  else printf("%d\n",ans);
  return 0;
}
```

D11 树链剖分（LCA）

```python
import sys
sys.setrecursionlimit(int(1e7))
N = 500010
fa = [0] * N
son = [0] * N
dep = [0] * N
siz = [0] * N
top = [0] * N
e = [[] for i in range(N)]
def dfs1(u, f):
    fa[u] = f
    siz[u] = 1
    dep[u] = dep[f] + 1
    for v in e[u]:
        if v == f:
            continue
        dfs1(v, u)
        siz[u] += siz[v]
        if siz[son[u]] < siz[v]:
            son[u] = v
def dfs2(u, t):
    top[u] = t
    if not son[u]:
        return
    dfs2(son[u], t)
    for v in e[u]:
        if v == fa[u] or v == son[u]:
            continue
        dfs2(v, v)
def lca(u, v):
    while top[u] != top[v]:
        if dep[top[u]] < dep[top[v]]:
            u, v = v, u
        u = fa[top[u]]
    if dep[u] < dep[v]:
        return u
    else:
        return v
n, m, s = map(int, input().split())
for i in range(1, n):
    a, b = map(int, input().split())
    e[a].append(b)
    e[b].append(a)
dfs1(s, 0)
dfs2(s, s)
for i in range(m):
    a, b = map(int, input().split())
    print(lca(a, b))
```

```cpp
// 树链剖分 O(mlogn)
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
```

```cpp
using namespace std;

const int N=500010;
int n,m,s,a,b;
vector<int> e[N];
int fa[N],son[N],dep[N],siz[N],top[N];

void dfs1(int u,int f){ //搞fa,son,dep
  fa[u]=f;siz[u]=1;dep[u]=dep[f]+1;
  for(int v:e[u]){
    if(v==f) continue;
    dfs1(v,u);
    siz[u]+=siz[v];
    if(siz[son[u]]<siz[v])son[u]=v;
  }
}
void dfs2(int u,int t){ //搞top
  top[u]=t;  //记录链头
  if(!son[u]) return; //无重儿子
  dfs2(son[u],t);     //搜重儿子
  for(int v:e[u]){
    if(v==fa[u]||v==son[u])continue;
    dfs2(v,v); //搜轻儿子
  }
}
int lca(int u,int v){
  while(top[u]!=top[v]){
    if(dep[top[u]]<dep[top[v]])swap(u,v);
    u=fa[top[u]];
  }
  return dep[u]<dep[v]?u:v;
}
int main(){
  scanf("%d%d%d",&n,&m,&s);
  for(int i=1; i<n; i++){
    scanf("%d%d",&a,&b);
    e[a].push_back(b);
    e[b].push_back(a);
  }
  dfs1(s,0);
  dfs2(s,s);
  while(m--){
    scanf("%d%d",&a,&b);
    printf("%d\n",lca(a,b));
  }
  return 0;
}
```

D13 LCA应用-树上距离

```python
## https://loj.ac/p/10130
# 相比树链剖分板子，只在有标注“#距离”处修改了
import sys
sys.setrecursionlimit(int(1e7))
N = 100010
fa = [0] * N
son = [0] * N
```

```python
dep = [0] * N
siz = [0] * N
top = [0] * N
# 距离
dis = [0] * N
e = [[] for i in range(N)]
def dfs1(u, f):
    fa[u] = f
    siz[u] = 1
    dep[u] = dep[f] + 1
    for v in e[u]:
        if v == f:
            continue
        # 距离
        dis[v] = dis[u] + 1
        dfs1(v, u)
        siz[u] += siz[v]
        if siz[son[u]] < siz[v]:
            son[u] = v
def dfs2(u, t):
    top[u] = t
    if not son[u]:
        return
    dfs2(son[u], t)
    for v in e[u]:
        if v == fa[u] or v == son[u]:
            continue
        dfs2(v, v)
def lca(u, v):
    while top[u] != top[v]:
        if dep[top[u]] < dep[top[v]]:
            u, v = v, u
        u = fa[top[u]]
    if dep[u] < dep[v]:
        return u
    else:
        return v
n = int(input())
for i in range(1, n):
    a, b = map(int, input().split())
    e[a].append(b)
    e[b].append(a)
dfs1(1, 0)
dfs2(1, 1)
m = int(input())
for i in range(m):
    a, b = map(int, input().split())
    # 距离
    d = dis[a] + dis[b] - dis[lca(a, b)] * 2
    print(d)
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
#include <vector>
using namespace std;
const int N=100010;
```

```cpp
int n,m,a,b,c;
vector<int> e[N];
int dep[N],fa[N],son[N],sz[N],dis[N];
int top[N];

void dfs1(int u,int father){
  fa[u]=father,dep[u]=dep[father]+1,sz[u]=1;
  for(int v:e[u]){
    if(v==father) continue;
    dis[v]=dis[u]+1;
    dfs1(v,u);
    sz[u]+=sz[v];
    if(sz[son[u]]<sz[v])son[u]=v;
  }
}
void dfs2(int u,int t){
  top[u]=t;
  if(!son[u]) return;
  dfs2(son[u],t);
  for(int v:e[u]){
    if(v==fa[u]||v==son[u])continue;
    dfs2(v,v);
  }
}
int lca(int x,int y){
  while(top[x]!=top[y]){
    if(dep[top[x]]<dep[top[y]])swap(x,y);
    x=fa[top[x]];
  }
  return dep[x]<dep[y]?x:y;
}
int main(){
  scanf("%d",&n);
  for(int i=1; i<n; i++){
    scanf("%d%d",&a,&b);
    e[a].push_back(b);
    e[b].push_back(a);
  }
  dfs1(1,0);
  dfs2(1,1);
  scanf("%d",&m);
  while(m--){
    scanf("%d%d",&a,&b);
    int d=dis[a]+dis[b]-dis[lca(a,b)]*2;
    printf("%d\n",d);
  }
  return 0;
}
```

E-动态规划

E4 最长上升子序列（二分优化）

```python
import bisect
n = int(input())
b = [0] * (n + 10)
a = list(map(int, input().split()))
b[0] = int(-2e9)
len = 0
for i in range(n):
    if b[len] < a[i]:
        len += 1
        b[len] = a[i]
    else:
        b[bisect.bisect_left(b, a[i], 1, len)] = a[i]
print(len)
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=100010;
int n, a[N];
int len, b[N]; //记录上升子序列

int main(){
  scanf("%d", &n);
  for(int i=0; i<n; i++) scanf("%d", &a[i]);

  b[0]=-2e9;                                  //哨兵
  for(int i=0; i<n; i++)
    if(b[len]<a[i]) b[++len]=a[i];            //新数大于队尾数，则插入队尾
    else *lower_bound(b,b+len,a[i])=a[i]; //替换第一个大于等于a[i]的数(贪心)

  printf("%d\n", len);
}
```

E5 最长公共子序列

```python
a = "ADABBC"
b = "DBPCA"
f = []
p = []
m = 0
n = 0

def LCS():
    global m, n, f, p
    m = len(a)
    n = len(b)
    f = [[0] * (n + 1) for _ in range(m + 1)]
    p = [[0] * (n + 1) for _ in range(m + 1)]
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if a[i-1] == b[j-1]:
```

```
                f[i][j] = f[i-1][j-1] + 1
                p[i][j] = 1  # 左上方
            else:
                if f[i][j-1] > f[i-1][j]:
                    f[i][j] = f[i][j-1]
                    p[i][j] = 2  # 左边
                else:
                    f[i][j] = f[i-1][j]
                    p[i][j] = 3  # 上边
    print(f[m][n])

def getLCS():
    global m, n, p, a
    i = m
    j = n
    k = f[m][n]
    s = [''] * k
    while i > 0 and j > 0:
        if p[i][j] == 1:
            s[k-1] = a[i-1]
            i -= 1
            j -= 1
            k -= 1
        elif p[i][j] == 2:
            j -= 1
        else:
            i -= 1
    print(''.join(s))

if __name__ == "__main__":
    LCS()
    getLCS()
```

```
#include <cstring>
#include <iostream>

char a[200] = "ADABBC";
char b[200] = "DBPCA";
int f[200][200];
int p[200][200];
int m, n;
void LCS() {
    int i, j;
    m = strlen(a);
    n = strlen(b);
    for (i = 1; i <= m; i++) {
        for (j = 1; j <= n; j++) {
            if (a[i-1] == b[j-1]) {
                f[i][j] = f[i-1][j-1] + 1;
                p[i][j] = 1; // 左上方
            } else if (f[i][j-1] > f[i-1][j]) {
                f[i][j] = f[i][j-1];
                p[i][j] = 2; // 左边
            } else {
                f[i][j] = f[i-1][j];
                p[i][j] = 3; // 上边
            }
```

```
        }
    }
    cout << f[m][n] << '\n';
}

void getLCS() {
    int i, j, k;
    char s[200];
    i = m;
    j = n;
    k = f[m][n];
    while (i > 0 && j > 0) {
        if (p[i][j] == 1) {
            s[k-1] = a[i-1];
            i--;
            j--;
            k--;
        } else if (p[i][j] == 2) {
            j--;
        } else {
            i--;
        }
    }
    for (i = 0; i < f[m][n]; i++) {
        cout << s[i];
    }
}
int main() {
    LCS();
    getLCS();
    return 0;
}
```

E6 最长公共子串

```
a = "BCCABCCB"
b = "AACCAB"
m = len(a)
n = len(b)
f = [[0] * (n + 1) for _ in range(m + 1)]
ans = 0
for i in range(1, m + 1):
    for j in range(1, n + 1):
        if a[i - 1] == b[j - 1]:
            f[i][j] = f[i - 1][j - 1] + 1
        else:
            f[i][j] = 0
        ans = max(ans, f[i][j])
print(ans)
```

```
#include<iostream>
#include<cstring>
using namespace std;

char a[200]="BCCABCCB";
char b[200]="AACCAB";
```

```
int f[201][201];

int main(){
  int ans=0;
  for(int i=1; i<=strlen(a); i++){
    for(int j=1; j<=strlen(b); j++){
      if(a[i-1]==b[j-1]) f[i][j]=f[i-1][j-1]+1;
      else f[i][j]=0;
      ans=max(ans,f[i][j]);
    }
  }
  printf("ans=%d\n",ans);
  return 0;
}
```

E8 01背包

```
N = 3410
M = 13000
n, m = map(int, input().split())
v = [0] * N
w = [0] * N
f = [0] * M
for i in range(n):
    v[i + 1], w[i + 1] = map(int, input().split())
for i in range(1, n + 1):
    for j in range(m, 0, -1):
        if j >= v[i]:
            f[j] = max(f[j], f[j - v[i]] + w[i])
        else:
            break
print(f[m])
```

```
// 逆序枚举，优化空间#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=3410,M=13000;
int n, m;
int v[N],w[N],f[M];

int main(){
  scanf("%d%d",&n,&m);
  for(int i=1; i<=n; i++)
    scanf("%d%d",&v[i],&w[i]);  //费用，价值

  for(int i=1; i<=n; i++)       //枚举物品
    for(int j=m; j>=v[i]; j--)  //枚举体积
      f[j]=max(f[j],f[j-v[i]]+w[i]);

  printf("%d\n",f[m]);
}
```

E9 完全背包

```python
N = 1010
n, m = map(int, input().split())
v = [0] * N
w = [0] * N
f = [0] * N
for i in range(n):
    v[i + 1], w[i + 1] = map(int, input().split())
for i in range(1, n + 1):
    for j in range(v[i], m + 1):
        f[j] = max(f[j], f[j - v[i]] + w[i])
print(f[m])
```

```cpp
// 优化决策+优化空间
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=1010;
int n, m;
int v[N],w[N],f[N];

int main(){
  scanf("%d%d",&n,&m);
  for(int i=1; i<=n; i++)
    scanf("%d%d",&v[i],&w[i]);  //费用，价值

  for(int i=1; i<=n; i++)       //枚举物品
    for(int j=v[i]; j<=m; j++)  //枚举体积
      f[j]=max(f[j],f[j-v[i]]+w[i]);

  printf("%d\n",f[m]);
}
```

E10 多重背包

```python
n, m = map(int, input().split())

v = [0] * 100005
w = [0] * 100005
f = [0] * 100005

cnt = 0
for _ in range(n):
    b, a, s = map(int, input().split())
    j = 1
    while j <= s:
        cnt += 1
        v[cnt] = j * a
        w[cnt] = j * b
        s -= j
        j <<= 1
    if s > 0:
```

```
        cnt += 1
        v[cnt] = s * a
        w[cnt] = s * b

for i in range(1, cnt + 1):
    for j in range(m, v[i] - 1, -1):
        f[j] = max(f[j], f[j - v[i]] + w[i])

print(f[m])
```

```
// 二进制分组优化
#include<iostream>
using namespace std;

const int N=100005;
int n,m,a,b,s;
int v[N],w[N];
int f[N];

int main(){
  cin>>n>>m;

  int cnt=0;
  for(int i=1;i<=n;i++){
    cin>>b>>a>>s;
    for(int j=1;j<=s;j<<=1){
      v[++cnt]=j*a; w[cnt]=j*b;
      s-=j;
    }
    if(s) v[++cnt]=s*a, w[cnt]=s*b;
  }

  for(int i=1;i<=cnt;i++)
    for(int j=m;j>=v[i];j--)
      f[j]=max(f[j],f[j-v[i]]+w[i]);
  cout<<f[m];
}
```

E11 滑动窗口

```
import sys
from collections import deque

input = sys.stdin.readline

n, k = map(int, input().split())
a = [0] * (n + 1)  # Using 1-based indexing to match the C++ code
a[1:] = list(map(int, input().split()))

# Maintain window minimum
q = deque()
for i in range(1, n + 1):
    while len(q) > 0 and a[q[-1]] >= a[i]:
        q.pop()
    q.append(i)
    while q[0] < i - k + 1:
```

```
        q.popleft()
    if i >= k:
        print(a[q[0]], end=' ')
print()

# Maintain window maximum
q = deque()
for i in range(1, n + 1):
    while len(q) > 0 and a[q[-1]] <= a[i]:
        q.pop()
    q.append(i)
    while q[0] < i - k + 1:
        q.popleft()
    if i >= k:
        print(a[q[0]], end=' ')
```

```
#include <iostream>
#include <deque>
using namespace std;

const int N=1000010;
int a[N];
deque<int> q;

int main(){
  int n, k; scanf("%d%d", &n, &k);
  for(int i=1; i<=n; i++) scanf("%d", &a[i]);

  // 维护窗口最小值
  q.clear();                                //清空队列
  for(int i=1; i<=n; i++){                  //枚举序列
    while(!q.empty() && a[q.back()]>=a[i]) q.pop_back(); //队尾出队(队列不空且新元素更优)
    q.push_back(i);                         //队尾入队(存储下标 方便判断队头出队)
    while(q.front()<i-k+1) q.pop_front(); //队头出队(队头元素滑出窗口)
    if(i>=k) printf("%d ",a[q.front()]);  //使用最值
  }
  puts("");

  // 维护窗口最大值
  q.clear();
  for(int i=1; i<=n; i++){
    while(!q.empty() && a[q.back()]<=a[i]) q.pop_back();
    q.push_back(i);
    while(q.front()<i-k+1) q.pop_front();
    if(i>=k) printf("%d ",a[q.front()]);
  }
}
```

E17 树形DP

```
import sys
sys.setrecursionlimit(int(1e7))
N = 6010
head = [0] * N
to = [0] * N
```

```python
ne = [0] * N
idx = 0
def add(a, b):
    global idx
    idx += 1
    to[idx] = b; ne[idx] = head[a]; head[a] = idx
w = [0] * N
fa = [0] * N
f = [[0 for i in range(2)] for j in range(N)]
def dfs(u):
    f[u][1] = w[u]
    i = head[u]
    while i != 0:
        v = to[i]
        dfs(v)
        f[u][0] += max(f[v][0], f[v][1])
        f[u][1] += f[v][0]
        i = ne[i]
n = int(input())
for i in range(1, n + 1):
    w[i] = int(input())
for i in range(n - 1):
    a, b = map(int, input().split())
    add(b, a)
    fa[a] = True
root = 1
while fa[root]:
    root += 1
dfs(root)
print(max(f[root][0], f[root][1]))
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N=6010;
int head[N],to[N],ne[N],idx;
void add(int a,int b){
  to[++idx]=b,ne[idx]=head[a],head[a]=idx;
}
int n,w[N],fa[N];
int f[N][2]; //0不选,1选

void dfs(int u){
  f[u][1]=w[u];
  for(int i=head[u];i;i=ne[i]){
    int v=to[i];
    dfs(v);
    f[u][0]+=max(f[v][0],f[v][1]);
    f[u][1]+=f[v][0];
  }
}
int main(){
  cin>>n;
  for(int i=1;i<=n;i++) cin>>w[i];
  for(int i=0,a,b;i<n-1;i++){
```

```
    cin>>a>>b;
    add(b,a);
    fa[a]=true;
  }
  int root=1;
  while(fa[root]) root++;
  dfs(root);
  cout<<max(f[root][0],f[root][1]);
}
```

F-字符串

EH过度劳累，此时的san值有点低，精神奔溃中，写水一点吧。

F3 KMP算法

```
N = 1000010
S = list(input().strip())
S = [""] + S
P = list(input().strip())
P = [""] + P
nxt = [0] * N
m = len(S) - 1
n = len(P) - 1
S.append("")
P.append("")
nxt[1] = 0
j = 0
for i in range(2, n + 1):
    while j and P[i] != P[j + 1]:
        j = nxt[j]
    if P[i] == P[j + 1]:
        j += 1
    nxt[i] = j
j = 0
for i in range(1, m + 1):
    while j and S[i] != P[j + 1]:
        j = nxt[j]
    if S[i] == P[j + 1]:
        j += 1
    if j == n:
        print(i - n + 1)
print(*nxt[1:n + 1])
```

```
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;

const int N=1000010;
int m,n;
char S[N],P[N];
int nxt[N];

int main(){
```

```
  cin>>S+1>>P+1;
  m=strlen(S+1),n=strlen(P+1);

  nxt[1]=0;
  for(int i=2,j=0;i<=n;i++){
    while(j && P[i]!=P[j+1]) j=nxt[j];
    if(P[i]==P[j+1]) j++;
    nxt[i]=j;
  }

  for(int i=1,j=0;i<=m;i++){
    while(j && S[i]!=P[j+1]) j=nxt[j];
    if(S[i]==P[j+1]) j++;
    if(j==n) printf("%d\n",i-n+1);
  }

  for(int i=1;i<=n;i++)printf("%d ",nxt[i]);
  return 0;
}
```

F6 Trie字典树

```
N = 100010
ch = [[0 for _ in range(26)] for _ in range(N)]
cnt = [0] * N
idx = 0

def insert(s):
    global idx
    p = 0
    for c in s:
        j = ord(c) - ord('a')
        if not ch[p][j]:
            idx += 1
            ch[p][j] = idx
        p = ch[p][j]
    cnt[p] += 1

def query(s):
    p = 0
    for c in s:
        j = ord(c) - ord('a')
        if not ch[p][j]:
            return 0
        p = ch[p][j]
    return cnt[p]

n, q = map(int, input().split())
for _ in range(n):
    s = input().strip()
    insert(s)
for _ in range(q):
    s = input().strip()
    print(query(s))
```

```
// O(n)#include <iostream>
```

```cpp
#include <cstring>
#include <algorithm>
using namespace std;

const int N=100010;
int n;
char s[N];
int ch[N][26],cnt[N],idx;

void insert(char *s){
  int p=0;
  for(int i=0; s[i]; i ++){
    int j=s[i]-'a';//字母映射
    if(!ch[p][j])ch[p][j]=++idx;
    p=ch[p][j];
  }
  cnt[p]++;//插入次数
}
int query(char *s){
  int p=0;
  for(int i=0; s[i]; i ++){
    int j=s[i]-'a';
    if(!ch[p][j]) return 0;
    p=ch[p][j];
  }
  return cnt[p];
}
int main(){
  scanf("%d",&n);
  while(n--){
    char op;
    scanf("%s%s",&op,s);
    if(op=='I')insert(s);
    else printf("%d\n",query(s));
  }
  return 0;
}
```

F7 最大异或对

```python
ch = [[0, 0]]  # 初始化Trie树，根节点为0

def insert(x):
    p = 0
    for i in range(30, -1, -1):
        j = (x >> i) & 1
        if ch[p][j] == 0:
            ch.append([0, 0])  # 创建新节点
            ch[p][j] = len(ch) - 1  # 更新指针到新节点
        p = ch[p][j]

def query(x):
    p = 0
    res = 0
    for i in range(30, -1, -1):
        j = (x >> i) & 1
        opposite = 1 - j
```

```
        if ch[p][opposite]:
            res += (1 << i)
            p = ch[p][opposite]
        else:
            p = ch[p][j]
    return res

n = int(input())
a = list(map(int, input().split()))
for num in a:
    insert(num)
ans = 0
for num in a:
    ans = max(ans, query(num))
print(ans)
```

```
// 01Trie 最大异或对
#include <iostream>
using namespace std;

const int N=100010;
int n, a[N];
int ch[N*31][2],cnt;

void insert(int x){
  int p=0;
  for(int i=30; i>=0; i--){
    int j=x>>i&1; //取出第i位
    if(!ch[p][j])ch[p][j]=++cnt;
    p=ch[p][j];
  }
}
int query(int x){
  int p=0,res=0;
  for(int i=30; i>=0; i--){
    int j=x>>i&1;
    if(ch[p][!j]){
      res += 1<<i; //累加位权
      p=ch[p][!j];
    }
    else p=ch[p][j];
  }
  return res;
}
int main(){
  cin>>n;
  for(int i=1; i<=n; i++)
    cin>>a[i],insert(a[i]);
  int ans=0;
  for(int i=1; i<=n; i++)
    ans=max(ans,query(a[i]));
  cout<<ans;
  return 0;
}
```

G-数学

G1 快速幂

```python
a, b, p = map(int, input().split())
ans = pow(a, b, p)
s = "{0}^{1} mod {2}={3}".format(a, b, p, ans)
print(s)
```

```cpp
#include <iostream>
using namespace std;

typedef long long LL;
int a,b,p;

int qpow(int a,int b,int p){ //快速幂
  int s=1;
  while(b){
    if(b&1) s=(LL)s*a%p;
    a=(LL)a*a%p;
    b>>=1;
  }
  return s;
}
int main(){
  cin>>a>>b>>p;
  int s=qpow(a,b,p);
  printf("%d^%d mod %d=%d\n",a,b,p,s);
  return 0;
}
```

G5 gcd及lcm问题

```python
# P1029 [NOIP 2001 普及组] 最大公约数和最小公倍数问题
import math
x, y = map(int, input().split())
t = x * y
ans = 0
for i in range(1, int(t ** 0.5) + 1):
    if t % i == 0 and math.gcd(t // i, i) == x:
        ans += 2
if x == y:
    ans -= 1
print(ans)
```

```cpp
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

typedef long long LL;
LL x,y,ans;

LL gcd(LL a, LL b){
```

```
  return b==0 ? a : gcd(b,a%b);
}
int main(){
    cin>>x>>y;
    LL t = x*y;
    for(LL i=1; i*i<=t; i++)
        if(t%i==0 && gcd(i,t/i)==x)
          ans += 2;
    if(x==y) ans--;
    cout << ans;
    return 0;
}
```

G8 线性筛质数

```
n, q = map(int, input().split())

# 使用欧拉筛（线性筛）来找出所有素数
vis = [True] * (n + 1)
prim = []
for i in range(2, n + 1):
    if vis[i]:
        prim.append(i)
    for p in prim:
        if i * p > n:
            break
        vis[i * p] = False
        if i % p == 0:
            break

for _ in range(q):
    k = int(input())
    print(prim[k - 1])
```

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int N = 100000010;
int vis[N];  //划掉合数
int prim[N]; //记录质数
int cnt; //质数个数

void get_prim(int n){ //线性筛法
  for(int i=2; i<=n; i++){
    if(!vis[i]) prim[++cnt] = i;
    for(int j=1; 1ll*i*prim[j]<=n; j++){
      vis[i*prim[j]] = 1;
      if(i%prim[j] == 0) break;
    }
  }
}
int main(){
    int n, q, k;
    scanf("%d %d", &n, &q);
```

```
    get_prim(n);
    while(q--){
        scanf("%d", &k);
        printf("%d\n", prim[k]);
    }
    return 0;
}
```

G13 费马小定理

```
a, p = map(int, input().split())
print(pow(a, p - 2, p))
```

```
#include<iostream>
using namespace std;

typedef long long LL;
int a, p;

int quickpow(LL a, int b, int p){
  int res = 1;
  while(b){
    if(b & 1) res = res*a%p;
    a = a*a%p;
    b >>= 1;
  }
  return res;
}
int main(){
  cin >> a >> p;
  if(a % p)
    printf("%d\n",quickpow(a,p-2,p));
  return 0;
}
```

G52 凸包算法

没有py代码，因为EH还不会。

```
#include <iostream>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

const int N=100010;
struct Point{double x,y;} p[N],s[N];
int n,top;

double cross(Point a,Point b,Point c){ //叉积
  return (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x);
}
double dis(Point a,Point b){ //距离
  return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
```

```
bool cmp(Point a, Point b){ //比较
  return a.x!=b.x ? a.x<b.x : a.y<b.y;
}
double Andrew(){
  sort(p+1,p+n+1,cmp); //排序
  for(int i=1; i<=n; i++){ //下凸包
    while(top>1&&cross(s[top-1],s[top],p[i])<=0)top--;
    s[++top]=p[i];
  }
  int t=top;
  for(int i=n-1; i>=1; i--){ //上凸包
    while(top>t&&cross(s[top-1],s[top],p[i])<=0)top--;
    s[++top]=p[i];
  }
  double res=0; //周长
  for(int i=1; i<top; i++) res+=dis(s[i],s[i+1]);
  return res;
}
int main(){
  scanf("%d",&n);
  for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
  printf("%.2lf\n", Andrew());
  return 0;
}
```

G53 旋转卡壳

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

#define N 50010
#define x first
#define y second
#define Point pair<int,int>
Point p[N],s[N];
int n;

int cross(Point a,Point b,Point c){ //叉积
  return (b.x-a.x)*(c.y-a.y)-(b.y-a.y)*(c.x-a.x);
}
int dis(Point a, Point b){ //距离平方
  return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
void Andrew(){
  sort(p+1,p+n+1);
  int t=0;
  for(int i=1; i<=n; i++){ //下凸包
    while(t>1&&cross(s[t-1],s[t],p[i])<=0)t--;
    s[++t]=p[i];
  }
  int k=t;
  for(int i=n-1; i>=1; i--){ //上凸包
    while(t>k&&cross(s[t-1],s[t],p[i])<=0)t--;
    s[++t]=p[i];
  }
```

```
  n=t-1; //n为凸包上的点数
}
int rotating_calipers(){ //旋转卡壳
  int res=0;
  for(int i=1,j=2; i<=n; i++){
    while(cross(s[i],s[i+1],s[j])<cross(s[i],s[i+1],s[j+1]))j=j%n+1;
    res=max(res,max(dis(s[i],s[j]),dis(s[i+1],s[j])));
  }
  return res;
}
int main(){
  scanf("%d",&n);
  for(int i=1; i<=n; i++) scanf("%d%d",&p[i].x,&p[i].y);
  Andrew();
  printf("%d\n",rotating_calipers());
  return 0;
}
```