

Contents

1	ubuntu	1
1.1	run	1
1.2	cp.sh	1
2	字串	1
2.1	最長迴文子字串	1
2.2	KMP	1
3	STL	2
3.1	BIT	2
3.2	deque	2
3.3	map	2
3.4	unordered_map	3
3.5	set	3
3.6	multiset	3
3.7	unordered_set	3
3.8	單調隊列	3
4	sort	3
4.1	大數排序	3
5	math	4
5.1	質數與因數	4
5.2	快速幂	4
5.3	歐拉函數	5
5.4	atan	5
5.5	大步小步	5
6	algorithm	5
6.1	basic	5
6.2	二分搜	6
6.3	三分搜	6
6.4	prefix sum	6
6.5	差分	7
6.6	greedy	7
6.7	floyd warshall	8
6.8	dinic	9
6.9	Nim Game	9
6.10	SPFA	10
6.11	dijkstra	10
6.12	SCC Tarjan	10
6.13	SCC Kosaraju	11
6.14	ArticulationPoints Tarjan	11
6.15	最小樹狀圖	12
6.16	二分圖最大匹配	14
6.17	Astar	14
6.18	JosephusProblem	15
6.19	KM	15
6.20	LCA 倍增法	16
6.21	LCA 樹壓平 RMQ	16
6.22	MCMF	17
6.23	莫隊	18
6.24	Dancing Links X	19
7	DataStructure	19
7.1	ChthollyTree	19
7.2	線段樹 1D	20
7.3	線段樹 2D	21
7.4	Trie	21
7.5	權值線段樹	22
8	geometry	23
8.1	intersection	23
8.2	半平面相交	23
8.3	凸包	24
9	DP	24
9.1	以價值為主的背包	24
9.2	抽屜	25
9.3	Barcode	25
9.4	Deque 最大差距	25
9.5	LCS 和 LIS	25
9.6	RangeDP	26
9.7	stringDP	26
9.8	TreeDP 有幾個 path 長度為 k	26
9.9	TreeDP reroot	26
9.10	WeightedLIS	27
9.11	dp1ist	27
10	Section2	37
10.1	thm	37

1 ubuntu

1.1 run

```
1 | ~$ bash cp.sh PA
```

1.2 cp.sh

```
1 #!/bin/bash
2 clear
3 g++ $1.cpp -DDBG -o $1
4 if [[ "$?" == "0" ]]; then
5     echo Running
6     ./$1 < $1.in > $1.out
7     echo END
8 fi
```

2 字串

2.1 最長迴文子字串

```
5 1 #include<bits/stdc++.h>
5 2 #define T(x) ((x)%2 ? s[(x)/2] : '. ')
6 3 using namespace std;
6 4
6 5 string s;
6 6 int n;
7 7
8 8 int ex(int l,int r){
9 9     int i=0;
10 10 while(l-i>=0&&r+i<n&&T(l-i)==T(r+i)) i++;
11 11 return i;
12 12 }
13 13
14 14 int main(){
15 15     cin>>s;
16 16     n=2*s.size()+1;
17 17     int mx=0;
18 18     int center=0;
19 19     vector<int> r(n);
20 20     int ans=1;
21 21     r[0]=1;
22 22     for(int i=1;i<n;i++){
23 23         int ii=center-(i-center);
24 24         int len=mx-i+1;
25 25         if(i>mx){
26 26             r[i]=ex(i,i);
27 27             center=i;
28 28             mx=i+r[i]-1;
29 29         }
30 30         else if(r[ii]==len){
31 31             r[i]=len+ex(i-len,i+len);
32 32             center=i;
33 33             mx=i+r[i]-1;
34 34         }
35 35         else r[i]=min(r[ii],len);
36 36         ans=max(ans,r[i]);
37 37     }
38 38     cout<<ans-1<<"\n";
39 39     return 0;
40 40 }
```

2.2 KMP

```
1 #define maxn 1000005
2 int nextArr[maxn];
3 void getNextArr(const string& str) {
4     nextArr[0] = 0;
```

```

5 | int prefixLen = 0;
6 | for (int i = 1; i < str.size(); ++i) {
7 |     prefixLen = nextArr[i - 1];
8 |     //如果不一樣就在之前算過的prefix中搜有沒有更短的前
9 |     while (prefixLen > 0 && str[prefixLen] !=
10 |            str[i])
11 |         prefixLen = nextArr[prefixLen - 1];
12 |     //一樣就繼承之前的前後綴長度+1
13 |     if (str[prefixLen] == str[i])
14 |         ++prefixLen;
15 |     nextArr[i] = prefixLen;
16 | }
17 | for (int i = 0; i < str.size() - 1; ++i) {
18 |     vis[nextArr[i]] = true;
19 | }

```

3 STL

3.1 BIT

```

1 | template <class T> class BIT {
2 | private:
3 |     int size;
4 |     vector<T> bit;
5 |     vector<T> arr;
6 |
7 | public:
8 |     BIT(int sz=0): size(sz), bit(sz+1), arr(sz) {}
9 |
10 |    /** Sets the value at index idx to val. */
11 |    void set(int idx, T val) {
12 |        add(idx, val - arr[idx]);
13 |    }
14 |
15 |    /** Adds val to the element at index idx. */
16 |    void add(int idx, T val) {
17 |        arr[idx] += val;
18 |        for (++idx; idx<=size; idx+=(idx & -idx))
19 |            bit[idx] += val;
20 |    }
21 |
22 |    /** @return The sum of all values in [0, idx]. */
23 |    T pre_sum(int idx) {
24 |        T total = 0;
25 |        for (++idx; idx>0; idx--=(idx & -idx))
26 |            total += bit[idx];
27 |        return total;
28 |    }
29 | };

```

3.2 deque

```

1 | deque 是 C++ 標準模板函式庫
2 | (Standard Template Library, STL)
3 | 中的雙向佇列容器 (Double-ended Queue) ,
4 | 跟 vector 相似, 不過在 vector
5 | 中若是要添加新元素至開端,
6 | 其時間複雜度為 O(N), 但在 deque 中則是 O(1)。
7 | 同樣也能在我們需要儲存更多元素的時候自動擴展空間,
8 | 讓我們不必煩惱佇列長度的問題。
9 | dq.push_back() //在 deque 的最尾端新增元素
10 | dq.push_front() //在 deque 的開頭新增元素
11 | dq.pop_back() //移除 deque 最尾端的元素
12 | dq.pop_front() //移除 deque 最開頭的元素
13 | dq.back() //取出 deque 最尾端的元素
14 | dq.front() //回傳 deque 最開頭的元素
15 | dq.insert()
16 | dq.insert(position, n, val)
    position: 插入元素的 index 值

```

```

17 | n: 元素插入次數
18 | val: 插入的元素值
19 | dq.erase() //刪除元素, 需要使用迭代器指定刪除的元素或位置,
20 | //同時也會返回指向刪除元素下一元素的迭代器。
21 | dq.clear() //清空整個 deque 佇列。
22 | dq.size() //檢查 deque 的尺寸
23 | dq.empty() //如果 deque 佇列為空返回 1;
24 | //若是存在任何元素, 則返回 0
25 | dq.begin() //返回一個指向 deque 開頭的迭代器
26 | dq.end() //指向 deque 結尾,
27 | //不是最後一個元素,
28 | //而是最後一個元素的下一個位置

```

3.3 map

```

1 | map: 存放 key-value pairs 的映射資料結構,
2 | 會按 key 由小到大排序。
3 | 元素存取
4 | operator[]: 存取指定的[i]元素的資料
5 |
6 | 迭代器
7 | begin(): 回傳指向map頭部元素的迭代器
8 | end(): 回傳指向map末尾的迭代器
9 | rbegin(): 回傳一個指向map尾部的反向迭代器
10 | rend(): 回傳一個指向map頭部的反向迭代器
11 |
12 | 遍歷整個map時, 利用 iterator 操作:
13 | 取key: it->first 或 (*it).first
14 | 取value: it->second 或 (*it).second
15 |
16 | 容量
17 | empty(): 檢查容器是否為空, 空則回傳 true
18 | size(): 回傳元素數量
19 | max_size(): 回傳可以容納的最大元素個數
20 |
21 | 修改器
22 | clear(): 刪除所有元素
23 | insert(): 插入元素
24 | erase(): 刪除一個元素
25 | swap(): 交換兩個map
26 |
27 | 查找
28 | count(): 回傳指定元素出現的次數
29 | find(): 查找一個元素
30 |
31 | //實作範例
32 | #include <bits/stdc++.h>
33 | using namespace std;
34 | int main(){
35 |     //declaration container and iterator
36 |     map<string, string> mp;
37 |     map<string, string>::iterator iter;
38 |     map<string, string>::reverse_iterator iter_r;
39 |
40 |     //insert element
41 |     mp.insert(pair<string, string>
42 |               ("r000", "student_zero"));
43 |     mp["r123"] = "student_first";
44 |     mp["r456"] = "student_second";
45 |
46 |     //traversal
47 |     for(iter=mp.begin(); iter!=mp.end(); iter++)
48 |         cout<<iter->first<<" "
49 |              <<iter->second<<endl;
50 |     for(iter_r=mp.rbegin(); iter_r!=mp.rend(); iter_r++)
51 |         cout<<iter_r->first<<" "
52 |              <<iter_r->second<<endl;
53 |
54 |     //find and erase the element
55 |     iter=mp.find("r123");
    mp.erase(iter);

```

```

56     iter=mp.find("r123");
57     if(iter!=mp.end())
58         cout<<"Find, the value is "
59             <<iter->second<<endl;
60     else cout<<"Do not Find"<<endl;
61     return 0;
62 }

```

3.4 unordered_map

1 unordered_map：存放 key-value pairs
 2 的「無序」映射資料結構。
 3 用法與map相同

3.5 set

1 set：集合，去除重複的元素，資料由小到大排序。
 2
 3 取值：使用iterator
 4 x = *st.begin();
 5 // set中的第一個元素(最小的元素)。
 6 x = *st.rbegin();
 7 // set中的最後一個元素(最大的元素)。
 8
 9 判斷是否為空的set：
 10 st.empty() 回傳true
 11 st.size() 回傳零
 12
 13 常用來搭配的member function：
 14 st.count(x);
 15 auto it = st.find(x);
 16 // binary search, $O(\log(N))$
 17 auto it = st.lower_bound(x);
 18 // binary search, $O(\log(N))$
 19 auto it = st.upper_bound(x);
 20 // binary search, $O(\log(N))$

3.6 multiset

1 與 set 用法雷同，但會保留重複的元素。
 2 資料由小到大排序。
 3 宣告：
 4 multiset<int> st;
 5 刪除資料：
 6 st.erase(val);
 7 //會刪除所有值為 val 的元素。
 8 st.erase(st.find(val));
 9 //只刪除第一個值為 val 的元素。

3.7 unordered_set

1 unordered_set 的實作方式通常是用雜湊表(hash table)，
 2 資料插入和查詢的時間複雜度很低，為常數級別 $O(1)$ ，
 3 相對的代價是消耗較多的記憶體，空間複雜度較高，
 4 無自動排序功能。
 5
 6 unordered_set 判斷元素是否存在
 7 unordered_set<int> myunordered_set;
 8 myunordered_set.insert(2);
 9 myunordered_set.insert(4);
 10 myunordered_set.insert(6);
 11 cout << myunordered_set.count(4) << "\n"; // 1
 12 cout << myunordered_set.count(8) << "\n"; // 0

3.8 單調隊列

```

1 //單調隊列
2 "如果一個選手比你小還比你強，你就可以退役了。"--單調隊列
3
4 example
5
6 給出一個長度為 n 的數組，
7 輸出每 k 個連續的數中的最大值和最小值。
8
9 #include <bits/stdc++.h>
10 #define maxn 1000100
11 using namespace std;
12 int q[maxn], a[maxn];
13 int n, k;
14
15 void getmin() {
16     // 得到這個隊列裡的最小值，直接找到最後的就行了
17     int head=0, tail=0;
18     for(int i=1; i<=k; i++) {
19         while(head<=tail&&a[q[tail]]>=a[i]) tail--;
20         q[++tail]=i;
21     }
22     for(int i=k; i<=n; i++) {
23         while(head<=tail&&a[q[tail]]>=a[i]) tail--;
24         q[++tail]=i;
25         while(q[head]<=i-k) head++;
26         cout<<a[q[head]]<<" ";
27     }
28     cout<<endl;
29 }
30
31 void getmax() { // 和上面同理
32     int head=0, tail=0;
33     for(int i=1; i<=k; i++) {
34         while(head<=tail&&a[q[tail]]<=a[i]) tail--;
35         q[++tail]=i;
36     }
37     for(int i=k; i<=n; i++) {
38         while(head<=tail&&a[q[tail]]<=a[i]) tail--;
39         q[++tail]=i;
40         while(q[head]<=i-k) head++;
41         cout<<a[q[head]]<<" ";
42     }
43     cout<<endl;
44 }
45
46 int main(){
47     cin>>n>>k; //每k個連續的數
48     for(int i=1; i<=n; i++) cin>>a[i];
49     getmin();
50     getmax();
51     return 0;
52 }

```

4 sort

4.1 大數排序

```

1 #python大數排序
2
3 while True:
4     try:
5         n = int(input()) # 有幾筆數字需要排序
6         arr = [] # 建立空串列
7         for i in range(n):
8             arr.append(int(input())) # 依序將數字存入串列
9         arr.sort() # 串列排序
10        for i in arr:
11            print(i) # 依序印出串列中每個項目
12    except:
13        break

```

5 math

5.1 質數與因數

```

1 埃氏篩法
2 int n;
3 vector<int> isprime(n+1,1);
4 isprime[0]=isprime[1]=0;
5 for(int i=2;i<=n;i++){
6     if(isprime[i])
7         for(int j=i*i;j<=n;j+=i) isprime[j]=0;
8 }
9
10 歐拉篩 O(n)
11 #define MAXN 47000 //sqrt(2^31)=46,340...
12 bool isPrime[MAXN];
13 int prime[MAXN];
14 int primeSize=0;
15 void getPrimes(){
16     memset(isPrime,true,sizeof(isPrime));
17     isPrime[0]=isPrime[1]=false;
18     for(int i=2;i<MAXN;i++){
19         if(isPrime[i]) prime[primeSize++]=i;
20         for(int
21             j=0;j<primeSize&&i*prime[j]<=MAXN;++j){
22             isPrime[i*prime[j]]=false;
23             if(i%prime[j]==0) break;
24         }
25     }
26
27 最大公因數 O(log(min(a,b)))
28 int GCD(int a,int b){
29     if(b==0) return a;
30     return GCD(b,a%b);
31 }
32
33 質因數分解
34 void primeFactorization(int n){
35     for(int i=0;i<(int)p.size();++i){
36         if(p[i]*p[i]>n) break;
37         if(n%p[i]) continue;
38         cout<<p[i]<<' ';
39         while(n%p[i]==0) n/=p[i];
40     }
41     if(n!=1) cout<<n<<' ';
42     cout<<'\n';
43 }
44
45 擴展歐幾里得算法
46 //ax+by=GCD(a,b)
47 #include <bits/stdc++.h>
48 using namespace std;
49
50 int ext_euc(int a,int b,int &x,int &y){
51     if(b==0){
52         x=1,y=0;
53         return a;
54     }
55     int d=ext_euc(b,a%b,y,x);
56     y-=a/b*x;
57     return d;
58 }
59
60 int main(){
61     int a,b,x,y;
62     cin>>a>>b;
63     ext_euc(a,b,x,y);
64     cout<<x<<' '<<y<<endl;
65     return 0;
66 }
67
68
69
70 歌德巴赫猜想

```

```

71 solution : 把偶數 N (6≤N≤10^6) 寫成兩個質數的和。
72 #include <iostream>
73 using namespace std;
74 #define N 2000000
75 int ox[N],p[N],pr;
76 void PrimeTable(){
77     ox[0]=ox[1]=1;
78     pr=0;
79     for(int i=2;i<N;i++){
80         if(!ox[i]) p[pr++]=i;
81         for(int j=0;i*p[j]<N&&j<pr;j++)
82             ox[i*p[j]]=1;
83     }
84 }
85
86 int main(){
87     PrimeTable();
88     int n;
89     while(cin>>n,n){
90         int x;
91         for(x=1;;x+=2)
92             if(!ox[x]&&!ox[n-x]) break;
93         printf("%d = %d + %d\n",n,x,n-x);
94     }
95 }
96
97 problem : 給定整數 N ,
98 求 N 最少可以拆成多少個質數的和。
99 如果 N 是質數,則答案為 1。
100 如果 N 是偶數(不包含2),則答案為 2 (強歌德巴赫猜想)。
101 如果 N 是奇數且 N-2 是質數,則答案為 2 (2+質數)。
102 其他狀況答案為 3 (弱歌德巴赫猜想)。
103 #include<bits/stdc++.h>
104 using namespace std;
105
106 bool isPrime(int n){
107     for(int i=2;i<n;++i){
108         if(i*i>n) return true;
109         if(n%i==0) return false;
110     }
111     return true;
112 }
113
114 int main(){
115     int n;
116     cin>>n;
117     if(isPrime(n)) cout<<"1\n";
118     else if(n%2==0||isPrime(n-2)) cout<<"2\n";
119     else cout<<"3\n";
120 }

```

5.2 快速幂

```

1 計算 a^b
2 #include<iostream>
3 #define ll long long
4 using namespace std;
5
6 const ll MOD=1000000007;
7 ll fp(ll a, ll b) {
8     int ans=1;
9     while(b>0){
10         if(b&1) ans=ans*a%MOD;
11         a=a*a%MOD;
12         b>>=1;
13     }
14     return ans;
15 }
16
17 int main() {
18     int a,b;
19     cin>>a>>b;
20     cout<<fp(a,b);
21 }

```

5.3 歐拉函數

```

1 //計算閉區間 [1,n] 中的正整數與 n 互質的個數
2
3 int phi(){
4     int ans=n;
5     for(int i=2;i*i<=n;i++){
6         if(n%i==0){
7             ans=ans-ans/i;
8             while(n%i==0) n/=i;
9         }
10    if(n>1) ans=ans-ans/n;
11    return ans;
12 }

```

5.4 atan

```

1 說明
2     atan() 和 atan2() 函數分別計算 x 和 y/x 的反正切。
3
4 回覆值
5     atan() 函數會傳回介於範圍 - /2 到 /2 弧度之間的值。
6     atan2() 函數會傳回介於 - 至 弧度之間的值。
7     如果 atan2() 函數的兩個引數都是零，
8     則函數會將 errno 設為 EDOM，並傳回值 0。
9
10 範例
11 #include <math.h>
12 #include <stdio.h>
13
14 int main(void){
15     double a,b,c,d;
16
17     c=0.45;
18     d=0.23;
19
20     a=atan(c);
21     b=atan2(c,d);
22
23     printf("atan(%lf)=%lf/n",c,a);
24     printf("atan2(%lf,%lf)=%lf/n",c,d,b);
25 }
26
27 /*
28 atan(0.450000)=0.422854
29 atan2(0.450000,0.230000)=1.098299
30 */

```

5.5 大步小步

```

1 題意
2 給定 B,N,P，求出 L 滿足  $B^L \equiv N \pmod{P}$ 。
3
4 題解
5 餘數的循環節長度必定為 P 的因數，因此
6    $B^0, B^1, B^2, \dots, B^{(P-1)}$ ，
7 也就是說如果有解則  $L < P$ ，枚舉 0,1,2,L-1
8 能得到結果，但會超時。
9
10 將 L 拆成  $mx+y$ ，只要分別枚舉 x,y 就能得到答案，
11 設  $m=\sqrt{P}$  能保證最多枚舉  $2\sqrt{P}$  次。
12
13  $B^{(mx+y)} \equiv N \pmod{P}$ 
14  $B^{mx} B^y \equiv N \pmod{P}$ 
15  $B^y \equiv N(B^{-m})^x \pmod{P}$ 
16
17 先求出  $B^0, B^1, B^2, \dots, B^{(m-1)}$ ，
18 再枚舉  $N(B^{-m}), N(B^{-m})^2, \dots$  查看是否有對應的  $B^y$ 。
19 這種算法稱為大步小步演算法，
20 大步指的是枚舉 x (一次跨 m 步)，

```

```

19 小步指的是枚舉 y (一次跨 1 步)。
20
21 複雜度分析
22 利用 map/unorder_map 存放  $B^0, B^1, B^2, \dots, B^{(m-1)}$ ，
23 枚舉 x 查詢 map/unorder_map 是否有對應的  $B^y$ ，
24 存放和查詢最多  $2\sqrt{P}$  次，時間複雜度為  $O(\sqrt{P} \log \sqrt{P}) / O(\sqrt{P})$ 。
25
26
27 #include <bits/stdc++.h>
28 using namespace std;
29 using LL = long long;
30 LL B, N, P;
31
32 LL fpow(LL a, LL b, LL c){
33     LL res=1;
34     for(;b>=1;){
35         if(b&1)
36             res=(res*a)%c;
37         a=(a*a)%c;
38     }
39     return res;
40 }
41
42 LL BSGS(LL a, LL b, LL p){
43     a%=p,b%=p;
44     if(a==0)
45         return b==0?-1:0;
46     if(b==1)
47         return 0;
48     map<LL, LL> tb;
49     LL sq=ceil(sqrt(p-1));
50     LL inv=fpow(a,p-sq-1,p);
51     tb[1]=sq;
52     for(LL i=1,tmp=1;i<sq;++i){
53         tmp=(tmp*a)%p;
54         if(!tb.count(tmp))
55             tb[tmp]=i;
56     }
57     for(LL i=0;i<sq;++i){
58         if(tb.count(b)){
59             LL res=tb[b];
60             return i*sq+(res==sq?0:res);
61         }
62         b=(b*inv)%p;
63     }
64     return -1;
65 }
66
67 int main(){
68     ios::sync_with_stdio(false);
69     cin.tie(0),cout.tie(0);
70     while(cin>>P>>B>>N){
71         LL ans=BSGS(B,N,P);
72         if(ans!=-1)
73             cout<<"no solution\n";
74         else
75             cout<<ans<<"\n";
76     }
77 }
78

```

6 algorithm

6.1 basic

```

1 min_element：找尋最小元素
2 min_element(first, last)
3 max_element：找尋最大元素
4 max_element(first, last)
5 sort：排序，預設由小排到大。
6 sort(first, last)
7 sort(first, last, cmp)：可自行定義比較運算子 cmp。
8 find：尋找元素。

```

```

9 find(first, last, val)
10 lower_bound : 尋找第一個小於 x 的元素位置，
11             如果不存在，則回傳 last。
12 lower_bound(first, last, val)
13 upper_bound : 尋找第一個大於 x 的元素位置，
14             如果不存在，則回傳 last。
15 upper_bound(first, last, val)
16 next_permutation : 將序列順序轉換成下一個字典序，
17                  如果存在回傳 true，反之回傳 false。
18 next_permutation(first, last)
19 prev_permutation : 將序列順序轉換成上一個字典序，
20                  如果存在回傳 true，反之回傳 false。
21 prev_permutation(first, last)

```

6.2 二分搜

```

1 int binary_search(int target) {
2     // For range [ok, ng) or (ng, ok], "ok" is for the
3     // index that target value exists, with "ng" doesn't.
4     int ok = maxn, ng = -1;
5     // For first lower_bound, ok=maxn and ng=-1,
6     // for last lower_bound, ok = -1 and ng = maxn
7     // (the "check" funtion
8     // should be changed depending on it.)
9     while(abs(ok - ng) > 1) {
10         int mid = (ok + ng) >> 1;
11         if(check(mid)) ok = mid;
12         else ng = mid;
13     }
14     // Be careful, "arr[mid]>=target" for first
15     // lower_bound and "arr[mid]<=target" for
16     // last lower_bound. For range (ng, ok],
17     // convert it into (ng, mid] and (mid, ok] than
18     // choose the first one, or convert [ok, ng) into
19     // [ok, mid) and [mid, ng) and than choose
20     // the second one.
21     return ok;
22 }
23
24 lower_bound(arr, arr + n, k);    //最左邊 ≥ k 的位置
25 upper_bound(arr, arr + n, k);    //最左邊 > k 的位置
26 upper_bound(arr, arr + n, k) - 1; //最右邊 ≤ k 的位置
27 lower_bound(arr, arr + n, k) - 1; //最右邊 < k 的位置
28 (lower_bound, upper_bound)      //等於 k 的範圍
29 equal_range(arr, arr+n, k);

```

6.3 三分搜

```

1 題意
2 給定兩射線方向和速度，問兩射線最近距離。
3
4 題解
5 假設 F(t) 為兩射線在時間 t 的距離，F(t) 為二次函數，
6 可用三分搜找二次函數最小值。
7
8 #include <bits/stdc++.h>
9 using namespace std;
10
11 struct Point{
12     double x, y, z;
13     Point() {}
14     Point(double _x, double _y, double _z):
15         x(_x), y(_y), z(_z){}
16     friend istream& operator>>(istream& is, Point& p)
17     {
18         is >> p.x >> p.y >> p.z;
19         return is;
20     }
21     Point operator+(const Point &rhs) const{
22         return Point(x+rhs.x, y+rhs.y, z+rhs.z);
23     }

```

```

23 Point operator-(const Point &rhs) const{
24     return Point(x-rhs.x, y-rhs.y, z-rhs.z);
25 }
26 Point operator*(const double &d) const{
27     return Point(x*d, y*d, z*d);
28 }
29 Point operator/(const double &d) const{
30     return Point(x/d, y/d, z/d);
31 }
32 double dist(const Point &rhs) const{
33     double res = 0;
34     res+=(x-rhs.x)*(x-rhs.x);
35     res+=(y-rhs.y)*(y-rhs.y);
36     res+=(z-rhs.z)*(z-rhs.z);
37     return res;
38 }
39 };
40
41 int main(){
42     ios::sync_with_stdio(false);
43     cin.tie(0), cout.tie(0);
44     int T;
45     cin>>T;
46     for(int ti=1; ti<=T; ++ti){
47         double time;
48         Point x1, y1, d1, x2, y2, d2;
49         cin>>time>>x1>>y1>>x2>>y2;
50         d1=(y1-x1)/time;
51         d2=(y2-x2)/time;
52         double L=0, R=1e8, m1, m2, f1, f2;
53         double ans = x1.dist(x2);
54         while(abs(L-R)>1e-10){
55             m1=(L+R)/2;
56             m2=(m1+R)/2;
57             f1=((d1*m1)+x1).dist((d2*m1)+x2);
58             f2=((d1*m2)+x1).dist((d2*m2)+x2);
59             ans = min(ans, min(f1, f2));
60             if(f1<f2) R=m2;
61             else L=m1;
62         }
63         cout<<"Case "<<ti<<": ";
64         cout<<fixed<<setprecision(4)<<sqrt(ans)<<"\n";
65     }
66 }

```

6.4 prefix sum

```

1 // 前綴和
2 陣列前n項的和。
3 b[i]=a[0]+a[1]+a[2]+...+a[i]
4 區間和 [l, r] : b[r]-b[l-1] (要保留b[l]所以-1)
5
6 #include<bits/stdc++.h>
7 using namespace std;
8 int main(){
9     int n;
10    cin>>n;
11    int a[n], b[n];
12    for(int i=0; i<n; i++) cin>>a[i];
13    b[0]=a[0];
14    for(int i=1; i<n; i++) b[i]=b[i-1]+a[i];
15    for(int i=0; i<n; i++) cout<<b[i]<<" ";
16    cout<<"\n";
17    int l, r;
18    cin>>l>>r;
19    cout<<b[r]-b[l-1]; //區間和
20 }

```

6.5 差分

```

1 // 差分
2 用途：在區間 [l, r] 加上一個數字v。
3 b[l] += v; (b[0~l] 加上v)

```



```

4 | b[r+1] -= v; (b[r+1~n] 減去v (b[r] 仍保留v) )
5 | 給的 a[] 是前綴和數列，建構 b[]，
6 | 因為 a[i] = b[0] + b[1] + b[2] + ... + b[i]，
7 | 所以 b[i] = a[i] - a[i-1]。
8 | 在 b[1] 加上 v，b[r+1] 減去 v，
9 | 最後再從 0 跑到 n 使 b[i] += b[i-1]。
10 | 這樣一來，b[] 是一個在某區間加上v的前綴和。
11 |
12 | #include <bits/stdc++.h>
13 | using namespace std;
14 | int a[1000], b[1000];
15 | // a: 前綴和數列, b: 差分數列
16 | int main(){
17 |     int n, l, r, v;
18 |     cin >> n;
19 |     for(int i=1; i<=n; i++){
20 |         cin >> a[i];
21 |         b[i] = a[i] - a[i-1]; //建構差分數列
22 |     }
23 |     cin >> l >> r >> v;
24 |     b[l] += v;
25 |     b[r+1] -= v;
26 |
27 |     for(int i=1; i<=n; i++){
28 |         b[i] += b[i-1];
29 |         cout << b[i] << ' ';
30 |     }
31 | }

```

6.6 greedy

```

1 | //貪心
2 | 貪心演算法的核心為，
3 | 採取在目前狀態下最好或最佳（即最有利）的選擇。
4 | 貪心演算法雖然能獲得當前最佳解，
5 | 但不保證能獲得最後（全域）最佳解，
6 | 提出想法後可以先試圖尋找有沒有能推翻原本的想法的反例，
7 | 確認無誤再實作。

```

刪數字問題

```

11 | //problem
12 | 給定一個數字 N( $\leq 10^4$ )，需要刪除 K 個數字，
13 | 請問刪除 K 個數字後最小的數字為何？
14 |
15 | //solution
16 | 刪除滿足第 i 位數大於第 i+1 位數的最左邊第 i 位數，
17 | 扣除高位數的影響較扣除低位數的大。

```

```

19 | //code
20 | int main(){
21 |     string s;
22 |     int k;
23 |     cin>>s>>k;
24 |     for(int i=0; i<k; ++i){
25 |         if((int)s.size()==0) break;
26 |         int pos =(int)s.size()-1;
27 |         for(int j=0; j<(int)s.size()-1; ++j){
28 |             if(s[j]>s[j+1]){
29 |                 pos=j;
30 |                 break;
31 |             }
32 |         }
33 |         s.erase(pos,1);
34 |     }
35 |     while((int)s.size()>0&&s[0]=='0')
36 |         s.erase(0,1);
37 |     if((int)s.size()) cout<<s<<'\n';
38 |     else cout<<0<<'\n';
39 | }

```

最小區間覆蓋長度

```

43 | //problem
44 | 給定 n 條線段區間為 [Li,Ri]，
45 | 請問最少要選幾個區間才能完全覆蓋 [0,S]?
46 |
47 | //solution
48 | 先將所有區間依照左界由小到大排序，
49 | 對於當前區間 [Li,Ri]，要從左界 >Ri 的所有區間中，
50 | 找到有著最大的右界的區間，連接當前區間。
51 |
52 | //problem
53 | 長度 n 的直線中有數個加熱器，
54 | 在 x 的加熱器可以讓 [x-r,x+r] 內的物品加熱，
55 | 問最少要幾個加熱器可以把 [0,n] 的範圍加熱。
56 |
57 | //solution
58 | 對於最左邊沒加熱的點a，選擇最遠可以加熱a的加熱器，
59 | 更新已加熱範圍，重複上述動作繼續尋找加熱器。
60 |
61 | //code
62 | int main(){
63 |     int n, r;
64 |     int a[1005];
65 |     cin>>n>>r;
66 |     for(int i=1; i<=n; ++i) cin>>a[i];
67 |     int i=1, ans=0;
68 |     while(i<=n){
69 |         int R=min(i+r-1,n), L=max(i-r+1,0)
70 |         int nextR=-1;
71 |         for(int j=R; j>=L; --j){
72 |             if(a[j]){
73 |                 nextR=j;
74 |                 break;
75 |             }
76 |         }
77 |         if(nextR==-1){
78 |             ans=-1;
79 |             break;
80 |         }
81 |         ++ans;
82 |         i=nextR+r;
83 |     }
84 |     cout<<ans<<'\n';
85 | }

```

最多不重疊區間

```

89 | //problem
90 | 給你 n 條線段區間為 [Li,Ri]，
91 | 請問最多可以選擇幾條不重疊的線段(頭尾可相連)?
92 |
93 | //solution
94 | 依照右界由小到大排序，
95 | 每次取到一個不重疊的線段，答案 +1。

```

```

97 | //code
98 | struct Line{
99 |     int L,R;
100 |     bool operator<(const Line &rhs) const{
101 |         return R<rhs.R;
102 |     }
103 | };
104 |
105 | int main(){
106 |     int t;
107 |     cin>>t;
108 |     Line a[30];
109 |     while(t--){
110 |         int n=0;
111 |         while(cin>>a[n].L>>a[n].R, a[n].L||a[n].R)
112 |             ++n;
113 |         sort(a,a+n);
114 |         int ans=1, R=a[0].R;
115 |         for(int i=1; i<n; ++i){
116 |             if(a[i].L>=R){
117 |                 ++ans;
118 |                 R=a[i].R;

```

```

119     }
120     }
121     cout<<ans<<'\n';
122 }
123 }
124
125
126 最小化最大延遲問題
127 //problem
128 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
129 期限是  $D_i$ ，第  $i$  項工作延遲的時間為  $Li=\max(0, Fi-D_i)$ ，
130 原本  $Fi$  為第  $i$  項工作的完成時間，
131 求一種工作排序使  $\max Li$  最小。

```

```

132 //solution
133 按照到期時間從早到晚處理。
134
135 //code
136 struct Work{
137     int t, d;
138     bool operator<(const Work &rhs)const{
139         return d<rhs.d;
140     }
141 };
142
143
144 int main(){
145     int n;
146     Work a[10000];
147     cin>>n;
148     for(int i=0;i<n;++i)
149         cin>>a[i].t>>a[i].d;
150     sort(a,a+n);
151     int maxL=0, sumT=0;
152     for(int i=0;i<n;++i){
153         sumT+=a[i].t;
154         maxL=max(maxL, sumT-a[i].d);
155     }
156     cout<<maxL<<'\n';
157 }

```

```

158
159
160 最少延遲數量問題
161 //problem
162 給定 N 個工作，每個工作的需要處理時長為  $T_i$ ，
163 期限是  $D_i$ ，求一種工作排序使得逾期工作數量最小。
164
165 //solution
166 期限越早到期的工作越先做。將工作依照到期時間從早到晚排序，
167 依序放入工作列表中，如果發現有工作預期，
168 就從目前選擇的工作中，移除耗時最長的工作。

```

上述方法為 Moore-Hodgson's Algorithm。

```

171 //problem
172 給定烏龜的重量和可承受重量，問最多可以疊幾隻烏龜？
173
174 //solution
175 和最少延遲數量問題是相同的問題，只要將題敘做轉換。
176 工作處理時長 → 烏龜重量
177 工作期限 → 烏龜可承受重量
178 多少工作不延期 → 可以疊幾隻烏龜

```

```

179 //code
180 struct Work{
181     int t, d;
182     bool operator<(const Work &rhs)const{
183         return d<rhs.d;
184     }
185 };
186
187
188 int main(){
189     int n=0;
190     Work a[10000];
191     priority_queue<int> pq;
192     while(cin>>a[n].t>>a[n].d)

```

```

193         ++n;
194     sort(a,a+n);
195     int sumT=0, ans=n;
196     for(int i=0;i<n;++i){
197         pq.push(a[i].t);
198         sumT+=a[i].t;
199         if(a[i].d<sumT){
200             int x=pq.top();
201             pq.pop();
202             sumT-=x;
203             --ans;
204         }
205     }
206     cout<<ans<<'\n';
207 }

```

```

208
209
210 任務調度問題
211 //problem
212 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
213 期限是  $D_i$ ，如果第  $i$  項工作延遲需要受到  $p_i$  單位懲罰，
214 請問最少會受到多少單位懲罰。
215
216 //solution
217 依照懲罰由大到小排序，
218 每項工作依序嘗試可不可以放在  $D_i-T_i+1, D_i-T_i, \dots, 1, 0$ ，
219 如果有空閒就放進去，否則延後執行。

```

```

220 //problem
221 給定 N 項工作，每項工作的需要處理時長為  $T_i$ ，
222 期限是  $D_i$ ，如果第  $i$  項工作在期限內完成會獲得  $a_i$ 
223 單位獎勵，
224 請問最多會獲得多少單位獎勵。
225
226 //solution
227 和上題相似，這題變成依照獎勵由大到小排序。

```

```

228 //code
229 struct Work{
230     int d, p;
231     bool operator<(const Work &rhs)const{
232         return p>rhs.p;
233     }
234 };
235
236
237 int main(){
238     int n;
239     Work a[100005];
240     bitset<100005> ok;
241     while(cin>>n){
242         ok.reset();
243         for(int i=0;i<n;++i)
244             cin>>a[i].d>>a[i].p;
245         sort(a,a+n);
246         int ans=0;
247         for(int i=0;i<n;++i){
248             int j=a[i].d;
249             while(j--){
250                 if(!ok[j]){
251                     ans+=a[i].p;
252                     ok[j]=true;
253                     break;
254                 }
255             }
256             cout<<ans<<'\n';
257         }
258     }

```

6.7 floyd warshall

```

1 int w[n][n];
2 int d[n][n];
3 int p[n][n];
4 // 由 i 點到 j 點的路徑，其中繼點為 p[i][j]。
5

```



```

6 void floyd_warshall(){ //O(V^3)
7     for(int i=0;i<n;i++){
8         for(int j=0;j<n;j++){
9             d[i][j]=w[i][j];
10            p[i][j]=-1; // 預設為沒有中繼點
11        }
12        for(int i=0;i<n;i++) d[i][i]=0;
13        for(int k=0;k<n;k++){
14            for(int i=0;i<n;i++){
15                for(int j=0;j<n;j++){
16                    if(d[i][k]+d[k][j]<d[i][j]){
17                        d[i][j]=d[i][k]+d[k][j];
18                        p[i][j]=k; // 由i點走到j點經過了k點
19                    }
20                }
21            }
22        // 這支函式並不會印出起點和終點，必須另行印出。
23        void find_path(int s,int t){ // 印出最短路徑
24            if(p[s][t]==-1) return; // 沒有中繼點就結束
25            find_path(s,p[s][t]); // 前半段最短路徑
26            cout<<p[s][t]; // 中繼點
27            find_path(p[s][t],t); // 後半段最短路徑
28        }

```

6.8 dinic

```

1 const int maxn = 1e5 + 10;
2 const int inf = 0x3f3f3f3f;
3
4 struct Edge {
5     int s, t, cap, flow;
6 };
7
8 int n, m, S, T;
9 int level[maxn], dfs_idx[maxn];
10 vector<Edge> E;
11 vector<vector<int>> G;
12
13 void init() {
14     S = 0;
15     T = n + m;
16     E.clear();
17     G.assign(maxn, vector<int>());
18 }
19
20 void addEdge(int s, int t, int cap) {
21     E.push_back({s, t, cap, 0});
22     E.push_back({t, s, 0, 0});
23     G[s].push_back(E.size()-2);
24     G[t].push_back(E.size()-1);
25 }
26
27 bool bfs() {
28     queue<int> q({S});
29
30     memset(level, -1, sizeof(level));
31     level[S] = 0;
32
33     while(!q.empty()) {
34         int cur = q.front();
35         q.pop();
36
37         for(int i : G[cur]) {
38             Edge e = E[i];
39             if(level[e.t]==-1 && e.cap>e.flow) {
40                 level[e.t] = level[e.s] + 1;
41                 q.push(e.t);
42             }
43         }
44     }
45     return level[T];
46 }
47
48 int dfs(int cur, int lim) {
49     if(cur==T || lim==0) return lim;

```

```

50
51     int result = 0;
52     for(int& i=dfs_idx[cur]; i<G[cur].size() && lim; i++) {
53         Edge& e = E[G[cur][i]];
54         if(level[e.s]+1 != level[e.t]) continue;
55
56         int flow = dfs(e.t, min(lim, e.cap-e.flow));
57         if(flow <= 0) continue;
58
59         e.flow += flow;
60         result += flow;
61         E[G[cur][i]^1].flow -= flow;
62         lim -= flow;
63     }
64     return result;
65 }
66
67 int dinic() { // O((V^2)E)
68     int result = 0;
69     while(bfs()) {
70         memset(dfs_idx, 0, sizeof(dfs_idx));
71         result += dfs(S, inf);
72     }
73     return result;
74 }

```

6.9 Nim Game

```

1 //兩人輪流取銅板，每人每次需在某堆取一枚以上的銅板，
2 //但不能同時在兩堆取銅板，直到最後，
3 //將銅板拿光的人贏得此遊戲。
4
5 #include <bits/stdc++.h>
6 #define maxn 23+5
7 using namespace std;
8
9 int SG[maxn];
10 int visited[1000+5];
11 int pile[maxn],ans;
12
13 void calculateSG(){
14     SG[0]=0;
15     for(int i=1;i<=maxn;i++){
16         int cur=0;
17         for(int j=0;j<i;j++){
18             for(int k=0;k<=j;k++){
19                 visited[SG[j]^SG[k]]=i;
20             }
21             while(visited[cur]==i) cur++;
22             SG[i]=cur;
23         }
24     }
25
26 int main(){
27     calculateSG();
28     int Case=0,n;
29     while(cin>>n,n){
30         ans=0;
31         for(int i=1;i<=n;i++) cin>>pile[i];
32         for(int i=1;i<=n;i++){
33             if(pile[i]&1) ans^=SG[n-i];
34             cout<<"Game "<<Case<<": ";
35             if(!ans) cout<<"-1 -1 -1\n";
36             else{
37                 bool flag=0;
38                 for(int i=1;i<=n;i++){
39                     if(pile[i]){
40                         for(int j=i+1;j<=n;j++){
41                             for(int k=j;k<=n;k++){
42                                 if((SG[n-i]^SG[n-j]^SG[n-k])==ans){
43                                     cout<<i-1<<" "<<j-1<<" "<<k-1<<endl;
44                                     flag=1;
45                                     break;
46                                 }

```

```

47         if(flag) break;
48     }
49     if(flag) break;
50 }
51 }
52 }
53 }
54 return 0;
55 }
56
57 /*
58 input
59 4 1 0 1 100
60 3 1 0 5
61 2 2 1
62 0
63 output
64 Game 1: 0 2 3
65 Game 2: 0 1 1
66 Game 3: -1 -1 -1
67 */

```

6.10 SPFA

```

1 struct Edge
2 {
3     int t;
4     long long w;
5     Edge(){};
6     Edge(int _t, long long _w) : t(_t), w(_w) {}
7 };
8
9 bool SPFA(int st) // 平均 $O(V + E)$  最糟 $O(VE)$ 
10 {
11     vector<int> cnt(n, 0);
12     bitset<MXV> inq(0);
13     queue<int> q;
14     q.push(st);
15     dis[st] = 0;
16     inq[st] = true;
17     while (!q.empty())
18     {
19         int cur = q.front();
20         q.pop();
21         inq[cur] = false;
22         for (auto &e : G[cur])
23         {
24             if (dis[e.t] <= dis[cur] + e.w)
25                 continue;
26             dis[e.t] = dis[cur] + e.w;
27             if (inq[e.t])
28                 continue;
29             ++cnt[e.t];
30             if (cnt[e.t] > n)
31                 return false; // negative cycle
32             inq[e.t] = true;
33             q.push(e.t);
34         }
35     }
36     return true;
37 }

```

6.11 dijkstra

```

1 #include<bits/stdc++.h>
2 #define maxn 50000+5
3 #define INF 0x3f3f3f3f
4 using namespace std;
5
6 struct edge{
7     int v,w;
8 };
9

```

```

10 struct Item{
11     int u,dis;
12     bool operator< (const Item &rhs) const{
13         return dis>rhs.dis;
14     }
15 };
16
17 vector<edge> G[maxn];
18 int dist[maxn];
19
20 void dijkstra(int s){ //  $O((V + E)\log(E))$ 
21     memset(dist,INF,sizeof(dist));
22     dist[s]=0;
23     priority_queue<Item> pq;
24     pq.push({s,0});
25     while(!pq.empty()){
26         Item now=pq.top();
27         pq.pop();
28         if(now.dis>dist[now.u]) continue;
29         for(edge e:G[now.u]){
30             if(dist[e.v]>dist[now.u]+e.w){
31                 dist[e.v]=dist[now.u]+e.w;
32                 pq.push({e.v,dist[e.v]});
33             }
34         }
35     }
36 }
37
38 int main(){
39     int t,cas=1;
40     cin>>t;
41     while(t--){
42         int n,m,s,t;
43         cin>>n>>m>>s>>t;
44         for(int i=0;i<=n;i++) G[i].clear();
45         int u,v,w;
46         for(int i=0;i<m;i++){
47             cin>>u>>v>>w;
48             G[u].push_back({v,w});
49             G[v].push_back({u,w});
50         }
51         dijkstra(s);
52         cout<<"Case #"<<cas++<<": ";
53         if(dist[t]==INF) cout<<"unreachable\n";
54         else cout<<dist[t]<<endl;
55     }
56 }

```

6.12 SCC Tarjan

```

1 #include <cstdio>
2 #include <vector>
3 #include <cstring>
4 #include <stack>
5 #include <algorithm>
6 using namespace std;
7 /*單純考SCC，每個SCC中找成本最小的蓋，如果有多個一樣小的要數出來*/
8 #define maxn 100005
9 #define MOD 1000000007
10 long long cost[maxn];
11 vector<vector<int>>> G;
12 int SCC = 0;
13 stack<int> sk;
14 int dfn[maxn];
15 int low[maxn];
16 bool inStack[maxn];
17 int dfsTime = 1;
18 long long totalCost = 0;
19 long long ways = 1;
20 void dfs(int u)
21 {
22     dfn[u] = low[u] = dfsTime;
23     ++dfsTime;
24     sk.push(u);
25     inStack[u] = true;

```

```

26 for (int v: G[u])
27 {
28     if (dfn[v] == 0)
29     {
30         dfs(v);
31         low[u] = min(low[u], low[v]);
32     }
33     else if (inStack[v])
34     {
35         //屬於同個SCC且是我的back edge
36         low[u] = min(low[u], dfn[v]);
37     }
38 }
39 //如果是SCC
40 if (dfn[u] == low[u])
41 {
42     long long minCost = 0x3f3f3f3f;
43     int currWays = 0;
44     ++SCC;
45     while (1)
46     {
47         int v = sk.top();
48         inStack[v] = 0;
49         sk.pop();
50         if (minCost > cost[v])
51         {
52             minCost = cost[v];
53             currWays = 1;
54         }
55         else if (minCost == cost[v])
56         {
57             ++currWays;
58         }
59         if (v == u)
60             break;
61     }
62     totalCost += minCost;
63     ways = (ways * currWays) % MOD;
64 }
65 }
66 int main()
67 {
68     int n;
69     scanf("%d", &n);
70     for (int i = 1; i <= n; ++i)
71         scanf("%lld", &cost[i]);
72     G.assign(n + 5, vector<int>());
73     int m;
74     scanf("%d", &m);
75     int u, v;
76     for (int i = 0; i < m; ++i)
77     {
78         scanf("%d %d", &u, &v);
79         G[u].emplace_back(v);
80     }
81
82     for (int i = 1; i <= n; ++i)
83     {
84         if (dfn[i] == 0)
85             dfs(i);
86     }
87
88     printf("%lld %lld\n", totalCost, ways % MOD);
89     return 0;
90 }

```

6.13 SCC Kosaraju

```

1 //做兩次dfs, O(V + E)
2 //g 是原圖, g2 是反圖
3 //s是dfs離開的節點
4 void dfs1(int u) {
5     vis[u] = true;
6     for (int v : g[u])
7         if (!vis[v]) dfs1(v);

```

```

8     s.push_back(u);
9 }
10
11 void dfs2(int u) {
12     group[u] = sccCnt;
13     for (int v : g2[u])
14         if (!group[v]) dfs2(v);
15 }
16
17 void kosaraju() {
18     sccCnt = 0;
19     for (int i = 1; i <= n; ++i)
20         if (!vis[i]) dfs1(i);
21     for (int i = n; i >= 1; --i)
22         if (!group[s[i]]) {
23             ++sccCnt;
24             dfs2(s[i]);
25         }
26 }

```

6.14 ArticulationPoints Tarjan

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 vector<vector<int>>> G;
5 int N;
6 int timer;
7 bool visited[105];
8 int visTime[105]; // 第一次visit的時間
9 int low[105];
10 // 最小能回到的父節點(不能是自己的parent)的visTime
11 int res;
12 //求割點數量
13 void tarjan(int u, int parent) {
14     int child = 0;
15     bool isCut = false;
16     visited[u] = true;
17     visTime[u] = low[u] = ++timer;
18     for (int v: G[u]) {
19         if (!visited[v]) {
20             ++child;
21             tarjan(v, u);
22             low[u] = min(low[u], low[v]);
23             if (parent != -1 && low[v] >= visTime[u])
24                 isCut = true;
25         }
26         else if (v != parent)
27             low[u] = min(low[u], visTime[v]);
28     }
29
30     //If u is root of DFS tree->有兩個以上的children
31     if (parent == -1 && child >= 2)
32         isCut = true;
33     if (isCut)
34         ++res;
35 }
36
37 int main()
38 {
39     char input[105];
40     char* token;
41     while (scanf("%d", &N) != EOF && N)
42     {
43         G.assign(105, vector<int>());
44         memset(visited, false, sizeof(visited));
45         memset(low, 0, sizeof(low));
46         memset(visTime, 0, sizeof(visTime));
47         timer = 0;
48         res = 0;
49         getchar(); // for \n
50         while (fgets(input, 105, stdin))
51         {
52             if (input[0] == '\0')
53                 break;
54             int size = strlen(input);

```

```

54     input[size - 1] = '\0';
55     --size;
56     token = strtok(input, " ");
57     int u = atoi(token);
58     int v;
59     while (token = strtok(NULL, " "))
60     {
61         v = atoi(token);
62         G[u].emplace_back(v);
63         G[v].emplace_back(u);
64     }
65 }
66 tarjan(1, -1);
67 printf("%d\n", res);
68 }
69 return 0;
70 }

```

6.15 最小樹狀圖

```

1  定義
2  有向圖上的最小生成樹 (Directed Minimum Spanning Tree)
3  稱為最小樹形圖。
4
5  const int maxn = 60 + 10;
6  const int inf = 0x3f3f3f3f;
7
8  struct Edge {
9      int s, t, cap, cost;
10 }; // cap 為頻寬 (optional)
11
12 int n, m, c;
13 int inEdge[maxn], idx[maxn], pre[maxn], vis[maxn];
14
15 // 對於每個點，選擇對它入度最小的那條邊
16 // 找環，如果沒有則 return;
17 // 進行縮環並更新其他點到環的距離。
18 int dirMST(vector<Edge> edges, int low) {
19     int result = 0, root = 0, N = n;
20
21     while(true) {
22         memset(inEdge, 0x3f, sizeof(inEdge));
23
24         // 找所有點的 in edge 放進 inEdge
25         // optional: low 為最小 cap 限制
26         for(const Edge& e : edges) {
27             if(e.cap < low) continue;
28             if(e.s!=e.t && e.cost<inEdge[e.t]) {
29                 inEdge[e.t] = e.cost;
30                 pre[e.t] = e.s;
31             }
32         }
33
34         for(int i=0; i<N; i++) {
35             if(i!=root && inEdge[i]==inf)
36                 return -1; //除了root 還有點沒有in edge
37         }
38
39         int seq = inEdge[root] = 0;
40         memset(idx, -1, sizeof(idx));
41         memset(vis, -1, sizeof(vis));
42
43         // 找所有的 cycle，一起編號為 seq
44         for(int i=0; i<N; i++) {
45             result += inEdge[i];
46             int cur = i;
47             while(vis[cur]!=i && idx[cur]==-1) {
48                 if(cur == root) break;
49                 vis[cur] = i;
50                 cur = pre[cur];
51             }
52             if(cur!=root && idx[cur]==-1) {
53                 for(int j=pre[cur]; j!=cur; j=pre[j])
54                     idx[j] = seq;

```

```

55         idx[cur] = seq++;
56     }
57 }
58
59 if(seq == 0) return result; // 沒有 cycle
60
61 for(int i=0; i<N; i++)
62     // 沒有被縮點的點
63     if(idx[i] == -1) idx[i] = seq++;
64
65 // 縮點並重新編號
66 for(Edge& e : edges) {
67     if(idx[e.s] != idx[e.t])
68         e.cost -= inEdge[e.t];
69     e.s = idx[e.s];
70     e.t = idx[e.t];
71 }
72 N = seq;
73 root = idx[root];
74 }
75 }
76
77 =====
78
79 Tarjan 的DMST 演算法
80 Tarjan 提出了一種能夠在
81 O(m+nlog n)時間內解決最小樹形圖問題的演算法。
82
83 流程
84 Tarjan 的演算法分為收縮與伸展兩個過程。
85 接下來先介紹收縮的過程。
86 我們要假設輸入的圖是滿足強連通的，
87 如果不滿足那就加入 o(n) 條邊使其滿足，
88 並且這些邊的邊權是無窮大的。
89
90 我們需要一個堆存儲結點的入邊編號，入邊權值，
91 結點總代價等相關信息，由於後續過程中會有堆的合併操作，
92 這裡採用左偏樹 與並查集實現。
93 演算法的每一步都選擇一個任意結點v，
94 需要保證v不是根節點，並且在堆中沒有它的入邊。
95 再將v的最小入邊加入到堆中，
96 如果新加入的這條邊使堆中的邊形成了環，
97 那麼將構成環的那些結點收縮，
98 我們不妨將這些已經收縮的結點命名為超級結點，
99 再繼續這個過程，如果所有的頂點都縮成了超級結點，
100 那麼收縮過程就結束了。
101 整個收縮過程結束後會得到一棵收縮樹，
102 之後就會對它進行伸展操作。
103
104 堆中的邊總是會形成一條路徑v0 <- v1<- ... <- vk，
105 由於圖是強連通的，這個路徑必然存在，
106 並且其中的 vi 可能是最初的單一結點，
107 也可能是壓縮後的超級結點。
108
109 最初有 v0=a，其中 a 是圖中任意的一個結點，
110 每次都選擇一條最小入邊 vk <- u，
111 如果 u 不是v0,v1,...,vk中的一個結點，
112 那麼就將結點擴展到 v k+1=u。
113 如果 u 是他們其中的一個結點 vi，
114 那麼就找到了一個關於 vi <- ... <- vk <- vi的環，
115 再將他們收縮為一個超級結點c。
116
117 向隊列 P 中放入所有的結點或超級結點，
118 並初始選擇任一節點 a，只要佇列不為空，就進行以下步驟：
119
120 選擇 a 的最小入邊，保證不存在自環，
121 並找到另一頭的結點 b。
122 如果結點b沒有被記錄過說明未形成環，
123 令 a <- b，繼續目前操作尋找環。
124
125 如果 b 被記錄過了，就表示出現了環。
126 總結點數加一，並將環上的所有結點重新編號，對堆進行合併，

```

```

127 以及結點/超級結點的總權值的更新。
128 更新權值操作就是將環上所有結點的入邊都收集起來，
129 並減去環上入邊的邊權。
130
131 typedef long long ll;
132 #define maxn 102
133 #define INF 0x3f3f3f3f
134
135 struct UnionFind {
136     int fa[maxn << 1];
137     UnionFind() { memset(fa, 0, sizeof(fa)); }
138     void clear(int n) {
139         memset(fa + 1, 0, sizeof(int) * n);
140     }
141     int find(int x) {
142         return fa[x] ? fa[x] = find(fa[x]) : x;
143     }
144     int operator[](int x) { return find(x); }
145 };
146
147 struct Edge {
148     int u, v, w, w0;
149 };
150
151 struct Heap {
152     Edge *e;
153     int rk, constant;
154     Heap *lch, *rch;
155
156     Heap(Edge *_e):
157         e(_e), rk(1), constant(0), lch(NULL), rch(NULL) {}
158
159     void push() {
160         if (lch) lch->constant += constant;
161         if (rch) rch->constant += constant;
162         e->w += constant;
163         constant = 0;
164     }
165 };
166
167 Heap *merge(Heap *x, Heap *y) {
168     if (!x) return y;
169     if (!y) return x;
170     if (x->e->w + x->constant > y->e->w + y->constant)
171         swap(x, y);
172     x->push();
173     x->rch = merge(x->rch, y);
174     if (!x->lch || x->lch->rk < x->rch->rk)
175         swap(x->lch, x->rch);
176     if (x->rch)
177         x->rk = x->rch->rk + 1;
178     else
179         x->rk = 1;
180     return x;
181 }
182
183 Edge *extract(Heap *&x) {
184     Edge *r = x->e;
185     x->push();
186     x = merge(x->lch, x->rch);
187     return r;
188 }
189
190 vector<Edge> in[maxn];
191 int n, m, fa[maxn << 1], nxt[maxn << 1];
192 Edge *ed[maxn << 1];
193 Heap *Q[maxn << 1];
194 UnionFind id;
195
196 void contract() {
197     bool mark[maxn << 1];
198     //將圖上的每一個節點與其相連的那些節點進行記錄
199     for (int i = 1; i <= n; i++) {
200         queue<Heap *> q;
201         for (int j = 0; j < in[i].size(); j++)
202             q.push(new Heap(&in[i][j]));
203         while (q.size() > 1) {

```

```

204             Heap *u = q.front();
205             q.pop();
206             Heap *v = q.front();
207             q.pop();
208             q.push(merge(u, v));
209         }
210         Q[i] = q.front();
211     }
212     mark[1] = true;
213     for (int a = 1, b = 1, p; Q[a]; b = a, mark[b] = true) {
214         //尋找最小入邊以及其端點，保證無環
215         do {
216             ed[a] = extract(Q[a]);
217             a = id[ed[a]->u];
218         } while (a == b && Q[a]);
219         if (a == b) break;
220         if (!mark[a]) continue;
221         //對發現的環進行收縮，以及環內的節點重新編號，
222         //總權值更新
223         for (a = b, n++; a != n; a = p) {
224             id.fa[a] = fa[a] = n;
225             if (Q[a]) Q[a]->constant -= ed[a]->w;
226             Q[n] = merge(Q[n], Q[a]);
227             p = id[ed[a]->u];
228             nxt[p == n ? b : p] = a;
229         }
230     }
231 }
232
233 ll expand(int x, int r);
234 ll expand_iter(int x) {
235     ll r = 0;
236     for (int u = nxt[x]; u != x; u = nxt[u]) {
237         if (ed[u]->w0 >= INF)
238             return INF;
239         else
240             r += expand(ed[u]->v, u) + ed[u]->w0;
241     }
242     return r;
243 }
244
245 ll expand(int x, int t) {
246     ll r = 0;
247     for (; x != t; x = fa[x]) {
248         r += expand_iter(x);
249         if (r >= INF) return INF;
250     }
251     return r;
252 }
253
254 void link(int u, int v, int w) {
255     in[v].push_back({u, v, w, w});
256 }
257
258 int main() {
259     int rt;
260     scanf("%d %d %d", &n, &m, &rt);
261     for (int i = 0; i < m; i++) {
262         int u, v, w;
263         scanf("%d %d %d", &u, &v, &w);
264         link(u, v, w);
265     }
266     //保證強連通
267     for (int i = 1; i <= n; i++)
268         link(i > 1 ? i - 1 : n, i, INF);
269     contract();
270     ll ans = expand(rt, n);
271     if (ans >= INF)
272         puts("-1");
273     else
274         printf("%lld\n", ans);
275     return 0;
276 }

```

6.16 二分圖最大匹配

```

1 #include <iostream>
2 #include <string>
3 #include <cmath>
4 #include <cstring>
5 #include <vector>
6 using namespace std;
7 /* 核心: 最大點獨立集 = |V| -
   /最大匹配數/, 用匈牙利演算法找出最大匹配數 */
8
9 struct Student {
10     int height;
11     char sex;
12     string musicStyle;
13     string sport;
14     bool canMatch(const Student& other) {
15         return ((abs(this->height - other.height) <=
16             40) && (this->musicStyle ==
17             other.musicStyle)
18             && (this->sport != other.sport));
19     }
20     friend istream& operator >> (istream& input,
21         Student& student);
22 };
23 vector<Student> boys;
24 vector<Student> girls;
25 vector<vector<int>>> G;
26 bool used[505];
27 int p[505]; //pair of boys and girls -> p[j] = i
   代表i男生連到j女生
28 istream& operator >> (istream& input, Student&
   student) {
29     input >> student.height >> student.sex >>
30     student.musicStyle >> student.sport;
31     return input;
32 }
33 bool match(int i) {
34     for (int j: G[i]) {
35         if (!used[j]) {
36             used[j] = true;
37             if (p[j] == -1 || match(p[j])) {
38                 p[j] = i;
39                 return true;
40             }
41         }
42     }
43     return false;
44 }
45 void maxMatch(int n) {
46     memset(p, -1, sizeof(p));
47     int res = 0;
48     for (int i = 0; i < boys.size(); ++i) {
49         memset(used, false, sizeof(used));
50         if (match(i)) ++res;
51     }
52     cout << n - res << '\n';
53 }
54 int main() {
55     int t, n;
56     scanf("%d", &t);
57     while (t--) {
58         scanf("%d", &n);
59         boys.clear();
60         girls.clear();
61         G.assign(n + 5, vector<int>());
62         Student student;
63         for (int i = 0; i < n; ++i) {
64             cin >> student;
65             if (student.sex == 'M')
66                 boys.emplace_back(student);
67             else
68                 girls.emplace_back(student);
69         }
70         for (int i = 0; i < boys.size(); ++i) {
71             for (int j = 0; j < girls.size(); ++j) {

```

```

69                 if (boys[i].canMatch(girls[j])) {
70                     G[i].emplace_back(j);
71                 }
72             }
73         }
74         maxMatch(n);
75     }
76     return 0;
77 }

```

6.17 Astar

```

1 #include <cstdio>
2 #include <cstring>
3 #include <vector>
4 #include <queue>
5 #include <algorithm>
6 using namespace std;
7 /*
8     A*求k短路
9      $f(x) = g(x) + h(x)$ 
10     $g(x)$  是實際cost
11     $h(x)$  是估計cost
12    在此 $h(x)$ 用所有點到終點的最短距離
13    則當用Astar找點
14    當該點 $cnt[u] == k$ 時即得到該點的第k短路
15 */
16 #define maxn 105
17 struct Edge {
18     int u, v, w;
19 };
20 struct Item_pqH {
21     int u, w;
22     bool operator <(const Item_pqH& other) const {
23         return this->w > other.w;
24     }
25 };
26 struct Item_astar {
27     int u, g, f;
28     bool operator <(const Item_astar& other) const {
29         return this->f > other.f;
30     }
31 };
32 vector<vector<Edge>>> G;
33 //反向圖，用於建 $h(u)$ 
34 vector<vector<Edge>>> invertG;
35 int h[maxn];
36 bool visited[maxn];
37 int cnt[maxn];
38 //用反向圖去求出每一點到終點的最短距離，並以此當作 $h(u)$ 
39 void dijkstra(int s, int t)
40 {
41     memset(visited, 0, sizeof(visited));
42     priority_queue<Item_pqH> pq;
43     pq.push({s, 0});
44     h[s] = 0;
45     while (!pq.empty()) {
46         Item_pqH curr = pq.top();
47         pq.pop();
48         visited[curr.u] = true;
49         for (Edge& edge: invertG[curr.u]) {
50             if (!visited[edge.v]) {
51                 if (h[edge.v] > h[curr.u] + edge.w) {
52                     h[edge.v] = h[curr.u] + edge.w;
53                     pq.push({edge.v, h[edge.v]});
54                 }
55             }
56         }
57     }
58 }
59 int Astar(int s, int t, int k) {
60     memset(cnt, 0, sizeof(cnt));
61     priority_queue<Item_astar> pq;
62     pq.push({s, 0, h[s]});
63     while (!pq.empty()) {

```



```

64     Item_astar curr = pq.top();
65     pq.pop();
66     ++cnt[curr.u];
67     //終點出現k次，此時即可得k短路
68     if (cnt[t] == k)
69         return curr.g;
70     for (Edge& edge: G[curr.u]) {
71         if (cnt[edge.v] < k) {
72             pq.push({edge.v, curr.g + edge.w,
73                     curr.g + edge.w + h[edge.v]});
74         }
75     }
76     return -1;
77 }
78 int main() {
79     int n, m;
80     while (scanf("%d %d", &n, &m) && (n != 0 && m != 0)) {
81         G.assign(n + 5, vector<Edge>());
82         invertG.assign(n + 5, vector<Edge>());
83         int s, t, k;
84         scanf("%d %d %d", &s, &t, &k);
85         int u, v, w;
86         for (int i = 0; i < m; ++i) {
87             scanf("%d %d %d", &u, &v, &w);
88             G[u].emplace_back(Edge{u, v, w});
89             invertG[v].emplace_back(Edge{v, u, w});
90         }
91         memset(h, 0x3f, sizeof(h));
92         dijkstra(t, s);
93         printf("%d\n", Astar(s, t, k));
94     }
95     return 0;
96 }

```

6.18 JosephusProblem

```

1 #include <cstdio>
2 //JosephusProblem，只是規定要先砍1號
3 //所以當作有n - 1個人，目標的13順移成12
4 //再者從0開始比較好算，所以目標12順移成11
5 int getWinner(int n, int k)
6 {
7     int winner = 0;
8     for (int i = 1; i <= n; ++i)
9         winner = (winner + k) % i;
10    return winner;
11 }
12 int main()
13 {
14     int n;
15     while (scanf("%d", &n) != EOF && n)
16     {
17         --n;
18         for (int k = 1; k <= n; ++k)
19         {
20             if (getWinner(n, k) == 11)
21             {
22                 printf("%d\n", k);
23                 break;
24             }
25         }
26     }
27     return 0;
28 }

```

6.19 KM

```

1 #include <cstdio>
2 #include <cstring>
3 #include <algorithm>
4 using namespace std;

```

```

5 /*題意：
6 給定一個w矩陣，現在分成row、column兩個1維陣列
7 W[i][j]=k即代表column[i] + row[j]要>=k
8 求row[] 與 column[]的所有值在滿足矩陣w的要求之下
9 row[] + column[]所有元素相加起來要最小
10 利用KM求二分圖最大權匹配
11 Lx -> vertex labeling of x
12 Ly -> vertex labeling of y
13 一開始Lx[i] = max(W[i][j]), Ly = 0
14 Lx[i] + Ly[j] >= W[i][j]
15 要最小化全部的(Lx[i] + Ly[j])加總
16 不斷的調整vertex
17 labeling去找到一條交錯邊皆滿足Lx[i] + Ly[j]
18 == W[i][j]的增廣路
19 最後會得到正確的二分圖完美匹配中的最大權分配(先滿足最多匹配
20 意義是將最大化所有匹配邊權重和的問題改成最小化所有點的權重和)
21 #define maxn 505
22 int W[maxn][maxn];
23 int Lx[maxn], Ly[maxn];
24 bool S[maxn], T[maxn];
25 //L[i] = j -> S_i配給T_j, -1 for 還沒匹配
26 int L[maxn];
27 int n;
28 bool match(int i) {
29     S[i] = true;
30     for (int j = 0; j < n; ++j) {
31         // KM重點
32         // Lx + Ly >= selected_edge(x, y)
33         // 要想辦法降低Lx + Ly
34         // 所以選Lx + Ly == selected_edge(x, y)
35         if (Lx[i] + Ly[j] == W[i][j] && !T[j]) {
36             T[j] = true;
37             if ((L[j] == -1) || match(L[j])) {
38                 L[j] = i;
39                 return true;
40             }
41         }
42     }
43     return false;
44 }
45 // 修改二分圖上的交錯路徑上點的權重
46 // 此舉是在通過調整vertex
47 labeling看看能不能產生出新的增廣路(KM的增廣路要求Lx[i]
48 + Ly[j] == W[i][j])
49 // 在這裡優先從最小的diff調看，才能保證最大權重匹配
50 void update()
51 {
52     int diff = 0x3f3f3f3f;
53     for (int i = 0; i < n; ++i) {
54         if (S[i]) {
55             for (int j = 0; j < n; ++j) {
56                 if (!T[j])
57                     diff = min(diff, Lx[i] + Ly[j] - W[i][j]);
58             }
59         }
60     }
61     for (int i = 0; i < n; ++i) {
62         if (S[i]) Lx[i] -= diff;
63         if (T[i]) Ly[i] += diff;
64     }
65 }
66 void KM()
67 {
68     for (int i = 0; i < n; ++i) {
69         L[i] = -1;
70         Lx[i] = Ly[i] = 0;
71         for (int j = 0; j < n; ++j)
72             Lx[i] = max(Lx[i], W[i][j]);
73     }
74     while(1) {
75         memset(S, false, sizeof(S));
76         memset(T, false, sizeof(T));
77         if (match(i))

```

```

75         break;
76     else
77         update(); //去調整 vertex
              labeling以增加增廣路徑
78     }
79 }
80 }
81 int main()
82 {
83     while (scanf("%d", &n) != EOF) {
84         for (int i = 0; i < n; ++i)
85             for (int j = 0; j < n; ++j)
86                 scanf("%d", &W[i][j]);
87         KM();
88         int res = 0;
89         for (int i = 0; i < n; ++i) {
90             if (i != 0)
91                 printf(" %d", Lx[i]);
92             else
93                 printf("%d", Lx[i]);
94             res += Lx[i];
95         }
96         puts("");
97         for (int i = 0; i < n; ++i) {
98             if (i != 0)
99                 printf(" %d", Ly[i]);
100             else
101                 printf("%d", Ly[i]);
102             res += Ly[i];
103         }
104         puts("");
105         printf("%d\n", res);
106     }
107     return 0;
108 }

```

6.20 LCA 倍增法

```

1 #include <cstdio>
2 #include <vector>
3 #include <cstdlib>
4 using namespace std;
5 //倍增法預處理  $O(n \log n)$ ，查詢  $O(\log n)$ ，利用 lca 找樹上任兩點
6 #define maxn 100005
7 struct Edge
8 {
9     int u, v, w;
10 };
11 vector<vector<Edge>> G; // tree
12 int fa[maxn][31]; //fa[u][i] -> u 的第  $2^i$  個祖先
13 long long dis[maxn][31];
14 int dep[maxn]; //深度
15 void dfs(int u, int p) //預處理 fa
16 {
17     fa[u][0] = p; //因為 u 的第  $2^0 = 1$  的祖先就是 p
18     dep[u] = dep[p] + 1;
19     //第  $2^i$  的祖先是 (第  $2^{i-1}$  個祖先) 的第  $2^{i-1}$  的祖先
20     //ex: 第 8 個祖先是 (第 4 個祖先) 的第 4 個祖先
21     for (int i = 1; i < 31; ++i)
22     {
23         fa[u][i] = fa[fa[u][i-1]][i-1];
24         dis[u][i] = dis[fa[u][i-1]][i-1] +
25             dis[u][i-1];
26     }
27     //遍歷子節點
28     for (Edge& edge: G[u])
29     {
30         if (edge.v == p)
31             continue;
32         dis[edge.v][0] = edge.w;
33         dfs(edge.v, u);
34     }
35 }

```

```

35 long long lca(int x, int y)
    //此函數是找 lca 同時計算 x、y 的距離 -> dis(x, lca)
    + dis(lca, y)
36 {
37     //讓 y 比 x 深
38     if (dep[x] > dep[y])
39         swap(x, y);
40     int deltaDep = dep[y] - dep[x];
41     long long res = 0;
42
43     //讓 y 與 x 在同一個深度
44     for (int i = 0; deltaDep != 0; ++i, deltaDep >>= 1)
45         if (deltaDep & 1)
46             res += dis[y][i], y = fa[y][i];
47
48     if (y == x) //x = y -> x、y 彼此是彼此的祖先
49         return res;
50
51     //往上找，一起跳，但 x、y 不能重疊
52     for (int i = 30; i >= 0 && y != x; --i)
53     {
54         if (fa[x][i] != fa[y][i])
55         {
56             res += dis[x][i] + dis[y][i];
57             x = fa[x][i];
58             y = fa[y][i];
59         }
60     }
61     //最後發現不能跳了，此時 x 的第  $2^0 = 1$  個祖先 (或說 y 的第  $2^0 = 1$  的祖先) 即為 x、y 的 lca
62     res += dis[x][0] + dis[y][0];
63     return res;
64 }
65 int main()
66 {
67     int n, q;
68     while (~scanf("%d", &n) && n)
69     {
70         int v, w;
71         G.assign(n + 5, vector<Edge>());
72         for (int i = 1; i <= n - 1; ++i)
73         {
74             scanf("%d %d", &v, &w);
75             G[i + 1].push_back({i + 1, v + 1, w});
76             G[v + 1].push_back({v + 1, i + 1, w});
77         }
78         dfs(1, 0);
79         scanf("%d", &q);
80         int u;
81         while (q--)
82         {
83             scanf("%d %d", &u, &v);
84             printf("%lld%c", lca(u + 1, v + 1), (q) ?
85                 ' ' : '\n');
86         }
87     }
88     return 0;
89 }

```

6.21 LCA 樹壓平 RMQ

```

1 #include <cstdio>
2 #include <vector>
3 #include <cstdlib>
4 using namespace std;
5 //樹壓平求 LCA RMQ (sparse table)
     $O(n \log n)$  建立， $O(1)$  查詢，求任意兩點距離，
6 //如果用笛卡兒樹可以壓到  $O(n)$  建立， $O(1)$  查詢
7 //理論上可以過，但遇到直鏈的 case dfs 深度會 stack
    overflow
8 #define maxn 100005
9 struct Edge
10 {

```

```

11     int u, v, w;
12 };
13 int dep[maxn];
14 int pos[maxn];
15 long long dis[maxn];
16 int st[maxn * 2][32]; //sparse table
17 int realLCA[maxn * 2][32];
    //最小深度對應的節點，及真正的LCA
18 int Log[maxn]; //取代std::log2
19 int tp; // timestamp
20 vector<vector<Edge>> G; // tree
21 void calLog()
22 {
23     Log[1] = 0;
24     Log[2] = 1;
25     for (int i = 3; i < maxn; ++i)
26     {
27         Log[i] = Log[i / 2] + 1;
28     }
29 }
30 void buildST()
31 {
32     for (int j = 0; Log[tp]; ++j)
33     {
34         for (int i = 0; i + (1 << j) - 1 < tp; ++i)
35         {
36             if (st[i - 1][j] < st[i - 1][j + (1 << i - 1)])
37             {
38                 st[i][j] = st[i - 1][j];
39                 realLCA[i][j] = realLCA[i - 1][j];
40             }
41             else
42             {
43                 st[i][j] = st[i - 1][j + (1 << i - 1)];
44                 realLCA[i][j] = realLCA[i - 1][j + (1 << i - 1)];
45             }
46         }
47     }
48 } // O(nlogn)
49 int query(int l, int r) // [l, r] min
    depth即為lca的深度
50 {
51     int k = Log[r - l + 1];
52     if (st[l][k] < st[r - (1 << k) + 1][k])
53         return realLCA[l][k];
54     else
55         return realLCA[r - (1 << k) + 1][k];
56 }
57 void dfs(int u, int p) //euler tour
58 {
59     pos[u] = tp;
60     st[tp][0] = dep[u];
61     realLCA[tp][0] = dep[u];
62     ++tp;
63     for (int i = 0; i < G[u].size(); ++i)
64     {
65         Edge& edge = G[u][i];
66         if (edge.v == p)
67             continue;
68         dep[edge.v] = dep[u] + 1;
69         dis[edge.v] = dis[u] + edge.w;
70         dfs(edge.v, u);
71         st[tp++][0] = dep[u];
72     }
73 }
74 long long getDis(int u, int v)
75 {
76     if (pos[u] > pos[v])
77         swap(u, v);
78     int lca = query(pos[u], pos[v]);
79     return dis[u] + dis[v] - 2 * dis[query(pos[u], pos[v])];
80 }
81 int main()
82 {
83     int n, q;

```

```

84     calLog();
85     while (~scanf("%d", &n) && n)
86     {
87         int v, w;
88         G.assign(n + 5, vector<Edge>());
89         tp = 0;
90         for (int i = 1; i <= n - 1; ++i)
91         {
92             scanf("%d %d", &v, &w);
93             G[i].push_back({i, v, w});
94             G[v].push_back({v, i, w});
95         }
96
97         dfs(0, -1);
98         buildST();
99
100        scanf("%d", &q);
101        int u;
102        while (q--)
103        {
104            scanf("%d %d", &u, &v);
105            printf("%lld%c", getDis(u, v), (q) ? ' ' : '\n');
106        }
107    }
108    return 0;
109 }

```

6.22 MCMF

```

1 #include <cstdio>
2 #include <vector>
3 #include <cstring>
4 #include <queue>
5 using namespace std;
6 #define maxn 225
7 #define INF 0x3f3f3f3f
8 struct Edge
9 {
10     int u, v, cap, flow, cost;
11 };
12 //node size, edge size, source, target
13 int n, m, s, t;
14 vector<vector<int>> G;
15 vector<Edge> edges;
16 //SPFA用
17 bool inqueue[maxn];
18 //SPFA用的dis[]
19 long long dis[maxn];
20 //maxFlow一路扣回去時要知道parent
21 //<注> 在這題因為G[]中存的是edgeIndex in edges[]
22 //
    所以parent存的也是對應edges[]中的edgeIndex(主要是方便)
23 int parent[maxn];
24 //maxFlow時需要紀錄到node u時的bottleneck
25 //同時也代表著u該次流出去的量
26 long long outFlow[maxn];
27 void addEdge(int u, int v, int cap, int cost)
28 {
29     edges.emplace_back(Edge{u, v, cap, 0, cost});
30     edges.emplace_back(Edge{v, u, 0, 0, -cost});
31     m = edges.size();
32     G[u].emplace_back(m - 2);
33     G[v].emplace_back(m - 1);
34 }
35 //一邊求最短路的同时一邊MaxFlow
36 bool SPFA(long long& maxFlow, long long& minCost)
37 {
38     //memset(outFlow, 0x3f, sizeof(outFlow));
39     memset(dis, 0x3f, sizeof(dis));
40     memset(inqueue, false, sizeof(inqueue));
41     queue<int> q;
42     q.push(s);
43     dis[s] = 0;
44     inqueue[s] = true;

```

```

45 outFlow[s] = INF;
46 while (!q.empty())
47 {
48     int u = q.front();
49     q.pop();
50     inqueue[u] = false;
51     for (const int edgeIndex: G[u])
52     {
53         const Edge& edge = edges[edgeIndex];
54         if ((edge.cap > edge.flow) &&
55             (dis[edge.v] > dis[u] + edge.cost))
56         {
57             dis[edge.v] = dis[u] + edge.cost;
58             parent[edge.v] = edgeIndex;
59             outFlow[edge.v] = min(outFlow[u],
60                 (long long)(edge.cap -
61                     edge.flow));
62             if (!inqueue[edge.v])
63             {
64                 q.push(edge.v);
65                 inqueue[edge.v] = true;
66             }
67         }
68     }
69     //如果dis[t] > 0代表根本不賺還倒賠
70     if (dis[t] > 0)
71         return false;
72     maxFlow += outFlow[t];
73     minCost += dis[t] * outFlow[t];
74     //一路更新回去這次最短路流完後要維護的MaxFlow演算法相同
75     int curr = t;
76     while (curr != s)
77     {
78         edges[parent[curr]].flow += outFlow[t];
79         edges[parent[curr] ^ 1].flow -= outFlow[t];
80         curr = edges[parent[curr]].u;
81     }
82     return true;
83 }
84 long long MCMF()
85 {
86     long long maxFlow = 0;
87     long long minCost = 0;
88     while (SPFA(maxFlow, minCost))
89         ;
90     return minCost;
91 }
92 int main()
93 {
94     int T;
95     scanf("%d", &T);
96     for (int Case = 1; Case <= T; ++Case)
97     {
98         //總共幾個月，囤貨成本
99         int M, I;
100         scanf("%d %d", &M, &I);
101         //node size
102         n = M + M + 2;
103         G.assign(n + 5, vector<int>());
104         edges.clear();
105         s = 0;
106         t = M + M + 1;
107         for (int i = 1; i <= M; ++i)
108         {
109             int produceCost, produceMax, sellPrice,
110                 sellMax, inventoryMonth;
111             scanf("%d %d %d %d %d", &produceCost,
112                 &produceMax, &sellPrice, &sellMax,
113                 &inventoryMonth);
114             addEdge(s, i, produceMax, produceCost);
115             addEdge(M + i, t, sellMax, -sellPrice);
116             for (int j = 0; j <= inventoryMonth; ++j)
117             {
118                 if (i + j <= M)
119                     addEdge(i, M + i + j, INF, I * j);
120             }
121         }
122     }
123 }

```

```

116     }
117     printf("Case %d: %lld\n", Case, -MCMF());
118 }
119 return 0;
120 }

```

6.23 莫隊

```

1 #include <cstdio>
2 #include <cmath>
3 #include <algorithm>
4 using namespace std;
5 /*
6     利用prefix前綴XOR和
7     如果要求[x, y]的XOR和只要回答prefix[y] ^ prefix[x
8     - 1]即可在O(1)回答
9     同時維護cnt[i]代表[x, y]XOR和 == i的個數
10    如此我們知道[l, r]可以快速知道[l - 1, r], [l + 1,
11    r], [l, r - 1], [l, r + 1]的答案
12    就符合Mo's algorithm的思維O(N * sqrt(n))
13    每次轉移為O(1)
14    具體轉移方法在下面
15 */
16 #define maxn 100005
17 //在此prefix[i]是[1, i]的XOR和
18 int prefix[maxn];
19 //log_2(1000000) =
20 19.931568569324174087221916576937...
21 //所以開到1 << 20
22 //cnt[i]代表的是有符合nums[x, y] such that nums[x] ^
23    nums[x + 1] ^ .. ^ nums[y] == i
24 //的個數
25 long long cnt[1 << 20];
26 //塊大小 -> sqrt(n)
27 int sqrtQ;
28 struct Query
29 {
30     int l, r, id;
31     bool operator < (const Query& other) const
32     {
33         if (this->l / sqrtQ != other.l / sqrtQ)
34             return this->l < other.l;
35         //奇偶排序(優化)
36         if (this->l / sqrtQ & 1)
37             return this->r < other.r;
38         return this->r > other.r;
39     }
40 };
41 Query queries[maxn];
42 long long ans[maxn];
43 long long res = 0;
44 int k;
45 void add(int x)
46 {
47     res += cnt[k ^ prefix[x]];
48     ++cnt[prefix[x]];
49 }
50 void sub(int x)
51 {
52     --cnt[prefix[x]];
53     res -= cnt[k ^ prefix[x]];
54 }
55 int main() {
56     int n, m;
57     scanf("%d %d", &n, &m, &k);
58     sqrtQ = sqrt(n);
59     for (int i = 1; i <= n; ++i) {
60         scanf("%d", &prefix[i]);
61         prefix[i] ^= prefix[i - 1];
62     }
63     for (int i = 1; i <= m; ++i) {
64         scanf("%d %d", &queries[i].l, &queries[i].r);
65         //減1是因為prefix[i]是[1,
66         i]的前綴XOR和，所以題目問[l,

```

```

    r]我們要回答[l - 1, r]的答案
62     --querys[i].l;
63     querys[i].id = i;
64 }
65 sort(querys + 1, querys + m + 1);
66 int l = 1, r = 0;
67 for (int i = 1; i <= m; ++i) {
68     while (l < querys[i].l) {
69         sub(1);
70         ++l;
71     }
72     while (l > querys[i].l) {
73         --l;
74         add(1);
75     }
76     while (r < querys[i].r) {
77         ++r;
78         add(r);
79     }
80     while (r > querys[i].r) {
81         sub(r);
82         --r;
83     }
84     ans[querys[i].id] = res;
85 }
86 for (int i = 1; i <= m; ++i){
87     printf("%lld\n", ans[i]);
88 }
89 return 0;
90 }

```

6.24 Dancing Links X

```

1 struct DLX {
2     int seq, resSize;
3     int col[maxn], row[maxn];
4     int U[maxn], D[maxn], R[maxn], L[maxn];
5     int rowHead[maxn], colSize[maxn];
6     int result[maxn];
7
8     DLX(int r, int c) {
9         for(int i=0; i<=c; i++) {
10             L[i] = i-1, R[i] = i+1;
11             U[i] = D[i] = i;
12         }
13         L[R[seq=c]=0]=c;
14         resSize = -1;
15         memset(rowHead, 0, sizeof(rowHead));
16         memset(colSize, 0, sizeof(colSize));
17     }
18
19     void insert(int r, int c) {
20         row[++seq]=r, col[seq]=c, ++colSize[c];
21         U[seq]=c, D[seq]=D[c], U[D[c]]=seq, D[c]=seq;
22         if(rowHead[r]) {
23             L[seq]=rowHead[r], R[seq]=R[rowHead[r]];
24             L[R[rowHead[r]]]=seq, R[rowHead[r]]=seq;
25         } else {
26             rowHead[r] = L[seq] = R[seq] = seq;
27         }
28     }
29
30     void remove(int c) {
31         L[R[c]] = L[c], R[L[c]] = R[c];
32         for(int i=D[c]; i!=c; i=D[i]) {
33             for(int j=R[i]; j!=i; j=R[j]) {
34                 U[D[j]] = U[j];
35                 D[U[j]] = D[j];
36                 --colSize[col[j]];
37             }
38         }
39     }
40
41     void recover(int c) {
42         for(int i=U[c]; i!=c; i=U[i]) {
43             for(int j=L[i]; j!=i; j=L[j]) {

```

```

44                 U[D[j]] = D[U[j]] = j;
45                 ++colSize[col[j]];
46             }
47         }
48         L[R[c]] = R[L[c]] = c;
49     }
50
51     bool dfs(int idx=0) { // 判斷其中一解版
52         if(R[0] == 0) {
53             resSize = idx;
54             return true;
55         }
56
57         int c = R[0];
58         for(int i=R[0]; i; i=R[i]) {
59             if(colSize[i] < colSize[c]) c = i;
60         }
61         remove(c);
62         for(int i=D[c]; i!=c; i=D[i]) {
63             result[idx] = row[i];
64             for(int j=R[i]; j!=i; j=R[j])
65                 remove(col[j]);
66             if(dfs(idx+1)) return true;
67             for(int j=L[i]; j!=i; j=L[j])
68                 recover(col[j]);
69         }
70         recover(c);
71         return false;
72     }
73
74     void dfs(int idx=0) { // 判斷最小 dfs depth 版
75         if(R[0] == 0) {
76             resSize = min(resSize, idx); // 注意init值
77             return;
78         }
79
80         int c = R[0];
81         for(int i=R[0]; i; i=R[i]) {
82             if(colSize[i] < colSize[c]) c = i;
83         }
84         remove(c);
85         for(int i=D[c]; i!=c; i=D[i]) {
86             for(int j=R[i]; j!=i; j=R[j])
87                 remove(col[j]);
88             dfs(idx+1);
89             for(int j=L[i]; j!=i; j=L[j])
90                 recover(col[j]);
91         }
92         recover(c);
93     }
94 };

```

7 DataStructure

7.1 ChthollyTree

```

1 //重點：要求輸入資料隨機，否則可能被卡時間
2 struct Node {
3     long long l, r;
4     mutable long long val;
5     Node(long long l, long long r, long long val)
6         : l(l), r(r), val(val){}
7     bool operator < (const Node& other) const{
8         return this->l < other.l;
9     }
10 };
11 set<Node> chthollyTree;
12 //將[l, r] 拆成 [l, pos - 1], [pos, r]
13 set<Node>::iterator split(long long pos) {
14     //找第一個左端點大於等於pos的區間
15     set<Node>::iterator it =
16         chthollyTree.lower_bound(Node(pos, 0, 0));
17     //運氣很好直接找到左端點是pos的區間
18     if (it != chthollyTree.end() && it->l == pos)

```

```

18     return it;
19     //到這邊代表找到的是第一個左端點大於pos的區間
20     //it -
    //即可找到左端點等於pos的區間(不會是別的，因為沒有重疊)
21     --it;
22     long long l = it->l, r = it->r;
23     long long val = it->val;
24     chthollyTree.erase(it);
25     chthollyTree.insert(Node(l, pos - 1, val));
26     //回傳左端點是pos的區間iterator
27     return chthollyTree.insert(Node(pos, r, val)).first;
28 }
29 //區間賦值
30 void assign(long long l, long long r, long long val) {
31     //<注意>
    //end與begin的順序不能調換，因為end的split可能會改變
    //因為end可以在原本begin的區間中
32     set<Node>::iterator end = split(r + 1), begin =
        split(l);
33     //begin到end全部刪掉
34     chthollyTree.erase(begin, end);
35     //填回去[l, r]的區間
36     chthollyTree.insert(Node(l, r, val));
37 }
38 //區間加值(直接一個個區間去加)
39 void add(long long l, long long r, long long val) {
40     set<Node>::iterator end = split(r + 1);
41     set<Node>::iterator begin = split(l);
42     for (set<Node>::iterator it = begin; it != end;
        ++it)
43         it->val += val;
44 }
45 //查詢區間第k小 -> 直接把每個區間丟去vector排序
46 long long getKthSmallest(long long l, long long r,
    long long k) {
47     set<Node>::iterator end = split(r + 1);
48     set<Node>::iterator begin = split(l);
49     //pair -> first: val, second: 區間長度
50     vector<pair<long long, long long>> vec;
51     for (set<Node>::iterator it = begin; it != end;
        ++it) {
52         vec.push_back({it->val, it->r - it->l + 1});
53     }
54     sort(vec.begin(), vec.end());
55     for (const pair<long long, long long>& p: vec) {
56         k -= p.second;
57         if (k <= 0)
58             return p.first;
59     }
60     //不應該跑到這
61     return -1;
62 }
63 //快速冪
64 long long qpow(long long x, long long n, long long
    mod) {
65     long long res = 1;
66     x %= mod;
67     while (n)
68     {
69         if (n & 1)
70             res = res * x % mod;
71         n >>= 1;
72         x = x * x % mod;
73     }
74     return res;
75 }
76 //區間n次方和
77 long long sumOfPow(long long l, long long r, long
    long n, long long mod) {
78     long long total = 0;
79     set<Node>::iterator end = split(r + 1);
80     set<Node>::iterator begin = split(l);
81     for (set<Node>::iterator it = begin; it != end;
        ++it)
82         {
83

```

```

84         total = (total + qpow(it->val, n, mod) *
            (it->r - it->l + 1)) % mod;
85     }
86     return total;
87 }

```

7.2 線段樹 1D

```

1 #define MAXN 1000
2 int data[MAXN]; //原數據
3 int st[4 * MAXN]; //線段樹
4 int tag[4 * MAXN]; //懶標
5
6 inline int pull(int l, int r) {
7     // 隨題目改變sum、max、min
8     // l、r是左右樹的index
9     return st[l] + st[r];
10 }
11
12 void build(int l, int r, int i) {
13     // 在[l, r]區間建樹，目前根的index為i
14     if (l == r) {
15         st[i] = data[l];
16         return;
17     }
18     int mid = l + ((r - l) >> 1);
19     build(l, mid, i * 2);
20     build(mid + 1, r, i * 2 + 1);
21     st[i] = pull(i * 2, i * 2 + 1);
22 }
23
24 int query(int ql, int qr, int l, int r, int i) {
25     // [ql, qr]是查詢區間,[l, r]是當前節點包含的區間
26     if (ql <= l && r <= qr)
27         return st[i];
28     int mid = l + ((r - l) >> 1);
29     if (tag[i]) {
30         //如果當前懶標有值則更新左右節點
31         st[i * 2] += tag[i] * (mid - l + 1);
32         st[i * 2 + 1] += tag[i] * (r - mid);
33         tag[i * 2] += tag[i]; //下傳懶標至左節點
34         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
35         tag[i] = 0;
36     }
37     int sum = 0;
38     if (ql <= mid)
39         sum += query(ql, qr, l, mid, i * 2);
40     if (qr > mid)
41         sum += query(ql, qr, mid + 1, r, i * 2 + 1);
42     return sum;
43 }
44
45 void update(int ql, int qr, int l, int r, int i, int c) {
46     // [ql, qr]是查詢區間,[l, r]是當前節點包含的區間
47     // c是變化量
48     if (ql <= l && r <= qr) {
49         st[i] += (r - l + 1) * c;
50         //求和,此需乘上區間長度
51         tag[i] += c;
52         return;
53     }
54     int mid = l + ((r - l) >> 1);
55     if (tag[i] && l != r) {
56         //如果當前懶標有值則更新左右節點
57         st[i * 2] += tag[i] * (mid - l + 1);
58         st[i * 2 + 1] += tag[i] * (r - mid);
59         tag[i * 2] += tag[i]; //下傳懶標至左節點
60         tag[i * 2 + 1] += tag[i]; //下傳懶標至右節點
61         tag[i] = 0;
62     }
63     if (ql <= mid) update(ql, qr, l, mid, i * 2, c);
64     if (qr > mid) update(ql, qr, mid + 1, r, i * 2 + 1, c);
65     st[i] = pull(i * 2, i * 2 + 1);
66 }

```



```

66 //如果是直接改值而不是加值，query與update中的tag與st的
67 //改值從+=改成=

```

7.3 線段樹 2D

```

1 #include <cstdio>
2 #include <algorithm>
3 using namespace std;
4 //純2D segment tree 區間查詢單點修改最大最小值
5 #define maxn 2005 //500 * 4 + 5
6 int maxST[maxn][maxn], minST[maxn][maxn];
7 int N;
8 void modifyY(int index, int l, int r, int val, int
    yPos, int xIndex, bool xIsLeaf)
9 {
10     if (l == r)
11     {
12         if (xIsLeaf)
13         {
14             maxST[xIndex][index] =
15             minST[xIndex][index] = val;
16             return;
17         }
18         maxST[xIndex][index] = max(maxST[xIndex *
19             2][index], maxST[xIndex * 2 + 1][index]);
20         minST[xIndex][index] = min(minST[xIndex *
21             2][index], minST[xIndex * 2 + 1][index]);
22     }
23     else
24     {
25         int mid = (l + r) / 2;
26         if (yPos <= mid)
27             modifyY(index * 2, l, mid, val, yPos,
28                 xIndex, xIsLeaf);
29         else
30             modifyY(index * 2 + 1, mid + 1, r, val,
31                 yPos, xIndex, xIsLeaf);
32         maxST[xIndex][index] =
33             max(maxST[xIndex][index * 2],
34                 maxST[xIndex][index * 2 + 1]);
35         minST[xIndex][index] =
36             min(minST[xIndex][index * 2],
37                 minST[xIndex][index * 2 + 1]);
38     }
39 }
40 void modifyX(int index, int l, int r, int val, int
    xPos, int yPos)
41 {
42     if (l == r)
43     {
44         modifyY(1, 1, N, val, yPos, index, true);
45     }
46     else
47     {
48         int mid = (l + r) / 2;
49         if (xPos <= mid)
50             modifyX(index * 2, l, mid, val, xPos,
51                 yPos);
52         else
53             modifyX(index * 2 + 1, mid + 1, r, val,
54                 xPos, yPos);
55         modifyY(1, 1, N, val, yPos, index, false);
56     }
57 }
58 void queryY(int index, int l, int r, int yql, int
    yqr, int xIndex, int& vmax, int& vmin)
59 {
60     if (yql <= l && r <= yqr)
61     {
62         vmax = max(vmax, maxST[xIndex][index]);
63         vmin = min(vmin, minST[xIndex][index]);
64     }
65     else
66     {
67         int mid = (l + r) / 2;
68         if (yql <= mid)
69             queryY(index * 2, l, mid, yql, yqr,
70                 xIndex, vmax, vmin);
71         if (mid < yqr)
72             queryY(index * 2 + 1, mid + 1, r, yql,
73                 yqr, xIndex, vmax, vmin);
74     }
75 }
76 int main()
77 {
78     while (scanf("%d", &N) != EOF)
79     {
80         int val;
81         for (int i = 1; i <= N; ++i)
82         {
83             for (int j = 1; j <= N; ++j)
84             {
85                 scanf("%d", &val);
86                 modifyX(1, 1, N, val, i, j);
87             }
88         }
89         int q;
90         int vmax, vmin;
91         int xql, xqr, yql, yqr;
92         char op;
93         scanf("%d", &q);
94         while (q--)
95         {
96             getchar(); //for \n
97             scanf("%c", &op);
98             if (op == 'q')
99             {
100                 scanf("%d %d %d %d", &xql, &yql,
101                     &xqr, &yqr);
102                 vmax = -0x3f3f3f3f;
103                 vmin = 0x3f3f3f3f;
104                 queryX(1, 1, N, xql, xqr, yql, yqr,
105                     vmax, vmin);
106                 printf("%d %d\n", vmax, vmin);
107             }
108             else
109             {
110                 scanf("%d %d %d", &xql, &yql, &val);
111                 modifyX(1, 1, N, val, xql, yql);
112             }
113         }
114     }
115     return 0;
116 }

```

```

57     int mid = (l + r) / 2;
58     if (yql <= mid)
59         queryY(index * 2, l, mid, yql, yqr,
60             xIndex, vmax, vmin);
61     if (mid < yqr)
62         queryY(index * 2 + 1, mid + 1, r, yql,
63             yqr, xIndex, vmax, vmin);
64 }
65 void queryX(int index, int l, int r, int xql, int
    xqr, int yql, int yqr, int& vmax, int& vmin)
66 {
67     if (xql <= l && r <= xqr)
68     {
69         queryY(1, 1, N, yql, yqr, index, vmax,
70             vmin);
71     }
72     else
73     {
74         int mid = (l + r) / 2;
75         if (xql <= mid)
76             queryX(index * 2, l, mid, xql, xqr,
77                 yql, yqr, vmax, vmin);
78         if (mid < xqr)
79             queryX(index * 2 + 1, mid + 1, r, xql,
80                 xqr, yql, yqr, vmax, vmin);
81     }
82 }
83 int main()
84 {
85     while (scanf("%d", &N) != EOF)
86     {
87         int val;
88         for (int i = 1; i <= N; ++i)
89         {
90             for (int j = 1; j <= N; ++j)
91             {
92                 scanf("%d", &val);
93                 modifyX(1, 1, N, val, i, j);
94             }
95         }
96         int q;
97         int vmax, vmin;
98         int xql, xqr, yql, yqr;
99         char op;
100         scanf("%d", &q);
101         while (q--)
102         {
103             getchar(); //for \n
104             scanf("%c", &op);
105             if (op == 'q')
106             {
107                 scanf("%d %d %d %d", &xql, &yql,
108                     &xqr, &yqr);
109                 vmax = -0x3f3f3f3f;
110                 vmin = 0x3f3f3f3f;
111                 queryX(1, 1, N, xql, xqr, yql, yqr,
112                     vmax, vmin);
113                 printf("%d %d\n", vmax, vmin);
114             }
115             else
116             {
117                 scanf("%d %d %d", &xql, &yql, &val);
118                 modifyX(1, 1, N, val, xql, yql);
119             }
120         }
121     }
122     return 0;
123 }

```

7.4 Trie

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int maxn = 300000 + 10;
5 const int mod = 20071027;

```

```

6
7 int dp[maxn];
8 int mp[4000*100 + 10][26];
9 char str[maxn];
10
11 struct Trie {
12     int seq;
13     int val[maxn];
14
15     Trie() {
16         seq = 0;
17         memset(val, 0, sizeof(val));
18         memset(mp, 0, sizeof(mp));
19     }
20
21     void insert(char* s, int len) {
22         int r = 0;
23         for(int i=0; i<len; i++) {
24             int c = s[i] - 'a';
25             if(!mp[r][c]) mp[r][c] = ++seq;
26             r = mp[r][c];
27         }
28         val[r] = len;
29         return;
30     }
31
32     int find(int idx, int len) {
33         int result = 0;
34         for(int r=0; idx<len; idx++) {
35             int c = str[idx] - 'a';
36             if(!(r = mp[r][c])) return result;
37             if(val[r])
38                 result = (result + dp[idx + 1]) % mod;
39         }
40         return result;
41     }
42 };
43
44 int main() {
45     int n, tc = 1;
46
47     while(~scanf("%s%d", str, &n)) {
48         Trie tr;
49         int len = strlen(str);
50         char word[100+10];
51
52         memset(dp, 0, sizeof(dp));
53         dp[len] = 1;
54
55         while(n--) {
56             scanf("%s", word);
57             tr.insert(word, strlen(word));
58         }
59
60         for(int i=len-1; i>=0; i--)
61             dp[i] = tr.find(i, len);
62         printf("Case %d: %d\n", tc++, dp[0]);
63     }
64     return 0;
65 }
66
67 /*****
68 ****Input****
69 * abcd
70 * 4
71 * a b cd ab
72 ****Output***
73 * Case 1: 2
74 ****
75 */

```

7.5 權值線段樹

```

1 #include <iostream>
2 #include <cstring>

```

```

3 #include <algorithm>
4 using namespace std;
5 //權值線段樹 + 離散化 解決區間第k小問題
6 //其他網路上的解法: 2個heap, Treap, AVL tree
7 #define maxn 30005
8 int nums[maxn];
9 int getArr[maxn];
10 int id[maxn];
11 int st[maxn << 2];
12 void update(int index, int l, int r, int qx)
13 {
14     if (l == r)
15     {
16         ++st[index];
17         return;
18     }
19
20     int mid = (l + r) / 2;
21     if (qx <= mid)
22         update(index * 2, l, mid, qx);
23     else
24         update(index * 2 + 1, mid + 1, r, qx);
25     st[index] = st[index * 2] + st[index * 2 + 1];
26 }
27 //找區間第k個小的
28 int query(int index, int l, int r, int k)
29 {
30     if (l == r)
31         return id[l];
32     int mid = (l + r) / 2;
33     //k比左子樹小
34     if (k <= st[index * 2])
35         return query(index * 2, l, mid, k);
36     else
37         return query(index * 2 + 1, mid + 1, r, k -
38             st[index * 2]);
39 }
40
41 int main()
42 {
43     int t;
44     cin >> t;
45     bool first = true;
46     while (t--)
47     {
48         if (first)
49             first = false;
50         else
51             puts("");
52         memset(st, 0, sizeof(st));
53         int m, n;
54         cin >> m >> n;
55         for (int i = 1; i <= m; ++i)
56         {
57             cin >> nums[i];
58             id[i] = nums[i];
59         }
60         for (int i = 0; i < n; ++i)
61             cin >> getArr[i];
62
63         //離散化
64         //防止m == 0
65         if (m)
66             sort(id + 1, id + m + 1);
67         int stSize = unique(id + 1, id + m + 1) - (id
68             + 1);
69         for (int i = 1; i <= m; ++i)
70         {
71             nums[i] = lower_bound(id + 1, id + stSize
72                 + 1, nums[i]) - id;
73         }
74         int addCount = 0;
75         int getCount = 0;
76         int k = 1;
77         while (getCount < n)
78         {
79             if (getArr[getCount] == addCount)
80             {

```

```

76         printf("%d\n", query(1, 1, stSize,
77             k));
78         ++k;
79         ++getCount;
80     }
81     else
82     {
83         update(1, 1, stSize, nums[addCount +
84             1]);
85         ++addCount;
86     }
87     return 0;
88 }

```

8 geometry

8.1 intersection

```

1 using LL = long long;
2
3 struct Point2D {
4     LL x, y;
5 };
6
7 struct Line2D {
8     Point2D s, e;
9     LL a, b, c; // L: ax + by = c
10    Line2D(Point2D s, Point2D e): s(s), e(e) {
11        a = e.y - s.y;
12        b = s.x - e.x;
13        c = a * s.x + b * s.y;
14    }
15 };
16
17 // 用克拉克馬公式求二元一次解
18 Point2D intersection2D(Line2D l1, Line2D l2) {
19     LL D = l1.a * l2.b - l2.a * l1.b;
20     LL Dx = l1.c * l2.b - l2.c * l1.b;
21     LL Dy = l1.a * l2.c - l2.a * l1.c;
22
23     if(D) { // intersection
24         double x = 1.0 * Dx / D;
25         double y = 1.0 * Dy / D;
26     } else {
27         if(Dx || Dy) // Parallel lines
28             else // Same line
29     }
30 }

```

8.2 半平面相交

```

1 // Q: 給定一張凸包(已排序的點),
2 // 找出圖中離凸包外最遠的距離
3
4 const int maxn = 100 + 10;
5 const double eps = 1e-7;
6
7 struct Vector {
8     double x, y;
9     Vector(double x=0.0, double y=0.0): x(x), y(y) {}
10
11     Vector operator+(Vector v) {
12         return Vector(x+v.x, y+v.y);
13     }
14     Vector operator-(Vector v) {
15         return Vector(x-v.x, y-v.y);
16     }
17     Vector operator*(double val) {
18         return Vector(x*val, y*val);
19     }

```

```

20     double dot(Vector v) { return x*v.x + y*v.y; }
21     double cross(Vector v) { return x*v.y - y*v.x; }
22     double length() { return sqrt(dot(*this)); }
23     Vector unit_normal_vector() {
24         double len = length();
25         return Vector(-y/len, x/len);
26     }
27 };
28
29 using Point = Vector;
30
31 struct Line {
32     Point p;
33     Vector v;
34     double ang;
35     Line(Point p={}, Vector v={}): p(p), v(v) {
36         ang = atan2(v.y, v.x);
37     }
38     bool operator<(const Line& l) const {
39         return ang < l.ang;
40     }
41     Point intersection(Line l) {
42         Vector u = p - l.p;
43         double t = l.v.cross(u) / v.cross(l.v);
44         return p + v*t;
45     }
46 };
47
48 int n, m;
49 Line narrow[maxn]; // 要判斷的直線
50 Point poly[maxn]; // 能形成半平面交的凸包邊界點
51
52 // return true if point p is on the left of line l
53 bool onLeft(Point p, Line l) {
54     return l.v.cross(p-l.p) > 0;
55 }
56
57 int halfplaneIntersection() {
58     int l, r;
59     Line L[maxn]; // 排序後的向量隊列
60     Point P[maxn]; // s[i] 跟 s[i-1] 的交點
61
62     L[l=r=0] = narrow[0]; // notice: narrow is sorted
63     for(int i=1; i<n; i++) {
64         while(l<r && !onLeft(P[r-1], narrow[i])) r--;
65         while(l<r && !onLeft(P[l], narrow[i])) l++;
66
67         L[++r] = narrow[i];
68         if(l < r) P[r-1] = L[r-1].intersection(L[r]);
69     }
70
71     while(l<r && !onLeft(P[r-1], L[l])) r--;
72     if(r-l <= 1) return 0;
73
74     P[r] = L[r].intersection(L[l]);
75
76     int m=0;
77     for(int i=1; i<=r; i++) {
78         poly[m++] = P[i];
79     }
80
81     return m;
82 }
83
84 Point pt[maxn];
85 Vector vec[maxn];
86 Vector normal[maxn]; // normal[i] = vec[i] 的單位法向量
87
88 double bsearch(double l=0.0, double r=1e4) {
89     if(abs(r-l) < 1e-7) return l;
90
91     double mid = (l + r) / 2;
92
93     for(int i=0; i<n; i++) {
94         narrow[i] = Line(pt[i]+normal[i]*mid, vec[i]);
95     }
96

```

```

97     if(halfplaneIntersection())
98         return bsearch(mid, r);
99     else return bsearch(l, mid);
100 }
101
102 int main() {
103     while(~scanf("%d", &n) && n) {
104         for(int i=0; i<n; i++) {
105             double x, y;
106             scanf("%lf%lf", &x, &y);
107             pt[i] = {x, y};
108         }
109         for(int i=0; i<n; i++) {
110             vec[i] = pt[(i+1)%n] - pt[i];
111             normal[i] = vec[i].unit_normal_vector();
112         }
113         printf("%.6lf\n", bsearch());
114     }
115     return 0;
116 }
117 }

```

8.3 凸包

```

1 // Q: 平面上給定多個區域，由多個座標點所形成，再給定
2 // 多點(x,y)，判斷有落點的區域(destroyed)的面積總和。
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 const int maxn = 500 + 10;
7 const int maxCoordinate = 500 + 10;
8
9 struct Point {
10     int x, y;
11 };
12
13 int n;
14 bool destroyed[maxn];
15 Point arr[maxn];
16 vector<Point> polygons[maxn];
17
18 void scanAndSortPoints() {
19     int minX = maxCoordinate, minY = maxCoordinate;
20     for(int i=0; i<n; i++) {
21         int x, y;
22         scanf("%d%d", &x, &y);
23         arr[i] = (Point){x, y};
24         if(y < minY || (y == minY && x < minX)) {
25             // If there are floating points, use:
26             // if(y<minY || (abs(y-minY)<eps && x<minX)) {
27                 minX = x, minY = y;
28             }
29         }
30     }
31     sort(arr, arr+n, [minX, minY](Point& a, Point& b){
32         double theta1 = atan2(a.y - minY, a.x - minX);
33         double theta2 = atan2(b.y - minY, b.x - minX);
34         return theta1 < theta2;
35     });
36     return;
37 }
38
39 // returns cross product of u(AB) x v(AC)
40 int cross(Point& A, Point& B, Point& C) {
41     int u[2] = {B.x - A.x, B.y - A.y};
42     int v[2] = {C.x - A.x, C.y - A.y};
43     return (u[0] * v[1]) - (u[1] * v[0]);
44 }
45
46 // size of arr = n >= 3
47 // st = the stack using vector, m = index of the top
48 vector<Point> convex_hull() {
49     vector<Point> st(arr, arr+3);
50     for(int i=3, m=2; i<n; i++, m++) {
51         while(m >= 2) {
52             if(cross(st[m], st[m-1], arr[i]) < 0)

```

```

52             break;
53             st.pop_back();
54             m--;
55         }
56         st.push_back(arr[i]);
57     }
58     return st;
59 }
60
61 bool inPolygon(vector<Point>& vec, Point p) {
62     vec.push_back(vec[0]);
63     for(int i=1; i<vec.size(); i++) {
64         if(cross(vec[i-1], vec[i], p) < 0) {
65             vec.pop_back();
66             return false;
67         }
68     }
69     vec.pop_back();
70     return true;
71 }
72
73     1 | x1  x2  x3  x4  x5      xn |
74 A = - |   x   x   x   x   x ... x |
75     2 | y1  y2  y3  y4  y5      yn |
76 double calculateArea(vector<Point>& v) {
77     v.push_back(v[0]); // make v[n] = v[0]
78     double result = 0.0;
79     for(int i=1; i<v.size(); i++)
80         result += v[i-1].x*v[i].y - v[i-1].y*v[i].x;
81     v.pop_back();
82     return result / 2.0;
83 }
84
85 int main() {
86     int p = 0;
87     while(~scanf("%d", &n) && (n != -1)) {
88         scanAndSortPoints();
89         polygons[p++] = convex_hull();
90     }
91
92     int x, y;
93     double result = 0.0;
94     while(~scanf("%d%d", &x, &y)) {
95         for(int i=0; i<p; i++) {
96             if(inPolygon(polygons[i], (Point){x, y}))
97                 destroyed[i] = true;
98         }
99     }
100     for(int i=0; i<p; i++) {
101         if(destroyed[i])
102             result += calculateArea(polygons[i]);
103     }
104     printf("%.2lf\n", result);
105     return 0;
106 }

```

9 DP

9.1 以價值為主的背包

```

1 /*w 變得太大所以一般的01背包解法變得不可能
2 觀察題目w變成10^9
3 而v_i變成10^3
4 N不變10^2
5 試著湊湊看dp狀態
6 dp[maxn][maxv]是可接受的複雜度
7 剩下的是轉移式，轉移式變成
8 dp[i][j] = w ->
    當目前只考慮到第i個商品時，達到獲利j時最少的weight總和
    = w
9 所以答案是dp[n][1 ~ maxv]找價值最大且裝的下的*/
10 #define maxn 105
11 #define maxv 100005

```

```

12 long long dp[maxn][maxv];
13 long long weight[maxn];
14 long long v[maxn];
15 int main() {
16     int n;
17     long long w;
18     scanf("%d %lld", &n, &w);
19     for (int i = 1; i <= n; ++i) {
20         scanf("%lld %lld", &weight[i], &v[i]);
21     }
22     memset(dp, 0x3f, sizeof(dp));
23     dp[0][0] = 0;
24     for (int i = 1; i <= n; ++i) {
25         for (int j = 0; j <= maxv; ++j) {
26             if (j - v[i] >= 0)
27                 dp[i][j] = dp[i - 1][j - v[i]] +
28                     weight[i];
29             dp[i][j] = min(dp[i - 1][j], dp[i][j]);
30         }
31     }
32     long long res = 0;
33     for (int j = maxv - 1; j >= 0; --j) {
34         if (dp[n][j] <= w) {
35             res = j;
36             break;
37         }
38     }
39     printf("%lld\n", res);
40 }

```

9.2 抽屜

```

1 // dp[n][s][t] n: 幾個抽屜 s: 幾個是安全的 t: (0 or
2 // 1) 最上面的抽屜是U or L
3 // 分兩種 case
4 // case 1: dp[n][s][0] = dp[n - 1][s + 1][1] + dp[n -
5 // 1][s][0]
6 // 此時最上面放U, 則
7 // dp[n - 1][s + 1][1]: 現在要放的U會導致底下n -
8 // 1個抽屜最上面L變不安全, 為了得到n個抽屜s個安全, 所以要
9 // + 1
10 // dp[n - 1][s][0]: n -
11 // 1個抽屜有s個安全, 現在在其上面再放一個U不影響s的數量
12 // case 2: dp[n][s][1] = dp[n - 1][s - 1][1] + dp[n -
13 // 1][s - 1][0]
14 // 在最上面放L, 底下n - 1個抽屜有s -
15 // 1個安全, 無論上方是U、L皆不影響
16 // long long dp[70][70][2];
17 // 初始條件
18 dp[1][0][0] = dp[1][1][1] = 1;
19 for (int i = 2; i <= 66; ++i){
20     // i個抽屜0個安全且上方0 = (底下i -
21     // 1個抽屜且1個安全且最上面L) + (底下n -
22     // 1個抽屜0個安全且最上方為0)
23     dp[i][0][0] = dp[i - 1][1][1] + dp[i - 1][0][0];
24     for (int j = 1; j <= i; ++j) {
25         dp[i][j][0] = dp[i - 1][j + 1][1] + dp[i -
26         // 1][j][0];
27         dp[i][j][1] = dp[i - 1][j - 1][1] + dp[i -
28         // 1][j - 1][0];
29     }
30 }
31 //答案在 dp[n][s][0] + dp[n][s][1];

```

9.3 Barcode

```

1 int N, K, M;
2 long long dp[55][55];
3 // n -> 目前剩多少units
4 // k -> 目前剩多少bars

```

```

6 // m -> 1 bar最多多少units
7 long long dfs(int n, int k) {
8     if (k == 1) {
9         return (n <= M);
10    }
11    if (dp[n][k] != -1)
12        return dp[n][k];
13    long long result = 0;
14    for (int i = 1; i < min(M + 1, n); ++i) { // <
15        // min(M + 1, n)是因為n不能==0
16        result += dfs(n - i, k - 1);
17    }
18    return dp[n][k] = result;
19 }
20 int main() {
21     while (scanf("%d %d %d", &N, &K, &M) != EOF) {
22         memset(dp, -1, sizeof(dp));
23         printf("%lld\n", dfs(N, K));
24     }
25 }

```

9.4 Deque 最大差距

```

1 /*定義dp[l][r]是l ~ r時與先手最大差異值
2 Deque可以拿頭尾
3 所以轉移式中dp[l][r]與dp[l + 1][r]、dp[l][r - 1]有關
4 轉移式:
5 dp[l][r] = max{a[l] - solve(l + 1, r), a[r] -
6     solve(l, r - 1)}
7 裡面用減的主要是因為求的是相減且會一直換手, 所以正負正負...*/
8 #define maxn 3005
9 bool vis[maxn][maxn];
10 long long dp[maxn][maxn];
11 long long a[maxn];
12 long long solve(int l, int r) {
13     if (l > r)
14         return 0;
15     if (vis[l][r])
16         return dp[l][r];
17     vis[l][r] = true;
18     long long res = a[l] - solve(l + 1, r);
19     res = max(res, a[r] - solve(l, r - 1));
20     return dp[l][r] = res;
21 }
22 int main() {
23     ...
24     printf("%lld\n", solve(1, n));
25 }

```

9.5 LCS 和 LIS

```

1 //最長共同子序列(LCS)
2 給定兩序列 A,B, 求最長的序列 C,
3 C 同時為 A,B 的子序列。
4
5 //最長遞增子序列 (LIS)
6 給你一個序列 A, 求最長的序列 B,
7 B 是一個 (非) 嚴格遞增序列, 且為 A 的子序列。
8
9 //LCS 和 LIS 題目轉換
10 LIS 轉成 LCS
11 1. A 為原序列, B=sort(A)
12 2. 對 A,B 做 LCS
13 LCS 轉成 LIS
14 1. A, B 為原本的兩序列
15 2. 最 A 序列作編號轉換, 將轉換規則套用在 B
16 3. 對 B 做 LIS
17 4. 重複的數字在編號轉換時後要變成不同的數字,
18 越早出現的數字要越小
19 5. 如果有數字在 B 裡面而不在 A 裡面,

```

20 | 直接忽略這個數字不做轉換即可

9.6 RangeDP

```

1 //區間dp
2 int dp[55][55]; // dp[i][j] -> [i,
  j]切割區間中最小的cost
3 int cuts[55];
4 int solve(int i, int j) {
5     if (dp[i][j] != -1)
6         return dp[i][j];
7     //代表沒有其他切法，只能是cuts[j] - cuts[i]
8     if (i == j - 1)
9         return dp[i][j] = 0;
10    int cost = 0x3f3f3f3f;
11    for (int m = i + 1; m < j; ++m) {
12        //枚舉區間中間切點
13        cost = min(cost, solve(i, m) + solve(m, j) +
14            cuts[j] - cuts[i]);
15    }
16    return dp[i][j] = cost;
17 }
18 int main() {
19     int l;
20     int n;
21     while (scanf("%d", &l) != EOF && l){
22         scanf("%d", &n);
23         for (int i = 1; i <= n; ++i)
24             scanf("%d", &cuts[i]);
25         cuts[0] = 0;
26         cuts[n + 1] = 1;
27         memset(dp, -1, sizeof(dp));
28         printf("The minimum cutting is %d.\n",
29             solve(0, n + 1));
30     }
31     return 0;
32 }

```

9.7 stringDP

- Edit distance

S_1 最少需要經過幾次增、刪或換字變成 S_2

$$dp[i][j] = \begin{cases} i+1 & \text{if } j = -1 \\ j+1 & \text{if } i = -1 \\ \min \begin{cases} dp[i-1][j-1] \\ dp[i][j-1] \\ dp[i-1][j] \end{cases} & \text{if } S_1[i] \neq S_2[j] \end{cases} + 1$$

- Longest Palindromic Subsequence

$$dp[l][r] = \begin{cases} 1 & \text{if } l = r \\ \max\{dp[l+1][r-1], dp[l][r-1]\} & \text{if } S[l] \neq S[r] \end{cases}$$

9.8 TreeDP 有幾個 path 長度為 k

```

1 #define maxn 50005
2 #define maxk 505
3 //dp[u][u]的child且距離u長度k的數量
4 long long dp[maxn][maxk];
5 vector<vector<int>> G;
6 int n, k;
7 long long res = 0;
8 void dfs(int u, int p) {
9     //u自己
10    dp[u][0] = 1;
11    for (int v: G[u]) {
12        if (v == p)
13            continue;
14        dfs(v, u);
15        for (int i = 1; i <= k; ++i) {
16            //子樹v距離i-1的等於對於u來說距離i
17            dp[u][i] += dp[v][i-1];

```

```

18    }
19 }
20 //統計在u子樹中距離u為k的數量
21 res += dp[u][k];
22 //統計橫跨u但還是在u的子樹中合計長度為k的:
23 //考慮u有一子節點v，在v子樹中距離v長度為x的
24 //以及不在v子樹但在u子樹中(這樣才會是橫跨u)且距離u長度為k
  - x - 1的
25 //共有0.5 * (dp[v][x] * (dp[u][k-x-1] -
  dp[v][k-x-2]))
26 //以上算式是重點，可使複雜度下降，否則枚舉一定超時
27 //其中 dp[u][k-x-1]是所有u子樹中距離u為k-x
  - 1的節點
28 // - dp[v][k-x-2]是因為我們不要v子樹的節點且距離u為k-x-1
  的(要v子樹以外的)，
29 //那些點有dp[v][k-x-2]，最後0.5是由於計算中i
  -> j以及j -> i(i、j是不同節點)
30 //都會被算一遍，所以要 * 0.5
31 long long cnt = 0;
32 for (int v: G[u]) {
33     if (v == p)
34         continue;
35     for (int x = 0; x <= k - 2; ++x) {
36         cnt += dp[v][x] * (dp[u][k-x-1] -
37             dp[v][k-x-2]);
38     }
39     res += cnt / 2;
40 }
41 int main() {
42     scanf("%d %d", &n, &k);
43     G.assign(n + 5, vector<int>());
44     int u, v;
45     for (int i = 1; i < n; ++i) {
46         scanf("%d %d", &u, &v);
47         G[u].emplace_back(v);
48         G[v].emplace_back(u);
49     }
50     dfs(1, -1);
51     printf("%lld\n", res);
52     return 0;
53 }

```

9.9 TreeDP reroot

```

1 /*
2 Re-root經典題
3 1. 選0作為root
4 2. 以0為root去求出所有節點的subtreeSize
5 3. 觀察到re-root後的關係式
6 配合思考圖片
7 f(0)與f(2)的關係
8 f(2) = f(0) + a - b
9 a = n - b, (subtree(2)以外的節點)
10 b = subtreeSize(2), (subtree(2))
11 所以f(n)是n為root到所有點的距離
12 f(2) = f(0) + n - 2 * subtreeSize(2)
13 這就是快速得到答案的轉移式
14 f(child) = f(parent) + n - 2 * subtreeSize(child)
15 流程
16 1. root = 0去求各項subtreeSize
17 2. 求f(root)
18 3. 以f(0)去求出re-root後的所有f(v), v != 0
19 整體來說
20 暴力解 O(n ^ 2)
21 re-root dp on tree O(n + n + n) -> O(n)
22 */
23 class Solution {
24 public:
25     vector<int> sumOfDistancesInTree(int n,
26         vector<vector<int>>& edges) {

```



```

26     this->res.assign(n, 0);
27     G.assign(n + 5, vector<int>());
28     for (vector<int>& edge: edges) {
29         G[edge[0]].emplace_back(edge[1]);
30         G[edge[1]].emplace_back(edge[0]);
31     }
32     memset(this->visited, 0,
33            sizeof(this->visited));
34     this->dfs(0);
35     memset(this->visited, 0,
36            sizeof(this->visited));
37     this->res[0] = this->dfs2(0, 0);
38     return this->res;
39 }
40 private:
41     vector<vector<int>>> G;
42     bool visited[30005];
43     int subtreeSize[30005];
44     vector<int> res;
45     //求subtreeSize
46     int dfs(int u) {
47         this->visited[u] = true;
48         for (int v: this->G[u]) {
49             if (!this->visited[v]) {
50                 this->subtreeSize[u] += this->dfs(v);
51             }
52         }
53         //自己
54         this->subtreeSize[u] += 1;
55         return this->subtreeSize[u];
56     }
57     //求res[0], 0到所有點的距離
58     int dfs2(int u, int dis) {
59         this->visited[u] = true;
60         int sum = 0;
61         for (int v: this->G[u]) {
62             if (!visited[v]) {
63                 sum += this->dfs2(v, dis + 1);
64             }
65         }
66         //要加上自己的距離
67         return sum + dis;
68     }
69     //算出所有的res
70     void dfs3(int u, int n) {
71         this->visited[u] = true;
72         for (int v: this->G[u]) {
73             if (!visited[v]) {
74                 this->res[v] = this->res[u] + n - 2 *
75                     this->subtreeSize[v];
76                 this->dfs3(v, n);
77             }
78         }
79     };

```

9.10 Weighted LIS

```

1  /*概念基本上與LIS相同，但不能用greedy的LIS，所以只能用dp
2  但有個問題是dp版要 $O(n^2)$ 
3   $n$ 最大200000一定超時，所以這題要改一下dp的LIS
4  在DP版中有一層迴圈是要往前搜height[j] < height[i](j
5  in 1 ~ i - 1)的然後挑B[j]最大的
6  這for loop造成 $O(n^2)$ 
7  注意到子問題是在1 ~ i - 1中挑出B[j]最大的
8  這一步可以用線段樹優化
9  所以最後可以在 $O(n \log n)$ 完成*/
10 #define maxn 200005
11 long long dp[maxn];
12 long long height[maxn];
13 long long B[maxn];

```

```

13 long long st[maxn << 2];
14 void update(int p, int index, int l, int r, long long
15 v) {
16     if (l == r) {
17         st[index] = v;
18         return;
19     }
20     int mid = (l + r) >> 1;
21     if (p <= mid)
22         update(p, (index << 1), l, mid, v);
23     else
24         update(p, (index << 1) + 1, mid + 1, r, v);
25     st[index] = max(st[index << 1], st[(index << 1) +
26     1]);
27 }
28 long long query(int index, int l, int r, int ql, int
29 qr) {
30     if (ql <= l && r <= qr)
31         return st[index];
32     int mid = (l + r) >> 1;
33     long long res = -1;
34     if (ql <= mid)
35         res = max(res, query(index << 1, l, mid, ql,
36 qr));
37     if (mid < qr)
38         res = max(res, query((index << 1) + 1, mid +
39 1, r, ql, qr));
40     return res;
41 }
42 int main() {
43     int n;
44     scanf("%d", &n);
45     for (int i = 1; i <= n; ++i)
46         scanf("%lld", &height[i]);
47     for (int i = 1; i <= n; ++i)
48         scanf("%lld", &B[i]);
49     long long res = B[1];
50     update(height[1], 1, 1, n, B[1]);
51     for (int i = 2; i <= n; ++i) {
52         long long temp;
53         if (height[i] - 1 >= 1)
54             temp = B[i] + query(1, 1, n, 1, height[i]
55 - 1);
56         else
57             temp = B[i];
58         update(height[i], 1, 1, n, temp);
59         res = max(res, temp);
60     }
61     printf("%lld\n", res);
62     return 0;
63 }

```

9.11 dplist

1	-----
2	
3	
4	-----
5	
6	
7	-----
8	
9	
10	-----
11	
12	
13	-----
14	
15	
16	-----
17	
18	
19	-----
20	
21	
22	-----

23									
24									
25	-----								
26									
27									
28	-----								
29									
30									
31	-----								
32									
33									
34	-----								
35									
36									
37	-----								
38									
39									
40	-----								
41									
42									
43	-----								
44									
45									
46	-----								
47									
48									
49	-----								
50									
51									
52	-----								
53									
54									
55	-----								
56									
57									
58	-----								
59									
60									
61	-----								
62									
63									
64	-----								
65									
66									
67	-----								
68									
69									
70	-----								
71									
72									
73	-----								
74									
75									
76	-----								
77									
78									
79	-----								
80									
81									
82	-----								
83									
84									
85	-----								
86									
87									
88	-----								
89									
90									
91	-----								
92									
93									
94	-----								
95									
96									
97	-----								
98									
99									

100	-----								
101									
102									
103	-----								
104									
105									
106	-----								
107									
108									
109	-----								
110									
111									
112	-----								
113									
114									
115	-----								
116									
117									
118	-----								
119									
120									
121	-----								
122									
123									
124	-----								
125									
126									
127	-----								
128									
129									
130	-----								
131									
132									
133	-----								
134									
135									
136	-----								
137									
138									
139	-----								
140									
141									
142	-----								
143									
144									
145	-----								
146									
147									
148	-----								
149									
150									
151	-----								
152									
153									
154	-----								
155									
156									
157	-----								
158									
159									
160	-----								
161									
162									
163	-----								
164									
165									
166	-----								
167									
168									
169	-----								
170									
171									
172	-----								
173									
174									
175	-----								
176									

177									
178									
179									
180									
181									
182									
183									
184									
185									
186									
187									
188									
189									
190									
191									
192									
193									
194									
195									
196									
197									
198									
199									
200									
201									
202									
203									
204									
205									
206									
207									
208									
209									
210									
211									
212									
213									
214									
215									
216									
217									
218									
219									
220									
221									
222									
223									
224									
225									
226									
227									
228									
229									
230									
231									
232									
233									
234									
235									
236									
237									
238									
239									
240									
241									
242									
243									
244									
245									
246									
247									
248									
249									
250									
251									
252									
253									

254									
255									
256									
257									
258									
259									
260									
261									
262									
263									
264									
265									
266									
267									
268									
269									
270									
271									
272									
273									
274									
275									
276									
277									
278									
279									
280									
281									
282									
283									
284									
285									
286									
287									
288									
289									
290									
291									
292									
293									
294									
295									
296									
297									
298									
299									
300									
301									
302									
303									
304									
305									
306									
307									
308									
309									
310									
311									
312									
313									
314									
315									
316									
317									
318									
319									
320									
321									
322									
323									
324									
325									
326									
327									
328									
329									
330									

331							
332							
333							
334							
335							
336							
337							
338							
339							
340							
341							
342							
343							
344							
345							
346							
347							
348							
349							
350							
351							
352							
353							
354							
355							
356							
357							
358							
359							
360							
361							
362							
363							
364							
365							
366							
367							
368							
369							
370							
371							
372							
373							
374							
375							
376							
377							
378							
379							
380							
381							
382							
383							
384							
385							
386							
387							
388							
389							
390							
391							
392							
393							
394							
395							
396							
397							
398							
399							
400							
401							
402							
403							
404							
405							
406							
407							

408							
409							
410							
411							
412							
413							
414							
415							
416							
417							
418							
419							
420							
421							
422							
423							
424							
425							
426							
427							
428							
429							
430							
431							
432							
433							
434							
435							
436							
437							
438							
439							
440							
441							
442							
443							
444							
445							
446							
447							
448							
449							
450							
451							
452							
453							
454							
455							
456							
457							
458							
459							
460							
461							
462							
463							
464							
465							
466							
467							
468							
469							
470							
471							
472							
473							
474							
475							
476							
477							
478							
479							
480							
481							
482							
483							
484							

485							
486							
487	-----						
488							
489							
490	-----						
491							
492							
493	-----						
494							
495							
496	-----						
497							
498							
499	-----						
500							
501							
502	-----						
503							
504							
505	-----						
506							
507							
508	-----						
509							
510							
511	-----						
512							
513							
514	-----						
515							
516							
517	-----						
518							
519							
520	-----						
521							
522							
523	-----						
524							
525							
526	-----						
527							
528							
529	-----						
530							
531							
532	-----						
533							
534							
535	-----						
536							
537							
538	-----						
539							
540							
541	-----						
542							
543							
544	-----						
545							
546							
547	-----						
548							
549							
550	-----						
551							
552							
553	-----						
554							
555							
556	-----						
557							
558							
559	-----						
560							
561							

562	-----						
563							
564							
565	-----						
566							
567							
568	-----						
569							
570							
571	-----						
572							
573							
574	-----						
575							
576							
577	-----						
578							
579							
580	-----						
581							
582							
583	-----						
584							
585							
586	-----						
587							
588							
589	-----						
590							
591							
592	-----						
593							
594							
595	-----						
596							
597							
598	-----						
599							
600							
601	-----						
602							
603							
604	-----						
605							
606							
607	-----						
608							
609							
610	-----						
611							
612							
613	-----						
614							
615							
616	-----						
617							
618							
619	-----						
620							
621							
622	-----						
623							
624							
625	-----						
626							
627							
628	-----						
629							
630							
631	-----						
632							
633							
634	-----						
635							
636							
637	-----						
638							

639									
640									
641									
642									
643									
644									
645									
646									
647									
648									
649									
650									
651									
652									
653									
654									
655									
656									
657									
658									
659									
660									
661									
662									
663									
664									
665									
666									
667									
668									
669									
670									
671									
672									
673									
674									
675									
676									
677									
678									
679									
680									
681									
682									
683									
684									
685									
686									
687									
688									
689									
690									
691									
692									
693									
694									
695									
696									
697									
698									
699									
700									
701									
702									
703									
704									
705									
706									
707									
708									
709									
710									
711									
712									
713									
714									
715									

716									
717									
718									
719									
720									
721									
722									
723									
724									
725									
726									
727									
728									
729									
730									
731									
732									
733									
734									
735									
736									
737									
738									
739									
740									
741									
742									
743									
744									
745									
746									
747									
748									
749									
750									
751									
752									
753									
754									
755									
756									
757									
758									
759									
760									
761									
762									
763									
764									
765									
766									
767									
768									
769									
770									
771									
772									
773									
774									
775									
776									
777									
778									
779									
780									
781									
782									
783									
784									
785									
786									
787									
788									
789									
790									
791									
792									

793							
794							
795							
796							
797							
798							
799							
800							
801							
802							
803							
804							
805							
806							
807							
808							
809							
810							
811							
812							
813							
814							
815							
816							
817							
818							
819							
820							
821							
822							
823							
824							
825							
826							
827							
828							
829							
830							
831							
832							
833							
834							
835							
836							
837							
838							
839							
840							
841							
842							
843							
844							
845							
846							
847							
848							
849							
850							
851							
852							
853							
854							
855							
856							
857							
858							
859							
860							
861							
862							
863							
864							
865							
866							
867							
868							
869							

870							
871							
872							
873							
874							
875							
876							
877							
878							
879							
880							
881							
882							
883							
884							
885							
886							
887							
888							
889							
890							
891							
892							
893							
894							
895							
896							
897							
898							
899							
900							
901							
902							
903							
904							
905							
906							
907							
908							
909							
910							
911							
912							
913							
914							
915							
916							
917							
918							
919							
920							
921							
922							
923							
924							
925							
926							
927							
928							
929							
930							
931							
932							
933							
934							
935							
936							
937							
938							
939							
940							
941							
942							
943							
944							
945							
946							

947							
948							
949	-----						
950							
951							
952	-----						
953							
954							
955	-----						
956							
957							
958	-----						
959							
960							
961	-----						
962							
963							
964	-----						
965							
966							
967	-----						
968							
969							
970	-----						
971							
972							
973	-----						
974							
975							
976	-----						
977							
978							
979	-----						
980							
981							
982	-----						
983							
984							
985	-----						
986							
987							
988	-----						
989							
990							
991	-----						
992							
993							
994	-----						
995							
996							
997	-----						
998							
999							
1000	-----						
1001							
1002							
1003	-----						
1004							
1005							
1006	-----						
1007							
1008							
1009	-----						
1010							
1011							
1012	-----						
1013							
1014							
1015	-----						
1016							
1017							
1018	-----						
1019							
1020							
1021	-----						
1022							
1023							

1024	-----						
1025							
1026							
1027	-----						
1028							
1029							
1030	-----						
1031							
1032							
1033	-----						
1034							
1035							
1036	-----						
1037							
1038							
1039	-----						
1040							
1041							
1042	-----						
1043							
1044							
1045	-----						
1046							
1047							
1048	-----						
1049							
1050							
1051	-----						
1052							
1053							
1054	-----						
1055							
1056							
1057	-----						
1058							
1059							
1060	-----						
1061							
1062							
1063	-----						
1064							
1065							
1066	-----						
1067							
1068							
1069	-----						
1070							
1071							
1072	-----						
1073							
1074							
1075	-----						
1076							
1077							
1078	-----						
1079							
1080							
1081	-----						
1082							
1083							
1084	-----						
1085							
1086							
1087	-----						
1088							
1089							
1090	-----						
1091							
1092							
1093	-----						
1094							
1095							
1096	-----						
1097							
1098							
1099	-----						
1100							

1101									
1102									
1103									
1104									
1105									
1106									
1107									
1108									
1109									
1110									
1111									
1112									
1113									
1114									
1115									
1116									
1117									
1118									
1119									
1120									
1121									
1122									
1123									
1124									
1125									
1126									
1127									
1128									
1129									
1130									
1131									
1132									
1133									
1134									
1135									
1136									
1137									
1138									
1139									
1140									
1141									
1142									
1143									
1144									
1145									
1146									
1147									
1148									
1149									
1150									
1151									
1152									
1153									
1154									
1155									
1156									
1157									
1158									
1159									
1160									
1161									
1162									
1163									
1164									
1165									
1166									
1167									
1168									
1169									
1170									
1171									
1172									
1173									
1174									
1175									
1176									
1177									

1178									
1179									
1180									
1181									
1182									
1183									
1184									
1185									
1186									
1187									
1188									
1189									
1190									
1191									
1192									
1193									
1194									
1195									
1196									
1197									
1198									
1199									
1200									
1201									
1202									
1203									
1204									
1205									
1206									
1207									
1208									
1209									
1210									
1211									
1212									
1213									
1214									
1215									
1216									
1217									
1218									
1219									
1220									
1221									
1222									
1223									
1224									
1225									
1226									
1227									
1228									
1229									
1230									
1231									
1232									
1233									
1234									
1235									
1236									
1237									
1238									
1239									
1240									
1241									
1242									
1243									
1244									
1245									
1246									
1247									
1248									
1249									
1250									
1251									
1252									
1253									
1254									

1255							
1256							
1257							
1258							
1259							
1260							
1261							
1262							
1263							
1264							
1265							
1266							
1267							
1268							
1269							
1270							
1271							
1272							
1273							
1274							
1275							
1276							
1277							
1278							
1279							
1280							
1281							
1282							
1283							
1284							
1285							
1286							
1287							
1288							
1289							
1290							
1291							
1292							
1293							
1294							
1295							
1296							
1297							
1298							
1299							
1300							
1301							
1302							
1303							
1304							
1305							
1306							
1307							
1308							
1309							
1310							
1311							
1312							
1313							
1314							
1315							
1316							
1317							
1318							
1319							
1320							
1321							
1322							
1323							
1324							
1325							
1326							
1327							
1328							
1329							
1330							
1331							

1332							
1333							
1334							
1335							
1336							
1337							
1338							
1339							
1340							
1341							
1342							
1343							
1344							
1345							
1346							
1347							
1348							
1349							
1350							
1351							
1352							
1353							
1354							
1355							
1356							
1357							
1358							
1359							
1360							
1361							
1362							
1363							
1364							
1365							
1366							
1367							
1368							
1369							
1370							
1371							
1372							
1373							
1374							
1375							
1376							
1377							
1378							
1379							
1380							
1381							
1382							
1383							
1384							
1385							
1386							
1387							
1388							
1389							
1390							
1391							
1392							
1393							
1394							
1395							
1396							
1397							
1398							
1399							
1400							
1401							
1402							
1403							
1404							
1405							
1406							
1407							
1408							

1409									
1410									
1411									
1412									
1413									
1414									
1415									
1416									
1417									
1418									
1419									
1420									
1421									
1422									
1423									
1424									
1425									
1426									
1427									
1428									
1429									
1430									
1431									
1432									
1433									
1434									
1435									
1436									
1437									
1438									
1439									
1440									
1441									
1442									
1443									
1444									
1445									
1446									
1447									
1448									
1449									
1450									
1451									
1452									
1453									
1454									
1455									
1456									
1457									
1458									
1459									
1460									
1461									
1462									
1463									
1464									
1465									
1466									
1467									
1468									
1469									
1470									
1471									
1472									
1473									
1474									
1475									
1476									
1477									
1478									
1479									
1480									
1481									
1482									
1483									
1484									
1485									

1486									
1487									
1488									
1489									
1490									
1491									
1492									
1493									
1494									
1495									
1496									
1497									
1498									
1499									
1500									
1501									
1502									
1503									
1504									
1505									
1506									
1507									
1508									
1509									
1510									
1511									
1512									
1513									
1514									
1515									
1516									
1517									
1518									
1519									
1520									
1521									
1522									
1523									
1524									
1525									
1526									
1527									
1528									
1529									
1530									
1531									
1532									
1533									
1534									

10 Section2

10.1 thm

- 中文測試
- $\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}$
- $\binom{x}{y} = \frac{x!}{y!(x-y)!}$
- $\int_0^\infty e^{-x} \, dx$
- $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$