ACM 模板

2018 年 11 月 22 日

目录

1 字符串处理 **4**
1.1 SAM 4
1.2 AC 自动机 5
1.3 z-algorithm 13
1.4 后缀数组 14
1.5 最长上升子序列 18
1.6 Manacher 18
1.7 KMP 19
1.8 ex-KMP 20
1.9 Sunday 21
1.10 字符串哈希 22
1.11 字符串最大最小表示 23

2 排序算法 **24**
2.1 归并排序 24

3 数学 **25**
3.1 扩展欧几里得算法 25
3.2 求逆元 25
3.3 Miller robin 素数检验 26
3.4 快速傅里叶变换 27
3.5 快速数论变换 29
3.6 求原根 31
3.7 BM 黑盒线代 33
3.8 FWT 35
3.9 Simpson 积分 36
3.10 扩展欧拉定理 36
3.11 杜教筛 37
3.12 Pell 方程 37
3.13 莫比乌斯反演 38
3.14 常用素数 38
3.15 快速取模乘 39
3.16 多项式算法 (逆元,exp,ln) 41
3.17 gcd 45
3.18 Eratosthenes 素数筛 45
3.19 扩展 BSGS 46
3.20 欧拉函数 48
3.21 矩阵快速幂 49
3.22 组合数 53
3.23 长整型无脑取模乘 59

4 数据结构 **59**
4.1 KD-Tree 59
4.2 李超树 62
4.3 左偏树 & 优先队列 63
4.4 树链剖分 66
4.5 Treap 75
4.6 线段树 82
4.7 主席树 84
4.8 二叉查找树 88
4.9 Splay 92
4.10 线段树 101
4.11 二叉堆 & 优先队列 103
4.12 树状数组 106
4.13 二维树状数组 107
4.14 可持久化 Trie 树 108
4.15 并查集 110
4.16 栈 & 队列 110
4.16.1 单调栈 110
4.16.2 队列 111
4.16.3 单调队列 112

5 动态规划 **113**
5.1 数位 dp 113
5.2 状态压缩 114
5.3 背包问题 114
5.3.1 01 背包 114
5.3.2 多重背包 114
5.3.3 完全背包 115

6 图论 **115**
6.1 二分图 115
6.1.1 KM 115
6.1.2 匈牙利 120
6.2 拓扑排序 122
6.2.1 DFS 拓扑排序 122
6.2.2 Kahn 拓扑排序 123
6.3 最短路 125
6.3.1 堆优化 Dijkstra 125
6.3.2 SPFA 126
6.3.3 次短路 Dijkstra 127
6.3.4 AStar 128
6.4 网络流 130

6.4.1 Dinic 130
6.4.2 MCMF 132
6.5 连通图 134
6.5.1 割边 134
6.5.2 连通图 Tarjan 135

7 树 **136**
7.1 LCA 136
7.1.1 RMQ-ST 136
7.1.2 Tarjan 并查集 137
7.1.3 倍增算法 140
7.2 最小生成树 141
7.2.1 O(elog2v)-primMST 141
7.2.2 O(elogv)-prim+heap MST 142

8 计算几何 **144**
8.1 基础定义 144
8.2 点 144
8.3 线 145
8.4 凸包 147
8.5 三角形 151
8.6 圆 152
8.7 O(n)-求凸包 & 旋转卡壳 155
8.8 两圆相交面积 157
8.9 两直线交点 158
8.10 点在直线上的垂点 158
8.11 矩形面积交 158
8.12 线段类 159
8.13 计算三角形外心 161

9 STL **163**
9.1 accumulate 163

10 其他 **163**
10.1 约瑟夫问题 163
10.2 RMQ-ST 164
10.3 一些理论 164
10.4 随机数和文件输出 165
10.5 随机遍历数组 165
10.6 尺取 166
10.7 std::unordered_map 避免 TLE 166
10.8 输入输出挂 167

10.9 CDQ 分治 169
10.10 vim 配置 171
10.11 程序对拍器 172
10.12 简易对排器 173

1 字符串处理

1.1 SAM

```
const int maxn = int(1e5) + 7;
struct SAM {
   struct Node {
      int len, pre; // 到达当前状态的最大步，当前状态的父亲
      std::map<int, int> next;
      void init(int _len, int _pre) { // 状态节点初始化
         len = _len, pre = _pre;
         next.clear();
      }
   } node[maxn << 1];
   int size, last;
   void init() { // sam初始化
      for (int i = 1; i < size; i++) node[i].next.clear();
      node[0].init(0, -1);
      size = 1, last = 0;
   }
   void extend(int ch) {
      int cur = size++, u = last;
      node[cur].len = node[last].len + 1;
      while (u != -1 && !node[u].next.count(ch)) {
         node[u].next[ch] = cur;
         u = node[u].pre;
      }
      if (u == -1) node[cur].pre = 0; // 到达虚拟节点
      else {
         int v = node[u].next[ch];
         if (node[v].len == node[u].len + 1) node[cur].pre = v; //
             状态连续
         else {
            int clone = size++; // 拷贝
            node[clone] = node[v];
            node[clone].len = node[u].len + 1;
            while (u != -1 && node[u].next[ch] == v) {
```

```
                node[u].next[ch] = clone; // 把指向v的全部替换为指向
                    clone
                u = node[u].pre;
            }
            node[v].pre = node[cur].pre = clone;
        }
    }
    last = cur; // 更新last
  }
} sam;
```

1.2 AC 自动机

(by qi)

```
// luogu3796
#include<iostream>
#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>
#include<algorithm>
using namespace std;
const int maxn=1000500;
const int sigsize=26;
struct AC
{
  int ch[maxn][sigsize],f[maxn],sz;
  vector<int> val[maxn];
  AC(){
    memset(ch[0],0,sizeof ch[0]);
    sz=0;
    val[0].clear();
  }
  int idx(char c){return c-'a';};
  void insert(char *s,int x){
    if(x==0){
```

```
23          memset(ch[0],0,sizeof ch[0]);
24          sz=0;
25          val[0].clear();
26      }
27      int u=0;
28      for (int i = 0; s[i] != '\0'; ++i)
29      {
30          int c=idx(s[i]);
31          if(ch[u][c]) u=ch[u][c];
32          else{
33              ++sz;
34              memset(ch[sz],0,sizeof ch[sz]);
35              val[sz].clear();
36              ch[u][c]=sz;
37              u=sz;
38          }
39      }
40      val[u].push_back(x);
41  }
42  void get_fail(){
43      f[0]=0;
44      queue<int> q;
45      q.push(0);
46      while(!q.empty()){
47          int u=q.front();
48          q.pop();
49          for (int i = 0; i < 26; ++i)
50          {
51              if(!ch[u][i]) continue;
52              int v=ch[u][i],pre=f[u];
53              while(pre&&!ch[pre][i]) pre=f[pre];
54              f[v]=ch[pre][i];
55              if(!u) f[v]=0;
56              q.push(v);
57          }
58      }
```

```
59     }
60     void query(char *s,int cnt[]){
61        int u=0;
62        for (int i = 0; s[i] != '\0'; ++i)
63        {
64           int c=idx(s[i]);
65           while(u&&!ch[u][c]) u=f[u];
66           if(ch[u][c]) u=ch[u][c];
67           int v=u;
68           while(v){
69              for (int i = 0; i < val[v].size(); ++i)
70              {
71                 cnt[val[v][i]]++;
72              }
73              v=f[v];
74           }
75        }
76     }
77 }ac;
78 char T[1000050];
79 int main(int argc, char const *argv[])
80 {
81     int n;
82     while(~scanf("%d", &n)&&n){
83        char s[n+1][80];
84        for (int i = 0; i < n; ++i)
85        {
86           scanf("%s",s[i]);
87           ac.insert(s[i],i);
88        }
89        ac.get_fail();
90        scanf("%s",T);
91        int cnt[n];
92        memset(cnt,0,sizeof cnt);
93        ac.query(T,cnt);
94        int M=0;
```

```
95        for (int i = 0; i < n; ++i)
96        {
97           M=max(M,cnt[i]);
98        }
99        printf("%d\n", M);
100       for (int i = 0; i < n; ++i)
101       {
102          if(cnt[i]==M)
103             printf("%s\n", s[i]);
104       }
105    }
106    return 0;
107 }
```

(by shui)

```
1  #include <bits/stdc++.h>
2  const int maxn = int(1e6) + 7;
3  struct AC {
4  #define WIDTH 26
5  private:
6     int size, root;
7     struct Node { int end, fail, next[WIDTH]; } node[maxn];
8     int Node() {
9        size++;
10       node[size].end = 0, node[size].fail = -1;
11       for (int i = 0; i < WIDTH; i++) node[size].next[i] = -1;
12       return size;
13    }
14 public:
15    AC() { size = 0, root = Node();}
16    void clear() {
17       size = 0;
18       root = Node();
19    }
20    void insert(char *str, int t = 0) {
21       for (t = root; *str; str++) {
```

```
22          int pos = *str - 'a';
23          if (node[t].next[pos] == -1) node[t].next[pos] = Node();
24          t = node[t].next[pos];
25       }
26       node[t].end++;
27    }
28    void build_fail_pointer() {
29       std::queue<int> que;
30       for (int i = 0; i < WIDTH; i++) if (~node[root].next[i]) {
31          que.push(node[root].next[i]);
32          node[node[root].next[i]].fail = root;
33       }
34       while (!que.empty()) {
35          int cur = que.front();
36          que.pop();
37          for (int i = 0; i < WIDTH; i++) {
38             if (~node[cur].next[i]) {
39                int next = node[cur].next[i], fail = node[cur].fail
                     ;
40                que.push(next);
41                while (~fail) {
42                   if (~node[fail].next[i]) {
43                      node[next].fail = node[fail].next[i];
44                      break;
45                   }
46                   fail = node[fail].fail;
47                }
48                if (fail == -1) node[next].fail = root;
49             }
50          }
51       }
52    }
53    int ac_automaton(char *str) {
54       int result = 0, cur = root;
55       while (*str) {
56          int pos = *str - 'a';
```

```
57          if (~node[cur].next[pos]) {
58             int next = node[cur].next[pos];
59             while (next != root && node[next].end >= 0) {
60                result += node[next].end, node[next].end = -1;
61                next = node[next].fail;
62             }
63             cur = node[cur].next[pos], str++;
64          } else {
65             if (cur == root) str++;
66             else cur = node[cur].fail;
67          }
68       }
69       return result;
70    }
71 } ac;
72
73 int t, n;
74 char str[maxn];
75 int main() {
76    for (scanf("%d", &t); t; t--) {
77       ac.clear();
78       scanf("%d", &n);
79       for (int i = 1; i <= n; i++) {
80          scanf("%s", str);
81          ac.insert(str);
82       }
83       scanf("%s", str);
84       ac.build_fail_pointer();
85       printf("%d\n", ac.ac_automaton(str));
86    }
87    return 0;
88 }
89
90 #include <bits/stdc++.h>
91 const int maxn = int(1e6) + 7;
92 struct AC {
```

```
93  #define TREE_WIDTH 26
94  private:
95     struct Node {
96        int end;
97        Node *fail, *next[TREE_WIDTH];
98        Node() {
99           this->end = 0, this->fail = nullptr;
100          for (int i = 0; i < TREE_WIDTH; i++) { this->next[i] =
                nullptr; }
101       }
102    } *root;
103    void clear(Node *t) {
104       for (int i = 0; i < TREE_WIDTH; i++) if (t->next[i]) clear(
             t->next[i]);
105       delete t;
106    }
107
108 public:
109    AC() { root = new Node; }
110    ~AC() { clear(root); }
111    void clear() {
112       clear(root);
113       root = new Node;
114    }
115    void insert(char *s, int pos = 0) {
116       Node *t = root;
117       while (*s) {
118          pos = *s - 'a';
119          if (t->next[pos] == nullptr) t->next[pos] = new Node;
120          t = t->next[pos];
121          s++;
122       }
123       t->end++;
124    }
125    void build_fail_pointer() {
126       std::queue<Node *> que;
```

```
127     for (int i = 0; i < TREE_WIDTH; i++) if (root->next[i]) {
128        que.push(root->next[i]);
129        root->next[i]->fail = root;
130     }
131     Node *pre = nullptr, *son = nullptr, *fail = nullptr;
132     while (!que.empty()) {
133        pre = que.front();
134        que.pop();
135        for (int i = 0; i < TREE_WIDTH; i++) {
136           if (pre->next[i]) {
137              son = pre->next[i];
138              que.push(son);
139              fail = pre->fail;
140              while (fail) {
141                 if (fail->next[i]) {
142                    son->fail = fail->next[i];
143                    break;
144                 }
145                 fail = fail->fail;
146              }
147              if (!fail) son->fail = root;
148           }
149        }
150     }
151  }
152  int ac_automaton(char *str) {
153     int result = 0, pos;
154     Node *pre = root, *cur = nullptr;
155     while (*str) {
156        pos = *str - 'a';
157        if (pre->next[pos]) {
158           cur = pre->next[pos];
159           while (cur != root) {
160              if (cur->end >= 0) result += cur->end, cur->end =
                    -1;
161              else break;
```

```
                cur = cur->fail;
            }
            pre = pre->next[pos], str++;
         } else {
            if (pre == root) str++;
            else pre = pre->fail;
         }
      }
      return result;
   }
} ac;
int t, n;
char str[maxn];
int main() {
// freopen("../in.txt", "r", stdin);
   for (scanf("%d", &t); t; t--) {
      ac.clear();
      scanf("%d", &n);
      for (int i = 1; i <= n; i++) {
         scanf("%s", str);
         ac.insert(str);
      }
      scanf("%s", str);
      ac.build_fail_pointer();
      printf("%d\n", ac.ac_automaton(str));
   }
   return 0;
}
```

1.3 z-algorithm

```
// z[i]代表 i 开始的后缀与整个串的最长公共前缀
// time complexity O(n)
const int maxn=100050;
struct z_Algorithm {
   int z[maxn];
```

```
6     void init(char *str,int n) {
7        z[1]=n;
8        for(int i = 2,l=0,r=0; i <= n; ++i) {
9           if(i<=r) z[i]=min(z[i-l+1],r-i+1);
10          else z[i]=0;
11          while(str[i+z[i]]==str[1+z[i]]) ++z[i];
12          if(i+z[i]-1>r) r=i+z[i]-1,l=i;
13       }
14    }
15 };
```

1.4 后缀数组

(by shui)

```
1  #include <bits/stdc++.h>
2  const int maxn = int(1e6) + 7;
3  namespace DA { // sa[i]代表排名第i的后缀的下标, height[i]代表相邻排名的
      最长公共前缀
4     int base[maxn], tmp[maxn], sa[maxn], rank[maxn], height[maxn];
         // 下标均从1开始
5     bool cmp(const int *s, int u, int v, int l, int n) {
6        return s[u] == s[v] && (u + l > n ? 0 : s[u + l]) == (v + l
            > n ? 0 : s[v + l]);
7     } // 注意是大于s
8     void clear(int m) { for (int i = 1; i <= m; i++) base[i] = 0; }
9     void init(const char *s, int n, int m) { // 注意字符集的最小值应该
         从1开始
10       int i, k = 0, *pre = rank, *cur = height;
11       for (i = 1, clear(m); i <= n; i++) base[pre[i] = s[i]]++;
12       for (i = 2; i <= m; i++) base[i] += base[i - 1];
13       for (i = n; i; i--) sa[base[pre[i]]--] = i;
14       for (int l = 1, p = 1; p < n && l <= n; l <<= 1, m = p) {
15          for (p = 0, i = n - l + 1; i <= n; i++) cur[++p] = i;
16          for (i = 1; i <= n; i++) if (sa[i] > l) cur[++p] = sa[i]
               - l;
17          for (i = 1; i <= n; i++) tmp[i] = pre[cur[i]];
```

```
18          for (i = 1, clear(m); i <= n; i++) base[tmp[i]]++;
19          for (i = 2; i <= m; i++) base[i] += base[i - 1];
20          for (i = n; i; i--) sa[base[tmp[i]]--] = cur[i];
21          for (std::swap(pre, cur), p = 1, pre[sa[1]] = 1, i = 2; i
                <= n; i++)
22             pre[sa[i]] = cmp(cur, sa[i - 1], sa[i], l, n) ? p :
                  ++p;
23       } // pre[sa[1]] = 1, 不要敲错
24       for (i = 1; i <= n; i++) rank[sa[i]] = i;
25       for (i = 1; i <= n; height[rank[i++]] = k) // height[rank[i
            ]] = k, i++
26          for (k ? k-- : 0; s[i + k] == s[sa[rank[i] - 1] + k]; k
               ++);
27    }
28 }
29 char str[maxn];
30 int main() {
31    while (~scanf("%s", str + 1)) {
32       int len = int(strlen(str + 1));
33       DA::solve(str, len, 130);
34       puts("--------------------Sa--------------------");
35       for (int i = 1; i <= len; ++i) printf("sa[%2d ] = %2d\t%s\n"
            , i, DA::sa[i], str + DA::sa[i]);
36       puts("-----------------Height-------------------");
37       for (int i = 1; i <= len; ++i) printf("height[%2d ]= %2d \n"
            , i, DA::height[i]);
38       puts("-----------------rank---------------------");
39       for (int i = 1; i <= len; ++i) printf("rank[%2d ] = %2d\n",
            i, DA::rank[i]);
40       puts("--------------------END--------------------");
41    }
42    return 0;
43 }
```

(by qi)

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

```
using namespace std;
typedef long long ll;
const int maxn=100050;
char str[maxn];
namespace suffixArray {
   int sa[maxn],x[maxn],c[maxn],t[maxn],n;
   int height[maxn],*rank=x;
   char *str;
   bool cmp(int u,int v,int l) {
      return x[u]==x[v]&&x[u+l]==x[v+l];
   }
   void calHeight();
   void da(char *_str,int _n) {
      int m=255;
      str=_str,n=_n;x[n+1]=x[0]=0,str[n+1]=str[0]=0,t[n+1]=0;// !! 注意计算中t与x数组交换多次，所以也要设定n+1处值
      for(int i = 0; i <= m; ++i) c[i]=0;
      for(int i = 1; i <= n; ++i) c[x[i]=str[i]]++;
      for(int i = 1; i <= m; ++i) c[i]+=c[i-1];
      for(int i = n; i >= 1; --i) sa[c[x[i]]--]=i;

      for(int l = 1; l <= n; l<<=1) {
         int cnt=0;
         for(int i = n-l+1; i <= n; ++i) t[++cnt]=i;
         for(int i = 1; i <= n; ++i) {
            if(sa[i]>l) t[++cnt]=sa[i]-l;
         }

         for(int i = 0; i <= m; ++i) c[i]=0;
         for(int i = 1; i <= n; ++i) c[x[i]]++;
         for(int i = 1; i <= m; ++i) c[i]+=c[i-1];
         for(int i = n; i >= 1; --i) sa[c[x[t[i]]]--]=t[i];

         m=0,t[sa[1]]=++m;
         for(int i = 2; i <= n; ++i) {
            if(cmp(sa[i],sa[i-1],l)) t[sa[i]]=m;
```

```
37              else t[sa[i]]=++m;
38           }
39           swap(x,t);
40           if(m==n) break;
41        }
42        calHeight();
43     }
44     void calHeight() {
45        int h=1;
46        for(int i = 1; i <= n; ++i) {
47           --h;
48           if(h<0) h=0;
49           while(str[i+h]==str[sa[rank[i]-1]+h]) ++h;
50           height[rank[i]]=h;
51        }
52     }
53     void debug(int *arr=sa) {
54        for(int i = 1; i <= n; ++i) {
55           printf("%d%s", arr[i],i==n?"\n":" ");
56        }
57     }
58  }
59  int main() {
60     scanf("%s", str+1);
61     int n=strlen(str+1);
62     suffixArray::da(str,n);
63     using suffixArray::sa;
64     using suffixArray::height;
65     for(int i = 1; i <= n; ++i) {
66        printf("%d%s", sa[i],i==n?"\n":" ");
67     }
68     for(int i = 2; i <= n; ++i) {
69        printf("%d%s", height[i],i==n?"\n":" ");
70     }
71     return 0;
72  }
```

1.5 最长上升子序列

```
// 下标从0开始，最长不下降子序列的话就改成upper_bound，大于改大于等于
int array[maxn], f[maxn]; // f[i]： 第i个位置最小可以是多少
int LIS() {
   f[0] = array[0];
   int len = 1;
   for(int i=1 ; array[i] ; i++) {
      if(array[i] > f[len-1]) f[len++] = array[i];
      else f[lower_bound(f, f+len, array[i]) - f] = array[i];
   }
   return len;
}
```

1.6 Manacher

```
#include <bits/stdc++.h>
const int maxn = int(1e6) + 7;
int p[maxn << 1]; // p[i] 在增广串中以i为中心的最长回文串的半径，半径长度包括str[i]本身
int Manacher(char *str) {
   int len = int(strlen(str)), max_len = -1; // max_len 最长回文串的长度
   int max_right = 0, pos = 0; // max_right 当前已访问过的所有回文串能触及的最右端的下标
   for (int i = len; i >= 0; i--) str[i + i + 2] = str[i], str[i + i + 1] = '#';
   str[0] = '$'; // 下标是从1开始的
   for (int i = 1; str[i]; i++) {
      p[i] = (max_right > i ? std::min(p[pos + pos - i], max_right - i) : 1);
      while (str[i + p[i]] == str[i - p[i]]) p[i]++;
      if (max_right < i + p[i]) pos = i, max_right = i + p[i];
      max_len = std::max(max_len, p[i] - 1);
   }
   return max_len;
}
```

```
char str[maxn << 1]; // 字符串长度要两倍于原串
int main() {
   std::cin >> str;
   std::cout << Manacher(str) << std::endl;
   return 0;
}
```

1.7 KMP

```
#include <bits/stdc++.h>
const int maxn = int(2e7)+7;
struct KMP {
   int next[maxn]; // next下表:[0, strlen(p)]
   int process(const char *p) {
      int i = 0, j = next[0] = -1, m = int(strlen(p));
      while (i < m) {
         if (j < 0 || p[i] == p[j]) {
            i++, j++;
            next[i] = (p[i] == p[j] ? next[j] : j);
            // next[i] = j; // 求循环节的时候需要这样写，对于字符串中的第i个位置
            // 如果i % (i - next[i]) == 0，最小循环节长度为i - next[i] (i >= 1)
         } else j = next[j];
      }
      return m;
   }
   int kmp(const char *str, const char *p) { // s:匹配串 p:模式串，下标均从0开始
      int i = 0, j = 0, n = int(strlen(str)), m = int(strlen(p));
      // int cnt = 0; // p串出现的次数
      while (i < n && j < m) {
         if (j < 0 || str[i] == p[j]) i++, j++;
         else j = next[j];
         if(j == m) {
            return i - m; // 返回第一次出现的下标，从0开始
```

```
            // j = next[j], cnt++; // 可重叠出现次数
            // j = 0, cnt++; // 不可重叠出现次数
         }
      }
      return -1;
      // return cnt;
   }
} kmp;
char p[maxn], s[maxn];
int main() {
   std::cin >> s >> p;
   kmp.process(p);
   std::cout << kmp.kmp(s, p) << std::endl;
   return 0;
}
```

1.8 ex-KMP

```
void GetNext(string & T, int & m, int jump[]) {
   int a = 0, p = 0;
   jump[0] = m;
   for (int i = 1; i < m; i++) {
      if (i >= p || i + jump[i - a] >= p) {
         if (i >= p) p = i;
         while (p < m && T[p] == T[p - i]) p++;
         jump[i] = p - i, a = i;
      }
      else jump[i] = jump[i - a];
   }
}
void GetExtend(string & S, int & n, string & T, int & m, int
   extend[], int jump[]) {
   int a = 0, p = 0;
   GetNext(T, m, jump);
   for (int i = 0; i < n; i++) {
```

```
17          if (i >= p || i + jump[i - a] >= p) { // i >= p 的作用：举个典
                型例子，S 和 T 无一字符相同
18              if (i >= p) p = i;
19              while (p < n && p - i < m && S[p] == T[p - i]) p++;
20              extend[i] = p - i, a = i;
21          }
22          else extend[i] = jump[i - a];
23      }
24  }
25  int main() {
26      int jump[100];
27      int extend[100];
28      string S, T;
29      int n, m;
30      while (cin >> S >> T) {
31          n = S.size();
32          m = T.size();
33          GetExtend(S, n, T, m, extend, jump);
34          // 打印 jump
35          cout << "next: ";
36          for (int i = 0; i < m; i++)
37              cout << jump[i] << " ";
38          // 打印 extend
39          cout << "\nextend: ";
40          for (int i = 0; i < n; i++)
41              cout << extend[i] << " ";
42          cout << endl << endl;
43      }
44      return 0;
45  }
```

1.9 Sunday

```
1  const int maxn = 1007;
2  int jump[maxn];
3
```

```
void Sunday(char *s, char *p) {
   int lens = int(strlen(s)), lenp = int(strlen(p));
   for (int i = 0; i < lenp; i++) jump[p[i]] = i;
   int i = 0, j, k;
   while (j = i, k = 0, i <= lens - lenp) {
      while (j < lens && k < lenp && s[j] == p[k]) j++, k++;
      if (k == lenp) printf("Find \"p\" in \"s\" at %d\n", i);
      if (i + lenp < lens) i += (lenp - jump[s[i + lenp]]);
      else return ;
   }
   printf("Not found\n");
}
```

1.10 字符串哈希

```
//多项式哈希
typedef unsigned long long ull;
const int N = 100000 + 5;
const ull base = 163;
char s[N];
ull hash[N];

void init() {//处理hash值, 字符串下标从1开始
   p[0] = 1;
   hash[0] = 0;
   int n = strlen(s + 1);
   for(int i = 1; i <=100000; i ++) p[i] = p[i-1] * base;
   for(int i = 1; i <= n; i ++) hash[i] = hash[i - 1] * base + (s
      [i] - 'a' + 1);
}

ull get(int l, int r, ull g[]) {//取出g里l - r里面的字符串的hash值
   return g[r] - g[l - 1] * p[r - l + 1];
}

```

```
/**********/

unsigned int BKDRHash(char *str){
   unsigned int seed = 131; // 31 131 1313 13131 131313 etc..
   unsigned int hash = 0;
   while (*str)
      hash = hash * seed + (*str++);
   return (hash & 0x7FFFFFFF);
}

void ha(char *p, int len, unsigned ll h[]) {
   h[len] = 0;
   for (int i = len - 1; i >= 0; i--) {
      h[i] = h[i + 1] * 1008611 + p[i];
   }
}
```

1.11 字符串最大最小表示

```
//return is the index from 0
int getmin(char *s) {
   int i = 0, j = 1, n = int(strlen(s));
   while (i < n && j < n) {
      int k = 0;
      while (s[(i+k)%n] == s[(j+k)%n] && k < n) k++;
      if (k == n) return min(i, j);
      if (s[(i+k)%n] > s[(j+k)%n]) i = max(i+k+1, j+1);
      else j = max(j+k+1, i+1);
   }
   return min(i, j)%n;
}

int getmax(char *s) {
   int i = 0, j = 1, n = int(strlen(s));
   while (i < n && j < n) {
      int k = 0;
```

```
        while (s[(i+k)%n] == s[(j+k)%n] && k < n) k++;
        if (k == n) return min(i, j);
        if (s[(i+k)%n] < s[(j+k)%n]) i = max(i+k+1, j+1);
        else j = max(j+k+1, i+1);
    }
    return min(i, j)%n;
}
```

2 排序算法

2.1 归并排序

```
void mergearray(int a[], int first, int mid, int last, int temp[])
{
    int i = first, j = mid + 1;
    int m = mid, n = last;
    int k = 0;
    while (i <= m && j <= n) {
        if (a[i] <= a[j]) temp[k++] = a[i++];
        else temp[k++] = a[j++];
    }
    while (i <= m) temp[k++] = a[i++];
    while (j <= n) temp[k++] = a[j++];
    for (i = 0; i < k; i++) a[first + i] = temp[i];
}

void mergesort(int a[], int first, int last, int temp[]) {
    if (first < last) {
        int mid = (first + last) / 2;
        mergesort(a, first, mid, temp); //左边有序
        mergesort(a, mid + 1, last, temp); //右边有序
        mergearray(a, first, mid, last, temp); //再将二个有序数列合并
    }
}

bool MergeSort(int a[], int n) {
```

```
25    int *p = new int[n];
26    if (p == NULL)
27       return false;
28    mergesort(a, 0, n - 1, p);
29    delete[] p;
30    return true;
31 }
```

3 数学

3.1 扩展欧几里得算法

```
1  typedef long long ll;
2  ll exgcd(ll a,ll b,ll &x,ll &y){
3     if(b==0){
4        x=1,y=0;
5        return a;
6     }
7     ll d=exgcd(b,a%b,x,y);
8     ll t=x;
9     x=y;
10    y=t-a/b*y;
11    return d;
12 }
```

3.2 求逆元

```
1  // 利用exgcd，必须满足a与mod互质才存在a的逆元
2  ll inv_ele(ll a,ll mod){
3     ll x,y;
4     ll d=exgcd(a,mod,x,y);
5     if(d==1) return (x+mod)%mod;
6     return -1;
7  }
8
9  // 利用Pow(a,mod-2)必须满足mod为质数
10
```

```
11
12 // 线性时间求逆元
13 ll inv[maxn];
14 void init()
15 {
16    inv[1]=1;
17    for(int i = 2; i < maxn; ++i) inv[i]=(-MOD/i+MOD)*inv[MOD%i]%
      MOD;
18 }
```

3.3 Miller robin 素数检验

```
1  // 18位素数: 154590409516822759
2  // 19位素数: 2305843009213693951 (梅森素数)
3  // 19位素数: 4384957924686954497
4  typedef long long ll;
5  struct Miller_Rabin
6  {
7     int prime[5]={2,3,5,233,331};
8     ll qmul(ll x,ll y,ll mod){
9        x%=mod,y%=mod;
10       ll ans=(x*y-(ll)((long double)x/mod*y+1e-3)*mod);
11       ans=(ans%mod+mod)%mod;
12       return ans;
13    }
14    ll qpow(ll x,ll n,ll mod){
15       ll ans=1;
16       while(n){
17          if(n&1) ans=qmul(ans,x,mod);
18          x=qmul(x,x,mod);
19          n>>=1;
20       }
21       return ans;
22    }
23    bool isprime_std(ll p) {
24       if(p < 2) return 0;
```

```
25        if(p != 2 && p % 2 == 0) return 0;
26        ll s = p - 1;
27        while(! (s & 1)) s >>= 1;
28        for(int i = 0; i < 5; ++i) {
29           if(p == prime[i]) return 1;
30           ll t = s, m = qpow(prime[i], s, p);
31           while(t != p - 1 && m != 1 && m != p - 1) {
32              m = qmul(m, m, p);
33              t <<= 1;
34           }
35           if(m != p - 1 && !(t & 1)) return 0;
36        }
37        return 1;
38     }
39  };
```

3.4 快速傅里叶变换

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   const ll MOD=1e9+7;
5   const int maxn=100050;
6   const double Pi=acos(-1);
7   struct FFT {
8      struct C {
9         double x,y;
10        C operator +(const C&t) const {
11           return (C){x+t.x,y+t.y};
12        }
13        C operator -(const C&t) const {
14           return (C){x-t.x,y-t.y};
15        }
16        C operator *(const C&t) const {
17           return (C){x*t.x-y*t.y,x*t.y+y*t.x};
18        }
```

```
}A[maxn*4],B[maxn*4];
void run(int *a,int n,int *b,int m) {
   int N=1;
   while(N<n+m-1) N<<=1;
   for(int i = 0; i < n; ++i) A[i].x=a[i];
   for(int i = 0; i < m; ++i) B[i].x=b[i];
   fft(A,N,1);
   fft(B,N,1);
   for(int i = 0; i < N; ++i) A[i]=A[i]*B[i];
   fft(A,N,-1);
   for(int i = 0; i < n+m; ++i) {
      a[i]=A[i].x+0.5;
   }
}
int idx[maxn*4];
void rev(C *a,int n) {
   static int _n=-1;
   if(_n!=n) {
      _n=n;
      int L=31-__builtin_clz(n);
      for(int i = 0; i < n; ++i)
         idx[i]=(idx[i>>1]>>1)|((i&1)<<(L-1));
   }
   for(int i = 0; i < n; ++i) {
      if(idx[i]>i) swap(a[i],a[idx[i]]);
   }
}
void fft(C *a,int n,int op) {
   rev(a,n);
   for(int l = 2; l <= n; l<<=1) {
      C wn={cos(Pi*2/l*op),sin(Pi*2/l*op)};
      for(int i = 0; i < n; i+=l) {
         C w={1,0};
         for(int j = i; j < i+(l>>1); ++j) {
            C u=a[j],v=a[j+(l>>1)];
            a[j]=u+v*w;
```

```
55                  a[j+(l>>1)]=u-v*w;
56                  w=w*wn;
57              }
58          }
59      }
60      if(op==-1) {
61          for(int i = 0; i < n; ++i) a[i].x/=n,a[i].y/=n;
62      }
63   }
64 }fft;
65 int a[maxn*2],b[maxn];
66 int main() {
67   int n,m;
68   scanf("%d%d", &n,&m);
69   for(int i = 0; i <= n; ++i) scanf("%d", a+i);
70   for(int i = 0; i <= m; ++i) scanf("%d", b+i);
71   fft.run(a,n+1,b,m+1);
72   for(int i = 0; i <= n+m; ++i) printf("%d%s", a[i],i==n+m?"\n":"
         ");
73   return 0;
74 }
```

3.5 快速数论变换

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int maxn=100050;
5  const int MOD=998244353;
6  struct NTT
7  {
8     ll Pow(ll x,ll n) {
9        ll ans=1,base=x%MOD;
10       while(n) {
11          if(n&1) ans=ans*base%MOD;
12          base=base*base%MOD;
```

```
13          n>>=1;
14       }
15       return ans;
16    }
17    ll getInv(ll x) {
18       return x==1?1:getInv(MOD%x)*(MOD-MOD/x)%MOD;
19    }
20    int t[maxn*2];
21    int _n,idx[maxn*4];
22    void rev(vector<int>&a,int n) {
23       if(_n!=n) {
24          _n=n;
25          for(int i = 0; i < n; ++i) {
26             idx[i]=(idx[i>>1]>>1)+(i&1)*(n>>1);
27          }
28       }
29       for(int i = 0; i < n; ++i) {
30          if(i<idx[i]) swap(a[i],a[idx[i]]);
31       }
32    }
33
34    // 计算数组a的点值表达并存在数组a中
35    void ntt(vector<int>&a,int n,int op) {
36       rev(a,n);
37       for(int i = 1; i < n; i<<=1) {
38          ll gn,g=1;
39          if(op==1) gn=Pow(3,(MOD-1)/i/2);
40          else gn=Pow(3,(MOD-1)/i/2*(i*2-1));
41          for(int j = 0; j < n; j+=(i<<1)) {
42             g=1;
43             for(int k = j; k < j+i; ++k) {
44                ll u=a[k],v=a[k+i];
45                a[k]=(u+g*v)%MOD;
46                a[k+i]=(u-g*v)%MOD;
47                g=g*gn%MOD;
48             }
```

```
49          }
50       }
51       if(op==-1) {
52          ll t=getInv(n);
53          for(int i = 0; i < n; ++i) {
54             a[i]=(ll)a[i]*t%MOD;
55          }
56       }
57    }
58
59    // a，b分别是两个多项式的系数数组，计算两个多项式乘积并存到a中
60    void mul(vector<int>&a,vector<int>&b) {
61       int n=1;
62       while(n<a.size()+b.size()-1) n<<=1;
63       a.resize(n),b.resize(n);
64       ntt(a,n,1);
65       ntt(b,n,1);
66       for(int i = 0; i < n; ++i) a[i]=(ll)a[i]*b[i]%MOD;
67       ntt(a,n,-1);
68       while(a.size()>1&&a.back()==0) a.pop_back();
69    }
70 }ntt;
```

3.6 求原根

```
1  // 调用getG返回n的第一个原根
2  // ！不支持非素数
3  namespace Primitive_root {
4     int phi(int n) {
5        int ans=n;
6        for(int i = 2; i*i <= n; ++i) {
7           if(n%i==0) {
8              ans=ans/i*(i-1);
9              while(n%i==0) n/=i;
10          }
11       }
```

```
12        if(n>1) ans=ans/n*(n-1);
13        return ans;
14    }
15    ll Pow(ll x, ll n, ll mod) {
16        ll ans=1,base=x%mod;
17        while(n) {
18            if(n&1) ans=ans*base%mod;
19            base=base*base%mod;
20            n>>=1;
21        }
22        return ans;
23    }
24    vector<int> fac;
25    bool test(int g,int eu,int mod) {
26        for(auto e:fac) {
27            if(Pow(g,eu/e,mod)==1) return false;
28        }
29        return true;
30    }
31    int getG(int n) {
32        int eu=phi(n),eu0=eu;
33        fac.clear();
34        for(int i = 2; i*i <= n; ++i) {
35            if(eu%i==0) {
36                fac.push_back(i);
37                while(eu%i==0) eu/=i;
38            }
39        }
40        if(eu>1) fac.push_back(eu);
41        eu=eu0;
42        int g=2;
43        while(!test(g,eu,n)) ++g;
44        return g;
45    }
46 }
```

3.7 BM 黑盒线代

```
// Calculating kth term of linear recurrence sequence
// Complexity: init O(n^2log) query O(n^2logk)
// Requirement: const LOG const MOD
// Input(constructor): vector<int> — first n terms
// vector<int> — transition function
// Output(calc(k)): int — the kth term mod MOD
// Example: In: {1, 1} {2, 1} an = 2an−1 + an−2
//    Out: calc(3) = 3, calc(10007) = 71480733 (MOD 1e9+7)

struct LinearRec {

   int n;
   vector<int> first, trans;
   vector<vector<int> > bin;

   vector<int> add(vector<int> &a, vector<int> &b) {
      vector<int> result(n * 2 + 1, 0);
      //You can apply constant optimization here to get a ~10x
         speedup
      for (int i = 0; i <= n; ++i) {
         for (int j = 0; j <= n; ++j) {
            if ((result[i + j] += (long long)a[i] * b[j] % MOD) >=
                MOD) {
               result[i + j] -= MOD;
            }
         }
      }
      for (int i = 2 * n; i > n; --i) {
         for (int j = 0; j < n; ++j) {
            if ((result[i - 1 - j] += (long long)result[i] *
               trans[j] % MOD) >= MOD) {
               result[i - 1 - j] -= MOD;
            }
         }
```

```
32          result[i] = 0;
33       }
34       result.erase(result.begin() + n + 1, result.end());
35       return result;
36    }
37
38    LinearRec(vector<int> &first, vector<int> &trans):first(first),
          trans(trans) {
39       n = first.size();
40       vector<int> a(n + 1, 0);
41       a[1] = 1;
42       bin.push_back(a);
43       for (int i = 1; i < LOG; ++i) {
44          bin.push_back(add(bin[i - 1], bin[i - 1]));
45       }
46    }
47
48    int calc(int k) {
49       vector<int> a(n + 1, 0);
50       a[0] = 1;
51       for (int i = 0; i < LOG; ++i) {
52          if (k >> i & 1) {
53             a = add(a, bin[i]);
54          }
55       }
56       int ret = 0;
57       for (int i = 0; i < n; ++i) {
58          if ((ret += (long long)a[i + 1] * first[i] % MOD) >= MOD)
               {
59             ret -= MOD;
60          }
61       }
62       return ret;
63    }
64 };
```

3.8 FWT

```
const ll MOD=1e9+7;
ll getInv(ll x) {
   return x==1?x:(MOD-MOD/x)*getInv(MOD%x)%MOD;
}
int N,inv2=getInv(2);
void FWT_or(int *a,int opt)
{
   for(int i=1;i<N;i<<=1)
      for(int p=i<<1,j=0;j<N;j+=p)
         for(int k=0;k<i;++k)
            if(opt==1)a[i+j+k]=(a[j+k]+a[i+j+k])%MOD;
            else a[i+j+k]=(a[i+j+k]+MOD-a[j+k])%MOD;
}
void FWT_and(int *a,int opt)
{
   for(int i=1;i<N;i<<=1)
      for(int p=i<<1,j=0;j<N;j+=p)
         for(int k=0;k<i;++k)
            if(opt==1)a[j+k]=(a[j+k]+a[i+j+k])%MOD;
            else a[j+k]=(a[j+k]+MOD-a[i+j+k])%MOD;
}
void FWT_xor(int *a,int opt)
{
   for(int i=1;i<N;i<<=1)
      for(int p=i<<1,j=0;j<N;j+=p)
         for(int k=0;k<i;++k)
         {
            int X=a[j+k],Y=a[i+j+k];
            a[j+k]=(X+Y)%MOD;a[i+j+k]=(X+MOD-Y)%MOD;
            if(opt==-1)a[j+k]=1ll*a[j+k]*inv2%MOD,a[i+j+k]=1ll*a[
               i+j+k]*inv2%MOD;
         }
}
```

3.9 Simpson 积分

```
// 调用asr(积分左端点，积分右端点，精度）返回积分结果
namespace Simpson_Integral {
    double F(double x) {
        return b/a*sqrt(a*a-x*x);
    }
    double simpson(double a,double b){
        double c = a + (b-a)/2;
        return (F(a)+4*F(c)+F(b))*(b-a)/6;
    }
    double asr(double a,double b,double eps,double A){
        double c = a + (b-a)/2;
        double L = simpson(a,c), R = simpson(c,b);
        if(fabs(L + R - A) <= 15*eps)return L + R + (L + R - A)
            /15.0;
        return asr(a,c,eps/2,L) + asr(c,b,eps/2,R);
    }
    double asr(double a,double b,double eps){
        return asr(a,b,eps,simpson(a,b));
    }
}
```

3.10 扩展欧拉定理

$$a^b \equiv \begin{cases} a^{b\%\phi(p)} & gcd(a,p)=1 \\ a^b & gcd(a,p)\neq 1, b<\phi(p) \\ a^{b\%\phi(p)+\phi(p)} & gcd(a,p)\neq 1, b\geq \phi(p) \end{cases} \tag{1}$$

3.11 杜教筛

$$
\begin{aligned}
(f*g)(n) &= \sum_{d|n} f(d)g(n/d) \\
\sum_{i=1}^{n}(f*g)(i) &= \sum_{i=1}^{n}\sum_{d|i} f(d)g(i/d) \\
&= \sum_{d=1}^{n} g(d) \sum_{i=1}^{\lfloor n/d \rfloor} f(i) \\
&= \sum_{d=1}^{n} g(d)s(\lfloor n/d \rfloor) \\
&= \sum_{d=2}^{n} g(d)s(\lfloor n/d \rfloor) + g(1)s(n) \\
g(1)s(n) &= \sum_{i=1}^{n}(f*g)(i) - \sum_{d=2}^{n} g(d)s(n/d)
\end{aligned}
$$

```
// 计算欧拉函数前缀和模板
// time complexity: O(n^(2/3))
// 需预处理n^(2/3)内的所有答案
unorderd_map<ll,ll> mp;
ll solve(ll n) {
   if(n<N) return phi_sum[n];
   if(mp.count(n)) return mp[n];
   ll nn=n%MOD;
   ll ans=nn*(nn+1)%MOD*inv2%MOD;
   for(ll i = 2; i <= n; ++i) {
      ll t=n/i,nxt=n/t;
      if(nxt>n) nxt=n;
      ans-=(ll)(nxt-i+1)%MOD*solve(t)%MOD;
      i=nxt;
   }
   return mp[n]=ans;
}
```

3.12 Pell 方程

```
// time complexity O(log(n))
inline bool solve(ll n,ll &p,ll &q){
```

```
    ll N,p1,p2,q1,q2,a0,a1,a2,g1,g2,h1,h2;
    g1=q2=p1=0;
    h1=q1=p2=1;
    a0=a1=sqrt(n*1.0);
    ll ans = a0*a0;
    N = n;
    if(ans==N) return false;
    while(1){
        g2 = a1*h1-g1;
        h2 = (N-g2*g2)/h1;
        a2 = (g2+a0)/h2;
        p = a1*p2+p1;
        q = a1*q2+q1;
        if(p*p==(N*q*q+1)) break;
        g1 = g2;
        h1 = h2;
        a1 = a2;
        p1 = p2;
        p2 = p;
        q1 = q2;
        q2 = q;
    }
    return true;
}//nth xn + yn*sqrt(n) = (x1+y1*sqrt(n)) ^ n
```

3.13 莫比乌斯反演

$$f(n)=\sum_{d|n}g(d)\Longleftrightarrow g(n)=\sum_{d|n}\mu(d)f(\frac{n}{d})$$

3.14 常用素数

```
1009 1013
10007 10009
100003 100019
1000003 1000033
10000019 10000079
```

```
100000007 100000037
10000000019 10000000033
1000000000039 1000000000061
100000000000031 100000000000067
10000000000000061 10000000000000069
100000000000000003 100000000000000009
```

3.15 快速取模乘

```
// 高端版，速度很快
#include <bits/stdc++.h>
typedef long long ll;
typedef unsigned long long ull;
typedef __uint128_t u128;
int T, k;
ull A0, A1, M0, M1, C, M;
namespace Modll { // 加减乘请务必严格按照模版中的写法来写，不要有任何魔改
   static ull mod, inv, r2;
   struct mll { // 开始进行运算之前务必调用一下set_mod函数
      ull val;
      mll(ull tmp = 0) : val(reduce(u128(tmp) * r2)) {}
      static void set_mod(ull m) {
         mod = m;
         assert(mod & 1);
         inv = m;
         for (int i = 0; i < 5; i++) inv *= 2 - inv * m;
         r2 = -u128(m) % m;
      }
      static ull reduce(u128 x) {
         ull y = ull(x >> 64) - ull((u128(ull(x) * inv) * mod) >>
             64);
         return ll(y) < 0 ? y + mod : y;
      }
      mll &operator += (mll tmp) {
         val += tmp.val - mod;
         if (ll(val) < 0) val += mod;
```

```
27          return *this;
28       }
29       mll operator + (mll tmp) const { return mll(*this) += tmp; }
30       mll &operator -= (mll tmp) {
31          val -= tmp.val;
32          if (ll(val) < 0) val += mod;
33          return *this;
34       }
35       mll operator - (mll tmp) const { return mll(*this) -= tmp;
            }
36       mll &operator *= (mll tmp) {
37          val = reduce(u128(val) * tmp.val);
38          return *this;
39       }
40       mll operator * (mll tmp) const { return mll(*this) *= tmp; }
41       ull value() const { return reduce(val); }
42    };
43 } using Modll::mll;
44 int main() {
45 #ifdef LOCAL
46    freopen("../in", "r", stdin);
47 #endif
48    for (scanf("%d", &T); T; T--) {
49       std::cin >> A0 >> A1 >> M0 >> M1 >> C >> M >> k;
50       mll::set_mod(M);
51       mll a0 = A0, a1 = A1, m0 = M0, m1 = M1, c = C, ans = a0 * a1
            , tmp;
52       for (int i = 2; i <= k; i++) {
53          tmp = m0 * a1 + m1 * a0 + c;
54          ans *= tmp;
55          a0 = a1, a1 = tmp;
56       }
57       printf("%llu\n", ans.value());
58    }
59 }
```

3.16 多项式算法 (逆元,exp,ln)

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll MOD=998244353;
const int maxn=1000050;
ll getInv(ll x) {
   return x==1?x:getInv(MOD%x)*(MOD-MOD/x)%MOD;
}
ll Pow(ll x,ll n) {
   ll ans=1,base=x%MOD;
   while(n) {
      if(n&1) ans=ans*base%MOD;
      base=base*base%MOD;
      n>>=1;
   }
   return ans;
}
vector<int> idx;
void rev(vector<int> &a,int n) {
   if(idx.size()!=n) {
      idx.assign(n,0);
      for(int i = 0; i < n; ++i) {
         idx[i]=(idx[i>>1]>>1)+(i&1)*(n>>1);
      }
   }
   for(int i = 0; i < n; ++i) {
      if(i<idx[i]) swap(a[i],a[idx[i]]);
   }
}
void ntt(vector<int>& a,int n,int op) {
   rev(a,n);
   for(int l = 2; l <= n; l<<=1) {
      ll wn=Pow(3,(MOD-1)/l);
      if(op==-1) wn=getInv(wn);
```

```
35        for(int i = 0; i < n; i+=l) {
36           ll w=1;
37           for(int j = i; j < i+(l>>1); ++j) {
38              ll u=a[j],v=a[j+(l>>1)];
39              a[j]=(u+v*w)%MOD;
40              a[j+(l>>1)]=(u-v*w)%MOD;
41              w=w*wn%MOD;
42           }
43        }
44     }
45     if(op==-1) {
46        ll t=getInv(n);
47        for(int i = 0; i < n; ++i) a[i]=(ll)a[i]*t%MOD;
48     }
49  }
50  void deb(vector<int>&a) {
51     for(auto &e:a) cout<<(e+MOD)%MOD<<" ";
52     cout<<endl;
53  }
54  int inv[maxn*2];
55  vector<int> polydiff(vector<int>&a,int n) {
56     vector<int> ans(n);
57     for(int i = 0; i < n-1; ++i) {
58        ans[i]=(ll)(i+1)*a[i+1]%MOD;
59     }
60     return ans;
61  }
62  vector<int> polyInv(vector<int> &a,int n) {
63     vector<int> ans(1,getInv(a[0]));
64     for(int l = 2; l <= n; l<<=1) {
65        vector<int> b(a.begin(),a.begin()+l);
66        ans.resize(l*2),b.resize(l*2);
67        ntt(ans,l*2,1),ntt(b,l*2,1);
68        for(int i = 0; i < l*2; ++i) ans[i]=(ll)(2ll-(ll)b[i]*ans[i
             ]%MOD)*ans[i]%MOD;
69        ntt(ans,l*2,-1);
```

```
70        for(int i = l; i < l*2; ++i) ans[i]=0;
71     }
72     ans.resize(n);
73     return ans;
74  }
75  vector<int> polyln(vector<int>& a,int n) {
76     vector<int> da=polydiff(a,n),inva=polyInv(a,n);
77     da.resize(n*2),inva.resize(n*2);
78     ntt(da,n*2,1),ntt(inva,n*2,1);
79     for(int i = 0; i < n*2; ++i) da[i]=(ll)da[i]*inva[i]%MOD;
80     ntt(da,n*2,-1);
81     vector<int> ans(n);
82     for(int i = 1; i < n; ++i) ans[i]=(ll)da[i-1]*inv[i]%MOD;
83     return ans;
84  }
85  vector<int> polyExp(vector<int>& a,int n) {
86     vector<int> ans(1,1);
87     ans.resize(2);
88     for(int l = 2; l <= n; l<<=1) {
89        vector<int> lnf0=polyln(ans,l),b(a.begin(),a.begin()+l);
90        ans.resize(l*2),lnf0.resize(l*2),b.resize(l*2);
91        ntt(ans,l*2,1),ntt(lnf0,l*2,1),ntt(b,l*2,1);
92        for(int i = 0; i < l*2; ++i) ans[i]=(ll)ans[i]*(1ll-(ll)
             lnf0[i]+b[i])%MOD;
93        ntt(ans,l*2,-1);
94        for(int i = l; i < l*2; ++i) ans[i]=0;
95     }
96     ans.resize(n);
97     return ans;
98  }
99  void init() {
100    inv[1]=1;
101    for(int i = 2; i < maxn*2; ++i) inv[i]=(ll)inv[MOD%i]*(MOD-MOD
         /i)%MOD;
102 }
103 int a[maxn];
```

```
vector<int> get(int l,int r) {
   if(l==r) return vector<int>{1,a[l]};
   int mid=(l+r)>>1;
   vector<int> a=get(l,mid),b=get(mid+1,r);
   int n=1;
   while(n<=(int)a.size()+(int)b.size()-2) n<<=1;
   a.resize(n),b.resize(n);
   ntt(a,n,1),ntt(b,n,1);
   for(int i = 0; i < n; ++i) a[i]=(ll)a[i]*b[i]%MOD;
   ntt(a,n,-1);
   while(a.size()>1&&a.back()==0) a.pop_back();
   return a;
}
int main() {
   init();
   int n,m;
   scanf("%d%d", &n,&m);
   for(int i = 1; i <= n; ++i) scanf("%d", a+i);
   int N=1;
   while(N<=m) N<<=1;
   vector<int> g(N);
   for(int i = 0; i < N; ++i) {
      if(i==0) g[i]=1;
      else g[i]=(ll)g[i-1]*inv[i+1]%MOD;
   }
   g=polyln(g,N);
   vector<int> f=get(1,n);
   f.resize(N);
   f=polyln(f,N);
   for(int i = 1; i < N; ++i) {
      ll t=-(ll)f[i]*i%MOD;
      if(i&1) t=-t;
      g[i]=(ll)g[i]*t%MOD;
   }
   g=polyExp(g,N);
   ll ans=g[m];
```

```
140     for(int i = 1; i <= m; ++i) ans=ans*i%MOD;
141     ans=(ans+MOD)%MOD;
142     cout<<ans<<endl;
143     return 0;
144 }
```

3.17 gcd

```
1  ll gcd(ll a, ll b) {
2     if (a < b) return gcd(b, a);
3     if (b == 0) return a;
4     if (!(a & 1) && !(b & 1)) return 2 * gcd(a >> 1, b >> 1);
5     if (!(a & 1)) return gcd(a >> 1, b);
6     if (!(b & 1)) return gcd(a, b >> 1);
7     return gcd((a + b) >> 1, (a - b) >> 1);
8  }
9
10 ll gcd(ll p, ll q) { return q ? gcd(q, p % q) : p; }
11
12 inline ll gcd(ll p, ll q, ll tmp = 0) {
13    while (q) tmp = p % q, p = q, q = tmp;
14    return p;
15 }
16
17 ll gcd(ll a, ll b) {
18    while (b ^= a ^= b ^= a %= b);
19    return a;
20 }
```

3.18 Eratosthenes 素数筛

```
1  #include <bits/stdc++.h>
2  const int maxn = 1007;
3  int minimal_prime_factor[maxn], prime[maxn];
4  int primes(int upper) {
5     // i的最小质因子为minimal_prime_factor[i]
```

```
    memset(minimal_prime_factor, 0, sizeof(minimal_prime_factor));
    int cnt_prime = 0; // 质数数量
    for (int i = 2; i <= upper; i++) {
        if (minimal_prime_factor[i] == 0) { // i是质数
            minimal_prime_factor[i] = i;
            prime[++cnt_prime] = i;
        }
        // 给当前的数i乘上一个质因子
        int tmp = upper / i;
        for (int j = 1; j <= cnt_prime; j++) {
            // i有比prime[j]更小的因子，或者超出upper的范围
            if (prime[j] > minimal_prime_factor[i] || prime[j] > tmp)
                break;
            // prime[j]是合数i * prime[j]的最小质因子
            minimal_prime_factor[i * prime[j]] = prime[j];
        }
    }
}
int main() {
    int cnt_prime = primes(1000);
    for (int i = 1; i <= cnt_prime; i++) printf("%d ", prime[i]);
    return 0;
}
```

3.19 扩展 BSGS

```
#include <bits/stdc++.h>
typedef long long ll;
using namespace std;
map<ll, ll> mp;
ll power(ll p, ll q, ll c) {
    ll ans = 1;
    while (q) {
        if (q & 1) ans = ans * p % c;
        q >>= 1;
        p = p * p % c;
```

```
    }
    return ans;
}

ll gcd(ll a, ll b) { return (b) ? gcd(b, a % b) : a; }

// 求解使得a^x == b (Mod c)的最小非负整数x

inline int EX_BSGS(ll a, ll b, ll p) {
    if (p == 1) return 0 * puts("0");
    a %= p, b %= p;
    if (!a && !b) return 0 * puts("1");
    if (b == 1) return 0 * puts("0");
    if (!a) return 0 * puts("No Solution");
    ll d, step = 0, k = 1;
    while ((d = gcd(a, p)) != 1) {
        if (b % d) return 0 * puts("No Solution");
        step++;
        p /= d, b /= d, k = k * a / d % p;
        if (b == k) return 0 * printf("%lld\n", step);
    }
    ll m = ceil(sqrt(p)), t = b;
    mp.clear();
    for (ll i = 0; i <= m; i++) {
        mp[t] = i;
        t = t * a % p;
    }
    ll A = power(a, m, p);
    t = k * A % p;
    for (ll i = 1; i <= m; i++) {
        ll ans = t;
        t = t * A % p;
        if (mp.find(ans) != mp.end()) {
            ans = i * m - mp[ans] + step;
            return 0 * printf("%lld\n", ans);
        }
```

```
47     }
48     puts("No Solution");
49 }
50 
51 int main() {
52     ll a, b, c;
53     scanf("%lld%lld%lld", &a, &b, &c);
54     EX_BSGS(a, b, c);
55     return 0;
56 }
```

3.20 欧拉函数

```
1  //欧拉函数离线
2  void Init () {
3      for (int i=0; i<maxn; i++) eular[i] = i;
4      for (int i=2; i<maxn; i++)
5          if (eular[i] == i) {
6              eular[i] = eular[i] / i * (i - 1);
7              for (int j=i+i; j<maxn; j+=i) eular[j] = eular[j] / i * (
                   i - 1);
8          }
9  }
10 
11 //欧拉函数在线
12 int eular(int n) {
13     int number = 1;
14     for(int i=2; i*i <= n; i++)
15         if(n%i == 0){
16             n /= i, number *= (i-1);
17             while(n%i == 0) n /= i, number *= i;
18         }
19     return n>1?number*(n-1):number;
20 }
21 
22 //欧拉函数（小于n的正整数中与n互质的数的数目）
```

3.21 矩阵快速幂

```
const int maxn = 107;
int Matrixsize = 2, mod = int(1e9)+7;
struct Matrix {
   int m[maxn][maxn];
   Matrix(int x = 0) {
      memset(m, 0, sizeof m);
      if (x) for(int i = 0; i < Matrixsize; i++) m[i][i] = 1;
   }
   Matrix operator * (const Matrix tmp) const {
      Matrix ret = 0;
      for(int i=0 ; i<Matrixsize ; i++)
         for(int k=0 ; k<Matrixsize ; k++)
            if(m[i][k]) for(int j=0 ; j<Matrixsize ; j++)
               ret.m[i][j] = ((1ll * m[i][k] * tmp.m[k][j]) % mod
                  + ret.m[i][j]) % mod;
      return ret;
   }
   Matrix qpow(long long q) {
      Matrix ret = 1, tmp = *this;
      for( ; q ; q>>=1) {
         if(q & 1) ret = ret * tmp;
         tmp = tmp * tmp;
      } return ret;
   }
};


/**
 *矩阵快速幂成员函数版
 **/
const int maxn = 107;
int Matrixsize = 2, mod = int(1e9)+7;
struct Matrix {
   int m[maxn][maxn];
```

```
34    Matrix(int i = 0) {
35       memset(m, 0, sizeof m);
36       if (i == 1) for (int I = 0; I < Matrixsize; I++) m[I][I] =
            1;
37    }
38    Matrix operator * (const Matrix tmp) const {
39       Matrix ret;
40       long long x;
41       for(int i=0 ; i<Matrixsize ; i++) {
42          for(int j=0 ; j<Matrixsize ; j++) {
43             x=0;
44             for(int k=0 ; k<Matrixsize ; k++) {
45                x+=((long long)m[i][k] * tmp.m[k][j]) % mod;
46             }
47             ret.m[i][j] = int(x % mod);
48          }
49       }
50       return ret;
51    }
52    Matrix operator+(const Matrix b) const {
53       Matrix a = *this;
54       for (int i = 0; i < Matrixsize; i++)
55          for (int j = 0; j < Matrixsize; j++)
56             a.m[i][j] = (a.m[i][j] + b.m[i][j]) % mod;
57       return a;
58    }
59    Matrix operator-(const Matrix b) const {
60       Matrix a = *this;
61       for (int i = 0; i < Matrixsize; i++)
62          for (int j = 0; j < Matrixsize; j++)
63             a.m[i][j] = (a.m[i][j] - b.m[i][j]) % mod;
64       return a;
65    }
66    Matrix operator-() const {
67       Matrix tmp = *this;
68       for (int i = 0; i < Matrixsize; i++)
```

```
69          for (int j = 0; j < Matrixsize; j++)
70              tmp.m[i][j] = -tmp.m[i][j];
71      return tmp;
72  }
73  void operator += (const Matrix tmp) const {
74      for (int i = 0; i < Matrixsize; i++)
75          for (int j = 0; j < Matrixsize; j++)
76              m[i][j] = (m[i][j] + tmp.m[i][j]) % mod;
77  }
78  bool operator != (const Matrix tmp) const {
79      for (int i = 0; i < Matrixsize; i++)
80          for (int j = 0; j < Matrixsize; j++)
81              if(m[i][j] != tmp.m[i][j]) return true;
82      return false;
83  }
84  bool operator == (const Matrix tmp) const {
85      for (int i = 0; i < Matrixsize; i++)
86          for (int j = 0; j < Matrixsize; j++)
87              if(m[i][j] != tmp.m[i][j]) return false;
88      return true;
89  }
90  Matrix qpow(long long n) {
91      Matrix ret = 1, tmp = *this;
92      while (n > 0) {
93          if (bool(n & 1)) ret = ret * tmp;
94          tmp = tmp * tmp;
95          n >>= 1;
96      }
97      return ret;
98  }
99  void show() {
100     Matrix tmp = *this;
101     for(int i=0 ; i<Matrixsize ; i++) {
102         for(int j=0 ; j<Matrixsize-1 ; j++) cout << tmp.m[i][j]
              << ' ';
103         cout << tmp.m[i][Matrixsize-1] << endl;
```

```
104         }
105         cout << endl;
106     }
107 };
108 
109 /**
110  *矩阵快速幂原版
111  **/
112 const int maxn = 110,mod = 1e9+7;
113 int n;
114 
115 struct mat {
116     int m[maxn][maxn];
117 };
118 mat unit;//unit 是单位矩阵，任何矩阵乘以单位矩阵都是这个矩阵本身
119 
120 mat operator * (mat a, mat b) {
121     mat ret;
122     long long x;
123     for(int i=0 ; i<n ; i++) {
124         for(int j=0 ; j<n ; j++) {
125             x=0;
126             for(int k=0 ; k<n ; k++) {
127                 x+=((long long)a.m[i][k] * b.m[j][k]) % mod;
128             }
129             ret.m[i][j] = x%mod;
130         }
131     }
132     return ret;
133 }
134 
135 void init_unit() {
136     for(int i=0 ; i<maxn ; i++) unit.m[i][i] = 1;
137     return ;
138 };
139 
```

```
mat qpow_mat(mat a, long long n) {
   mat ret = unit;
   while(n) {
      if(n&1) ret = ret*a;
      a = a*a;
      n >>= 1;
   }
   return ret;
}
```

3.22 组合数

```
/*
在线算法
*/

// O(n)在线求组合数

const int maxn = int(2e5) + 7, mod = int(1e9) + 7;
namespace combination {
   typedef long long ll;
   ll fac[maxn], inv[maxn];
   bool flag = false;
   ll pow(ll p, ll q) {
      ll ret = 1;
      while (q) {
         if (q & 1) ret = ret * p % mod;
         p = p * p % mod;
         q >>= 1;
      }
      return ret;
   }
   void init(int upper) {
      fac[0] = 1;
      for (int i = 1; i <= upper; i++) fac[i] = fac[i - 1] * i %
         mod;
```

```
24        inv[upper] = pow(fac[upper], mod - 2);
25        for (int i = upper - 1; i >= 0; i--) inv[i] = inv[i + 1] *
              (i + 1) % mod;
26    }
27    ll C(ll n, ll m) {
28        if (!flag) init(maxn - 7), flag = true;
29        if (n == m || m == 0) return 1ll;
30        return fac[n] * inv[m] % mod * inv[n - m] % mod;
31    }
32 }
33 using combination::C;
34
35 // Lucas
36
37 ll mod;
38 ll power_mod(ll p, ll q) {
39    ll ans = 1;
40    p %= mod;
41    while (q) {
42        if (q & 1) ans = ans * p % mod;
43        q >>= 1, p = p * p % mod;
44    }
45    return ans;
46 }
47
48 ll C(ll n, ll m) {
49    if (m > n) return 0;
50    ll ans = 1;
51    for (int i = 1; i <= m; i++) {
52        ll a = (n + i - m) % mod;
53        ll b = i % mod;
54        ans = ans * (a * power_mod(b, mod - 2) % mod) % mod;
55    }
56    return ans;
57 }
58
```

```
ll Lucas(ll n, ll m) {
   if (m == 0 || n == m) return 1;
   return C(n % mod, m % mod) * Lucas(n / mod, m / mod) % mod;
}

/*
扩展Lucas
*/

long long POW(long long a,long long b,long long mod) {
   long long ans = 1;
   while (b) {
      if (b & 1) ans = ans * a % mod;
      a = a * a % mod;
      b >>= 1;
   }
   return ans;
}

long long POW(long long a,long long b) {
   long long ans = 1;
   while (b) {
      if (b & 1) ans = ans * a;
      a = a * a;
      b >>= 1;
   }
   return ans;
}


long long exGcd(long long a,long long b,long long &x,long long &y)
    {
   long long t, d;
   if (!b) {
      x = 1;
      y = 0;
```

```
94          return a;
95       }
96       d = exGcd(b, a % b, x, y);
97       t = x;
98       x = y;
99       y = t - a / b * y;
100      return d;
101   }
102
103   bool modular(long long a[],long long m[],long long k) {
104      long long d, t, c, x, y, i;
105      for (i = 2; i <= k; i++) {
106         d = exGcd(m[1], m[i], x, y);
107         c = a[i] - a[1];
108         if (c % d) return false;
109         t = m[i] / d;
110         x = (c / d * x % t + t) % t;
111         a[1] = m[1] * x + a[1];
112         m[1] = m[1] * m[i] / d;
113      }
114      return true;
115   }
116
117
118
119   long long reverse(long long a,long long b) {
120      long long x, y;
121      exGcd(a, b, x, y);
122      return (x % b + b) % b;
123   }
124
125   long long C(long long n,long long m,long long mod) {
126      if (m > n) return 0;
127      long long ans = 1, i, a, b;
128      for (i = 1; i <= m; i++) {
129         a = (n + 1 - i) % mod;
```

```
130         b = reverse(i % mod, mod);
131         ans = ans * a % mod * b % mod;
132     }
133     return ans;
134 }
135 
136 long long C1(long long n,long long m,long long mod) {
137     if (m == 0) return 1;
138     return C(n % mod, m % mod, mod) * C1(n / mod, m / mod, mod) %
          mod;
139 }
140 
141 long long cal(long long n,long long p,long long t) {
142     if (!n) return 1;
143     long long x = POW(p, t), i, y = n / x, temp = 1;
144     for (i = 1; i <= x; i++) if (i % p) temp = temp * i % x;
145     long long ans = POW(temp, y, x);
146     for (i = y * x + 1; i <= n; i++) if (i % p) ans = ans * i % x;
147     return ans * cal(n / p, p, t) % x;
148 }
149 
150 long long C2(long long n,long long m,long long p,long long t) {
151     long long x = POW(p, t);
152     long long a, b, c, ap = 0, bp = 0, cp = 0, temp;
153     for (temp = n; temp; temp /= p) ap += temp / p;
154     for (temp = m; temp; temp /= p) bp += temp / p;
155     for (temp = n - m; temp; temp /= p) cp += temp / p;
156     ap = ap - bp - cp;
157     long long ans = POW(p, ap, x);
158     a = cal(n, p, t);
159     b = cal(m, p, t);
160     c = cal(n - m, p, t);
161     ans = ans * a % x * reverse(b, x) % x * reverse(c, x) % x;
162     return ans;
163 }
164 
```

```
//计算C(n,m)%mod
long long Lucas(long long n,long long m,long long mod) {
   long long i, t, cnt = 0;
   long long A[205], M[205];
   for (i = 2; i * i <= mod; i++)
      if (mod % i == 0) {
         t = 0;
         while (mod % i == 0) {
            t++;
            mod /= i;
         }
         M[++cnt] = POW(i, t);
         if (t == 1) A[cnt] = C1(n, m, i);
         else A[cnt] = C2(n, m, i, t);
      }
   if (mod > 1) {
      M[++cnt] = mod;
      A[cnt] = C1(n, m, mod);
   }
   modular(A, M, cnt);
   return A[1];
}

/*
离线打表
*/

long long C[maxn][maxn];

void get_C() {
   C[0][0] = 1;
   for(int i = 1;i <= maxn; i++) {
      C[i][0] = 1;
      for(int j = 1; j <= i; j++)
         C[i][j] = C[i-1][j] + C[i-1][j-1];
   }
```

```
}
```

3.23 长整型无脑取模乘

```
ll qmul(ll x, ll y, ll mod) {
    x %= mod, y %= mod;
    ll ans = (x * y - (ll)((long double)x / mod * y + 1e-3) * mod)
        ;
    ans = (ans % mod + mod) % mod;
    return ans;
}
```

4 数据结构

4.1 KD-Tree

```
#include <bits/stdc++.h>
const int MAXN=5e4+10;
#define ll long long
const ll INF=1e5;
using namespace std;
int root,pos,k,n;
typedef struct node{
    int p[6],c[2];ll maxx[6],minn[6];
    friend bool operator <(node aa,node bb){
        if(aa.p[pos]!=bb.p[pos])return aa.p[pos]<bb.p[pos];
        for(int i = 1;i<=k;i++) if(aa.p[i]!=bb.p[i]) return aa.p[i]<
            bb.p[i];
    }
}node;
node a[MAXN<<2];
void up(int x,int y){
    for(int i=1;i<=k;i++)a[x].maxx[i]=max(a[x].maxx[i],a[y].maxx[i
        ]),a[x].minn[i]=min(a[x].minn[i],a[y].minn[i]);
}
int build(int l,int r,int now){
    if(l>r)return 0;
```

```
20    int mid=(l+r)>>1;
21    pos=now,nth_element(a+l,a+mid,a+r+1);
22    for(int i=1;i<=k;i++)a[mid].maxx[i]=a[mid].minn[i]=a[mid].p[i];
23    a[mid].c[0]=a[mid].c[1]=0;
24    if(l<mid)a[mid].c[0]=build(l,mid-1,now%k+1),up(mid,a[mid].c
          [0]);
25    if(r>mid)a[mid].c[1]=build(mid+1,r,now%k+1),up(mid,a[mid].c[1])
          ;
26    return mid;
27 }
28 typedef struct Tmp{
29    int p[6];ll dis;
30    friend bool operator<(Tmp aa,Tmp bb){return aa.dis<bb.dis;}
31 }Tmp;
32 ll dist(node aa,node bb){
33    ll ans=0;
34    for(int i=1;i<=k;i++)ans+=1LL*(aa.p[i]-bb.p[i])*(aa.p[i]-bb.p[
          i]);
35    return ans;
36 }
37 priority_queue<Tmp>que;
38 node t;
39 stack<Tmp>s;
40 ll get_ans(node aa,node bb){
41    ll ans=0;
42    for(int i=1;i<=k;i++){
43       ans+=min(1LL*(aa.minn[i]-bb.p[i])*(aa.minn[i]-bb.p[i]),1LL
             *(bb.p[i]-aa.maxx[i])*(bb.p[i]-aa.maxx[i]));
44    }
45    return ans;
46 }
47 void query(int x,int now){
48    if(!x)return ;
49    ll res=dist(a[x],t);
50    if(res<(que.top()).dis){
51       que.pop();
```

```
52        Tmp tt;tt.dis=res;
53        for(int i=1;i<=k;i++)tt.p[i]=a[x].p[i];
54        que.push(tt);
55     }
56      ll tt=t.p[now]-a[x].p[now];
57      if(tt<=0){
58        query(a[x].c[0],now%k+1);
59         if(que.top().dis>tt*tt)
60            query(a[x].c[1],now%k+1);
61      }
62      else{
63        query(a[x].c[1],now%k+1);
64         if(que.top().dis>tt*tt)
65            query(a[x].c[0],now%k+1);
66      }
67  }
68  int main(){
69     while(scanf("%d%d",&n,&k)!=EOF){
70        for(int i = 1;i<=k;i++) a[0].minn[i] = INF,a[0].maxx[i] = -
             INF;
71        for(int i = 1;i<=n;i++) for(int j = 1;j<=k;j++) scanf("%d",&
             a[i].p[j]);
72        root=build(1,n,1);
73        int q;
74        scanf("%d",&q);
75        while(q--){
76           for(int i = 1;i<=k;i++) scanf("%d",&t.p[i]);
77           int K;
78           scanf("%d",&K);
79           Tmp t1;t1.dis=1e15;
80           for(int i = 1;i<=K;i++) que.push(t1);
81           query(root,1);
82           while(!que.empty())s.push(que.top()),que.pop();
83           printf("the closest %d points are:\n",K);
84           while(!s.empty()){
85              Tmp tt=s.top();
```

```
            for(int i=1;i<=k;i++){
               if(i!=k)printf("%d ",tt.p[i]);
               else printf("%d\n",tt.p[i]);
            }
            s.pop();
         }
      }
   }
   return 0;
}
```

4.2 李超树

```
#include<bits/stdc++.h>
//O(nlogn)
using namespace std;
const int maxm = 1e5+10;
double s[maxm],p[maxm];
int tree[maxm<<2];
double f(int num,int x){
   return p[num]*(x-1)+s[num];
}
void add(int ind,int l = 1,int r = maxm,int buf = 1){
   if(l==r){
      if(f(ind,l) > f(tree[ind],l)) tree[buf] = ind;
      return ;
   }
   int mid = (l+r)>>1;
   if(p[tree[buf]]<p[ind]){// the pre line's k is lower than now
      if(f(ind,mid)>f(tree[buf],mid)){
         add(tree[buf],l,mid,buf<<1);
         tree[buf] = ind;
      }else add(ind,mid+1,r,buf<<1|1);
   }
   if(p[tree[buf]]>p[ind]){
      if(f(ind,mid)>f(tree[buf],mid)){
```

```
24          add(tree[buf],mid+1,r,buf<<1|1);
25          tree[buf] = ind;
26       }else add(ind,l,mid,buf<<1);
27    }
28 }
29 double query(int pos,int l = 1,int r = maxm,int buf = 1){
30    if(l==r) return f(tree[buf],pos);
31    int mid = (l+r)>>1;
32    if(pos<=mid) return max(query(pos,l,mid,buf<<1),f(tree[buf],pos
         ));
33    else return max(query(pos,mid+1,r,buf<<1|1),f(tree[buf],pos));
34 }
35 int main(){
36    int n;
37    scanf("%d",&n);
38    int cnt = 1;
39    while(n--){
40       char nouse[10];
41       scanf("%s",nouse);
42       if(nouse[0]=='P'){
43          scanf("%lf%lf",s+cnt,p+cnt);//x = 1,y = s[cnt],k = p[cnt]
44          add(cnt);
45          cnt++;
46       }else{
47          int xxx;
48          scanf("%d",&xxx);
49          printf("%d\n",query(xxx));//x = xxx,y = query(xxx)
50       }
51    }
52    return 0;
53 }
```

4.3 左偏树 & 优先队列

```
1 template<typename T, typename _Compare=std::less<T> >
2 class leftist_tree {
```

```
private:
   struct node {
      T data;
      int deep;
      node *l, *r;

      node(const T &d) : data(d), deep(1), l(0), r(0) {}
   } *root;

   int _size;
   _Compare cmp;

   inline int deep(node *o) { return o ? o->deep : 0; }

   node *Merge(node *a, node *b) {
      if (!a || !b)return a ? a : b;
      if (cmp(a->data, b->data)) {
         std::swap(a,b);
      }
      a->r = Merge(a->r, b);
      if (deep(a->l) < deep(a->r)) {
         std::swap(a->l, a->r);
      }
      a->deep = deep(a->l) + 1;
      return a;
   }

   void _clear(node *&o) {
      if (o)_clear(o->l), _clear(o->r), delete o;
   }

public:
   leftist_tree() : root(0), _size(0) {}

   ~leftist_tree() { _clear(root); }

```

```
39    inline void clear() {
40       _clear(root);
41       root = 0;
42       _size = 0;
43    }
44
45    inline void merge(leftist_tree &o) {
46       root = Merge(root, o.root);
47       o.root = 0;
48       _size += o._size;
49       o._size = 0;
50    }
51
52    inline void swap(leftist_tree &o) {
53       node *t = root;
54       root = o.root;
55       o.root = t;
56       int st = _size;
57       _size = o._size;
58       o._size = st;
59    }
60
61    inline void push(const T &data) {
62       _size++;
63       root = Merge(root, new node(data));
64    }
65
66    inline void pop() {
67       if (_size)_size--;
68       node *tmd = Merge(root->l, root->r);
69       delete root;
70       root = tmd;
71    }
72
73    inline const T &top() { return root->data; }
74
```

```
75      inline int size() { return _size; }
76
77      inline bool empty() { return !_size; }
78  };
```

4.4 树链剖分

(by shui)

```
1   //bzoj 1036
2   #include <iostream>
3   #include <algorithm>
4   #include <set>
5   #include <string>
6   #include <vector>
7   #include <queue>
8   #include <map>
9   #include <stack>
10  #include <list>
11  #include <iomanip>
12  #include <functional>
13  #include <sstream>
14  #include <cstdio>
15  #include <cstring>
16  #include <cmath>
17  #include <ctime>
18  #include <cctype>
19
20  #define read read()
21  #define edl putchar('\n')
22  #define clr(a, b) memset(a,b,sizeof(a))
23  #define rep(i,a,b) for(int i = a ; i<=b ; i++)
24
25  using namespace std;
26
27  #define lson (i<<1)
28  #define rson (i<<1|1)
```

```
29
30 const int maxn = 30007, INF = 0x3f3f3f3f;
31
32 struct E {
33    int to, next;
34 }edge[maxn<<1];
35
36 int head[maxn], top[maxn], pre[maxn], deep[maxn], sz[maxn], p[maxn
      ], fp[maxn], son[maxn], num[maxn];
37 int pos, cnt_edge;
38
39 void init() {
40    cnt_edge = pos = 0;
41    clr(head, -1);
42    clr(son, -1);
43 }
44
45 void addedge(int u, int v) {
46    edge[++cnt_edge] = {v, head[u]};
47    head[u] = cnt_edge;
48 }
49
50 void dfs1(int u, int pre_, int d) {
51    deep[u] = d;
52    pre[u] = pre_;
53    sz[u] = 1;
54    for(int i = head[u] ; ~i ; i = edge[i].next) {
55       int v = edge[i].to;
56       if(v != pre_) {
57          dfs1(v, u, d+1);
58          sz[u] += sz[v];
59          if(son[u] == -1 || sz[v] > sz[son[u]]) {
60             son[u] = v;
61          }
62       }
63    }
```

```
}

void dfs2(int u, int sp) { // ????????????????????
   top[u] = sp;
   p[u] = pos++;
   fp[p[u]] = u;
   if(son[u] == -1) return ;
   dfs2(son[u], sp);
   for(int i = head[u] ; ~i ; i = edge[i].next) {
      int v = edge[i].to;
      if(v != son[u] && v != pre[u]) dfs2(v, v);
   }
}

struct {
   int l, r, sum, Max;
}node[maxn<<2];

void pushup(int i) {
   node[i].sum = node[lson].sum + node[rson].sum;
   node[i].Max = max(node[lson].Max, node[rson].Max);
}

void build(int i, int l, int r) {
   node[i].l = l;
   node[i].r = r;
   if(l == r) {
      node[i].sum = node[i].Max = num[fp[l]];
      return ;
   }
   int mid = (l+r)>>1;
   build(lson, l, mid);
   build(rson, mid+1, r);
   pushup(i);
}

```

```
100 void update(int i, int k, int val) {
101    if(node[i].l == node[i].r && node[i].l == k) {
102       node[i].sum = node[i].Max = val;
103       return ;
104    }
105    int mid = (node[i].l + node[i].r) >> 1;
106    if(k <= mid) update(lson, k, val);
107    else update(rson, k, val);
108    pushup(i);
109 }
110 
111 int queryMax(int i, int l, int r) {
112    if(node[i].l >= l && node[i].r <= r) {
113       return node[i].Max;
114    }
115    int mid = (node[i].l + node[i].r) >> 1;
116    if(r<=mid) return queryMax(lson, l, r);
117    if(l > mid) return queryMax(rson, l, r);
118    return max(queryMax(lson, l, r), queryMax(rson, l, r));
119 }
120 
121 int querySum(int i, int l, int r) {
122    if(node[i].l >= l && node[i].r <= r) {
123       return node[i].sum;
124    }
125    int mid = (node[i].l + node[i].r) >> 1;
126    if(r<=mid) return querySum(lson, l, r);
127    if(l > mid) return querySum(rson, l, r);
128    return querySum(lson, l, r) + querySum(rson, l, r);
129 }
130 
131 int findMax(int u, int v) { // ????????????
132    int f1 = top[u], f2 = top[v];
133    int tmp = -INF;
134    while(f1 != f2) {
135       if(deep[f1] < deep[f2]) {
```

```
136            swap(f1, f2);
137            swap(u, v);
138         }
139         cout << f1 << ' ' << u << " | " << p[f1] << ' ' << p[u] <<
               endl;
140         tmp = max(tmp, queryMax(1, p[f1], p[u]));
141         u = pre[f1];
142         f1 = top[u];
143      }
144      if(deep[u] > deep[v]) swap(u, v);
145      cout << u << ' ' << v << " | " << p[u] << ' ' << p[v] << endl;
146      return max(tmp, queryMax(1, p[u], p[v]));
147   }
148
149   int findSum(int u, int v) { // ?????????????
150      int f1 = top[u], f2 = top[v];
151      int tmp = 0;
152      while(f1 != f2) {
153         if(deep[f1] < deep[f2]) {
154            swap(f1, f2);
155            swap(u, v);
156         }
157         tmp += querySum(1, p[f1], p[u]);
158         u = pre[f1];
159         f1 = top[u];
160      }
161      if(deep[u] > deep[v]) swap(u, v);
162      return tmp + querySum(1, p[u], p[v]);
163   }
164
165   int main() {
166   #ifndef ONLINE_JUDGE
167      freopen("in.txt", "r", stdin);
168   #endif
169      int n, q, u, v;
170      char op[17];
```

```
171     while(~scanf("%d",&n)) {
172         init();
173         rep(i,2,n) {
174             scanf("%d%d",&u,&v);
175             addedge(u, v);
176             addedge(v, u);
177         }
178         rep(i,1,n) scanf("%d",num+i);
179         dfs1(1,0,0);
180         dfs2(1,1);
181         build(1,0,pos-1);
182         scanf("%d",&q);
183         while(q--) {
184             scanf(" %s%d%d", op, &u, &v);
185             if(op[0] == 'C') update(1, p[u], v);
186             else if(op[1] == 'M') printf("%d\n",findMax(u, v));
187             else printf("%d\n",findSum(u,v));
188         }
189     }
190     return 0;
191 }
```

(by qi)

```
1  // BZOJ-1036: [ZJOI2008]树的统计Count
2  #include <iostream>
3  #include <cstdio>
4  #include <algorithm>
5  #include <queue>
6  #include <cstring>
7  #include <vector>
8  using namespace std;
9  const int maxn=30050;
10 const int inf=0x3f3f3f3f;
11 struct Edge
12 {
13     int v,nxt;
```

```
}e[maxn*2];
int tot=0;
int w[maxn],h[maxn],n;
void addedge(int x,int y){
   ++tot;
   e[tot].v=y;
   e[tot].nxt=h[x];
   h[x]=tot;
}

int sz[maxn],dep[maxn],fa[maxn],son[maxn];
void dfs1(int x){
   sz[x]=1;
   dep[x]=dep[fa[x]]+1;
   for (int i = h[x]; i ; i=e[i].nxt)
   {
      if(e[i].v!=fa[x]){
         fa[e[i].v]=x;
         dfs1(e[i].v);
         sz[x]+=sz[e[i].v];
         if(sz[e[i].v]>sz[son[x]]) son[x]=e[i].v;
      }
   }
}
int pos[maxn],l=0,idx[maxn],top[maxn];
void dfs2(int x,int chain){
   ++l;
   pos[x]=l;
   idx[l]=x;
   top[x]=chain;
   if(son[x]) dfs2(son[x],chain);
   for (int i = h[x]; i ; i=e[i].nxt)
   {
      if(e[i].v!=fa[x]&&e[i].v!=son[x]){
         dfs2(e[i].v,e[i].v);
      }
```

```
50      }
51  }
52
53  int mx[maxn*4],sum[maxn*4];
54  void pushup(int id){
55      sum[id]=sum[id*2]+sum[id*2+1];
56      mx[id]=max(mx[id*2],mx[id*2+1]);
57  }
58  void build(int l,int r,int id){
59      if(l==r){
60          sum[id]=mx[id]=w[idx[l]];
61          return;
62      }
63      build(l,(l+r)/2,id*2);
64      build((l+r)/2+1,r,id*2+1);
65      pushup(id);
66  }
67  void update(int p,int val,int l,int r,int id){
68      if(l==p&&r==p){
69          sum[id]=mx[id]=val;
70          return ;
71      }
72      if(p<=(l+r)/2) update(p,val,l,(l+r)/2,id*2);
73      else update(p,val,(l+r)/2+1,r,id*2+1);
74      pushup(id);
75  }
76  int Qsum(int x,int y,int l,int r,int id){
77      if(x<=l&&y>=r) return sum[id];
78      int ans=0;
79      if(x<=(l+r)/2) ans+=Qsum(x,y,l,(l+r)/2,id*2);
80      if(y>(l+r)/2) ans+=Qsum(x,y,(l+r)/2+1,r,id*2+1);
81      return ans;
82  }
83  int Qmax(int x,int y,int l,int r,int id){
84      if(x<=l&&y>=r) return mx[id];
85      int ans=-inf;
```

```
86      if(x<=(l+r)/2) ans=max(ans,Qmax(x,y,l,(l+r)/2,id*2));
87      if(y>(l+r)/2) ans=max(ans,Qmax(x,y,(l+r)/2+1,r,id*2+1));
88      return ans;
89  }
90  int main(int argc, char const *argv[])
91  {
92      scanf("%d", &n);
93      for (int i = 0; i < n-1; ++i)
94      {
95          int x,y;
96          scanf("%d%d", &x,&y);
97          addedge(x,y);
98          addedge(y,x);
99      }
100     for (int i = 1; i <= n; ++i)
101     {
102         scanf("%d", w+i);
103     }
104     fa[1]=1;
105     dfs1(1);
106     dfs2(1,1);
107     build(1,n,1);
108     // for (int i = 1; i <= n; ++i)
109     // {
110     // printf("%d ", Qsum(i,i,1,n,1));
111     // }
112     // printf("\n");
113     int q;
114     scanf("%d", &q);
115     for (int i = 0; i < q; ++i)
116     {
117         char s[20];
118         int x,y;
119         scanf("%s %d %d", s,&x,&y);
120         if(s[0]=='C') update(pos[x],y,1,n,1);
121         else if(s[1]=='M'){
```

```
122             int ans=-inf;
123             while(top[x]!=top[y]){
124                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
125                 ans=max(ans,Qmax(pos[top[x]],pos[x],1,n,1));
126                 x=fa[top[x]];
127             // printf("%d\n", ans);
128             }
129             if(pos[x]>pos[y]) swap(x,y);
130             ans=max(ans,Qmax(pos[x],pos[y],1,n,1));
131             printf("%d\n", ans);
132         }
133         else{
134             int ans=0;
135             while(top[x]!=top[y]){
136                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
137                 ans+=Qsum(pos[top[x]],pos[x],1,n,1);
138                 x=fa[top[x]];
139             }
140             if(pos[x]>pos[y]) swap(x,y);
141             ans+=Qsum(pos[x],pos[y],1,n,1);
142             printf("%d\n", ans);
143         }
144     }
145     return 0;
146 }
```

4.5 Treap

```
1 Rotate的作用对象永远是父节点
2 Delete某个节点的时候只能从这个节点的父节点开始
```

```
1 template <class T, class Compare = std::less<T> >
2 class Treap {
3 private:
4     struct treap {
5         int size, fix;
6         T key;
```

```
7       Compare cmp;
8       treap *ch[2];
9
10      treap(T key) {
11          size = 1;
12          fix = rand();
13          this->key = key;
14          ch[0] = ch[1] = NULL;
15      }
16
17      int compare(T x) const {
18          if (x == key) return -1;
19          return cmp(x ,key) ? 0 : 1;
20      }
21
22      void Maintain() {
23          size = 1;
24          if (ch[0] != NULL) size += ch[0]->size;
25          if (ch[1] != NULL) size += ch[1]->size;
26      }
27  }*root;
28  Compare cmp;
29
30  void Rotate(treap *&t, int d) {
31      treap *k = t->ch[d ^ 1];
32      t->ch[d ^ 1] = k->ch[d];
33      k->ch[d] = t;
34      t->Maintain();
35      k->Maintain();
36      t = k;
37  }
38
39  void Insert(treap *&t, T x) {
40      if (t == NULL) t = new treap(x);
41      else {
42          //int d = t->compare(x);
```

```
43          int d = cmp(x ,t->key) ? 0 : 1;
44          Insert(t->ch[d], x);
45          if (t->ch[d]->fix > t->fix) Rotate(t, d ^ 1);
46       }
47       t->Maintain();
48    }
49
50    void Delete(treap *&t, T x) {
51       int d = t->compare(x);
52       if (d == -1) {
53          treap *tmp = t;
54          if (t->ch[0] == NULL) {
55             t = t->ch[1];
56             delete tmp;
57             tmp = NULL;
58          } else if (t->ch[1] == NULL) {
59             t = t->ch[0];
60             delete tmp;
61             tmp = NULL;
62          } else {
63             int k = t->ch[0]->fix > t->ch[1]->fix ? 1 : 0;
64             Rotate(t, k);
65             Delete(t->ch[k], x);
66          }
67       } else Delete(t->ch[d], x);
68       if (t != NULL) t->Maintain();
69    }
70
71    bool Find(treap *t, int x) {
72       while (t != NULL) {
73          int d = t->compare(x);
74          if (d == -1) return true;
75          t = t->ch[d];
76       }
77       return false;
78    }
```

```
79
80  T Kth(treap *t, int k) {
81     if (t == NULL || k <= 0 || k > t->size) return -1;
82     if (t->ch[0] == NULL) {
83        if (k == 1) return t->key;
84        return Kth(t->ch[1], k - 1);
85     }
86     if (t->ch[0]->size >= k) return Kth(t->ch[0], k);
87     if (t->ch[0]->size + 1 == k) return t->key;
88     return Kth(t->ch[1], k - 1 - t->ch[0]->size);
89  }
90
91  int Rank(treap *t, int x) {
92     int r;
93     if (t->ch[0] == NULL) r = 0;
94     else r = t->ch[0]->size;
95     if (x == t->key) return r + 1;
96     if (x < t->key) return Rank(t->ch[0], x);
97     return r + 1 + Rank(t->ch[1], x);
98  }
99
100 treap* PreSuc(treap *t, int x, int d) {// d = 0 : 前驱 , d = 1
       : 后驱
101    treap * pre = NULL;
102    while(t != NULL && t->v != x) {
103       int k = t->compare(x);
104       if(k == (d^1)) pre = t;
105       t = t->ch[k];
106    }
107    t = t->ch[d];
108    if(t == NULL) return pre;
109    else {
110       while(t->ch[d^1] != NULL) {
111          t = t->ch[d^1];
112       }
113       return t;
```

```
114         }
115     }
116 
117     void Deletetreap(treap *&t) {
118         if (t == NULL) return;
119         if (t->ch[0] != NULL) Deletetreap(t->ch[0]);
120         if (t->ch[1] != NULL) Deletetreap(t->ch[1]);
121         delete t;
122         t = NULL;
123     }
124 
125     void Print(treap *t) {
126         if (t == NULL) return;
127         Print(t->ch[0]);
128         cout << t->key << ' ';
129         Print(t->ch[1]);
130     }
131 
132 public:
133     Treap() {
134         root = NULL;
135     }
136     ~Treap() {
137         Deletetreap(root);
138     }
139     void insert(T x) {
140         Insert(root, x);
141     }
142     void clear() {
143         Deletetreap(root);
144     }
145     T kth(int x) {
146         return Kth(root, x);
147     }
148     void print() {
149         Print(root);
```

```
    }
    int size() {
        return root->size;
    }
};
```

```
struct treap {
    int v, info, fix, ch[2], size;

    treap() {}

    treap(int info, int v) : info(info), v(v) {
        ch[0] = ch[1] = -1;
        fix = rand();
        size = 1;
    }

    int compare(int x) {
        if (v == x) return -1;
        return x < v ? 0 : 1;
    }
} node[maxn];

int root, tot;

void Maintain(int t) {
    node[t].size = 1;
    if (node[t].ch[0] != -1) node[t].size += node[node[t].ch[0]].
        size;
    if (node[t].ch[1] != -1) node[t].size += node[node[t].ch[1]].
        size;
}

void Rotate(int &t, int d) {
    if (t == -1) return;
    int tmp = node[t].ch[d ^ 1];
```

```
29    node[t].ch[d ^ 1] = node[tmp].ch[d];
30    node[tmp].ch[d] = t;
31    Maintain(t);
32    Maintain(tmp);
33    t = tmp;
34 }
35
36 void Insert(int &t, int info, int v) {
37    if (t == -1) {
38       t = ++tot;
39       node[t] = treap(info, v);
40    } else {
41       //int d = node[t].compare(v);
42       int d = v < node[t].v ? 0 : 1;
43       Insert(node[t].ch[d], info, v);
44       if (node[t].fix < node[node[t].ch[d]].fix) Rotate(t, d ^ 1);
45    }
46    Maintain(t);
47 }
48
49 int Find(int t, int v) {
50    if (t == -1) return t;
51    int d = node[t].compare(v);
52    if (d == -1) return t;
53    return Find(node[t].ch[d], v);
54 }
55
56 int Findmax(int t) {
57    if (t == -1) return -1;
58    while (node[t].ch[1] != -1) {
59       t = node[t].ch[1];
60    }
61    return t;
62 }
63
64 int Findmin(int t) {
```

```
65    if (t == -1) return -1;
66    while (node[t].ch[0] != -1) {
67       t = node[t].ch[0];
68    }
69    return t;
70 }
71
72 void Delete(int &t, int x) {
73    if (t == -1) return;
74    int k = node[t].compare(x);
75    if (k == -1) {
76       if (node[t].ch[0] != -1 && node[t].ch[1] != -1) {
77          int d = node[node[t].ch[0]].fix < node[node[t].ch[1]].fix
                 ? 0 : 1;
78          Rotate(t, d);
79          Delete(node[t].ch[d]);
80       } else {
81          if (node[t].ch[0] == -1) t = node[t].ch[1];
82          else t = node[t].ch[0];
83       }
84    } else Delete(node[t].ch[k], x);
85    if (t != -1) Maintain(t);
86 }
87
88 void Print(int t) {
89    if (t == -1) return;
90    Print(node[t].ch[0]);
91    cout << node[t].v << ' ';
92    Print(node[t].ch[1]);
93 }
```

4.6 线段树

```
1 #include <cstdio>
2 #include <functional>
3 const int maxn = 1e6;
```

```cpp
template <typename T>
struct BasicZKW {
   T tree[maxn * 2];
   int n;
   std::function<T(T, T)> op;

   void init(int n_, const std::function<T(T, T)> &op_) { n = n_, op = op_; }
   T& operator[] (int idx) { return tree[idx + n]; }//[0, n)

   void build() {
      for(int i = n - 1; i > 0; i--) tree[i] = op(tree[i << 1], tree[i << 1 | 1]);
   }

   void update(int p, T val) {//a[p] = val
      tree[p += n] = val;
      for(p >>= 1; p > 0; p >>= 1) tree[p] = op(tree[p << 1], tree[p << 1 | 1]);
   }
   T query(int l, int r) {//[l, r]
      T res = tree[r += n];
      for(l += n; l < r; l >>= 1, r >>= 1) {
         if(l & 1) res = op(res, tree[l++]);
         if(r & 1) res = op(res, tree[--r]);
      }
      return res;
   }
};
BasicZKW<int> zkw;
int main() {
   zkw.init(5, [](int x, int y)->int{return x + y;});
   for(int i = 0; i < 5; i++) zkw[i] = i;
   zkw.build();
   printf("%d\n", zkw.query(0, 4));
   zkw.update(0, 1);
```

```
37    printf("%d\n", zkw.query(0, 4));
38    return 0;
39 }
```

4.7 主席树

(by shui)

```
1  /*
2     HDU-2665 给你n个数m个询问, 对于每个询问l, r, k, 让你输出区间第k大
3  */
4  #include <bits/stdc++.h>
5  using namespace std;
6  const int maxn = 100007;
7  struct {
8     int ls, rs, cnt;
9  } node[maxn * 20];
10
11 int cur, rt[maxn];
12 inline void init() { cur = 0; }
13 inline void pushup(int t) {
14    node[t].cnt = node[node[t].ls].cnt + node[node[t].rs].cnt;
15 }
16 int build(int l, int r) {
17    int k = cur++;
18    if (l == r) {
19       node[k].cnt = 0;
20       return k;
21    }
22    int mid = (l + r) >> 1;
23    node[k].ls = build(l, mid);
24    node[k].rs = build(mid + 1, r);
25    pushup(k);
26    return k;
27 }
28 int update(int t, int l, int r, int pos, int val) {
29    int k = cur++;
```

```
30    node[k] = node[t];
31    if (l == pos && r == pos) {
32       node[k].cnt += val;
33       return k;
34    }
35    int mid = (l + r) >> 1;
36    if (pos <= mid) node[k].ls = update(node[t].ls, l, mid, pos,
         val);
37    else node[k].rs = update(node[t].rs, mid + 1, r, pos, val);
38    pushup(k);
39    return k;
40 }
41 int query(int l, int r, int u, int v, int kth) {
42    if (l == r) return l;
43    int mid = (l + r) >> 1;
44    int res = node[node[v].ls].cnt - node[node[u].ls].cnt;
45    if (kth <= res) return query(l, mid, node[u].ls, node[v].ls,
         kth);
46    else return query(mid + 1, r, node[u].rs, node[v].rs, kth -
         res);
47 }
48
49 int a[maxn], sorta[maxn], n, m;
50
51 int main() {
52    int _; scanf("%d", &_);
53    while (_--) {
54       scanf("%d%d", &n, &m);
55       init();
56       for (int i = 1; i <= n; i++) {
57          scanf("%d", &a[i]);
58          sorta[i] = a[i];
59       }
60       sort(sorta+1, sorta+1+n);
61       int cnt = unique(sorta+1, sorta+1+n)-sorta-1;
62       rt[0] = build(1, cnt);
```

```
63        for (int i = 1; i <= n; i++) {
64           int p = lower_bound(sorta + 1, sorta + cnt + 1, a[i]) -
                sorta;
65           rt[i] = update(rt[i - 1], 1, cnt, p, 1);
66        }
67        while(m--) {
68           int l, r, k;
69           scanf("%d%d%d", &l, &r, &k);
70           int idx = query(1, cnt, rt[l - 1], rt[r], k);
71           printf("%d\n", sorta[idx]);
72        }
73     }
74     return 0;
75  }
```

(by qi)

```
1  // exam7036 查询区间最大的h,使得至少h个数不小于h
2  #include <iostream>
3  #include <cstdio>
4  #include <algorithm>
5  #include <cstring>
6  using namespace std;
7  const int maxn=100050;
8  int n,q,a[maxn];
9  int sz,sum[maxn*20],lson[maxn*20],rson[maxn*20];
10 int update(int x,int l,int r,int id){
11    int root=++sz;
12    if(l==x&&r==x){
13       sum[root]=sum[id]+1;
14       lson[root]=rson[root]=0;
15       return root;
16    }
17    int mid=(l+r)>>1;
18    if(x<=mid){
19       lson[root]=update(x,l,mid,lson[id]);
20       rson[root]=rson[id];
```

```
    }
    else{
        lson[root]=lson[id];
        rson[root]=update(x,mid+1,r,rson[id]);
    }
    sum[root]=sum[lson[root]]+sum[rson[root]];
    return root;
}
int query(int suf,int l,int r,int r1,int r2){
    if(l==r){
        return l;
    }
    int s=sum[rson[r2]]-sum[rson[r1]];
    int mid=(l+r)>>1;
    if(s+suf>=mid+1) return query(suf,mid+1,r,rson[r1],rson[r2]);
    else return query(s+suf,l,mid,lson[r1],lson[r2]);
}
int T[maxn];
int main(int argc, char const *argv[])
{
    while(~scanf("%d%d", &n,&q)){
        T[0]=0;
        sz=0;
        for (int i = 1; i <= n; ++i)
        {
            scanf("%d", &a[i]);
            T[i]=update(a[i],1,n,T[i-1]);
        }
        while(q--){
            int l,r;
            scanf("%d%d", &l,&r);
            printf("%d\n", query(0,1,n,T[l-1],T[r]));
        }
    }
    return 0;
}
```

4.8 二叉查找树

```
template <class T>
class Binary_Search_Tree {
private:
   int Capacity, Size, root;
   struct Base{
      T key;
      int num, l, r, pre;
      Base(){
         num = 0;
         l = r = pre = -1;
      }
   };
   Base *node;
   void Insert(int &x, T k, int p = -1) {
      if(x == -1) {
         x = ++Size;
         node[x].key = k;
         node[x].pre = p;
      }
      else if(k == node[x]) {
         node[x].num++;
      } else if(k < node[x].key) {
         Insert(node[x].l, k, x);
      } else Insert(node[x].r, k, x);
   }
   void Insert_NonRecur(int &x, T k) {
      int f = -1, t = x;
      while(t != -1) {
         f = t;
         if(k == node[t].key) node[t].num++;
         else if(k < node[t].key) t = node[t].l;
         else t = node[t].r;
      }
      node[++Size].key = k;
```

```
35      node[Size].l = node[Size].r = -1;
36      node[Size].pre = f;
37      if(f == -1) x = Size;
38      else {
39         if(k < node[f].key) {
40            node[f].l = Size;
41         } else {
42            node[f].r = Size;
43         }
44      }
45   }
46   int Search(int rt, T k) {
47      while(rt != -1 && k != node[rt].key) {
48         if(k < node[rt].key) rt = node[rt].l;
49         else rt = node[rt].r;
50      }
51      return rt;
52   }
53   T Minimum(int rt) {
54      while(node[rt].l != -1) {
55         rt = node[rt].l;
56      }
57      return rt;
58   }
59   T Maximum(int rt) {
60      while(node[rt].r != -1) {
61         rt = node[rt].r;
62      }
63      return rt;
64   }
65   int Successor(int rt) { //后继：查找给定结点在中序遍历中的后继结点
66      if(node[rt].r != -1) {
67         return Minimum(node[rt].r);
68      }
69      int f = node[rt].pre;
70      while(f != -1 && node[f].r == rt) {
```

```
71          rt = f;
72          f = node[f].pre;
73       }
74       return f;
75    }
76    int Predecessor(int rt) { //前驱：查找给定结点在中序遍历中的前驱结点
77       if(node[rt].l != -1) {
78          return Maximum(node[rt].l);
79       }
80       int f = node[rt].pre;
81       while(f != -1 && node[f].l == rt) {
82          rt = f;
83          f = node[f].pre;
84       }
85       return f;
86    }
87    void Delete(int &rt, int z) {
88       if(node[z].l == -1 && node[z].r == -1) {
89          if(node[z].pre != -1) {
90             if(node[node[z].pre].l == z) {
91                node[node[z].pre].l = -1;
92             } else node[node[z].pre].r = -1;
93          } else {
94             rt = -1;// 只剩一个结点的情况
95          }
96       } else if(node[z].l != -1 && node[z].r == -1) {
97          node[node[z].l].pre = node[z].pre;
98          if(node[z].pre != -1) {
99             if(node[node[z].pre].l == z) node[node[z].pre].l =
                  node[z].l;
100            else node[node[z].pre].r = node[z].l;
101         } else {
102            rt = node[z].l;// 删除左斜单支树的根结点
103         }
104      } else if(node[z].l == -1 && node[z].r != -1) {
105         node[node[z].r].pre = node[z].pre;
```

```
106            if(node[z].pre != -1) {
107               if(node[node[z].pre].l == z) node[node[z].pre].l =
                     node[z].r;
108               else node[node[z].pre].r = node[z].r;
109            } else {
110               rt = node[z].r;
111            }
112         } else {
113            int s = Successor(z);
114            node[z].key = node[s].key;
115            Delete(rt, s);
116         }
117      }
118
119   public:
120      Binary_Search_Tree (int capacity = 1007) {
121         root = -1, Size = 0;
122         Capacity = capacity;
123         node = new Base[Capacity];
124      }
125      Binary_Search_Tree (T *arr, int len, int capacity = 1007) {
126         root = -1, Size = 0;
127         Capacity = capacity;
128         node = new Base[Capacity];
129         srand((unsigned int)(time(NULL)));
130         for(int i=len-1 ; i>=0 ; i--) {
131            int j = rand() % (i+1);
132            Insert(root,arr[j]);
133            swap(arr[j], arr[i]);
134         }
135      }
136      void insert(T a) {
137         Insert_NonRecur(root, a);
138      }
139   };
```

4.9 Splay

```
template<class T, class Compare = std::less<T> >
class splay_tree {
private:
   struct node {
      T key;
      node *ch[2];
      Compare cmp;

      node() {}

      node(T key) : key(key) {
         ch[0] = ch[1] = NULL;
      }
   } *root;
   Compare cmp;

   void Deletetree(node *&t) {
      if (t == NULL) return;
      Deletetree(t->ch[0]);
      Deletetree(t->ch[1]);
      delete t;
      t = NULL;
   }
   node* Splayrotate(node *&t, T key) {
      if(t == NULL) return NULL;
      node *l, *r, *c, N;
      N.ch[0] = N.ch[1] = NULL;
      l = r = &N;
      while(true) {
         if(cmp(key, t->key)) {
            if(t->ch[0] == NULL) break;
            if(cmp(key, t->ch[0]->key)) { // rotate right
               c = t->ch[0];
               t->ch[0] = c->ch[1];
```

```
35              c->ch[1] = t;
36              t = c;
37              if(t->ch[0] == NULL) break;
38           }
39           r->ch[0] = t; // link right
40           r = t;
41           t = t->ch[0];
42        } else if(cmp(t->key, key)) {
43           if(t->ch[1] == NULL) break;
44           if(cmp(t->ch[1]->key, key)) { // rotate left
45              c = t->ch[1];
46              t->ch[1] = c->ch[0];
47              c->ch[0] = t;
48              t = c;
49              if(t->ch[1] == NULL) break;
50           }
51           l->ch[1] = t; // link left
52           l = t;
53           t = t->ch[1];
54        } else break;
55     }
56     l->ch[1] = t->ch[0]; // assemble
57     r->ch[0] = t->ch[1];
58     t->ch[0] = N.ch[1];
59     t->ch[1] = N.ch[0];
60     return t;
61  }
62  void Insert(node *&t, T z) {
63     node *y = NULL, *x = t;
64     int d;
65     while(x != NULL) {
66        y = x;
67        d = cmp(x->key, z);
68        x = x->ch[d];
69     }
70     if(y == NULL) t = new node(z);
```

```
71      else if(cmp(z, y->key)) y->ch[0] = new node(z);
72      else y->ch[1] = new node(z);
73   }
74
75   node* Remove(node* &t, T key) {
76      if(t == NULL || Search(t, key) == NULL) return NULL;// 查找键
           值为key的节点，找不到的话直接返回
77      node *x = Splayrotate(t, key);// 将key对应的节点旋转为根节点
78      if(t->ch[0] != NULL) {
79         Splayrotate(t, t->ch[0]->key);// 将t的前驱节点旋转为根节点
80         t->ch[1] = t->ch[1]->ch[1];// 移除t节点
81      } else t = t->ch[1];
82      delete x;
83      x = NULL;
84      return t;
85   }
86
87   node* Search(node *t, T key) {
88      int d;
89      while((t != NULL) && (t->key != key)) {
90         d = cmp(t->key, key);
91         t = t->ch[d];
92      }
93      return t;
94   }
95   node* FindMin_Max(node *t, int d) {
96      while(t->ch[d] != NULL) {
97         t = t->ch[d];
98      }
99      return t;
100  }
101  void Showitem(node *t, int key = -1, int d = -1) {
102     if(t == NULL) return ;
103     if(d==-1)
104        cout << setw(2) << t->key << " is root" << endl;
105     else
```

```
106         cout << setw(2) << key << "'s " << (d == 1 ? "right" : "
              left") << " is " << setw(2) << t->key << endl;
107
108      Showitem(t->ch[0], t->key, 0);
109      Showitem(t->ch[1], t->key, 1);
110   }
111public:
112   splay_tree() {
113      root = NULL;
114   }
115
116   ~splay_tree() {
117      Deletetree(root);
118   }
119   void splay(T x) {
120      Splayrotate(root, x);
121   }
122   void insert(T x) {
123      Insert(root, x);
124      Splayrotate(root, x);
125   }
126   node* search(T x) {
127      return Search(root, x);
128   }
129   node* findmin_max(int d = 0) {
130      return FindMin_Max(root, d);
131   }
132   void remove(T x) {
133      root = Remove(root, x);
134   }
135   void clear() {
136      Deletetree(root);
137   }
138   void item() {
139      Showitem(root);
140   }
```

```
};
```

```
#define L ch[x][0]
#define R ch[x][1]

const int maxn = int(3e5) + 7;
int ch[maxn][2], sz[maxn], pre[maxn], add[maxn], val[maxn], stk[
   maxn], root, tot, Size, top, num[maxn], n, m, k1, k2;
bool flip[maxn];

void pushdown(int x) {
   if (add[x]) {
      val[L] += add[x];
      val[R] += add[x];
      add[L] += add[x];
      add[R] += add[x];
      add[x] = 0;
   }
   if (flip[x]) {
      flip[L] ^= 1;
      flip[R] ^= 1;
      swap(L, R);
      flip[x] = false;
   }
}

void pushup(int x) {
   sz[x] = sz[L] + sz[R] + 1;
}

int get(int x) {
   return ch[pre[x]][1] == x;
}

void rotate(int &x, int d) {
   int y = pre[x], z = pre[y];
```

```
34    pushdown(z);
35    pushdown(y);
36    pushdown(x);
37    ch[y][d ^ 1] = ch[x][d];
38    pre[ch[x][d]] = y;
39    pre[x] = z;
40    if (z != 0) ch[z][get(y)] = x;
41    ch[x][d] = y;
42    pre[y] = x;
43    pushup(y);
44 }
45
46 void splay(int x, int g) {
47    pushdown(x);
48    int y, z, d;
49    while (pre[x] != g) {
50       y = pre[x], z = pre[y];
51       pushdown(z);
52       pushdown(y);
53       pushdown(x);
54       if (z == g) {
55          rotate(x, get(x) ^ 1);
56       } else {
57          d = get(y);
58          ch[y][d] == x ? rotate(y, d ^ 1) : rotate(x, d);
59          rotate(x, d ^ 1);
60       }
61    }
62    if (g == 0) root = x;
63    pushup(x);
64 }
65
66 int find(int k) { // Find the kth point's number
67    int x = root;
68    while (true) {
69       pushdown(x);
```

```
70        if (sz[L] == k) break;
71        if (sz[L] > k) x = L;
72        else {
73           k -= sz[L] + 1;
74           x = R;
75        }
76     }
77     return x;
78  }
79
80  void rotate_to_somewhere(int k, int g) { // rotate kth point to be
       g's child (right child usually)
81     int x = find(k);
82     splay(x, g);
83  }
84
85  void newnode(int &x, int v, int p) {
86     if (top != 0) x = stk[top--];
87     else x = ++tot;
88     Size++;
89     pre[x] = p;
90     sz[x] = 1;
91     L = R = add[x] = 0;
92     val[x] = v;
93     flip[x] = false;
94  }
95
96  void build(int &x, int l, int r, int p) {
97     if (l > r) return;
98     int m = (l + r) >> 1;
99     newnode(x, num[m], p);
100    build(L, l, m - 1, x);
101    build(R, m + 1, r, x);
102    pushup(x);
103 }
104
```

```
105 void init(int n) {
106    Size = top = tot = 0;
107    newnode(root, -1, 0);
108    newnode(ch[root][1], -1, root);
109    for (int i = 0; i < n; i++) scanf("%d", num + i);
110    build(ch[ch[root][1]][0], 0, n - 1, ch[root][1]);
111    pushup(ch[root][1]);
112    pushup(root);
113 }
114 
115 int erase(int k) { // erase the kth node
116    rotate_to_somewhere(k, 0);
117    rotate_to_somewhere(k - 1, root);
118    int l = ch[root][0], r = ch[root][1], ret = val[root];
119    stk[++top] = root;
120    Size--;
121    root = l;
122    pre[l] = 0;
123    ch[l][1] = r;
124    if (r != 0) pre[r] = l;
125    pushup(root);
126    return ret;
127 }
128 
129 void insert(int v, int k = 0) { // insert point v after the kth
       point
130    rotate_to_somewhere(k, 0);
131    int x, r = ch[root][1];
132    newnode(x, v, root);
133    ch[root][1] = x;
134    ch[x][1] = r;
135    if (r != 0) pre[r] = x;
136    pushup(x);
137    pushup(root);
138 }
139 
```

```
140 void show(int x = root) {
141    if (x == 0) return;
142    pushdown(x);
143    show(L);
144    printf("%d ", val[x]);
145    show(R);
146 }
147 
148 void move(int d) {
149    if (d == 1) {
150       int v = erase(Size - 2);
151       insert(v, 0);
152    }
153    else {
154       int v = erase(1);
155       insert(v, Size - 2);
156    }
157 }
158 
159 void reverse(int l, int r) {
160    rotate_to_somewhere(l - 1, 0);
161    rotate_to_somewhere(r + 1, root);
162    flip[ch[ch[root][1]][0]] ^= 1;
163    pushup(ch[root][1]);
164    pushup(root);
165 }
166 
167 void update(int l, int r, int k) {
168    rotate_to_somewhere(l - 1, 0);
169    rotate_to_somewhere(r + 1, root);
170    val[ch[ch[root][1]][0]] += k; // whatch out there!!!
171    add[ch[ch[root][1]][0]] += k;
172    pushup(ch[root][1]);
173    pushup(root);
174 }
175 
```

```
176 int query() {
177    return val[find(1)];
178 }
```

4.10 线段树

```
1  #include <bits/stdc++.h>
2
3  const int maxn = int(2e5) + 7;
4  namespace SegmentTree {
5  #define ls (t << 1)
6  #define rs (t << 1 | 1)
7     int len; // 线段树的区间长度，下标从1开始
8     typedef int type;
9     type *buf;
10
11    class SgmtTree {
12    private:
13       type node[maxn << 2], laz[maxn << 2]; //node是树节点，laz是
          lazy标记
14       void pushdown(int t, int l, int r) {
15          int mid = (l + r) >> 1;
16          laz[ls] += laz[t];
17          laz[rs] += laz[t];
18          node[ls] += laz[t] * (mid - l + 1);
19          node[rs] += laz[t] * (r - mid);
20          laz[t] = 0;
21       }
22
23       void pushup(int t) {
24          node[t] = node[ls] + node[rs];
25       }
26
27       void build(int t = 1, int l = 1, int r = len) {
28          if (l == r) {
29             node[t] = buf[l];
```

```
30            return;
31        }
32        int mid = (l + r) >> 1;
33        build(ls, l, mid);
34        build(rs, mid + 1, r);
35        pushup(t);
36    }
37
38  public:
39    void init(type *tmp, int len_) { // 用长度为len_的数组tmp去初始
          化线段树
40        len = len_;
41        buf = tmp;
42        memset(laz, 0, sizeof(laz));
43        build(1, 1, len); // 下标从1开始
44    }
45
46    void update(int b, int e, type num, int t = 1, int l = 1,
          int r = len) {
47        if (e < l || b > r) return;
48        if (b <= l && r <= e) {
49            laz[t] += num;
50            node[t] += num * (r - l + 1);
51            return;
52        }
53        if (laz[t]) pushdown(t, l, r);
54        int mid = (l + r) >> 1;
55        update(b, e, num, ls, l, mid);
56        update(b, e, num, rs, mid + 1, r);
57        pushup(t);
58    }
59
60    type query(int b, int e, int t = 1, int l = 1, int r = len)
          {
61        if (e < l || b > r) return type(0);
62        if (b <= l && r <= e) return node[t];
```

```
63          if (laz[t]) pushdown(t, l, r);
64          int mid = (l + r) >> 1;
65          if (e <= mid) query(b, e, ls, l, mid);
66          if (b > mid) query(b, e, rs, mid + 1, r);
67          return query(b, e, ls, l, mid) + query(b, e, rs, mid + 1,
                r);
68      }
69   } tree;
70 } using SegmentTree::tree;
71
72 int a[maxn], len = 5;
73
74 int main() {
75    for (int i = 1; i <= len; i++) a[i] = 1;
76    tree.init(a, len); // 初始化线段树，下标从1开始
77    std::cout << tree.query(1, len) << std::endl; // 查询区间和
78    tree.update(1, len, 1); // 区间[1, len]加一
79    std::cout << tree.query(1, len) << std::endl;
80    return 0;
81 }
```

4.11 二叉堆 & 优先队列

```
1  #include <bits/stdc++.h>
2  template<class type, class Compare = std::less<type> >
3  class priority_queue {
4  private:
5     type *Heap;// 数据
6     int Capacity, Size;// 总的容量，实际容量
7     Compare cmp;
8     void filterdown(int start, int end) {// 最大堆的向下调整算法
9        int index = start;
10       int cur = index << 1;// 先指向左儿子
11       type tmp = Heap[index];
12       while (cur <= end) {
```

```
13          if (cur < end && cmp(Heap[cur], Heap[cur + 1])) {// cur+1
               是右儿子
14             cur++;// 如果右儿子比较大，就将cur指向右儿子
15          }
16          if (!cmp(tmp, Heap[cur])) break;//调整结束
17          Heap[index] = Heap[cur];
18          index = cur;
19          cur <<= 1;
20       }
21       Heap[index] = tmp;
22    }
23
24    void filterup(int start) {// 最大堆的向上调整算法(从start开始向上直
         到1，调整堆)
25       int cur = start;
26       int pre = cur >> 1;
27       type tmp = Heap[cur];
28       while (cur > 1) {
29          if (!cmp(Heap[pre], tmp)) break;
30          Heap[cur] = Heap[pre];
31          cur = pre;
32          pre >>= 1;
33       }
34       Heap[cur] = tmp;
35    }
36
37    int getindex(type data) {// 返回data在二叉堆中的索引
38       for (int i = 1; i <= Size; i++) if (Heap[i] == data) return
            i;
39       return -1;
40    }
41
42    bool remove(type data) {// 删除最大堆中的data，成功返回true
43       if (Size == 0) return false;
44       int index = getindex(data);// 获取data在二叉堆中的索引
45       if (index == -1) return false;
```

```
46          Heap[index] = Heap[Size--];
47          filterdown(index, Size);
48          return true;
49      }
50
51      void print() {// 打印二叉堆
52          for (int i = 1; i < Size; i++) std::cout << Heap[i] << ' ';
53          std::cout << Heap[Size] << std::endl;
54      }
55
56  public:
57      priority_queue(int capactity = 1007) { // capactity 最大容量
58          Capacity = capactity;
59          Size = 0;
60          Heap = new type[capactity + 7];
61      }
62
63      ~priority_queue() { delete (Heap); }
64
65      void pop() {
66          if (Size == 0) return;
67          Heap[1] = Heap[Size--];
68          filterdown(1, Size);
69      }
70
71      type top() {
72          if (Size == 0) exit(1);
73          return Heap[1];
74      }
75
76      bool empty() { return Size == 0; }
77
78      bool push(type data) {// 将data插入到二叉堆中
79          if (Size == Capacity) return false;
80          Heap[++Size] = data;
81          filterup(Size);
```

```
82          return true;
83      }
84  };
```

4.12 树状数组

```
1   //单点修改，区间查询
2   //1.将某个数加上x，2.求[x，y]区间和
3   #include <bits/stdc++.h>
4   const int maxn = int(1e6) + 7;
5   struct Bit {
6       int data[maxn], len;
7       inline int lowbit(int x) { return x & -x; }
8       void init(int len_) { len = len_, memset(data, 0, sizeof(data))
            ; }
9       void add(int pos, int val) { while (pos <= len) data[pos] +=
            val, pos += lowbit(pos); }
10      int query(int pos) {
11          int ret = 0;
12          while (pos) ret += data[pos], pos -= lowbit(pos);
13          return ret;
14      }
15  } bit;
16  int n, m;
17  int main() {
18      scanf("%d%d", &n, &m);
19      bit.init(n);
20      for (int i = 1, buf; i <= n; i++) scanf("%d", &buf), bit.add(i,
            buf);
21      for (int i = 1, op, x, y; i <= m; i++) {
22          scanf("%d%d%d", &op, &x, &y);
23          if (op == 1) bit.add(x, y);
24          else if (op == 2) printf("%d\n", bit.query(y) - bit.query(x
                - 1));
25      }
26      return 0;
```

```
}

//区间修改，单点查询
//1.将区间[1，r]中每一个数都加上x，2.输出第x个数
#include <cstdio>
int n, m, c[500005];
#define lowbit(x) (x&-x)
void add(int pos, int x) { while(pos <= n) c[pos] += x, pos += lowbit(pos);}
int query(int pos) {
   int ans = 0;
   while(pos > 0) ans += c[pos], pos -= lowbit(pos);
   return ans;
}
int main() {
   scanf("%d%d", &n, &m);
   int x=0, y, op, k;
   for(int i=1 ; i<=n ; i++) scanf("%d",&y), add(i, y-x), x = y;
   while(m--) {
      scanf("%d", &op);
      if(op == 1) scanf("%d%d%d", &x, &y, &k), add(x,k), add(y+1,-k);
      else scanf("%d", &x), printf("%d\n", query(x));
   }
   return 0;
}
```

4.13 二维树状数组

```
struct BitTree {
   int data[307][307];
   void update(int x, int y, int w) { //将点(x，y)加上w
      for (int i = x; i <= n; i += i & -i)
         for (int j = y; j <= n; j += j & -j)
            data[i][j] += w;
   }
```

```
8     int query(int x, int y) { //求左上角为(1,1)右下角为(x,y) 的矩阵和
9        int ret = 0;
10       for (int i = x; i > 0; i -= i & -i)
11          for (int j = y; j > 0; j -= j & -j)
12             ret += data[i][j];
13       return ret;
14    }
15 };
```

4.14 可持久化 Trie 树

```
1  #include <iostream>
2  #include <cstdio>
3  #include <algorithm>
4  using namespace std;
5  const int maxn=50050;
6  int ch[maxn*20][2],sz=0;
7  int val[maxn*20],root[maxn];
8  void insert(int pos,int x){
9     int l=root[pos-1],r=++sz;
10    root[pos]=sz;
11    for (int i = 15; i >= 0; --i)
12    {
13       ch[r][0]=ch[l][0];
14       ch[r][1]=ch[l][1];
15       val[r]=pos;
16       ch[r][(x>>i)&1]=++sz;
17       r=ch[r][(x>>i)&1];
18       l=ch[l][(x>>i)&1];
19    }
20    val[r]=pos;
21 }
22 int query(int l,int r,int x){
23    int ans=0,u=root[r];
24    for (int i = 15; i >= 0; --i)
25    {
```

```
26        // printf("%d %d %d\n", i,x,ch[u][(x>>i)&1^1]);
27        if(val[ch[u][(x>>i)&1^1]]>=l) u=ch[u][(x>>i)&1^1],ans|=(1<<i
              );
28        else if(val[ch[u][(x>>i)&1]]>=l) u=ch[u][(x>>i)&1];
29        else break;
30     }
31     return ans;
32  }
33  int main(int argc, char const *argv[])
34  {
35     int n,m;
36     scanf("%d%d", &n,&m);
37     for (int i = 1; i <= n; ++i)
38     {
39        int x;
40        scanf("%d", &x);
41        insert(i,x);
42     }
43     for (int i = 0; i < m; ++i)
44     {
45        int l,r,x;
46        scanf("%d%d%d", &l,&r,&x);
47        printf("%d\n", query(l,r,x));
48     }
49     return 0;
50  }
51  /**************************************************************
52     Problem: 5679
53     User: upc_reserver201706
54     Language: C++
55     Result: 正确
56     Time:76 ms
57     Memory:13624 kb
58  **************************************************************/
```

4.15 并查集

```
const int maxn = (int)1e5+7;
int pre[maxn],height[maxn];
//pre就不解释了，height是当前树高

int Find(int x) {//while以及附带路徑壓縮的版本，遞歸在數據量比較大的時候可能爆
   if(x==pre[x]) return x;
   int p=x,q;
   while(x!=pre[x]) x=pre[x];
   while(p!=pre[p]) {
      q=pre[p];
      pre[p]=x;
      p=q;
   }
   return x;
}

void Add(int i, int j) {
   int x=Find(i),y=Find(j);
   if(x==y) return ;
   if(height[x]<height[y]) pre[x]=y;
   else {
      if(height[x]==height[y]) height[x]++;
      pre[y]=x;
   }
   return ;
}
```

4.16 栈 & 队列

4.16.1 单调栈

```
#include <bits/stdc++.h>
const int maxn = int(1e5) + 7;
template<typename type, typename _Compare = std::less<type>>
class monotony_stack {
```

```
5     _Compare cmp;
6     type data[maxn];
7     int cur;
8  public:
9     monotony_stack() { cur = 0; }
10    void clear() { cur = 0; }
11    void push(type val) {
12       while (cur && !cmp(data[cur], val)) cur--;
13       data[++cur] = val;
14    }
15    type top() { return data[cur]; }
16    type lower_top() { if (!empty()) return data[cur - 1]; }
17    void pop() { if (!empty()) cur--; }
18    bool empty() { return cur == 0; }
19    int size() { return cur; }
20 };
21 int t, n, num[maxn], smaller[maxn];
22 monotony_stack<int> stack;
23 int main() {
24    scanf("%d", &t);
25    while (t--) {
26       stack.clear();
27       scanf("%d", &n);
28       for (int i = 1; i <= n; i++) scanf("%d", num + i);
29       for (int i = 1; i <= n; i++) {
30          stack.push(num[i]);
31          if (stack.size() > 1) smaller[num[i]] = stack.lower_top()
               ;
32          else smaller[num[i]] = -1;
33       }
34       for (int i = 1; i <= n; i++) printf("%d%c", smaller[num[i]],
             i == n ? '\n' : ' ');
35    }
36 }
```

4.16.2 队列

```
template <typename type>
struct queue {
   const static int maxSize = 7;
   type data[maxSize];
   int head = 0, tail = 0;
   void push(type val) {
      data[tail] = val;
      tail = (tail + 1) % maxSize;
   }
   void pop() { head = (head + 1) % maxSize; }
   bool empty() { return head == tail; }
   int size() { return (tail - head + maxSize) % maxSize; }
   type front() { return data[head]; }
};
```

4.16.3 单调队列

```
template <typename type, typename compare = std::less<type>>
class monotony_queue { // 默认最大的在队首
   compare cmp;
   type data[maxn];
   int head, tail;
   int add(int num) { return (num + 1) % maxn; }
   int sub(int num) { return (num - 1 + maxn) % maxn; }
public:
   monotony_queue() { head = 0, tail = 0; }
   void clear() { head = 0, tail = 0; }
   void push(type val) {
      while (head != tail && cmp(data[sub(tail)], val)) tail = sub
         (tail);
      data[tail] = val, tail = add(tail);
   }
   void pop() { head = add(head); }
   bool empty() { return head == tail; }
   type front() { return data[head]; }
};
```

5 动态规划

5.1 数位 dp

```
/*
   HDU-3555 对于每个询问n输出0到n之间含有数字四九的数的个数(1 <= N <= 2^63-1)
*/
#include <bits/stdc++.h>
using namespace std;
long long dp[27][2][2], query;
int bit[27];
long long dfs(int pos, bool four = false, bool nine = false, bool limit = true) {
   if(pos < 0) return nine;
   if(!limit && ~dp[pos][four][nine]) return dp[pos][four][nine];
   int up = limit ? bit[pos] : 9;
   long long ans = 0;
   for(int i=0 ; i<=up ; i++)
      ans += dfs(pos-1, i==4, nine||(four&&i==9), limit&&i==up);
   return limit ? ans : dp[pos][four][nine] = ans;
}
long long solve(long long n) {
   int pos = 0;
   while(n) bit[pos++] = n%10, n/=10;
   return dfs(pos-1);
}
int main() {
   int t; scanf("%d", &t);
   memset(dp, -1, sizeof(dp));
   while(t--) {
      scanf("%lld", &query);
      printf("%lld\n", solve(query));
   }
   return 0;
}
```

5.2 状态压缩

```
//判断一个数字x二进制下第i位是不是等于1
return ((1 << ( i - 1 )) & x) > 0;

//将一个数字x二进制下第i位更改成1
x = x | (1 << (i - 1));

//把一个数字二进制下最靠右的第一个1去掉
x = x & (x - 1);

//求数字中1的个数(循环次数只和1的出现次数有关，原理是每次去掉最低位的1)
int cnt = 0; while (x) x &= x - 1, cnt++;

// 枚举状态S的所有子集
for (int x = S; x; x = (x-1)&S)
```

5.3 背包问题

5.3.1 01 背包

```
void ZeroOnePack(int c, int w) {
   for(int i=v ; i>=c ; i--)
      dp[i] = max(dp[i], dp[i-c]+w);
}
void solve() {
   for(int i=0 ; i<n ; i++)
      ZeroOnePack(c[i], w[i]);
}
```

5.3.2 多重背包

```
void CompletePack(int c, int w) {
   for(int i=0 ; i<=v ; i++)
      dp[i] = max(dp[i], dp[i-c]+w);
}
void ZeroOnePack(int c, int w) {
   for(int i=v ; i>=c ; i--)
```

```
        dp[i] = max(dp[i], dp[i-c]+w);
}
void MultiplePack(int c,int w,int cnt) {
    //如果总容量比这个物品的容量要小，那么这个物品可以直到取完，相当于完全背包
    if(v<=cnt*c) CompletePack(c, w);
    else {//否则就将多重背包转化为01背包
        int k = 1;
        while(k<cnt) {
            ZeroOnePack(k*c, k*w);
            cnt = cnt-k;
            k<<=1;
        }
        if(cnt) ZeroOnePack(cnt*cost, cnt*wei);
    }
}
```

5.3.3 完全背包

```
void CompletePack(int c, int w) {
    for(int i=0 ; i<=v ; i++)
        dp[i] = max(dp[i], dp[i-c]+w);
}
void solve() {
    for(int i=0 ; i<n ; i++)
        CompletePack(c[i], w[i]);
}
```

6 图论

6.1 二分图

6.1.1 KM

```
const int maxn = 307, INF = 0x3f3f3f3f;
//n和m的下标从0开始
int n, m;//n个女生（左），m个男生（右）
int love[maxn][maxn];//love[i][j]表示第i个女生对第j个男生的好感度
```

```
int ex_girl[maxn];//第i个女生的当前期望值为ex_girl[i]
int ex_boy[maxn];//同上
bool vis_girl[maxn];//标记在当前配对过程中girl[i]有没有被访问过
bool vis_boy[maxn];//同上
int match[maxn];//match[boy] = girl，说明这个boy和girl已经配对了，没有则为-1
int slack[maxn];//记录每个男生如果能被妹子倾心最少还需要多少期望值

/*
注意！！！！mp的初始值一定要给的很夸张！！！不然dfs过程中有可能tmp=0也能成立!
*/

bool dfs(int girl) {
    vis_girl[girl] = true;
    for(int boy = 0 ; boy < m ; boy++) {
        if(vis_boy[boy]) continue;//每一轮匹配，每个男生只尝试一次
        int gap = ex_girl[girl] + ex_boy[boy] - love[girl][boy];
        if(gap == 0) {//如果符合要求
            vis_boy[boy] = true;
            if(match[boy] == -1 || dfs(match[boy])) {
                //找到一个没有配对的男生 或者是 和这个男生配对的女生可以找到别人
                match[boy] = girl;
                return true;
            }
        } else {
            slack[boy] = min(slack[boy],gap);
            //可以理解为这个男生要能够得到配对的话最少还需要多少期望
        }
    }
    return false;
}

int KM() {
    clr(match, -1);//初始每个男生都没有匹配的女生
    clr(ex_boy, 0);//初始每个男生的期望值
```

```
    for(int i=0 ; i<n ; i++) {
        //每个女生的初始期望值是与她相连的男生最大好感度
        ex_girl[i] = love[i][0];
        for(int j=1 ; j<m ; j++) {
            ex_girl[i] = max(ex_girl[i], love[i][j]);
        }
    }
    for(int i=0 ; i<n ; i++) {
        //尝试为每一个女生解决归宿问题
        clr(slack, INF);//初始化为最大，随后取最小值
        while(true) {
            //为每个女生解决归宿问题的方法是:
            //如果找不到男票就降低自己的期望值，直到找到为止
            clr(vis_girl,0);
            clr(vis_boy,0);
            if(dfs(i)) break;//找到归宿，退出循环
            //如果找不到，就降低妹子的期望值
            int d = INF;//能够降低的最小期望值derta
            for(int j = 0; j<m ; j++) {
                //遍历所有一点在匈牙利树中另一点不在匈牙利树种的边
                //slack[i]的值是girl[i] 所在边的 端点权值之和-边的权值
                //取它们中的最小值，就是最小可以松弛的期望值
                if(!vis_boy[j]) d = min(d, slack[j]);
            }
            for(int j = 0; j<n ; j++) {
                //对每个访问过的女生减少期望
                if(vis_girl[j]) ex_girl[j] -= d;
            }
            for(int j = 0; j<m ; j++) {
                //所有访问过的男生增加期望
                if(vis_boy[j]) ex_boy[j] += d;
                //所有没访问过的男生距离自己被配对又更近了一步
                else slack[j] -= d;
            }
        }
    }
```

```
75      //配对完成，求出所有的好感度和
76      int res = 0;
77      for (int i = 0; i < m ; i++)
78          if(match[i] != -1) res += love[match[i]][i];
79      return res;
80  }
81
82
83  //HDU-2255
84  #include <iostream>
85  #include <algorithm>
86  #include <set>
87  #include <string>
88  #include <vector>
89  #include <queue>
90  #include <map>
91  #include <stack>
92  #include <list>
93  #include <iomanip>
94  #include <functional>
95  #include <sstream>
96  #include <cstdio>
97  #include <cstring>
98  #include <cmath>
99  #include <cctype>
100 #define edl putchar('\n')
101 #define clr(a,b) memset(a,b,sizeof a)
102 using namespace std;
103 const int maxn = 307, INF = 0x3f3f3f3f;
104
105 int love[maxn][maxn],n;
106 int ex_girl[maxn];
107 int ex_boy[maxn];
108 bool vis_girl[maxn];
109 bool vis_boy[maxn];
110 int match[maxn];
```

```
int slack[maxn];

bool dfs(int girl) {
   vis_girl[girl] = true;
   for(int boy = 0 ; boy < n ; boy++) {
      if(vis_boy[boy]) continue;
      int gap = ex_girl[girl] + ex_boy[boy] - love[girl][boy];
      if(gap == 0) {
         vis_boy[boy] = true;
         if(match[boy] == -1 || dfs(match[boy])) {
            match[boy] = girl;
            return true;
         }
      } else {
         slack[boy] = min(slack[boy],gap);
      }
   }
   return false;
}

int KM() {
   clr(match, -1);
   clr(ex_boy, 0);
   for(int i=0 ; i<n ; i++) {
      ex_girl[i] = love[i][0];
      for(int j=1 ; j<n ; j++) {
         ex_girl[i] = max(ex_girl[i], love[i][j]);
      }
   }
   for(int i=0 ; i<n ; i++) {
      fill(slack, slack+n, INF);
      while(true) {
         clr(vis_girl,0);
         clr(vis_boy,0);
         if(dfs(i)) break;
         int d = INF;
```

```
147             for(int j = 0; j<n ; j++)
148                 if(!vis_boy[j]) d = min(d, slack[j]);
149             for(int j = 0; j<n ; j++) {
150                 if(vis_girl[j]) ex_girl[j] -= d;
151                 if(vis_boy[j]) ex_boy[j] += d;
152                 else slack[j] -= d;
153             }
154         }
155     }
156     int res = 0;
157     for (int i = 0; i < n ; i++)
158         res += love[match[i]][i];
159     return res;
160 }
161 int main() {
162 #ifndef ONLINE_JUDGE
163     freopen("in.txt","r",stdin);
164 #endif
165     while(~scanf("%d",&n)) {
166         for(int i=0 ; i<n ; i++)
167             for(int j=0 ; j<n ; j++)
168                 scanf("%d",&love[i][j]);
169         printf("%d\n",KM());
170     }
171     return 0;
172 }
```

6.1.2 匈牙利

```
1 //跑有向图的时候得保证所有的边都满足从集合S指向集合T
2 const int maxn = 207;
3 struct {
4     int next, to;
5 }edge[maxn * maxn];
6 int head_edge[maxn * maxn], cnt_edge;
7 int n, link[maxn];
```

```
bool vis[maxn];

void addedge(int u, int v) {
   edge[cnt_edge] = {head_edge[u], v};
   head_edge[u] = cnt_edge++;
}

bool dfs(int u) {
   for(int i = head_edge[u] ; ~i ; i = edge[i].next) {
      int v = edge[i].to;
      if(vis[v]) continue;
      vis[v] = true;
      if(link[v] == -1 || dfs(link[v])) {
         link[v] = u;
         link[u] = v;
         return true;
      }
   }
   return false;
}

int match() {
   int ans = 0;
   clr(link, -1);
   for (int i = 1; i <= n; i++)
      if (link[i] == -1) {
         clr(vis, 0);
         if (dfs(i)) ans++;
      }
   return ans;
}

void init() {
   clr(head_edge, -1);
   cnt_edge = 0;
}
```

6.2 拓扑排序

6.2.1 DFS 拓扑排序

```cpp
#include <iostream>
#include <vector>
using namespace std;
const int maxn = int(1e5)+7;

vector<int> edge[maxn];
int tim,tp[maxn],n,m;
bool vis[maxn],in[maxn];

void dfs(int u) {
   vis[u] = true;
   for(auto v : edge[u]) if(!vis[v]) dfs(v);
   tp[tim--] = u;
}

void topological_sort() {
   tim = n;
   for(int i=1 ; i<=n ; i++) if(!vis[i]) dfs(i);
}

int main() {
#ifndef ONLINE_JUDGE
   freopen("in.txt","r",stdin);
#endif
   ios::sync_with_stdio(false);
   cin.tie(nullptr);
   cin >> n >> m;
   for(int i=0, u, v ; i<m ; i++) {
      cin >> u >> v;
      in[v]=true;
      edge[u].push_back(v);
   }
   topological_sort();
```

```
34      for(int i=1 ; i<n ; i++) cout << tp[i] << ", ";
35      cout << tp[n];
36      return 0;
37  }
```

6.2.2 Kahn 拓扑排序

```
1   #include <iostream>
2   #include <list>
3   #include <queue>
4   using namespace std;
5
6   /************************类声明************************/
7   class Graph
8   {
9       int V; // 顶点个数
10      list<int> *adj; // 邻接表
11      queue<int> q; // 维护一个入度为0的顶点的集合
12      int* indegree; // 记录每个顶点的入度
13  public:
14      Graph(int V); // 构造函数
15      ~Graph(); // 析构函数
16      void addEdge(int v, int w); // 添加边
17      bool topological_sort(); // 拓扑排序
18  };
19
20  /************************类定义************************/
21  Graph::Graph(int V)
22  {
23      this->V = V;
24      adj = new list<int>[V];
25
26      indegree = new int[V]; // 入度全部初始化为0
27      for(int i=0; i<V; ++i)
28          indegree[i] = 0;
29  }
```

```
30
31 Graph::~Graph()
32 {
33    delete [] adj;
34    delete [] indegree;
35 }
36
37 void Graph::addEdge(int v, int w)
38 {
39    adj[v].push_back(w);
40    ++indegree[w];
41 }
42
43 bool Graph::topological_sort()
44 {
45    for(int i=0; i<V; ++i)
46       if(indegree[i] == 0)
47          q.push(i); // 将所有入度为0的顶点入队
48
49    int count = 0; // 计数，记录当前已经输出的顶点数
50    while(!q.empty())
51    {
52       int v = q.front(); // 从队列中取出一个顶点
53       q.pop();
54
55       cout << v << " "; // 输出该顶点
56       ++count;
57       // 将所有v指向的顶点的入度减1，并将入度减为0的顶点入栈
58       list<int>::iterator beg = adj[v].begin();
59       for( ; beg!=adj[v].end(); ++beg)
60          if(!(--indegree[*beg]))
61             q.push(*beg); // 若入度为0，则入栈
62    }
63
64    if(count < V)
65       return false; // 没有输出全部顶点，有向图中有回路
```

```
66      else
67          return true; // 拓扑排序成功
68  }
```

6.3 最短路

6.3.1 堆优化 Dijkstra

```
1   const int maxn = 1007, INF = 0x3f3f3f3f;
2   vector<pair<int, int> > edge[maxn];
3   int m, n, dist[maxn];
4   bool vis[maxn];
5
6   struct cmp {//重载优先队列比较方法
7       bool operator () (pair<int, int> y, pair<int, int> x) {//这里第
            一个是y，第二个是x
8           if(x.second == y.second) return x.first < y.first;
9           return x.second < y.second;
10      }
11  };
12
13  int dijkstra_heap(int start, int end) {//返回从start到end的最短路径
14      clr(dist, INF);
15      clr(vis, 0);
16      priority_queue<pair<int, int>, vector<pair<int, int> >, cmp>
            que;
17      //以自定义的比较方式创建优先队列
18      dist[start] = 0;
19      que.push(make_pair(start,0));
20      while(!que.empty()) {
21          pair<int, int> now = que.top();
22          que.pop();
23          int u = now.first, v, w;
24          if(vis[u]) continue;//每个点只访问一次
25          vis[u] = true;
26          for(vector<pair<int ,int> >::iterator i=edge[u].begin() ; i
                !=edge[u].end() ; i++) {
```

```
27          v = (*i).first, w = (*i).second;
28          if(dist[v] > dist[u] + w) {
29             dist[v] = dist[u] + w;
30             que.push(make_pair(v, dist[v]));//这里入队的是v和dist[v]
                  ，别写错了!
31          }
32       }
33    }
34    return dist[end];
35 }
```

6.3.2 SPFA

```
1  const int maxn = 105, INF = 0x3f3f3f3f;
2  int map[maxn][maxn], dist[maxn];
3  bool inque[maxn];
4  int n, m;
5  int spfa(int n, int start) {
6     clr(dist, INF);
7     clr(inque, 0);
8     dist[start] = 0;
9     queue<int> q;
10    q.push(start);
11    inque[start] = true;
12    while(!q.empty()) {
13       int index = q.front();
14       q.pop();
15       for(int i=1 ; i<=n ; i++)
16          if(dist[i] > dist[index] + mp[index][i]) {
17             dist[i] = dist[index] + mp[index][i];
18             if(!inque[i]) {
19                q.push(i);
20                inque[i] = true;
21             }
22          }
23       inque[index] = false;
```

```
    }
    return dist[n];
}
```

6.3.3 次短路 Dijkstra

```
struct node {
    int u, d;
    node(int u = 0, int d = 0):u(u), d(d) {}
    bool operator < (const node &tmp) const {
        return d > tmp.d;
    }
};
int dijkstra_Secondary(int start, int end) {
    priority_queue<node> que;
    memset(dist, INF, sizeof(dist));
    memset(dist_, INF, sizeof(dist_));
    dist[start] = 0;
    que.emplace(start, dist[start]);
    while(!que.empty()) {
        int u = que.top().u, d_ = que.top().d;
        que.pop();
        if(dist_[u] < d_) continue; //取出的不是次短距离，抛弃
        for(int i = head[u] ; ~i ; i = edge[i].next) {
            int v = edge[i].to, d = d_ + edge[i].val;
            if(dist[v] > d) { //更新最短距离
                swap(dist[v], d);
                que.emplace(v, dist[v]);
            }
            if(dist_[v] > d && dist[v] <= d) {//更新次短距离
                //如果只返回次短路，就删掉上一行的等于号
                dist_[v] = d;
                que.emplace(v, dist_[v]);
            }
        }
    }
```

```
31     return dist_[end];//如果存在两条最短路，那么返回的就是最短路
32 }
```

6.3.4 AStar

```
1  #include <bits/stdc++.h>
2  const int maxn = int(1e4) + 7, maxm = int(1e5) + 7, inf = 0
       x3f3f3f3f;
3  int n, m, k, t, S, T, f[maxn];
4  struct Graph {
5      int head[maxn], cnt;
6      struct { int next, v, cost; } edge[maxm];
7      void init() {
8          memset(head, 0xff, sizeof(head));
9          cnt = 0;
10     }
11     void addedge(int u, int v, int cost) {
12         edge[cnt] = {head[u], v, cost};
13         head[u] = cnt++;
14     }
15 } graph, inv;
16 struct Node {
17     int index, cost, dist;
18     Node(int index, int dist, int cost = 0):index(index), dist(dist
           ), cost(cost) {}
19     bool operator < (const Node &tmp) const {
20         return dist + cost > tmp.dist + tmp.cost;
21     }
22 };
23 void bfs(int S, const Graph &g) {
24     std::bitset<maxn> vis;
25     std::priority_queue<Node> que;
26     memset(f, 0x3f, sizeof(f));
27     f[S] = 0;
28     que.emplace(S, 0);
29     while (!que.empty()) {
```

```
30        Node cur = que.top();
31        que.pop();
32        int u = cur.index;
33        if (vis[u]) continue;
34        vis[u] = true;
35        for (int i = g.head[u]; ~i; i = g.edge[i].next) {
36           int v = g.edge[i].v, cost = g.edge[i].cost;
37           if (f[v] > f[u] + cost) {
38              f[v] = f[u] + cost;
39              if (!vis[v]) que.emplace(v, f[v]);
40           }
41        }
42     }
43  }
44
45  int Astar(int S, int T, int k, const Graph &g) {
46     bfs(T, inv);
47     int cnt[maxn] = {0};
48     std::priority_queue<Node> que;
49     que.emplace(S, 0, f[S]);
50     while (!que.empty()) {
51        Node cur = que.top();
52        que.pop();
53        int u = cur.index;
54        if (++cnt[u] > k) continue;
55        if (u == T && cnt[u] == k) return cur.dist;
56        if(cur.dist >= t) return inf;
57        for (int i = g.head[u]; ~i; i = g.edge[i].next) {
58           int v = g.edge[i].v, cost = g.edge[i].cost;
59           if (cnt[v] < k) que.emplace(v, cur.dist + cost, f[v]);
60        }
61     }
62     return inf;
63  }
64  int main() {
65     while (graph.init(), inv.init(), ~scanf("%d%d", &n, &m)) {
```

```
66        scanf("%d%d%d%d", &S, &T, &k, &t);
67        for (int i = 1, u, v, cost; i <= m; i++) {
68           scanf("%d%d%d", &u, &v, &cost);
69           graph.addedge(u, v, cost);
70           inv.addedge(v, u, cost);
71        }
72        puts(Astar(S, T, k, graph) <= t ? "yareyaredawa" : "
             Whitesnake!");
73     }
74     return 0;
75  }
```

6.4 网络流

6.4.1 Dinic

```
1  #include <bits/stdc++.h>
2  const int maxn = 1007, maxm = int(4e6) + 7, inf = 0x3f3f3f3f;
3  struct { int next, v, flow; } edge[maxm << 1];
4  struct Graph {
5     int head[maxn], cnt;
6     Graph() { memset(head, 0xff, sizeof(head)), cnt = 0; }
7     void addedge(int u, int v, int flow) {
8        edge[cnt] = {head[u], v, flow};
9        head[u] = cnt++;
10       edge[cnt] = {head[v], u, 0};
11       head[v] = cnt++;
12    }
13 } graph;
14 struct Dinic {
15    int dist[maxn], cur[maxn], que[maxn * maxn];
16    bool bfs(int S, int T) {
17       memset(dist, 0xff, sizeof(dist));
18       dist[S] = 0;
19       int head = 0, tail = 0;
20       que[tail++] = S;
21       while (head < tail) {
```

```
22          int u = que[head++];
23          for (int i = graph.head[u]; ~i; i = edge[i].next) {
24             int v = edge[i].v, flow = edge[i].flow;
25             if (dist[v] == -1 && flow > 0) {
26                dist[v] = dist[u] + 1;
27                if (v == T) return true;
28                que[tail++] = v;
29             }
30          }
31       }
32       return false;
33    }
34    int dfs(int u, int low, int T) {
35       if (u == T) return low;
36       for (int &i = cur[u]; ~i; i = edge[i].next) {
37          int v = edge[i].v, flow = edge[i].flow;
38          if (dist[v] == dist[u] + 1 && flow > 0) {
39             int min = dfs(v, flow < low ? flow : low, T);
40             if (min > 0) {
41                edge[i].flow -= min;
42                edge[i ^ 1].flow += min;
43                return min;
44             }
45          }
46       }
47       return 0;
48    }
49    int solve(int S, int T) {
50       int ans = 0, tmp;
51       while (bfs(S, T)) {
52          memcpy(cur, graph.head, sizeof(cur));
53          while (tmp = dfs(S, inf, T), tmp > 0) ans += tmp;
54       }
55       return ans;
56    }
57 } dinic;
```

```
int main() {
   int n, m, s, t;
   scanf("%d%d%d%d", &n, &m, &s, &t);
   for (int i = 0, u, v, flow; i < m; i++) {
      scanf("%d%d%d", &u, &v, &flow);
      graph.addedge(u, v, flow);
   }
   printf("%d\n", dinic.solve(s, t));
   return 0;
}
```

6.4.2 MCMF

```
#include <bits/stdc++.h>
typedef long long ll;
const int maxn = 407, maxm = 15007 << 1, inf = 0x3f3f3f3f;
struct { int next, u, v, flow, cost; } edge[maxm];
struct Graph {
   int head[maxn], cnt;
   Graph() { memset(head, 0xff, sizeof(head)), cnt = 0; }
   void addedge(int u, int v, int flow, int cost) {
      edge[cnt] = {head[u], u, v, flow, cost};
      head[u] = cnt++;
      edge[cnt] = {head[v], v, u, 0, -cost};
      head[v] = cnt++;
   }
} graph;
struct MCMF {
   int dist[maxn], pre[maxn], low[maxn], vis[maxn], que[maxn *
      maxn], clk;
   bool bfs(int S, int T) {
      vis[S] = ++clk, low[S] = inf, dist[S] = 0, memset(dist, 0x3f
         , sizeof(dist));
      int head = 0, tail = 0;
      que[tail++] = S;
      while (head < tail) {
```

```
22          int u = que[head++];
23          vis[u] = -1;
24          for (int i = graph.head[u]; ~i; i = edge[i].next) {
25             int v = edge[i].v, flow = edge[i].flow, cost = edge[i
                ].cost;
26             if (dist[v] > dist[u] + cost && flow > 0) {
27                dist[v] = dist[u] + cost;
28                pre[v] = i;
29                low[v] = std::min(low[u], flow);
30                if (vis[v] != clk) {
31                   que[tail++] = v;
32                   vis[v] = clk;
33                }
34             }
35          }
36       }
37       return dist[T] < inf;
38    }
39    std::pair<ll, ll> solve(int S, int T) {
40       ll flow = 0, cost = 0;
41       while (bfs(S, T)) {
42          flow += low[T];
43          cost += 1ll * low[T] * dist[T];
44          for (int u = T; u != S; u = edge[pre[u]].u) {
45             edge[pre[u]].flow -= low[T];
46             edge[pre[u] ^ 1].flow += low[T];
47          }
48       }
49       return std::make_pair(flow, cost);
50    }
51 } mcmf;
52 int main() {
53    int n, m;
54    scanf("%d%d", &n, &m);
55    for (int i = 1, u, v, flow, cost; i <= m; i++) {
56       scanf("%d%d%d%d", &u, &v, &flow, &cost);
```

```
57          graph.addedge(u, v, flow, cost);
58      }
59      std::pair<ll, ll> ans = mcmf.solve(1, n);
60      printf("%lld %lld\n", ans.first, ans.second);
61      return 0;
62  }
```

6.5 连通图

6.5.1 割边

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int maxn = 3000+7, INF = 0x3f3f3f3f;
4   int dfn[maxn], low[maxn], n, m, result, Time;
5   //bool cut[maxn][maxn];
6   vector<pair<int, int> > edge[maxn];
7   void addedge(int u, int v, int w) {
8       edge[u].emplace_back(v, w);
9       edge[v].emplace_back(u, w);
10  }
11  void findcut(int u, int pre) {
12      dfn[u] = low[u] = Time++;
13      for(auto i : edge[u]) {
14          int v = i.first, w = i.second;
15          if(dfn[v] == -1) {
16              findcut(v, u);
17              low[u] = min(low[u], low[v]);
18              if(low[v] > dfn[u]) {
19                  //cut[u][v] = cut[v][u] = true;
20                  result = min(result, w);
21              }
22          }
23          else if(pre != v) low[u] = min(low[u], dfn[v]);
24      }
25  }
26  int main() {
```

```
27    while(cin >> n >> m) {
28       memset(dfn, -1, sizeof dfn);
29       result = INF, Time = 1;
30       for(int i=0, u, v, w ; i<m ; i++) {
31          scanf("%d%d%d", &u, &v, &w);
32          addedge(u,v,w);
33       }
34       for(int i=1 ; i<=n ; i++) if(dfn[i] == -1) findcut(i,i);
35       cout << result << '\n';
36    }
37    return 0;
38 }
```

6.5.2 连通图 Tarjan

```
1  int top;//这个是用作栈顶的指针
2  int Stack[maxn];//维护的一个栈
3  bool instack[maxn];//instack[i]为真表示i在栈中
4  int DFN[maxn],LOW[maxn];
5  int Belong[maxn];//Belong[i] = a; 表示i这个点属于第a个连通分量
6  int Bcnt,Dindex;//Bcnt用来记录连通分量的个数，Dindex表示到达某个点的时间
7  void tarjan(int u) {
8     int v;
9     DFN[u]=LOW[u] = ++ Dindex;//这里要注意 Dindex是初始化为0，这里就不能
          Dindex++; 不然第一个点的DFN和LOW就为0
10    Stack[++ top] = u;
11    instack[u] = true;
12    for (edge *e = V[u] ; e ; e = e->next) {//对所有可达边的搜索
13       v = e->t;
14       if (!DFN[v]) {//这个if 就是用来更新LOW[u]
15          tarjan(v);
16          if (LOW[v] < LOW[u]) LOW[u] = LOW[v];
17       }
18       else if (instack[v] && DFN[v] < LOW[u]) LOW[u] = DFN[v];
19    }
20    if (DFN[u] == LOW[u]) {//这里表示找完一个强连通啦
```

```
21          Bcnt ++;//强连通个数加1
22          do {
23              v = Stack[top --];
24              instack[v] = false;
25              Belong[v] = Bcnt;
26          }
27          while (u != v);//一直到v=u都是属于第Bcnt个强连通分量
28      }
29  }
30  void solve() {
31      top = Bcnt = Dindex = 0;
32      memset(DFN,0,sizeof(DFN));
33      for (int i = 1; i <= N ; i ++) if (!DFN[i]) tarjan(i); //这里是
            一定要对所有点tarjan才能求出所有的点的强连通分量
34  }
```

7 树

7.1 LCA

7.1.1 RMQ-ST

```
1   #define maxn 1007
2   #define maxn 100005
3   struct node
4   {
5       int x;
6   };
7   vector<node> V[maxn]; //储存树的结构，也可以使用邻接表
8   int E[maxn * 2], D[maxn * 2], first[maxn]; //标号数列 深度数列 某个标
        号第一次出现的位置
9   int vis[maxn], dis[maxn], n, m, top = 1, root[maxn], st; //dis[]
        若边有权值可求距离 root[] 求整棵树的根
10  int dp[30][maxn * 2]; //储存某区间最小值的下标
11
12  //DFS树得到欧拉序列
13  void dfs(int u, int dep) {
```

```
14      vis[u] = 1, E[top] = u, D[top] = dep, first[u] = top++;
15      for (int i = 0; i < V[u].size(); i++)
16          if (!vis[V[u][i].x]) {//储存时双向，因此判断是否为父节点
17              int v = V[u][i].x;
18              dfs(v, dep + 1);
19              E[top] = u, D[top++] = dep; //dfs回溯过程必须储存，否则原理就
                    错误了
20          }
21  }
22
23  //ST预处理
24  void ST(int num) {
25      for (int i = 1; i <= num; i++) dp[0][i] = i;//初始状态
26      for (int i = 1; i <= log2(num); i++) //控制区间长度
27          for (int j = 1; j <= num; j++) //控制区间左端点
28              if (j + (1 << i) - 1 <= num) {
29                  int a = dp[i - 1][j], b = dp[i - 1][j + (1 << i >> 1)
                        ];
30                  if (D[a] < D[b]) dp[i][j] = a;//储存D数组下标，以便找到对
                        应的E数组
31                  else dp[i][j] = b;
32              }
33  }
34
35  //RMQ查询
36  int RMQ(int x, int y) {
37      int k = (int) log2(y - x + 1.0);
38      int a = dp[k][x], b = dp[k][y - (1 << k) + 1];
39      if (D[a] < D[b]) return a;//前后两段比较
40      return b;
41  }
```

7.1.2 Tarjan 并查集

```
1   //poj 1986
2   #include <bits/stdc++.h>
```

```
#define read read()
#define edl putchar('\n')
#define clr(a,b) memset(a,b,sizeof a)
int read{ int x=0;char c=getchar();while(c<'0' || c>'9')c=getchar
   ();while(c>='0' && c<='9'){ x=x*10+c-'0';c=getchar(); }return x
   ;}
void write(int x){ int y=10,len=1;while(y<=x) {y*=10;len++;}while(
   len--){y/=10;putchar(x/y+48);x%=y;}}
using namespace std;

const int maxn = int(1e5)+7;
int n,m,k,ans[maxn],cnt;

int pre[maxn];
int find(int x) { return (pre[x] == x ? x : pre[x] = find(pre[x]))
   ; }

vector<pair<int,int> > edge[maxn], query[maxn];
void addedge(int u, int v, int w) {
   edge[u].push_back(make_pair(v,w));
   edge[v].push_back(make_pair(u,w));
}

void add(int f, int s) {
   int ff = find(f), fs = find(s);
   if(ff == fs) return ;
   pre[fs] = ff;
}

int cost[maxn];
bool vis[maxn];
void tarjan(int u, int p) {
   int len = int(edge[u].size());
   for(int i=0, v, w ; i<len ; i++) {
      v = edge[u][i].first, w = edge[u][i].second;
      if (v == p) continue;
```

```
35          cost[v] = cost[u] + w;
36          tarjan(v, u);
37          add(u, v);
38       }
39       vis[u] = true;
40       if(cnt == k) return ;
41       int lenq = int(query[u].size());
42       for(int i=0 ; i<lenq ; i++) {
43          int v = query[u][i].first;
44          if(vis[v]) {
45             ans[query[u][i].second] = cost[u] + cost[v] - 2 * cost[
                find(v)];
46          }
47       }
48    }
49
50    void init() {
51       for(int i=1 ; i<=n ; i++) {
52          pre[i] = i;
53          edge[i].clear();
54          query[i].clear();
55       }
56       cnt = 0;
57       clr(vis,0);
58       clr(cost,0);
59    }
60
61    int main() {
62    #ifndef ONLINE_JUDGE
63       freopen("in.txt", "r", stdin);
64    #endif
65       while(cin >> n >> m) {
66          init();
67          for(int i=0, u, v, w ; i<m ; i++) {
68             u = read, v = read, w = read;
69             addedge(u,v,w);
```

```
70        }
71        k = read;
72        for(int i=0, u, v ; i<k ; i++) {
73           u = read, v = read;
74           query[u].push_back(make_pair(v,i));
75           query[v].push_back(make_pair(u,i));
76           ans[i] = 0;
77        }
78        tarjan(1,0);
79        for(int i=0 ; i<k ; i++) write(ans[i]),edl;
80     }
81     return 0;
82  }
```

7.1.3 倍增算法

```
1  #include <bits/stdc++.h>
2  const int maxn = int(1e5) + 7, maxm = int(4e5) + 7;
3  struct Lca { // 点的下标从1开始
4     int dep[maxn], up[maxn][32], cnt, head[maxn];
5     struct { int next, to, val; } edge[maxm];
6     void addedge(int u, int v, int w) {
7        edge[cnt] = {head[u], v, w};
8        head[u] = cnt++;
9     }
10    Lca() {
11       cnt = 0;
12       memset(head, 0xff, sizeof(head));
13       memset(up, 0xff, sizeof(up));
14    }
15    void init(int n) {
16       std::queue<int> que;
17       que.emplace(dep[1] = 1); // 下标从1开始
18       int u, v;
19       while (!que.empty()) {
20          u = que.front(), que.pop();
```

```
21          for (int i = head[u]; ~i; i = edge[i].next) {
22             v = edge[i].to;
23             if (dep[v]) continue ;
24             up[v][0] = u, dep[v] = dep[u] + 1, que.push(v);
25          }
26       }
27       for (int j = 1; j <= 20; j++)
28          for (int i = 1; i <= n; i++)
29             if (~up[i][j - 1])
30                up[i][j] = up[up[i][j - 1]][j - 1];
31    }
32    int query(int u, int v) {
33       if (dep[u] < dep[v]) std::swap(u, v);
34       int tmp = dep[u] - dep[v];
35       for (int j = 0; tmp; j++)
36          if (tmp & (1 << j)) tmp ^= (1 << j), u = up[u][j];
37       if (u == v) return v;
38       for (int j = 20; j >= 0; j--) {
39          if (up[u][j] != up[v][j]) {
40             u = up[u][j], v = up[v][j];
41          }
42       }
43       return up[u][0];
44    }
45 } lca;
```

7.2 最小生成树

7.2.1 O(elog2v)-primMST

```
1  //点v，边e
2  //邻接矩阵O(v^2)，邻接表O(e log_{2}v)
3  int prime(int cur) {//當前的根
4     int index; //用來臨時存儲最小邊對應的下標
5     int sum = 0; //當前數的權值
6     memset(visit, false, sizeof(visit)); //初始化節點訪問情🄵
7     visit[cur] = true; //將根節點設置🄵已訪問
```

```
    for(int i = 0; i < m; i ++){
        dist[i] = graph[cur][i]; //dist记录的是树到点i的距离
        //将树根cur视为一颗树，将树到未连接点之间的距离存入dist中
    }
    for(int i = 1; i < m; i ++){
        int mincost = INF; //初始化最小值🄵INF
        for(int j = 0; j < m; j ++){
            if(!visit[j] && dist[j] < mincost){
            //如果當前邊的另一個點不在生成樹中而且距離小於mincost
                mincost = dist[j]; //更新mincost和其對應邊和點的下標
                index = j;
            }
        }
        visit[index] = true; //標記當前最小的邊🄴已訪問
        sum += mincost; //最小生成樹的權+=當前這條最小邊的權
        for(int j = 0; j < m; j ++){
            if(!visit[j] && graph[index][j] < dist[j]){
            //如果新加入的这个点到j点的距离小于当前的树到j点的距离
                dist[j] = graph[index][j]; //更新新生成树到j点的距离
            }
        }
    }
    return sum; //返回當前根對應最小生成樹的權值
}
```

7.2.2 O(elogv)-prim+heap MST

```
const int maxn = int(2e4)+7;
int dist[maxn], head_edge[maxn], cnt_edge;
bitset<maxn> vis;
struct {int next, to, cost;}edge[maxn];
void addedge(int u, int v, int c) {
    edge[cnt_edge] = {head_edge[u], v, c};
    head_edge[u] = cnt_edge++;
}
struct node {
```

```
10      int u, d;
11      bool operator < (const node &tmp) const {
12          return d > tmp.d;
13      }
14  }now;
15  int prim(int cur) {
16      int ans = 0, u, v, d;
17      vis.reset();
18      memset(dist, 0x3f3f3f3f, sizeof(dist));
19      priority_queue<node> q;
20      q.push({cur, 0});
21      dist[cur] = 0;
22      while(!q.empty()) {
23          now = q.top(), q.pop();
24          u = now.u, d = now.d;
25          if(vis[u] || d > dist[u]) continue;
26          vis[u] = true;
27          ans += dist[u];
28          for(int i=head_edge[u] ; ~i ; i=edge[i].next) {
29              v = edge[i].to;
30              if(dist[v] > edge[i].cost) {
31                  dist[v] = edge[i].cost;
32                  q.push({v, dist[v]});
33              }
34          }
35      }
36      return ans;
37  }
38  void init() {
39      cnt_edge = 0;
40      memset(head_edge, -1, sizeof(head_edge));
41  }
```

8 计算几何

8.1 基础定义

```
///const
const double eps = 1e-8;
const int maxm = 1000+10;
const double pi = acos(-1.0);
///determain the sign
int cmp(double x) {
   if(fabs(x)<eps)
      return 0;
   if(x>0)
      return 1;
   return -1;
}
```

8.2 点

```
namespace _point {
struct point {
   double x,y;
   point(double _a = 0,double _b = 0):x(_a),y(_b) {}
   void input() {
      scanf("%lf%lf",&x,&y);
   }
   friend point operator+(const point &a,const point &b) {
      return point(a.x+b.x,a.y+b.y);
   }
   friend point operator-(const point &a,const point &b) {
      return point(a.x-b.x,a.y-b.y);
   }
   friend bool operator==(const point &a,const point &b) {
      return cmp(a.x-b.x)==0&&cmp(a.y-b.y)==0;
   }
   friend point operator*(const point &a,const double &b) {
      return point(a.x*b,a.y*b);
```

```
19      }
20      friend point operator*(const double &a,const point &b) {
21          return point(b.x*a,b.y*a);
22      }
23      friend point operator/(const point &a,const double &b) {
24          return point(a.x/b,a.y/b);
25      }
26      double len() {
27          return sqrt(x*x+y*y);
28      }
29      double angle(){
30          return atan2(y,x);
31      }
32  };
33  double across(const point &a,const point &b) {///叉积
34      return a.x*b.y-a.y*b.x;
35  }
36  double dot(const point &a,const point &b) {///点积
37      return a.x*b.x+a.y*b.y;
38  }
39  double dist(const point &a,const point &b) {///距离
40      return (a-b).len();
41  }
42  point rotate_point(const point &p,double angle) {///旋转
43      double tx = p.x,ty = p.y;
44      return point(tx*cos(angle)-ty*sin(angle),tx*sin(angle)+ty*cos(
        angle));
45  }
46  }
47  using namespace _point;
```

8.3 线

```
1   namespace _line {
2   struct line {
3       point a,b;
```

```
4     line(point x=point(0,0),point y=point(0,0)):a(x),b(y) {}
5     void input(){
6        a.input();
7        b.input();
8     }
9  };
10 bool point_on_segment(point p,point s,point t) { ///1
11    return cmp(across(p-s,t-s))==0&&cmp(dot(p-s,p-t))<=0;
12 }
13 bool parallel(line a,line b) {///判平行
14    return !cmp(across(a.a-a.b,b.a-b.b));
15 }
16 point point_across_line(line l, point p) { ///点关于直线的对称点
17    point p1 = l.a;
18    point p2 = l.b;
19    double _x, _y;
20    if(cmp(p1.x - p2.x) == 0) {
21       _x = 2 * p1.x - p.x;
22       _y = p.y;
23    } else if(cmp(p1.y - p2.y) == 0) {
24       _x = p.x;
25       _y = 2 * p1.y - p.y;
26    } else {
27       double k1 = (p1.y - p2.y) / (p1.x - p2.x);
28       double b1 = p1.y - k1 * p1.x;
29       double k2 = -1 / k1;
30       double b2 = p.y - k2 * p.x;
31       _x = (b2 - b1) / (k1 - k2);
32       _y = k2 * _x + b2;
33    }
34    return point(2 * _x - p.x, 2 * _y - p.y);
35 }
36 bool segment_across_line(line a,line b){ ///判断线段和直线相交
37    if(across(b.a-a.a,b.b-a.a)*across(b.a-a.b,b.b-a.b)>eps)
38       return false;
39    return true;
```

```
}
bool segment_across_segment(line l1,line l2){///线段和线段相交
   return
      max(l1.a.x,l1.b.x) >= min(l2.a.x,l2.b.x) &&
      max(l2.a.x,l2.b.x) >= min(l1.a.x,l1.b.x) &&
      max(l1.a.y,l1.b.y) >= min(l2.a.y,l2.b.y) &&
      max(l2.a.y,l2.b.y) >= min(l1.a.y,l1.b.y) &&
      cmp(across((l2.a-l1.a),(l1.b-l1.a)))*cmp(across((l2.b-l1.a)
         ,(l1.b-l1.a))) <= 0 &&
      cmp(across((l1.a-l2.a),(l2.b-l2.a)))*cmp(across((l1.b-l2.a)
         ,(l2.b-l2.a))) <= 0;
}
bool point_of_line_to_line(line a,line b,point &ret){///直线和直线的
   交点
   if(parallel(a,b)) return false;
   double s1 = across(a.a-b.a,b.b-b.a);
   double s2 = across(a.b-b.a,b.b-b.a);
   ret = point(s1*a.b-s2*a.a)/(s1-s2);
   return true;
}
}
using namespace _line;
```

8.4 凸包

```
namespace _polygon {
struct polygon {
   int n;
   vector<point> a;
   polygon(int _n = 0) {///初始化
      a.resize(_n);
      n = _n;
   }
   double length() {///OK
      double sum = 0;
      a[n] = a[0];
```

```
12      for(int i = 0; i<n; i++)
13          sum+=(a[i+1]-a[i]).len();
14      return sum;
15  }
16  double area() {///凸包面积
17      double ret = 0;
18      a[n] = a[0];
19      for(int i = 0; i<n; i++) {
20          ret+=across(a[i+1],a[i]);
21      }
22      return ret/2.0;
23  }
24  int point_in_polygon(point t) {///判断点是否在凸包内
25      int num = 0;
26      a[n] = a[0];
27      for(int i = 0; i<n; i++) {
28          if(point_on_segment(t,a[i],a[i+1]))
29              return 2;///on
30          int k = cmp(across(a[i+1]-a[i],t-a[i]));
31          int d1 = cmp(a[i].y-t.y);
32          int d2 = cmp(a[i+1].y-t.y);
33          if(k>0&&d1<=0&&d2>0)
34              num++;
35          if(k<0&&d2<=0&&d1>0)
36              num++;
37      }
38      return num!=0;
39  }
40  point mess_center() {///凸包重心
41      point ans(0,0);
42      if(cmp(area())==0)
43          return ans;
44      a[n] = a[0];
45      for(int i = 0; i<n; i++) {
46          ans = ans+(a[i]+a[i+1])*across(a[i+1],a[i]);
47      }
```

```
48        return ans/area()/6.0;
49     }
50     int border_point_num() {
51        int ret = 0;
52        a[n] = a[0];
53        for(int i = 0; i<n; i++) {
54           ret+=__gcd(abs(int(a[i+1].x-a[i].x)),abs(int(a[i+1].y-a[
                 i].y)));
55        }
56        return ret;
57     }
58     int inside_point_num() {
59        return int(area())+1-border_point_num()/2;
60     }
61  };
62  bool cmp_less(const point &a,const point &b) {///OK
63     ///is a lower or lefter than b
64     return cmp(a.x-b.x)<0||(cmp(a.x-b.x)==0&&cmp(a.y-b.y)<0);
65  }
66  polygon convex_full(vector<point> las) {///求凸包
67     polygon ret(2*las.size()+5);
68     sort(las.begin(),las.end(),cmp_less);
69     las.erase(unique(las.begin(),las.end()),las.end());
70     int m = 0;
71     int si = las.size();
72     for(int i = 0; i<si; i++) {
73        while(m>1&&cmp(across(ret.a[m-1]-ret.a[m-2],las[i]-ret.a[m
              -2]))<=0)
74           m--;
75        ret.a[m++] = las[i];
76     }
77     int k = m;
78     for(int i = si-2; i>=0; i--) {
79        while(m>k&&cmp(across(ret.a[m-1]-ret.a[m-2],las[i]-ret.a[m
              -2]))<=0)
80           m--;
```

```
81          ret.a[m++] = las[i];
82       }
83       ret.a.resize(m);
84       ret.n = m;
85       if(si>1){
86          ret.a.resize(m-1);
87          ret.n--;
88       }
89       return ret;
90    }
91    int point_in_polygon_logn(const polygon &a,const point &b) {///在
         logn时间内判断点是否在凸包内
92       int n = a.a.size();
93       point g = (a.a[0]+a.a[n/3]+a.a[(n<<1)/3])/3.0;
94       int l = 0,r = n;
95       while(l+1<r) {
96          int mid = (l+r)>>1;
97          if(cmp(across(a.a[l]-g,a.a[mid]-g))>0) {
98             if(cmp(across(a.a[l]-g,b-g))>=0&&cmp(across(a.a[mid]-g,b
                -g))>=0)
99                r = mid;
100            else
101               l = mid;
102         } else {
103            if(cmp(across(a.a[l]-g,b-g))<0&&cmp(across(a.a[mid]-g,b
                -g))>=0)
104               l = mid;
105            else
106               r = mid;
107         }
108      }
109      r%=n;
110      int z = cmp(across(a.a[r]-b,a.a[l]-b)) - 1;
111      if(z==-2)
112         return 1;
113      return z;
```

```
}
double convex_diameter(polygon &a,int &first,int &second){///凸包直径和对锺点的下标
   vector<point> &p = a.a;
   int n = p.size();
   double maxd = 0.0;
   if(n==1) {
      first = second = 0;
      return 0;
   }
   #define nex(i) ((i+1)%n)
   for(int i = 0,j = 1;i<n;i++){
      while(cmp( across(p[nex(i)]-p[i] , p[j]-p[i]) - across(p[nex(i)]-p[i],p[nex(j)]-p[i]) )<0)
         j = nex(j);
      double d = (p[i]-p[j]).len();
      if(d>maxd) {
         maxd = d;
         first = i,second = j;
      }
      d = (p[nex(i)]-p[nex(j)]).len();
      if(d>maxd){
         maxd = d;
         first = i,second = j;
      }
   }
   return maxd;
}
}
using namespace _polygon;
```

8.5 三角形

```
namespace _triangle{
struct triangle{
   point t0,t1,t2;
```

```
    triangle(point _t0 = point(0,0),point _t1 = point(0,0),point
       _t2 = point(0,0)):t0(_t0),t1(_t1),t2(_t2){}
    double area(){
       return fabs(across(t1-t0,t2-t1))/2;
    }
};
}
using namespace _triangle;
```

8.6 圆

```
namespace _circle{
    struct circle{
       point o;
       double r;
       circle(point _o=point(0,0),double _r=0.0):o(_o),r(_r){}
    };
    void point_of_circle_across_line(point a,point b,circle o,point
        ret[],int &num){///直线和圆的交点
       double x0 = o.o.x , y0 = o.o.y;
       double x1 = a.x , y1 = a.y;
       double x2 = b.x , y2 = b.y;
       double r = o.r;
       double dx = x2-x1 , dy = y2-y1;
       double A = dx*dx+dy*dy;
       double B = 2*dx*(x1-x0)+2*dy*(y1-y0);
       double C = (x1-x0)*(x1-x0)+(y1-y0)*(y1-y0)-r*r;
       double delta = B*B-4*A*C;
       num = 0;
       if(cmp(delta)>=0){
          double t1 = (-B-sqrt(delta))/(A*2);
          double t2 = (-B+sqrt(delta))/(A*2);
          ret[num++] = point(x1+t1*dx,y1+t1*dy);
          ret[num++] = point(x1+t2*dx,y1+t2*dy);
       }
    }
```

```
25  void point_of_circle_across_segment(point a,point b,circle o,
       point ret[],int &num){///线段和圆的交点
26     double x0 = o.o.x , y0 = o.o.y;
27     double x1 = a.x , y1 = a.y;
28     double x2 = b.x , y2 = b.y;
29     double r = o.r;
30     double dx = x2-x1 , dy = y2-y1;
31     double A = dx*dx+dy*dy;
32     double B = 2*dx*(x1-x0)+2*dy*(y1-y0);
33     double C = (x1-x0)*(x1-x0)+(y1-y0)*(y1-y0)-r*r;
34     double delta = B*B-4*A*C;
35     num = 0;
36     if(cmp(delta)>=0){
37        double t1 = (-B-sqrt(delta))/(A*2);
38        double t2 = (-B+sqrt(delta))/(A*2);
39        if(cmp(t1-1)<=0&&cmp(t1)>=0)
40           ret[num++] = point(x1+t1*dx,y1+t1*dy);
41        if(cmp(t2-1)<=0&&cmp(t2)>=0)
42           ret[num++] = point(x1+t2*dx,y1+t2*dy);
43     }
44  }
45  circle outer_circle_of_triangle(point t0,point t1,point t2){///
       三角形外接圆
46     circle tmp;
47     double a=(t0-t1).len();
48     double b=(t0-t2).len();
49     double c=(t1-t2).len();
50     tmp.r=a*b*c/4/triangle(t0,t1,t2).area();
51     double a1=t1.x-t0.x;
52     double b1=t1.y-t0.y;
53     double c1=(a1*a1+b1*b1)/2;
54     double a2=t2.x-t0.x;
55     double b2=t2.y-t0.y;
56     double c2=(a2*a2+b2*b2)/2;
57     double d=a1*b2-a2*b1;
58     tmp.o.x=t0.x+(c1*b2-c2*b1)/d;
```

```
59      tmp.o.y=t0.y+(a1*c2-a2*c1)/d;
60      return tmp;
61   }
62   circle min_circle_cover(point p[],int n){///最小圆覆盖
63      random_shuffle(p,p+n);//随机排序取点
64      circle ret;
65      ret.o=p[0];
66      ret.r=0;
67      for(int i=1;i<n;i++){
68         if((p[i]-ret.o).len()>ret.r+eps){
69            ret.o=p[i];
70            ret.r=0;
71            for(int j=0;j<i;j++){
72               if((p[j]-ret.o).len()>ret.r+eps){
73                  ret.o = (p[i]+p[j])/2;
74                  ret.r=(p[i]-p[j]).len()/2;
75                  for(int k=0;k<j;k++)
76                     if((p[k]-ret.o).len()>ret.r+eps)
77                        ret=outer_circle_of_triangle(p[i],p[j],p[k
                              ]);
78               }
79            }
80         }
81      }
82      return ret;
83   }
84   double area_circle_across_circle(circle a, circle b) {///圆的交
85      double d = (a.o-b.o).len();
86      if (d >= a.r + b.r)
87         return 0;
88      if (d <= fabs(a.r - b.r)) {
89         double r = a.r < b.r ? a.r : b.r;
90         return pi * r * r;
91      }
92      double ang1 = acos((a.r * a.r + d * d - b.r * b.r) / 2. / a
         .r / d);
```

```
93        double ang2 = acos((b.r * b.r + d * d - a.r * a.r) / 2. / b
              .r / d);
94        double ret = ang1 * a.r * a.r + ang2 * b.r * b.r - d * a.r
              * sin(ang1);
95        return ret;
96     }
97     bool point_inon_circle(point p,circle c){
98        double dis = (p-c.o).len();
99        return dis<=c.r;
100    }
101 }
102 using namespace _circle;
```

8.7 O(n)-求凸包 & 旋转卡壳

```
1  #include <bits/stdc++.h>
2  struct Point {
3     double x, y;
4  } point[int(2e5) + 7];
5
6  double dist(Point p1, Point p2) { // 两点间距离
7     return sqrt((p1.x - p2.x) * (p1.x - p2.x) + (p1.y - p2.y) * (
          p1.y - p2.y));
8  }
9
10 double cross_product(Point p0, Point p1, Point p2) { // 叉乘
11    return (p1.x - p0.x) * (p2.y - p0.y) - (p2.x - p0.x) * (p1.y
          - p0.y);
12 }
13
14 bool cmp(const Point &p1, const Point &p2) { // 按极角排序
15    double temp = cross_product(point[0], p1, p2);
16    if (fabs(temp) < 1e-6) return dist(point[0], p1) < dist(point
          [0], p2);
17    return temp > 0;
18 }
```

```
19
20 vector<Point> graham_scan(int n) { // 求凸包
21    vector<Point> ch;
22    int top = 2;
23    int index = 0;
24    for (int i = 1; i < n; ++i) {
25       if (point[i].y < point[index].y || (point[i].y == point[
            index].y && point[i].x < point[index].x)) index = i;
26    }
27    swap(point[0], point[index]);
28    ch.push_back(point[0]);
29    sort(point + 1, point + n, cmp);
30    ch.push_back(point[1]);
31    ch.push_back(point[2]);
32    for (int i = 3; i < n; ++i) {
33       while (top > 0 && cross_product(ch[top - 1], point[i], ch[
            top]) >= 0) {
34          --top;
35          ch.pop_back();
36       }
37       ch.push_back(point[i]);
38       ++top;
39    }
40    return ch;
41 }
42
43 double rotating_caliper(vector<Point> v) { // 旋转卡壳求凸包最大直径
44    double max_dis = 0.0;
45    int n = int(v.size());
46    if (n == 2) max_dis = dist(v[0], v[1]);
47    else {
48       v.push_back(v[0]);
49       int j = 2;
50       for (int i = 0; i < n; ++i) {
51          while (cross_product(v[i], v[i + 1], v[j]) <
               cross_product(v[i], v[i + 1], v[j + 1])) {
```

```
52              j = (j + 1) % n;
53          }
54          max_dis = max(max_dis, max(dist(v[j], v[i]), dist(v[j], v
                [i + 1])));
55      }
56   }
57   return max_dis;
58}
59
60int main() {
61   int n;
62   scanf("%d", &n);
63   for (int i = 0; i < n; i++) scanf("%lf%lf", &point[i].x, &point
        [i].y);
64   printf("%lf\n", rotating_caliper(graham_scan(n)));
65   return 0;
66}
```

8.8 两圆相交面积

```
1 struct Circle { double x, y, r; };
2
3 double area(Circle a, Circle b) {
4    double d = sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y
         - b.y));
5    double mi = min(a.r, b.r);
6    double ans;
7    if (d > a.r + b.r || d == a.r + b.r) ans = 0.0;
8    else if (d < abs(a.r - b.r) || d == abs(a.r - b.r)) ans = acos
        (-1.0) * mi * mi;
9    else {
10      double a1, a2, S1, S2, S3, p = (a.r + b.r + d) / 2.0;
11      a1 = acos((a.r * a.r + d * d - b.r * b.r) / (2.0 * a.r * d)
           );
12      a2 = acos((b.r * b.r + d * d - a.r * a.r) / (2.0 * b.r * d)
           );
```

```
13        S1 = a1 * a.r * a.r, S2 = a2 * b.r * b.r;
14        S3 = 2 * sqrt(p * (p - a.r) * (p - b.r) * (p - d));
15        ans = S1 + S2 - S3;
16    }
17    return ans;
18 }
```

8.9 两直线交点

```
1  Point intersection(Point u1, Point u2, Point v1, Point v2) {
2     Point ret = u1;
3     double tmp = ((u1.x - v1.x) * (v1.y - v2.y) - (u1.y - v1.y) *
          (v1.x - v2.x))
4               / ((u1.x - u2.x) * (v1.y - v2.y) - (u1.y - u2.y) * (
                  v1.x - v2.x));
5     ret.x += (u2.x - u1.x) * tmp;
6     ret.y += (u2.y - u1.y) * tmp;
7     return ret;
8  }
```

8.10 点在直线上的垂点

```
1  Point ptoline(Point p, Point line1, Point line2) { // 点在直线上的垂
      点
2     Point t = p;
3     t.x += line1.y - line2.y, t.y += line2.x - line1.x;
4     return intersection(p, t, line1, line2); // 两直线交点
5  }
```

8.11 矩形面积交

```
1  struct Rec {
2     int x1, x2, y1, y2;
3     void init() { scanf("%d%d%d%d", &x1, &y1, &x2, &y2); }
4     int area() { return abs(x2 - x1) * abs(y2 - y1); }
5     int intersection(const Rec &tmp) const {
```

```
6         int tmpx = max((x2 - x1) + (tmp.x2 - tmp.x1) - (max(x2, tmp
              .x2) - min(x1, tmp.x1)), 0);
7         int tmpy = max((y2 - y1) + (tmp.y2 - tmp.y1) - (max(y2, tmp
              .y2) - min(y1, tmp.y1)), 0);
8         return tmpx * tmpy;
9     }
10 };
```

8.12 线段类

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int cmp(double tmp) {
4     if(fabs(tmp) < 1e-9) return 0;
5     return tmp < 0 ? -1 : 1;
6  }
7  struct Segment { // 线段类
8     struct Point { // 点类
9        double x, y;
10       Point(double x = 0, double y = 0):x(x), y(y) {}
11       bool operator == (const Point &tmp) const {
12          return cmp(tmp.x - x) == 0 && cmp(tmp.y - y) == 0;
13       }
14    } u, v; // 线段的起点和终点
15    //两种构造函数
16    Segment(Point u = 0, Point v = 0):u(u), v(v) {}
17    Segment(double x = 0, double y = 0, double a = 0, double b = 0)
          :u(x, y), v(a, b) {}
18    // 叉乘
19    static double CrossProduct(Point i, Point j, Point k) { // i->
          k 叉乘 i->j
20       return (k.x - i.x) * (j.y - i.y) - (k.y - i.y) * (j.x - i.
             x);
21    }
22    static double DotProduct(Point i, Point j, Point k) { // i->k
          点乘 i->j
```

```
23      return (k.x - i.x) * (j.x - i.x) - (k.y - i.y) * (j.y - i.
            y);
24  }
25  // 判断点是否在线上，这个判定函数不具有一般性，只有在叉乘为零时才能使用
26  static bool OnSegment_(Segment l, Point c) {
27      Point a = l.u, b = l.v;
28      return c.x >= min(a.x, b.x) && c.x <= max(a.x, b.x) &&
29           c.y >= min(a.y, b.y) && c.y <= max(a.y, b.y);
30  }
31  static bool OnSegment(Segment l, Point c) { // 如果叉积等于0且点积
        小于0，说明在线段内部
32      if(c == l.u || c == l.v) return true;
33      return cmp(CrossProduct(l.u, l.v, c)) == 0 && cmp(DotProduct
            (c, l.u, l.v)) < 0;
34  }
35  // 判断两线段是否相交，只有严格不相交时才返回false(两线段不存在共点)
36  static bool isIntersect(Segment a, Segment b) {
37      double ab_u = CrossProduct(a.u, a.v, b.u);
38      double ab_v = CrossProduct(a.u, a.v, b.v);
39      double ba_u = CrossProduct(b.u, b.v, a.u);
40      double ba_v = CrossProduct(b.u, b.v, a.v);
41      if(ab_u * ab_v < 0 && ba_u * ba_v < 0) return true; // 如果两
            个线段相互跨越
42      if(ab_u == 0 && OnSegment_(a, b.u)) return true; // 如果叉积
            为零，且点在线上
43      if(ab_v == 0 && OnSegment_(a, b.v)) return true;
44      if(ba_u == 0 && OnSegment_(b, a.u)) return true;
45      return ba_v == 0 && OnSegment_(b, a.v);
46  }
47  // 返回线段的斜率(正切值)
48  double tan() {
49      double tmp = u.x - v.x;
50      return tmp == 0.0 ? 1e9+7 : (u.y - v.y)/tmp;
51  }
52  // 在两线段不平行的前提下求线段交点
53  static Point GetCrossPoint(Segment a, Segment b) {
```

```
54        double tmpLeft, tmpRight, retx, rety;
55        tmpLeft = (b.v.x - b.u.x) * (a.u.y - a.v.y) - (a.v.x - a.u
              .x) * (b.u.y - b.v.y);
56        tmpRight = (a.u.y - b.u.y) * (a.v.x - a.u.x) * (b.v.x - b.u
              .x) +
57              b.u.x * (b.v.y - b.u.y) * (a.v.x - a.u.x) -
58              a.u.x * (a.v.y - a.u.y) * (b.v.x - b.u.x);
59        retx = tmpRight / tmpLeft;
60        tmpLeft = (a.u.x - a.v.x) * (b.v.y - b.u.y) - (a.v.y - a.u
              .y) * (b.u.x - b.v.x);
61        tmpRight = a.v.y * (a.u.x - a.v.x) * (b.v.y - b.u.y) +
62              (b.v.x- a.v.x) * (b.v.y - b.u.y) * (a.u.y - a.v.y) -
63              b.v.y * (b.u.x - b.v.x) * (a.v.y - a.u.y);
64        rety = tmpRight / tmpLeft;
65        return {retx, rety};
66    }
67 };
68
69 int main() {
70    cout << Segment::isIntersect(Segment(0,0,1,1), Segment(0,2,2,0)
          ) << endl; // true
71    cout << Segment::isIntersect(Segment(0,0,1,1), Segment(1,1,2,2)
          ) << endl; // true
72    cout << Segment::isIntersect(Segment(0,0,2,1), Segment(1,0,6,3)
          ) << endl; // false
73    Segment a(0, 0, 1, 1), b(1, 1, 2, 2);
74    if(a.tan() == b.tan()) cout << -1 << endl;
75    else {
76       Segment::Point tmp = Segment::GetCrossPoint(a, b);
77       cout << tmp.x << ' ' << tmp.y << endl;
78    }
79    return 0;
80 }
```

8.13 计算三角形外心

```
//C++
float S(float x1,float y1,float x2,float y2,float x3,float y3){
   return ((x1-x3)*(y2-y3)-(y1-y3)*(x2-x3) );
}

float cal_center_x(float x1,float y1,float x2,float y2,float x3,
   float y3){
   return (S(x1*x1+y1*y1,y1, x2*x2+y2*y2, y2,x3*x3+y3*y3,y3)/(2*S(
      x1,y1,x2,y2,x3,y3)) );
}

float cal_center_y(float x1,float y1,float x2,float y2,float x3,
   float y3){
   return (S(x1, x1*x1+y1*y1, x2, x2*x2+y2*y2, x3, x3*x3+y3*y3) /
      (2*S(x1,y1,x2,y2,x3,y3)));
}

//Java:
public class Main {
   private static BigDecimal S(BigDecimal x1,BigDecimal y1,
      BigDecimal x2,BigDecimal y2,BigDecimal x3,BigDecimal y3) {
      return ((x1.subtract(x3)).multiply(y2.subtract(y3)).subtract
         ((y1.subtract(y3)).multiply(x2.subtract(x3))));
   }
   private static BigDecimal cal_center_x(BigDecimal x1,BigDecimal
       y1,BigDecimal x2,BigDecimal y2,BigDecimal x3,BigDecimal y3)
       {
      return ((S((x1.multiply(x1)).add((y1.multiply(y1))), y1, (x2
         .multiply(x2)).add((y2.multiply(y2))), y2, (x3.multiply(
         x3)).add((y3.multiply(y3))), y3)).divide(S(x1,y1,x2,y2,x3
         ,y3).multiply(BigDecimal.valueOf(2))));
   }
   private static BigDecimal cal_center_y(BigDecimal x1,BigDecimal
       y1,BigDecimal x2,BigDecimal y2,BigDecimal x3,BigDecimal y3)
       {
```

```
23          return ((S(x1, (x1.multiply(x1)).add((y1.multiply(y1))), x2,
                (x2.multiply(x2)).add((y2.multiply(y2))), x3, (x3.
               multiply(x3)).add((y3.multiply(y3))))).divide(S(x1,y1,x2,
               y2,x3,y3).multiply(BigDecimal.valueOf(2))));
24      }
25  }
```

9 STL

9.1 accumulate

```
1  // 定义在<numeric>中，作用有两个，1.累加求和 2.自定义类型数据的处理
2  #include <numeric>
3  /* 累加求和 */
4  int sum = std::accumulate(vec.begin(), vec.end() , 42);
5  // 三个形参：前两个指定要累加的元素范围，第三个形参是累加的初值。
6  string sum = accumulate(vec.begin(), vec.end(), string(""));
7  // 把vec里的所有string都连接起来
8
9  /*自定义数据类型的处理*/
10 struct Grade {
11     string name;
12     int grade;
13 } subject[2] = {{"English", 80}, {"Biology", 70}};
14
15 int main() {
16     int sum = accumulate(subject, subject + 2, 0, [](int tmp, Grade
           b){return a + b.grade; });
17     cout << sum << endl;
18     return 0;
19 }
```

10 其他

10.1 约瑟夫问题

```
1  // time Complexity O(log(n))
```

```
int kth(int n, int m, int k)
{
   if (m == 1) return k;
   for (k = k*m+m-1; k >= n; k = k-n+(k-n)/(m-1));
      return k;
}
```

10.2 RMQ-ST

```
void init(int n) {//预处理复杂度：O(n*lgn)
   //n为原数组的个数，编号为1~n
   memset(maxq, 0, sizeof maxq);
   for(int i=1 ; i<=n ; i++)
      maxq[i][0] = num[i];
   //bitn为n的二进制位数，取下整(int)(log(n)/log(2))
   int bitn = int(log(double(n))/log(2.0));
   for (int j = 1; j <= bitn; ++j) {
      for (int i = 1; i <= n; ++i) {
         if(i+(1<<(j-1)) > n) break;
         maxq[i][j] = max(maxq[i][j-1], maxq[i+(1<<(j-1))][j-1]);
      }
   }
}
int querymax(int l, int r) {
   int bitn = int(log(double(r-l+1))/log(2.0));
   return max(maxq[l][bitn], maxq[r-(1<<bitn)+1][bitn]);
   //这里注意右区间是r-(1<<bitn)+1，不要忘了加一
}
```

10.3 一些理论

```
/*
一些理论：
1.对于三个整数a，b，n，有：(a % (n * b)) % b = a % b
2.给你一个无向图，让你求最小环的长度
   先求最小生成树，然后枚举每一条没有被加入生成树的边，求这些边的两端点在最小
      生成树上的距离。
```

```
3.从a到b之间有多条路径，规定每条路径的权值为路径上最小的边的权值，现在让你求a到b的所有路径里的最大权
    先求最大生成树，然后取最小树边
*/
```

10.4 随机数和文件输出

```cpp
#include <iostream>
#include <random>
#include <fstream>
using namespace std;
int main() {
    ofstream fout;//定义流文件输出
    fout.open("test.out");//打开文件
    random_device rd;//定义一种随机数引擎（产生器）
    default_random_engine gen = default_random_engine(rd());//设置默认引擎
    uniform_int_distribution<int> dis1(1,10);//设置类型范围
    uniform_real_distribution<double> dis2(1,10);//设置类型范围
    auto rand1 = bind(dis1,gen);
    //调用rand1()或者是dis1(gen)来产生1到10之间的整数
    auto rand2 = bind(dis2,gen);
    //调用rand2()或者是dis2(gen)来产生1到10之间的浮点数
    fout << "Hello World!" << endl;
    fout.close();//关闭文件
    return 0;
}
```

10.5 随机遍历数组

```cpp
#include <iostream>
#include <cstdlib> // srand rand
#include <ctime> // time
#include <algorithm>
using namespace std;

```

```

/**
 * 随机遍历数组
 */
void Traverse_Random(int arr[], int n) {
   srand(time(NULL));
   for(int i=n-1; i>=0; --i) {
      int j = rand() % (i+1);
      cout << arr[j] << " "; // 输出
      swap(arr[j], arr[i]);
   }
}

int main() {
   int arr[9] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
   Traverse_Random(arr, 9);
   getchar();
   return 0;
}
```

10.6 尺取

```
int l = 0, r= 0, sum = 0, ans=n+1;
while(true){
   while(r < n && sum < s) sum+=a[r++];//从前面开始
   if(sum < s) break;//区间和小于s;
   ans = min(r-l,ans);//ans就是最小长度
   sum -= a[l++];//尺取法的前进;
}
```

10.7 std::unordered_map 避免 TLE

```
unordered_map<int,int>mp;
mp.reserve(1024);
mp.max_load_factor(0.25);
```

10.8 输入输出挂

(by shui)

```
namespace Fread {
   const int CACHE = 1 << 16;
   char buf[CACHE];
   size_t curl, curr;
   inline char get() {
      if (curl == curr) {
         curl = 0, curr = fread(buf, 1, CACHE, stdin);
         if (curr == 0) return EOF;
      }
      return buf[curl++];
   }
   template <typename type>
   inline bool read(type &x, char c = '0') {
      int flag = 1;
      do {
         c = get();
         if (c == '-') flag = -1;
      } while (c < '0' || c > '9');
      do x = x * 10 + (c ^ '0'); while (c = get(), '0' <= c && c
         <= '9');
      x *= flag;
      return c != EOF;
   }
}
namespace FastIO {
   const int SIZE = 1 << 16;
   char buf[SIZE], str[64];
   int l = SIZE, r = SIZE;
   int read(char *s) {
      while (r) {
         for (; l < r && buf[l] <= ' '; l++);
         if (l < r) break;
         l = 0, r = int(fread(buf, 1, SIZE, stdin));
```

```
33        }
34        int cur = 0;
35        while (r) {
36           for (; l < r && buf[l] > ' '; l++) s[cur++] = buf[l];
37           if (l < r) break;
38           l = 0, r = int(fread(buf, 1, SIZE, stdin));
39        }
40        s[cur] = '\0';
41        return cur;
42     }
43     template<typename type>
44     bool read(type &x, int len = 0, int cur = 0, bool flag = false)
           {
45        if (!(len = read(str))) return false;
46        if (str[cur] == '-') flag = true, cur++;
47        for (x = 0; cur < len; cur++) x = x * 10 + str[cur] - '0';
48        if (flag) x = -x;
49        return true;
50     }
51     template <typename type>
52     type read(int len = 0, int cur = 0, bool flag = false, type x =
           0) {
53        if (!(len = read(str))) return false;
54        if (str[cur] == '-') flag = true, cur++;
55        for (x = 0; cur < len; cur++) x = x * 10 + str[cur] - '0';
56        return flag ? -x : x;
57     }
58  } using FastIO::read;
```

(by qi)

```
1  namespace Input
2  {
3     const int BUF = 65536;
4     char buf[BUF + 1];
5     char *head = buf, *tail = buf;
6  }
```

```
inline char inputchar()
{
   using namespace Input;
   if(head == tail)
      *(tail = (head = buf) + fread(buf, 1, BUF, stdin)) = 0;
   return *head++;
}
inline void inputnum(int &ret)
{
   char ch = inputchar();
   while(ch < '0' || ch > '9')
      ch = inputchar();
   ret = ch - '0';
   ch = inputchar();
   while(ch >= '0' && ch <= '9')
   {
      ret = ret * 10 + ch - '0';
      ch = inputchar();
   }
}
// 使用了快速读入之后不能再使用cin或者scanf
```

10.9 CDQ 分治

```
// luogu3810 求满足x<=a,y<=b,z<=c的点的个数
// time complexity: O(nlog^2(n))
#include <bit/stdc++.h>
using namespace std;
const int maxn=100050;
struct node {
   int x,y,z,w;
   int sum;
   void read() {
      scanf("%d%d%d", &x,&y,&z);
   }
   bool operator <(const node&t)const {
```

```
13        if(x!=t.x) return x<t.x;
14        if(y!=t.y) return y<t.y;
15        return z<t.z;
16     }
17     bool operator !=(const node&t)const {
18        return x!=t.x||y!=t.y||z!=t.z;
19     }
20 }a[maxn],t[maxn];
21 int M,n;
22 bool cmpy(const node&a,const node&t) {
23     return a.y<t.y;
24 }
25 int sum[maxn*2];
26 void add(int x,int val) {
27     while(x<=M) {
28        sum[x]+=val;
29        x+=(x&-x);
30     }
31 }
32 int query(int x) {
33     int ans=0;
34     while(x) {
35        ans+=sum[x];
36        x-=(x&-x);
37     }
38     return ans;
39 }
40 void solve(int l,int r) {
41     if(l==r) return;
42     int mid=(l+r)>>1;
43     solve(l,mid),solve(mid+1,r);
44     // sort(a+1,a+mid+1,cmpy),sort(a+mid+1,a+r+1,cmpy);
45     int p=l;
46     for(int i = mid+1; i <= r; ++i) {
47        while(p<=mid&&a[p].y<=a[i].y) add(a[p].z,a[p].w),++p;
48        a[i].sum+=query(a[i].z);
```

```
49    }
50    for(int i = l; i < p; ++i) add(a[i].z,-a[i].w);
51    merge(a+l,a+mid+1,a+mid+1,a+r+1,t,cmpy);
52    for(int i = l; i <= r; ++i) a[i]=t[i-l];
53 }
54 int ans[maxn];
55 int main() {
56    scanf("%d%d", &n,&M);
57    int n0=n;
58    for(int i = 1; i <= n; ++i) {
59       a[i].read();
60    }
61    sort(a+1,a+1+n);
62    int tot=0;
63    for(int i = 1; i <= n; ++i) {
64       if(!tot||t[tot]!=a[i]) t[++tot]=a[i],t[tot].w=1;
65       else t[tot].w++;
66    }
67    swap(a,t);
68    n=tot;
69    solve(1,n);
70    for(int i = 1; i <= n; ++i) {
71       ans[a[i].sum+a[i].w-1]+=a[i].w;
72    }
73    for(int i = 0; i < n0; ++i) {
74       printf("%d\n", ans[i]);
75    }
76    return 0;
77 }
```

10.10 vim 配置

```
1 set cin nu ts=4 sw=4 sts=4 noswapfile
2 imap {<cr> {<cr>}<c-o>O<left><right>
3
4 map <F9> :call CR()<CR>
```

```
func! CR()
exec "w"
exec "! clear && g++ -std=c++11 -W % -o a && ./a "
endfunc

map <F5> :call Run()<CR>
func! Run()
exec "w"
exec "! clear && g++ -std=c++11 -W % -o a && ./a < in"
endfunc

map <C-A> ggVG"+y
```

10.11 程序对拍器

```
echo "compiling..."
if !(g++ gen.cpp -o gen.out && g++ code1.cpp -o code1.out && g++
   code2.cpp -o code2.out)
then
   echo "compilation failed, program exiting..."
   exit
fi

printf "enter number of attempts:"
read N

RAND_FILE=rand.txt
i=1
while [ $i -le $N ];
do
   FILENAME=attempt_${i}.txt
   printf "attempt `echo $i`: generating..."
   echo $i > $RAND_FILE
   ./gen.out < $RAND_FILE > $FILENAME
   printf "code1 running..."
   ./code1.out < $FILENAME > code1_output.txt
```

```
    printf "code2 running..."
    ./code2.out < $FILENAME > code2_output.txt

    if diff code1_output.txt code2_output.txt
    then
        printf "passed.\n"
        rm $FILENAME
    else
        printf "OUTPUT NOT IDENTICAL!\n"
    fi

    i=$((i+1))
done

rm code1_output.txt code2_output.txt
```

10.12 简易对排器

```
#!/bin/bash
g++ gen.cpp -o gen
g++ std.cpp -o std
g++ user.cpp -o user
while true; do
    ./gen > data.in
    ./std < data.in > std.out
    ./user < data.in > user.out
    if diff std.out user.out; then
        date
    else
        printf "WA\n"
        exit 0
    fi
done
```

中文 english