Programming contest notebook

Luiz Felipe Abrão Reis

Contents

1 Strings

1.1 KMP $< \mathcal{O}(N + K) >$ - I Love Strings!

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

char s[100010], w[1001];
int k, q, t[1001];

void build(char *w) {
    int pos = 2, cnd = 0;
    int sz = strlen(w);
    t[0] = -1; t[1] = 0;
    while(pos < sz) {
        if (w[pos-1] == w[cnd]) t[pos++] = ++cnd;
        else if (cnd > 0) cnd = t[cnd];
        else t[pos++] = 0;
    }
}

int kmp(char *w, char *s) {
    int m = 0, i = 0;
    int sz = strlen(s);
    int szt = strlen(w);
    while (m+i < sz) {
        if (w[i] == s[m+i]) {
            if (i++ == szt-1) return m;
        } else {
            if (t[i] > -1) m = m+i-t[i], i = t[i];
            else m = m+1, i = 0;
        }
    }
    return -1;
}

int main() {
    scanf("%d", &k);
    while (k--) {
        scanf("%s", s);
        scanf("%d", &q);
        while (q--) {
            scanf("%s", w);
            build(w);
            printf("%c\n", (kmp(w, s) != -1)?'y':'n');
        }
    }
}
```

1.2 Aho Corasick $< \mathcal{O}(N + K + Z) >$ - Cultivando Strings

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

const int MAX = 1000100;
const int MAXC = 26;
int trie[MAX][MAXC];
int match[MAX], fail[MAX];
int str[MAX];
char words[10003][1003];
int p[10009], sz[10009], mem[10009];
int n;

void build() {
    // only memset if the state will be used
    memset(trie[0], 0, sizeof(int)*MAXC);
    int state = 0;
    for(int i = 0; i < n; i++) {
        char *s = words[i];
        int v = 0;
```

```
20          for(int j = 0; s[j]; j++) {
21              if(!trie[v][s[j]-'a']) {
22                  trie[v][s[j]-'a'] = ++state;
23                  // only memset if the state will be used
24                  memset(trie[state], 0, sizeof(int)*MAXC);
25                  match[state] = 0;
26              }
27              v = trie[v][s[j]-'a'];
28          }
29          match[v] = 1;
30          str[v] = i;
31      }
32      queue<int> q;
33      for(int i = 0; i < MAXC; i++)
34          if(trie[0][i])
35              q.push(trie[0][i]), fail[trie[0][i]] = 0;
36      while(!q.empty()) {
37          int stt = q.front(); q.pop();
38          for(int i = 0; i < MAXC; i++) {
39              if(!trie[stt][i]) continue;
40              int &p = fail[trie[stt][i]];
41              p = fail[stt];
42              while(p && !trie[p][i]) p = fail[p];
43              p = trie[p][i];
44              if(match[p]) match[trie[stt][i]] |= 2;
45              q.push(trie[stt][i]);
46          }
47      }
48  }
49
50  int pd() {
51      int ans = 0;
52      // when a word of length l is being calculated, all words of length m
53      // with l < m, have been previously calculated
54      for(int i = 0; i < n; i++) {
55          int k = p[i];
56          char *s = words[k]; int m = sz[k];
57          int v = 0;
58          mem[k] = 1;
59          for(int j = 0; j < m; j++) {
60              v = trie[v][s[j]-'a'];
61              int state = v;
62              if(j == m - 1) state = fail[state];
63              // track end of words using fail function
64              while(match[state]) {
65                  if(match[state] & 1) mem[k] = max(mem[k], 1 + mem[str[state]]);
66                  state = fail[state];
67              }
68          }
69          ans = max(ans, mem[k]);
70      }
71      return ans;
72  }
73
74  int cmp(const int i, const int j) { return sz[i] < sz[j]; }
75
76  int main() {
77      while(scanf("%d", &n) == 1 && n) {
78          for(int i = 0; i < n; i++) {
79              scanf("%s", words[i]);
80              sz[i] = strlen(words[i]);
81              p[i] = i;
82          }
83          // sort in order to execute a bottom-up dp
84          sort(p, p + n, cmp);
85
86          build();
87
88          printf("%d\n", pd());
89      }
90  }
```

2 Math

2.1 Modular Inverse $< \mathcal{O}(\log N) >$ - Jupiter Ataca!

```
#include <cstdio>
#include <cstring>
#include <algorithm>

#define MAX 100010

using namespace std;

long long tree[100010];
long long vect[100010];
long long b, p, l, n;

long long gcd (long long a, long long b) {
    while (b) { long long t = a%b; a = b; b = t; }
    return a;
}

long long lcm (long long a, long long b) {
    return a / gcd(a, b)*b;
}

long long powermod(long long a, long long b, long long m) {
    long long ret = 1;
    while (b) {
        if (b&1) ret = (ret*a) % m;
        a = (a*a) % m;
        b>>=1;
    }
    return ret % m;
}

long long extended_euclid(long long a, long long b, long long &x, long long &y) {
    long long xx = y = 0;
    long long yy = x = 1;
    while (b) {
        long long q = a/b;
        long long t = b; b = a%b; a = t;
        t = xx; xx = x-q*xx; x = t;
        t = yy; yy = y-q*yy; y = t;
    }
    return a;
}

long long mod_inverse (long long a, long long n) {
    long long x, y;
    long long g = extended_euclid(a, n, x, y);
    if (g>1) return -1;
    return (x+n) % n;
}

void update(long long idx ,long long val){
    long long old = vect[idx];
    vect[idx] = val;
    while (idx <= MAX){
        tree[idx] = (tree[idx] + p - old + val) % p;
        idx += (idx & -idx);
    }
}

long long read(long long idx){
    long long sum = 0;
    while (idx > 0){
        sum = (sum + tree[idx]) % p;
        idx -= (idx & -idx);
    }
    return sum;
}

```

```
69  int main () {
70      while (1) {
71          scanf("%lld %lld %lld %lld", &b, &p, &l, &n);
72          if (!b && !p && !l && !n) break;
73          memset(tree, 0, sizeof(tree));
74          memset(vect, 0, sizeof(vect));
75          //printf("byte(%d) prime(%d) length(%d) queries(%d)\n", b, p, l, n);
76          for (long long i = 0; i < n; i++) {
77              // comandos
78              char cmd[2];
79              long long x, y;
80              scanf("%s %lld %lld", cmd, &x, &y);
81              //printf("%c, byte/init(%d), value/end(%d)\n", cmd[0], x, y);
82              if (cmd[0] == 'E') {
83                  // calc base
84                  long long base = powermod(b, l-x, p);
85                  //printf("base = %lld\n", base);
86                  long long value = (y*base) % p;
87                  //printf("value = %lld(%lld)\n", value, y*base);
88                  update(x, value);
89              } else {
90                  long long interval = (read(y) + p - read(x-1)) % p;
91                  //printf("interval sum %lld %lld = %lld - %lld = %lld\n", y, x-1,
        read(y), read(x-1), interval);
92                  long long mod_inv = mod_inverse(powermod(b, l-y, p), p);
93                  //printf("mod inv = %lld\n", mod_inv);
94                  long long hash = interval * mod_inv;
95                  printf("%lld\n", hash % p);
96              }
97          }
98          printf("-\n");
99      }
100     return 0;
101 }
```

2.2 Pisano Period - Crescimento das Populações de Bacilos

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main () {
5      int t, n, r;
6      char in[1000001];
7      int fib[1501] = {0, 1};
8      // pre calc fib
9      for (int i = 2; i <= 1500; i++) fib[i] = (fib[i-1] + fib[i-2])%1000;
10     scanf("%d", &t);
11     while (t--) {
12         scanf("%s", in);
13         n = strlen(in); r = 0;
14         for (int i = 0; i < n; i++)
15             r = (r*10 + in[i]-'0')%1500;
16         printf("%03d\n", fib[r]);
17     }
18     return 0;
19 }
```

2.3 Sieve of Eratosthenes $< \mathcal{O}(N \log\log N) >$ - Conte os fatores

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int siev[1000000];
5  vector<int> p;
6
7  int main() {
8      int n;
9      memset(siev, -1, sizeof(siev));
10     for (int i=2; i<=1000000; i++)
```

```
11          if (siev[i])
12              for (int j = i+i; j <= 1000000; j+=i) siev[j] = 0;
13      for (int i = 2; i <=1000000; i++)
14          if (siev[i]) p.push_back(i);
15      while (scanf("%d", &n) && n) {
16          int c = 0;
17          for (int i = 0; i<p.size(); i++)
18              if (!(n%p[i])) c++;
19          printf("%d : %d\n", n, c);
20      }
21  }
```

3 Graphs

3.1 Least Common Ancestor $< \mathcal{O}(N * \log N), \mathcal{O}(\log N) >$ - Ant's Colony

```cpp
#include <cstdio>
#include <vector>

using namespace std;

long long int dist[100001];
int n, q, temp1, temp2;
vector<int> adj[100001];
int L[100001], P[100001][17], T[100001];

void dfs (int v) {
  for (int i = 0; i < adj[v].size(); i++) {
    //printf("dist(%lld) = %lld\n", v, dist[v]);
      dist[adj[v][i]] = dist[v] + dist[adj[v][i]];
      dfs(adj[v][i]);
  }
}

void processanc () {
  for (int i=0; i< n; i++)
    for(int j=0;1<<j < n; j++)
      P[i][j] = -1;

  for (int i=0; i<n;i++)
    P[i][0] = T[i];

  for (int j =1; 1<<j < n; j++)
    for (int i=0;i<n;i++)
      if(P[i][j -1] != -1)
        P[i][j] = P[P[i][j-1]][j-1];
}

int lca(int p, int q) {
  int tmp, log, i;

  if(L[p] < L[q])
    tmp = p, p =q, q=tmp;

  for (log = 1; 1 << log <= L[p]; log++);
  log--;

  for(i=log; i >=0; i--)
    if(L[p] - (1<<i) >= L[q])
      p = P[p][i];

  if (p == q)
    return p;

  for (i=log; i>= 0; i--)
    if(P[p][i] != -1 && P[p][i] != P[q][i])
      p = P[p][i], q = P[q][i];

  return T[p];
}

long long int calcdist (int a, int b){
  return dist[a] + dist[b] - 2 * dist[lca(a, b)];
}

int main () {
  start:
  dist[0] = 0;
  L[0] = 0;
  T[0] = -1;
  scanf("%d", &n);
  for (int i =0; i < n; i++) adj[i].clear();
  if (n == 0) return 0;
  for (int i=1; i < n; i++) {
```

```
69        scanf("%d %d", &temp1, &temp2);
70        adj[temp1].push_back(i);
71        dist[i] = temp2;
72        // prepara pre processamento da lca
73        T[i] = temp1;
74        L[i] = L[temp1]+1;
75      }
76
77      processanc();
78      dfs(0);
79
80      /*for ( long long int i =0; i < n; i++)
81        printf("%lld\n", dist[i]);*/
82
83      scanf("%d", &q);
84
85      for (int i=0; i<q; i++) {
86        scanf("%d %d", &temp1, &temp2);
87          if (i) printf(" ");
88        printf("%lld", calcdist(temp1, temp2));
89      }
90      printf("\n");
91      goto start;
92
93      return 0;
94    }
```

3.2 Least Common Ancestor $< \mathcal{O}(N * \log N)$, $\mathcal{O}(\log N) >$ - Nlogonian Tickets

```
1   #include <cstdio>
2   #include <vector>
3   #include <cstring>
4
5   using namespace std;
6
7   #define MAX(a,b) ((a)>(b))?(a):(b)
8
9   int n, q, temp1, temp2, temp3, aux;
10  vector< pair<int, int> > adj[100001];
11  int L[100001], P[100001][17], H[100001][17], T[100001];
12
13  void processanc () {
14    for (int i=0; i< n; i++)
15      for(int j=0;1<<j < n; j++)
16        P[i][j] = -1;
17
18    for (int i=0; i<n;i++)
19      P[i][0] = T[i];
20
21    for (int j =1; 1<<j < n; j++)
22      for (int i=0;i<n;i++)
23        if(P[i][j -1] != -1) {
24          H[i][j] = MAX(H[i][j - 1], H[P[i][j - 1]][j-1]);
25          P[i][j] = P[P[i][j-1]][j-1];
26        }
27  }
28
29  int lca(int p, int q) {
30    int tmp, log, i, t = 0;
31
32    if(L[p] < L[q])
33      tmp = p, p =q, q=tmp;
34
35    for (log = 1; 1 << log <= L[p]; log++);
36    log--;
37
38    for(i=log; i >=0; i--)
39      if(L[p] - (1<<i) >= L[q]) {
40        if (t < H[p][i]) t = H[p][i];
41        p = P[p][i];
```

```
42      }
43
44    if (p == q)
45      return t;
46
47    for (i=log; i>= 0; i--)
48      if(P[p][i] != -1 && P[p][i] != P[q][i]) {
49        if (t < H[p][i]) t = H[p][i];
50        if (t < H[q][i]) t = H[q][i];
51        p = P[p][i], q = P[q][i];
52      }
53
54    if (t < H[p][0]) t = H[p][0];
55    if (t < H[q][0]) t = H[q][0];
56    return t;
57  }
58
59  bool vis[100001];
60
61  void root (int v) {
62    if (vis[v]) return;
63    vis[v] = true;
64    for (int i=0; i < adj[v].size(); i++) {
65      if (vis[adj[v][i].first]) continue;
66      L[adj[v][i].first] = L[v]+1;
67      T[adj[v][i].first] = v;
68      H[adj[v][i].first][0] = adj[v][i].second;
69      root (adj[v][i].first);
70    }
71  }
72
73  int main () {
74    start:
75    scanf("%d", &n);
76    memset(vis, false, sizeof(vis));
77    for (int i =0; i <= n; i++) adj[i].clear();
78    if (n == 0) return 0;
79    for (int i=1; i < n; i++) {
80      scanf("%d %d %d", &temp1, &temp2, &temp3);
81      temp1--;
82      temp2--;
83      adj[temp2].push_back(pair<int, int> (temp1, temp3));
84      adj[temp1].push_back(pair<int, int> (temp2, temp3));
85    }
86
87    L[0] = 0;
88    T[0] = -1;
89    H[0][0] = 0;;
90
91    root(0);
92
93    processanc();
94
95    scanf("%d", &q);
96
97    for (int i=0; i<q; i++) {
98      scanf("%d %d", &temp1, &temp2);
99      printf("%d\n", lca(--temp1, --temp2));
100   }
101   goto start;
102
103   return 0;
104 }
```

3.3 Strongly Connected Components $< \mathcal{O}(V + E) >$ - Ir e Vir

```
1  #include <stdio.h>
2  #include <vector>
3  #include <stack>
4  #include <string.h>
5
```

```
6  using namespace std;
7
8  #define MIN(a, b) ((a) < (b))?(a):(b)
9
10 int N, M, v, w, _w, p, C, idx;
11 int I[2001], LL[2001];
12 bool OS[2001];
13 stack<int> S;
14 vector<int> G[2001];
15
16 void strongconnect (int _v) {
17     I[_v] = idx;
18     LL[_v] = idx;
19     idx++;
20     S.push(_v);
21     OS[_v] = true;
22
23     for (int i = 0; i < G[_v].size(); i++) {
24         if (I[G[_v][i]] == -1) {
25             strongconnect(G[_v][i]);
26             LL[_v] = MIN(LL[_v], LL[G[_v][i]]);
27         } else if (OS[G[_v][i]]) {
28             LL[_v] = MIN(LL[_v], I[G[_v][i]]);
29         }
30     }
31
32     if (LL[_v] == I[_v]) {
33         // start new scc
34         C++;
35         do {
36             _w = S.top();
37             S.pop();
38             OS[_w] = false;
39         } while (_w != _v);
40     }
41
42 }
43
44 int main () {
45     scanf("%d %d", &N, &M);
46     while (N != 0 || M != 0) {
47         idx = 0;
48         C = 0;
49         for (int i = 1; i <= N; i++) G[i].clear();
50         memset(I, -1, sizeof(I));
51         memset(LL, -1, sizeof(LL));
52         memset(OS, false, sizeof(OS));
53         for (int i = 0; i < M; i++) {
54             scanf("%d %d %d", &v, &w, &p);
55             G[v].push_back(w);
56             if (p == 2) G[w].push_back(v);
57         }
58
59         for (int i = 1; i <= N; i++) {
60             if (I[i] == -1) {
61                 strongconnect(i);
62             }
63         }
64
65         printf("%d\n", (C>1)?0:1);
66         scanf("%d %d", &N, &M);
67     }
68     return 0;
69 }
```

3.4 Max Flow $< \mathcal{O}(V * E) >$ - Internet Bandwidth

```
1 #include <stdio.h>
2 #include <vector>
3 #include <queue>
4 #include <string.h>
```

```
5
6  using namespace std;
7
8  #define MIN(a, b) ((a) < (b))?(a):(b)
9  #define INFINITY 0x3f3f3f3f
10
11 int n, s, t, c, x, y, b, id = 0;
12 long long int mflow = 0, pathcap;
13 vector<int> G[101];
14 long long int cap[101][101];
15
16 int findpath() {
17     queue<int> Q;
18     int F[101], w, p, n;
19     bool V[101];
20     memset(V, false, sizeof(V));
21     memset(F, -1, sizeof(F));
22     Q.push(s);
23     V[s] = true;
24
25     while (!Q.empty()) {
26         w = Q.front();
27         Q.pop();
28         for (int i = 0; i < G[w].size(); i++) {
29             n = G[w][i];
30             if (!V[n] && cap[w][n] > 0) {
31                 Q.push(n);
32                 V[n] = true;
33                 F[n] = w;
34                 if (n == t) goto end_while;
35             }
36         }
37     }
38     end_while:
39     w = t;
40     pathcap = INFINITY;
41     while (F[w] != -1) {
42         p = F[w];
43         pathcap = MIN(pathcap, cap[p][w]);
44         w = p;
45     }
46     w = t;
47     while (F[w] != -1) {
48         p = F[w];
49         cap[p][w] -= pathcap;
50         cap[w][p] += pathcap;
51         w = p;
52     }
53     if (pathcap == INFINITY) return 0;
54     else return pathcap;
55 }
56
57 int main () {
58     scanf("%d", &n);
59     while (n != 0) {
60         memset(cap, 0, sizeof(cap));
61         mflow = 0;
62         printf("Network %d\n", ++id);
63         scanf("%d %d %d", &s, &t, &c);
64         for (int i = 0; i < c; i++) {
65             scanf("%d %d %d", &x, &y, &b);
66             G[x].push_back(y);
67             G[y].push_back(x);
68             cap[x][y] += b;
69             cap[y][x] += b;
70         }
71
72         while (true) {
73             pathcap = findpath();
74             if (pathcap == 0) break;
75             else mflow += pathcap;
76         }
77
```

```
78          printf("The bandwidth is %lld.\n\n", mflow);
79          scanf("%d", &n);
80      }
81      return 0;
82  }
```

3.5 Dijkstra $< \mathcal{O}(V + E * \log E) >$ - Engarrafamento

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <queue>
4
5  using namespace std;
6
7  int G[110][110];
8  bool V[110];
9  int N, M, D = -1;
10 int a, b, t, c, d;
11 priority_queue< pair<int, int>, vector< pair<int, int> >, greater< pair<int, int>
       > > Q;
12
13 int main () {
14     scanf("%d %d", &N, &M);
15     while (N != 0 || M != 0) {
16         D = -1;
17         Q = priority_queue< pair<int, int>, vector< pair<int, int> >, greater<
       pair<int, int> > >();
18         memset(G, -1, sizeof(G));
19         memset(V, false, sizeof(V));
20         for (int i = 0; i < M; i++) {
21             scanf("%d %d %d", &a, &b, &t);
22             G[a][b] = t;
23         }
24         scanf("%d %d", &a, &b);
25
26         Q.push(make_pair(0, a));
27         while (!Q.empty()) {
28             if (!V[Q.top().second]) {
29                 c = Q.top().second;
30                 d = Q.top().first;
31                 V[c] = true;
32                 if (c == b) {
33                     D = d;
34                     break;
35                 }
36
37                 for (int i = 0; i <= N; i++)
38                     if (G[c][i] != -1)
39                         Q.push(make_pair(d + G[c][i], i));
40             }
41             Q.pop();
42         }
43
44         printf("%d\n", D);
45
46         scanf("%d %d", &N, &M);
47     }
48     return 0;
49 }
```

3.6 Dijkstra with bitmask - Desafio das Moedas Prateadas

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  typedef pair<int, int> pii;
5  typedef pair<pii, pii> piiii;
6
7  const int MAXN = 1005;
```

```
8  const int MAXK = 12;
9  const int INF = 0x3f3f3f3f;
10
11 int n, m, k, coin[MAXN];
12 vector<pii> adj[MAXN], g[MAXK+1];
13 bool vis[MAXN];
14 int mem[4][(((1<<MAXK)-1)<<1)+10][MAXK+1];
15
16 void dij(int r) {
17     int addz = 1;
18     memset(vis, 0, sizeof(vis));
19     priority_queue< pair<int, int>, vector< pair<int, int> >, greater< pair<int,
       int> > > Q;
20     Q.push(make_pair(0, r));
21     while(!Q.empty()) {
22         int d = Q.top().first;
23         int v = Q.top().second;
24         Q.pop();
25         if (v == 0 && v == r && addz && d > 0) {
26             g[coin[r]].push_back(make_pair(v, d));
27             addz = 0;
28         }
29         if (vis[v]) continue;
30         vis[v] = 1;
31         //printf("dij for %d: coin is %d and v is %d | dist is %d\n", r, coin[v],
       v, d);
32         if (coin[v] && v != r) g[coin[r]].push_back(make_pair(coin[v], d));
33         if (v == 0 && v != r) {
34             g[coin[r]].push_back(make_pair(v, d));
35             continue;
36         }
37         for (int i = 0; i < adj[v].size(); i++)
38             Q.push(make_pair(d + adj[v][i].second, adj[v][i].first));
39     }
40 }
41
42 int pd() {
43     priority_queue <pair<pair<long long, int>, pair<int, int> >, vector <pair<
       pair<long long, int>, pair<int, int> > >, greater < pair<pair<long long, int
       >, pair<int, int> > > > Q;
44     memset(mem, 0, sizeof mem);
45     Q.push(make_pair(make_pair(0, 0), make_pair(0, 0)));
46     while(!Q.empty()){
47         long long d = Q.top().first.first;
48         int vol = Q.top().first.second;
49         int bm = Q.top().second.first;
50         int u = Q.top().second.second;
51         Q.pop();
52
53         if(u==0) vol++;
54         //printf("dist %lld vol %d, bm %x(%x), u %d\n", d, vol, bm,(((1<<k)-1)
       <<1) , u);
55         if(vol == 4 && u == 0 && bm == ((1<<k)-1)<<1) return d;
56         if(vol == 4) continue;
57
58         if(mem[vol][bm][u]) continue;
59         mem[vol][bm][u] = 1;
60
61         for(int i=0; i<g[u].size(); i++){
62             int v = g[u][i].first;
63             int w = g[u][i].second;
64             int bt = bm;
65             if(v) bt |= (1<<v);
66             Q.push(make_pair(make_pair(d+w, vol), make_pair(bt, v)));
67         }
68     }
69
70     return -1;
71 }
72
73 int main() {
74     memset(coin, 0, sizeof(coin));
75     int a, b, t;
```

```
76      scanf("%d %d %d", &n, &m, &k);
77      for (int i = 0; i < m; i++) {
78          scanf("%d %d %d", &a, &b, &t);
79          a--, b--;
80          adj[a].push_back(make_pair(b, t));
81      }
82      for (int i = 1; i <= k; i++) {
83          scanf("%d", &t);
84          coin[--t] = i;
85      }
86      dij(0);
87      for (int i = 1; i < n; i++) {
88          if(coin[i]) dij(i);
89      }
90      /*for (int i = 0; i <= k; i++) {
91          printf("%d adj list: ", i);
92          for (int j = 0; j < g[i].size(); j++) {
93              printf("%d(%d) ", g[i][j].first, g[i][j].second);
94          }
95          printf("\n");
96      }*/
97      long long ans = pd();
98      if (ans == -1) printf("impossivel\n");
99      else printf("%lld\n", ans);
100     return 0;
101 }
```

3.7 Articulation Points $< \mathcal{O}(V + E) >$ - Manutenção

```
1  #include <cstdio>
2  #include <cstring>
3  #include <vector>
4
5  #define MAX 401
6
7  using namespace std;
8
9  int n, m, time_s, vis[MAX], ans[MAX];
10 vector <int> adj[MAX];
11
12 int dfs(int u) {
13     int low = time_s, cc = 0;
14     vis[u] = time_s++;
15     for (int i = 0; i < adj[u].size(); i++) {
16         if (!vis[adj[u][i]]) {
17             cc++;
18             int low_i = dfs(adj[u][i]);
19             low = min(low, low_i);
20             if (vis[u] <= low_i && (u != 1 || cc > 1)) ans[u] = 1;
21         } else {
22             low = min(low, vis[adj[u][i]]);
23         }
24     }
25     return low;
26 }
27
28 void get_art() {
29     time_s = 1;
30     memset(ans, 0, sizeof(ans));
31     memset(vis, 0, sizeof(vis));
32     dfs(1);
33 }
34
35 int main() {
36     scanf("%d %d", &n, &m);
37     for (int t = 1; n; ++t) {
38         int x, y, p = 0;
39         for (int i = 0; i < MAX; i++) adj[i].clear();
40
41         for (int i = 0; i < m; i++) {
42             scanf("%d %d", &x, &y);
```

```
43              adj[x].push_back(y);
44              adj[y].push_back(x);
45          }
46
47          get_art();
48
49          printf("Teste %d\n", t);
50          for (int i = 1; i <= n; i++) {
51              if (ans[i]) {
52                  if (p++) printf(" ");
53                  printf("%d", i);
54              }
55          }
56          if (!p) printf("nenhum");
57          printf("\n\n");
58          scanf("%d %d", &n, &m);
59      }
60  }
```

3.8 Bellman-Ford $< \mathcal{O}(V * E^2) >$ - Haunted Graveyard

```
1   #include <cstdio>
2   #include <cstring>
3   #include <vector>
4   #include <queue>
5
6   #define INF 0x3f3f3f3f
7   #define MAX 30
8
9   using namespace std;
10
11  int w, h, g, e;
12  int dist[MAX][MAX], r[MAX][MAX], vis[MAX][MAX][MAX][MAX];
13  vector< pair< pair<int, int> , int> > adj[MAX][MAX];
14
15  int bellman_ford(int x, int y, int dx, int dy) {
16      int vc = w*h;
17
18      memset(dist, 0x3f, sizeof(dist));
19      dist[x][y] = 0;
20
21      for (int a = 1; a < vc; a++) {
22          for (int i = 0; i < w; i++) {
23              for (int j = 0; j < h; j++) {
24                  if (i == dx && j == dy) continue;
25                  for (int k = 0; k < adj[i][j].size(); k++) {
26                      if (dist[i][j] != INF && dist[i][j] + adj[i][j][k].second <
        dist[adj[i][j][k].first.first][adj[i][j][k].first.second]) {
27                          dist[adj[i][j][k].first.first][adj[i][j][k].first.second]
        = dist[i][j] + adj[i][j][k].second;
28                      }
29                  }
30              }
31          }
32      }
33
34      for (int i = 0; i < w; i++) {
35          for (int j = 0; j < h; j++) {
36              if (i == dx && j == dy) return 0; // the moment he reaches it he
        exits
37              for (int k = 0; k < adj[i][j].size(); k++) {
38                  if (dist[i][j] != INF && dist[i][j] + adj[i][j][k].second < dist[
        adj[i][j][k].first.first][adj[i][j][k].first.second]) {
39                      return -1;
40                  }
41              }
42          }
43      }
44  }
45
46  int main() {
```

```
47      int x, y, _x, _y, t, ans;
48      scanf("%d %d", &w, &h);
49      while (w && h) {
50          memset(r, 0, sizeof(r));
51          for (int i = 0; i < MAX; i++) {
52              for (int j = 0; j < MAX; j++) {
53                  adj[i][j].clear();
54              }
55          }
56
57          scanf("%d", &g);
58          for (int i = 0; i < g; i++) {
59              scanf("%d %d", &x, &y);
60              r[x][y] = 1;
61          }
62
63          scanf("%d", &e);
64          for (int i = 0; i < e; i++) {
65              scanf("%d %d %d %d %d", &x, &y, &_x, &_y, &t);
66              adj[x][y].push_back(make_pair(make_pair(_x, _y), t));
67              r[x][y] = 1;
68          }
69
70          for (int i = 0; i < w; i++) {
71              for (int j = 0; j < h; j++) {
72                  if (!r[i][j]) {
73                      if (i > 0) adj[i][j].push_back(make_pair(make_pair(i - 1, j),
        1));
74                      if (i < w - 1) adj[i][j].push_back(make_pair(make_pair(i + 1,
        j), 1));
75                      if (j > 0) adj[i][j].push_back(make_pair(make_pair(i, j - 1),
        1));
76                      if (j < h - 1) adj[i][j].push_back(make_pair(make_pair(i, j +
        1), 1));
77                  }
78              }
79          }
80
81          ans = bellman_ford(0, 0, w-1, h-1);
82
83          if (ans == -1) printf("Never\n");
84          else if (dist[w-1][h-1] == INF) printf("Impossible\n");
85          else printf("%d\n", dist[w-1][h-1]);
86
87          scanf("%d %d", &w, &h);
88      }
89      return 0;
90  }
```

3.9 Dominators $< \mathcal{O}(V + E) >$ - Bytelandian Information Agency

```
1   #include <cstdio>
2   #include <algorithm>
3   #include <vector>
4   #include <list>
5
6   #define MAX 5001
7
8   using namespace std;
9
10  int n, m, k, ans[MAX];
11  vector<int> adj[MAX];
12
13  int N;
14  vector<int> pred[MAX], bucket[MAX];
15  int semi[MAX], parent[MAX], vert[MAX];
16  int dom[MAX], label[MAX], anc[MAX];
17
18  void dfs(int u) {
19      semi[u] = ++N;
20      vert[N] = label[u] = u;
```

```
21      anc[u] = 0;
22      for (int i=0;i<(int)adj[u].size();i++) {
23          int w = adj[u][i];
24          if (semi[w]==0) {
25              parent[w] = u;
26              dfs(w);
27          }
28          pred[w].push_back(u);
29      }
30  }
31
32  void compress(int u) {
33      if (anc[anc[u]] != 0) {
34          compress(anc[u]);
35          if (semi[label[anc[u]]] < semi[label[u]])
36              label[u] = label[anc[u]];
37          anc[u] = anc[anc[u]];
38      }
39  }
40
41  int eval(int u) {
42      if (anc[u]==0) return u;
43      compress(u);
44      return label[u];
45  }
46
47  void link(int u, int v) {
48      anc[v] = u;
49  }
50
51  void dominators(int ini) {
52      for (int i=0;i<=n;i++) {
53          pred[i].clear();
54          bucket[i].clear();
55          semi[i] = 0;
56      }
57
58      N=0;
59      dfs(ini);
60      for (int i=N;i>=2;i--) {
61          int w = vert[i];
62          for (int j=0;j<(int)pred[w].size();j++) {
63              int v = pred[w][j];
64              int u = eval(v);
65              if (semi[u] < semi[w])
66                  semi[w] = semi[u];
67          }
68          bucket[vert[semi[w]]].push_back(w);
69          link(parent[w],w);
70          for (int j=0;j<(int)bucket[parent[w]].size();j++) {
71              int v = bucket[parent[w]][j];
72              int u = eval(v);
73              dom[v] = (semi[u] < semi[v]) ? u : parent[w];
74          }
75          bucket[parent[w]].clear();
76      }
77      for (int i=2;i<=N;i++) {
78          int w = vert[i];
79          if (dom[w] != vert[semi[w]])
80              dom[w] = dom[dom[w]];
81      }
82      dom[ini] = 0;
83  }
84
85  int main() {
86      while (scanf("%d %d", &n, &m) == 2) {
87          for (int i = 1; i <= n; i++) adj[i].clear();
88
89          for (int i = 0; i < m; i++) {
90              int a, b;
91              scanf("%d %d", &a, &b);
92              adj[a].push_back(b);
93          }
```

```
94
95          dominators(1);
96
97          int dc = 0;
98          for (int i = 1; i <= n; i++)
99              if (dom[i] != 0) ans[dc++] = dom[i];
100
101         sort(ans, ans + dc);
102         dc = unique(ans, ans + dc) - ans;
103         printf("%d\n", dc);
104         for (int i = 0; i < dc; i++) {
105             if (i) printf(" ");
106             printf("%d", ans[i]);
107         }
108         printf("\n");
109     }
110     return 0;
111 }
```

3.10 Bridges $< \mathcal{O}(V + E) >$ - Série de Tubos

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define MAX 1001
5
6  int n, m, time_s, vis[MAX], par[MAX], ans;
7  vector <pair<int, int> > bridge;
8  vector <int> adj[MAX];
9
10 int dfs(int u) {
11     int low = time_s, cc = 0;
12     vis[u] = time_s++;
13     for (int i = 0; i < adj[u].size(); i++) {
14         if (!vis[adj[u][i]]) {
15             par[adj[u][i]] = u;
16             int low_i = dfs(adj[u][i]);
17             low = min(low, low_i);
18             if (low_i > vis[u]) {
19                 bridge.push_back(make_pair(u, adj[u][i]));
20                 ans = 1; // if there is a bridge, set the answer
21             }
22         } else if (adj[u][i] != par[u]) {
23             low = min(low, vis[adj[u][i]]);
24         }
25     }
26     return low;
27 }
28
29 void get_bridges() {
30     time_s = 1;
31     ans = 0;
32     bridge.clear();
33     memset(vis, 0, sizeof(vis));
34     memset(par, 0, sizeof(par));
35     dfs(1);
36 }
37
38 int main() {
39     while (scanf("%d %d", &n, &m) && (n || m)) {
40         int x, y, p = 0;
41         for (int i = 0; i < MAX; i++) adj[i].clear();
42
43         for (int i = 0; i < m; i++) {
44             scanf("%d %d", &x, &y);
45             adj[x].push_back(y);
46             adj[y].push_back(x);
47         }
48
49         get_bridges();
50
```

```
51          if (!ans) printf("S\n");
52          else printf("N\n");
53      }
54  }
```

3.11 Stable Marriage $< \mathcal{O}(N^2) >$ - Lobos Stark

```
1   /*
2   function stableMatching {
3       Initialize all men and women to free
4       while there is a free man m who still has a woman w to propose to {
5          w = first woman on m list to whom m has not yet proposed
6          if w is free
7            (m, w) become engaged
8          else some pair (m1, w) already exists
9            if w prefers m to m1
10             m1 becomes free
11           (m, w) become engaged
12           else
13             (m1, w) remain engaged
14      }
15  }
16  */
17
18  #include <bits/stdc++.h>
19  using namespace std;
20
21  int n, cid = 1, wid = 1, cc, cw;
22  list<int> pc[201];
23  int pw[201][201];
24  string str;
25  unordered_map<string, int> c, w;
26  string nc[201], nw[201];
27  vector<pair<int, int> > marriage;
28
29  void stable_marriage() {
30      marriage.clear();
31      int m[201];
32      list<int> fc; // free children
33      memset(m, 0, sizeof(m)); // matches (wolf -> child)
34      for (int i = 1; i <= n; i++) fc.push_back(i); // mark every child as free
35      while (fc.size() > 0) { // while there is a free child
36          int _c = fc.front(); fc.pop_front();
37          int _w = pc[_c].front(); pc[_c].pop_front(); // best match in child list
38          if (!m[_w]) {
39              m[_w] = _c; // if wolf is free match them
40          } else {
41              if (pw[_w][_c] < pw[_w][m[_w]]) { // wolf prefers current child
42                  fc.push_back(m[_w]); // set other child as free
43                  m[_w] = _c; // match wolf and current child
44              } else {
45                  fc.push_front(_c); // c remains free
46              }
47          }
48      }
49      for (int i = 1; i <= n; i++)
50          marriage.push_back(make_pair(m[i], i));
51  }
52
53  int main() {
54      ios_base::sync_with_stdio(false);
55      cin >> n;
56      for (int i = 0; i < n; i++) {
57          cin >> str;
58          if (!c[str]) {
59              cc = c[str] = cid++;
60              nc[cc] = str;
61          }
62          else cc = c[str];
63          for (int j = 0; j < n; j++) {
64              cin >> str;
```

```
65              if (!w[str]) {
66                  cw = w[str] = wid++;
67                  nw[cw] = str;
68              }
69              else cw = w[str];
70              pc[cc].push_back(cw);
71          }
72      }
73      for (int i = 0; i < n; i++) {
74          cin >> str;
75          if (!w[str]) {
76              cw = w[str] = wid++;
77              nw[cw] = str;
78          }
79          else cw = w[str];
80          for (int j = 0; j < n; j++) {
81              cin >> str;
82              if (!c[str]) {
83                  cc = c[str] = cid++;
84                  nc[cc] = str;
85              }
86              else cc = c[str];
87              pw[cw][cc] = j;
88          }
89      }
90      stable_marriage();
91      sort(marriage.begin(), marriage.end());
92      for (int i = 0; i < marriage.size(); i++)
93          cout << nc[marriage[i].first] << ' ' << nw[marriage[i].second] << '\n';
94      return 0;
95  }
```

3.12 Topological Sort $< \mathcal{O}(V + E) >$ - Orkut

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   #define MAXN 31
5
6   int n, m, pid, tc = 0;
7   string nome[MAXN], aux;
8   unordered_map<string, int> id;
9   vector<int> adj[MAXN], ts;
10  int vis[MAXN];
11
12  int visit(int u) {
13      if (vis[u] == 1) return 0; // temporary mark found, not a DAG
14      if (!vis[u]) {
15          vis[u] = 1;
16          for (int i = 0; i < adj[u].size(); i++)
17              if (!visit(adj[u][i])) return 0;
18          vis[u] = 2;
19          ts.push_back(u);
20      }
21      return 1;
22  }
23
24  int toposort() {
25      memset(vis, 0, sizeof(vis));
26      for (int i = 1; i <= n; i++)
27          if (!vis[i]) if (!visit(i)) return 0;
28      return 1;
29  }
30
31  int main() {
32      ios_base::sync_with_stdio(false);
33      while (cin >> n && n) {
34          for (int i = 0; i <= n; i++) adj[i].clear();
35          ts.clear();
36          cout << "Teste " << ++tc << '\n';
37          for (int i = 1; i <= n; i++) {
```

```
38              cin >> nome[i];
39              id[nome[i]] = i;
40          }
41          for (int i = 1; i <= n; i++) {
42              cin >> aux;
43              pid = id[aux];
44              cin >> m;
45              while (m--) {
46                  cin >> aux;
47                  adj[pid].push_back(id[aux]);
48              }
49          }
50          if (!toposort()) cout << "impossivel";
51          else for (int i = 0; i < ts.size(); i++) {
52              if (i) cout << ' ';
53              cout << nome[ts[i]];
54          }
55          cout << "\n\n";
56      }
57      return 0;
58  }
```

3.13 BFS with bitmask - O Labirinto de Ninguém

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   char mapa[101][101];
5   char vis[101][101][1<<8];
6   int h = -1, w;
7
8   void set_key(char c, int *b) {
9       int i = c - 'a';
10      *b = *b|(1<<i);
11  }
12
13  int has_key(char c, int b) {
14      int i = c - 'A';
15      return ((b>>i)&1);
16  }
17
18  int bfs(int i, int j) {
19      int x, y, b, m;
20      int dx[] = {-1, 0, 1, 0};
21      int dy[] = {0, 1, 0, -1};
22      char c;
23      queue< pair< pair<int, int> , pair<int, int> > > fila;
24      fila.push(make_pair(make_pair(i, j), make_pair(0, 0)));
25      while (!fila.empty()) {
26          x = fila.front().first.first;
27          y = fila.front().first.second;
28          b = fila.front().second.first;
29          m = fila.front().second.second;
30          fila.pop();
31          if (x < 0 || y < 0 || x >= h || y >= w) continue;
32          if (vis[x][y][b]) continue;
33          vis[x][y][b] = (char)1;
34          c = mapa[x][y];
35          if (c == '*') return m;
36          if (c == '#') continue;
37          if (c >= 'a' && c <= 'g') set_key(c, &b);
38          if (c >= 'A' && c <= 'G') if (!has_key(c, b)) continue;
39          for (int k = 0; k < 4; k++)
40              fila.push(make_pair(make_pair(x+dx[k], y+dy[k]), make_pair(b, m+1)));
41      }
42      return -1;
43  }
44
45  int main() {
46      int x, y, ans;
47      while (scanf("%s", mapa[++h]) != EOF);
```

```
48      w = strlen(mapa[0]);
49      // locate start
50      for (int i = 0; i < h; i++)
51          for (int j = 0; j < w; j++)
52              if (mapa[i][j] == '@') {x = i; y = j;}
53      memset(vis, 0, sizeof(vis));
54      ans = bfs(x, y);
55      if (ans == -1) printf("--\n");
56      else printf("%d\n", ans);
57      return 0;
58  }
```

3.14 Min Cost Max Flow $< \mathcal{O}((V+E) * \max F) >$ - Data Flow

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   #define pb push_back
5
6   typedef long long ll;
7
8   const ll INF = 1e14;
9
10  struct edge {
11      int u, v;
12      ll cap, flow, cost;
13      ll rem() { return cap - flow; }
14  };
15
16  int n, m, d, k, pre[102], u[5001], v[5001], c[5001];
17  vector<int> adj[102];
18  vector<edge> e;
19  ll dist[102], cap[102], min_cost, max_flow;
20  bool in_queue[102];
21
22  void add_edge(int u, int v, ll cap, ll cost) {
23      adj[u].pb(e.size()); e.pb((edge){u, v, cap, 0, cost});
24      adj[v].pb(e.size()); e.pb((edge){v, u, 0, 0, -cost});
25  }
26
27  void flow(int s, int t) {
28      memset(in_queue, 0, sizeof (in_queue));
29      min_cost = max_flow = 0;
30      while (1) {
31          for (int i=0; i<n; i++) dist[i] = INF; dist[s] = 0;
32          memset(pre, -1, sizeof(pre)); pre[s] = 0;
33          memset(cap, 0, sizeof(cap)); cap[s] = INF;
34          queue<int> q; q.push(s); in_queue[s] = 1;
35          while (!q.empty()) {
36              int u = q.front(); q.pop(); in_queue[u] = 0;
37              for (auto it:adj[u]) {
38                  edge &E = e[it];
39                  if (E.rem() && dist[E.v] > dist[u] + E.cost) {
40                      dist[E.v] = dist[u] + E.cost;
41                      pre[E.v] = it;
42                      cap[E.v] = min(cap[u], E.rem());
43                      if (!in_queue[E.v]) q.push(E.v), in_queue[E.v] = 1;
44                  }
45              }
46          }
47          if (pre[t] == -1) break;
48          max_flow += cap[t];
49          min_cost += cap[t]*dist[t];
50          for (int v = t; v != s; v = e[pre[v]].u) {
51              e[pre[v]].flow += cap[t];
52              e[pre[v]^1].flow -= cap[t];
53          }
54      }
55  }
56
57
```

```
int main() {
    while (scanf("%d %d", &n, &m) != EOF) {
        for (int i=0; i<=n; i++) adj[i].clear();
        e.clear();
        for (int i=0; i<m; i++) scanf("%d %d %d", u+i, v+i, c+i);
        scanf("%d %d", &d, &k);
        n++;
        add_edge(0, 1, d, 0);
        for (int i=0; i<m; i++) {
            add_edge(u[i], v[i], k, c[i]);
            add_edge(v[i], u[i], k, c[i]);
        }
        flow(0, n-1);
        if (max_flow == d) printf("%lld\n", min_cost);
        else printf("Impossible.\n");
    }
    return 0;
}
```

3.15 Bipartite Matching $< \mathcal{O}(\sqrt{V} * E) >$ - Final de Fisiologia Canina

```
#include <cstdio>
#include <cstring>
#include <vector>
#include <queue>

#define INF 0x3f3f3f3f
using namespace std;

vector<int> G[200002];
queue<int> Q;
int vis[200002];
int dist[200002];
int match[200002];
int n, m, q, p, size;

int bfs () {
    for (int u = 1; u <= n; u++) {
        if (!match[u]) {
            dist[u] = 0;
            Q.push(u);
        } else {
            dist[u] = INF;
        }
    }
    dist[0] = INF;
    while (!Q.empty()) {
        int u = Q.front();
        Q.pop();
        if (dist[u] < dist[0]) {
            for (int i = 0; i < G[u].size(); i++) {
                int v = G[u][i];
                if (dist[match[v]] == INF) {
                    dist[match[v]] = dist[u] + 1;
                    Q.push(match[v]);
                }
            }
        }
    }
    return dist[0] != INF;
}

int dfs(int u) {
    if (u) {
        for (int i = 0; i < G[u].size(); i++) {
            int v = G[u][i];
            if (dist[match[v]] == dist[u] + 1) {
                if (dfs(match[v])) {
                    match[v] = u;
                    match[u] = v;
                    return 1;
```

```
                    }
                }
            }
            dist[u] = INF;
            return 0;
        }
        return 1;
    }

    int hopcroft_karp() {
        memset(match, 0, sizeof(match));
        int size = 0;
        while (bfs()) {
            for (int u = 1; u <= n; u++) {
                if (match[u] == 0) {
                    if (dfs(u)) size++;
                }
            }
        }
        return size;
    }

    int main () {
        scanf("%d %d", &n, &m);
        size = 0;
        for(int i = 1; i <= n; i++) {
            scanf("%d", &q);
            for (int j = 0; j < q; j++) {
                scanf("%d", &p);
                p += 100000;
                G[i].push_back(p);
                G[p].push_back(i);
            }
        }
        printf("%d\n", hopcroft_karp());
        return 0;
    }
```

3.16 Union Find $< \mathcal{O}(1) * OP >$ - Virtual Friends

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

int p[100010], r[100010], s[100010];
int t, f, id;
unordered_map<string, int> name;
string n1, n2;

void create_set(int x) { p[x] = x; r[x] = 0; s[x] = 1; }
int find_set(int x) { if (x != p[x]) p[x] = find_set(p[x]); return p[x]; }
void merge_sets(int x, int y) {
    int px = find_set(x);
    int py = find_set(y);
    if (px == py) return; // do not merge same set
    if (r[px] > r[py]) { p[py] = px; s[px] += s[py]; }
    else { p[px] = py; s[py] += s[px]; }
    if (r[px] == r[py]) r[py]++;
}

int main() {
    cin >> t;
    while (t--) {
        id = 1; name.clear();
        cin >> f;
        while (f--) {
            cin >> n1 >> n2;
            if (name.find(n1) == name.end()) {
                name[n1] = id;
                create_set(id++);
            }
            if (name.find(n2) == name.end()) {
```

```
32                      name[n2] = id;
33                      create_set(id++);
34                  }
35                  merge_sets(name[n1], name[n2]);
36                  printf("%d\n", s[find_set(name[n1])]);
37              }
38      }
39      return 0;
40  }
```

4 Geometry

4.1 Convex Hull $< \mathcal{O}(N \log N) >$ - Camadas de Cebola

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

typedef struct {
  int x, y;
} point;

bool compare(const point &p1, const point &p2) {
  return p1.x < p2.x || (p1.x == p2.x && p1.y < p2.y);
}

int cross(const point &o, const point &a, const point &b) {
  return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x - o.x);
}

// on real convex hull, should return the points (vector<point>)
// used vector is not used on real c_hull
vector<int> convex_hull(vector<point> &P, vector<bool> &used) {
  int n = P.size(), k = 0;
  vector<int> H(2*n); // vector<point>

  sort(P.begin(), P.end(), compare);

  for (int i = 0; i < n; ++i) {
    if (used[i]) continue; // not used on real c_hull
    while (k >= 2 && cross(P[H[k-2]], P[H[k-1]], P[i]) < 0) k--;
    H[k++] = i; // i should be P[i]
  }

  for (int i = n-2, t = k+1; i >= 0; i--) {
    if (used[i]) continue; // not used on real c_hull
    while (k >= t && cross(P[H[k-2]], P[H[k-1]], P[i]) < 0) k--;
    H[k++] = i; // i should be P[i]
  }

  H.resize(k);
  return H;
}

int main() {
  int n;
  while (1) {
    scanf("%d", &n);
    if (n == 0) break;
    vector<point> P(n);
    vector<bool> used(n);
    for (int i = 0; i < n; i++) {
      used[i] = false;
      scanf("%d %d", &(P[i].x), &(P[i].y));
    }

    int ans = 0;
    while (true) {
      // should be vector<point> when using real c_hull
      vector<int> C = convex_hull(P, used);
      if (!C.size()) break;
      for (int i = 0; i < C.size(); i++) { // not used on real c_hull
        used[C[i]] = true; // not used on real c_hull
      } // not used on real c_hull
      ans++;
    }

    if (ans&1) printf("Take this onion to the lab!\n");
    else printf("Do not take this onion to the lab!\n");
  }
  return 0;
}
```

4.2 Max Enclosing Circle (Circle Sweep) $< \mathcal{O}(N \log N) >$ - Phone Cell

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

#define EPS 1e-7
#define SQ(a) ((a)*(a))

typedef struct {
    double ang;
    int p;
} angle;

bool cmpa(const angle &a1, const angle &a2) { return a1.ang < a2.ang; }

int n, r;
int x[2002], y[2002];

double ang(const int i, const int j) {
    if(x[i]==x[j]) {
        if(y[j]>y[i]) return M_PI/2; else return 3*M_PI/2;
    } else if(y[i]==y[j]) {
        if(x[j]>x[i]) return 0; else return M_PI;
    }
    return atan2(y[j]-y[i], x[j]-x[i]);
}

int enc_circle() {
    vector<angle> angles;
    int ans = 1;
    double d, td, tang;
    angle a1, a2;
    for (int i = 0; i < n; i++) {
        angles.clear();
        // get angid points to compare to
        for (int j = 0; j < n; j++) {
            if (i == j) continue;
            if (x[i] == x[j] && y[i] == y[j]) continue;
            d = SQ(x[j] - x[i]) + SQ(y[j] - y[i]);
            if (SQ(2*r) + EPS < d) continue;
            td = sqrt(d);
            // get point relative to pi
            tang = ang(i, j);
            d = acos(td/(2*r));
            // get angle to center of circun
            a1.ang = tang+d+EPS/10;
            a1.p = j;
            a2.ang = tang-d-EPS/10;
            a2.p = j;
            angles.push_back(a1);
            angles.push_back(a2);
        }
        // normalize angles to 0 and 2PI, 0-360
        for (int j = 0; j < angles.size(); j++) {
            while (angles[j].ang < 0) angles[j].ang += 2*M_PI;
            while (angles[j].ang >= 2*M_PI) angles[j].ang -= 2*M_PI;
        }
        sort(angles.begin(), angles.end(), cmpa);
        int cnt = 0;
        // check points that are already in the starting circle
        vector<bool> in; in.resize(n);
        for (int j = 0; j < n; j++) {
            if (SQ(x[j] - (x[i]+r)) + SQ(y[j] - y[i]) <= SQ(r)+EPS) {
                in[j] = 1;
                cnt++;
            } else {
                in[j] = 0;
            }
        }
        // update answer with initial points
        ans = max(ans, cnt);
        for (int j = 0; j < angles.size(); j++) {
            if (in[angles[j].p]) cnt--;
```

```
72              else cnt++;
73              in[angles[j].p] = 1-in[angles[j].p];
74              ans = max(ans, cnt);
75          }
76      }
77      return ans;
78  }
79
80  int main() {
81      while (scanf("%d %d", &n, &r) && (n || r)) {
82          for (int i = 0; i < n; i++) scanf("%d %d", &(x[i]), &(y[i]));
83          printf("It is possible to cover %d points.\n", enc_circle());
84      }
85      return 0;
86  }
```

4.3 Closest Points $< \mathcal{O}(N \log N) >$ - Problema dos Pares Mais Próximos

```
1   #include <bits/stdc++.h>
2   using namespace std;
3
4   #define INF 10000
5
6   typedef struct {
7     double x, y;
8   } point;
9
10  bool cmpx(const point &p1, const point &p2) {
11    return p1.x < p2.x || (p1.x == p2.x && p1.y < p2.y);
12  }
13
14  bool cmpy(const point &p1, const point &p2) {
15    return p1.y < p2.y || (p1.y == p2.y && p1.x < p2.x);
16  }
17
18  double dist(const point &a, const point &b) {
19      return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
20  }
21
22  int n;
23  point p[10010], strip[10010];
24
25  double closest_pair(int l, int r) {
26      if (l >= r) return INF;
27      if (l == r - 1) return dist(p[l], p[r]);
28      int mid = (l+r)/2;
29      double dl = closest_pair(l, mid);
30      double dr = closest_pair(mid+1, r);
31      double d = min(dl, dr);
32      // get points on strip
33      int c = 0;
34      for (int i = l; i <= r; i++)
35          if (fabs(p[i].x - p[mid].x) < d) strip[c++] = p[i];
36      sort(strip, strip+c, cmpy);
37      // h < 7 as there are at max 6 points on the strip rectangle
38      for (int i = 0; i < c; i++)
39          for (int j = i+1, h = 0; j < c && h < 7; j++, h++)
40              d = min(d, dist(strip[i], strip[j]));
41      return d;
42  }
43
44  int main() {
45      while (scanf("%d", &n) && n) {
46          for (int i = 0; i < n; i++) scanf("%lf %lf", &(p[i].x), &(p[i].y));
47          sort(p, p+n, cmpx);
48          double ans = closest_pair(0, n-1);
49          if (ans >= 10000) printf("INFINITY\n");
50          else printf("%.4lf\n", ans);
51      }
52      return 0;
53  }
```

4.4 Rotating Calipers $< \mathcal{O}(N) >$ - Trash Removal

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

#define EPS 1e-9

typedef struct {
    int x, y;
} point;

bool cmpx(const point &p1, const point &p2) {
    return p1.x < p2.x || (p1.x == p2.x && p1.y < p2.y);
}

double cross(const point &o, const point &a, const point &b) {
  return (a.x - o.x) * (b.y - o.y) - (a.y - o.y) * (b.x - o.x);
}

double dist_line(const point &p, const point &a, const point &b) {
    return cross(b, a, p)/sqrt((b.y-a.y)*(b.y-a.y) + (b.x-a.x)*(b.x-a.x));
}

point p[101];
int n, tc = 0;

int main() {
    while (scanf("%d", &n) && n) {
        for (int i=0; i<n; i++) {
            scanf("%d %d", &(p[i].x), &(p[i].y));
        }
        // convex hull via monotone
        int k = 0;
        vector<point> h(2*n);
        sort(p, p+n, cmpx);
        for (int i = 0; i < n; ++i) {
            while (k >= 2 && cross(h[k-2], h[k-1], p[i]) <= 0) k--;
            h[k++] = p[i];
        }
        for (int i = n-2, t = k+1; i >= 0; i--) {
            while (k >= t && cross(h[k-2], h[k-1], p[i]) <= 0) k--;
            h[k++] = p[i];
        }
        h.resize(k);
        printf("k %d k depois %d\n", k, k-(n>1));
        k = k-(n>1);
        // rotate calipers
        double ans = 1e15;
        int j = 1;
        h[0] = h[k];
        for (int i = 1; i <= k; i++) {
            while (cross(h[i-1], h[i], h[j%k+1]) > cross(h[i-1], h[i], h[j]))
                j = j%k+1;
            printf("i %d j %d\n", i, j);
            ans = min (ans, dist_line(h[j], h[i], h[i-1]));
        }
        printf("Case %d: %.2lf\n", ++tc, ans);
    }
}
```

4.5 Line Sweep $< \mathcal{O}(N \log N) >$ - Janela

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

#define MAXN 100010
#define H 100
#define W 200
#define MAXW 600

bool in[MAXN];
```

```
pair<int, int> p[2*MAXN];


int main() {
    int n = 3, t, count = 0, last = 0, ans = 0;
    for (int i = 0; i < n; i++) {
        scanf("%d", &t);
        p[i].first = t, p[i].second = i;
        p[i+n].first = t+W, p[i+n].second = i;
    }
    memset(in, false, sizeof(bool)*n);
    sort(p, p+2*n);
    for (int i = 0; i < 2*n; i++) {
        //printf("%s event for %d(%d)\n", (in[p[i].second])?"Out":"Entry", p[i].second, p[i].first);
        //printf("Current position %d, count %d, last %d\n", p[i].first, count, last);
        if (!in[p[i].second]) {
            // entry event
            if (count == 0) ans += (p[i].first-last)*H;
            in[p[i].second] = true;
            count++;
        } else {
            // out event
            count--;
            in[p[i].second] = false;
            last = p[i].first;
        }
    }
    ans += (MAXW-last)*H;
    printf("%d\n", ans);
    return 0;
}
```

4.6 Angle Sweep (Circunference covered) $< \mathcal{O}(N \log N) >$ - Planet Destruction

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

#define mp make_pair
#define EPS 1e-12

#define N 10010

long long R, k;
long long x[N], y[N], r[N], v[N];

inline double dist(int i){
  return sqrt(x[i]*x[i] + y[i]*y[i]);
}

bool in[N<<1];

double cob(double t){
  vector <pair <double, int> > evt;
  int c = 0;

  for(int i=0; i<k; i++)
  if((dist(i)-R)/r[i] < t){
    double a1, a2;

    if(v[i]*(t - (dist(i)-R)/r[i]) >= M_PI*R){
      return 1;
    }
    a1 = atan2(y[i], x[i]) - v[i]*(t - (dist(i)-R)/r[i])/R;
    a2 = atan2(y[i], x[i]) + v[i]*(t - (dist(i)-R)/r[i])/R;

    if(a1<0 && a2>=0){
      in[i] = 1;
      c++;
```

```
35      }else in[i] = 0;
36
37      while(a1 < 0) a1 += 2*M_PI;
38      while(a1 >= 2*M_PI) a1 -= 2*M_PI;
39      while(a2 < 0) a2 += 2*M_PI;
40      while(a2 >= 2*M_PI) a2 -= 2*M_PI;
41
42      evt.push_back(mp(a1, i));
43      evt.push_back(mp(a2, i));
44    }
45
46    sort(evt.begin(), evt.end());
47    double r = 0, pi = 0;
48    for(int i=0; i<evt.size(); i++){
49      if(in[evt[i].second]){
50        c--;
51        if(!c) r += (evt[i].first - pi);
52      }else{
53        if(!c) pi = evt[i].first;
54        c++;
55      }
56      in[evt[i].second] = 1-in[evt[i].second];
57    }
58    if(c) r += (2*M_PI - pi);
59
60    return r/(2*M_PI);
61  }
62
63  int main(){
64    int t;
65    scanf("%d", &t);
66    while(t--){
67      scanf("%lld %lld", &R, &k);
68      double a = 0, b = 0;
69      for(int i=0; i<k; i++){
70        scanf("%lld %lld %lld %lld", x+i, y+i, r+i, v+i);
71        b = max(b, (dist(i)-R)/r[i] + M_PI*R/v[i]);
72      }
73
74      for(int i=0; i<65; i++){
75        double m = (a+b)/2;
76        if(fabs(cob(m)-1) > EPS) a = m;
77        else b = m;
78      }
79
80      printf("%.4lf\n", (a+b)/2);
81    }
82
83    return 0;
84  }
```

4.7 Angle Sweep with Intersections - Hide and Seek

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define EPS 1e-9
5  #define SQ(a) ((a)*(a))
6  #define pb push_back
7
8  int cmp(double a, double b=0) { return a < b-EPS? -1 : a > b+EPS? 1 : 0; }
9
10 struct point {
11     double x, y;
12     point(double x, double y): x(x), y(y) {}
13     point() {}
14     point operator+(const point &p) const { return point(x + p.x, y + p.y); }
15   point operator-(const point &p) const { return point(x - p.x, y - p.y); }
16   point operator*(double t) const { return point(x*t, y*t); }
17   point operator/(double t) const { return point(x/t, y/t); }
18   double operator*(const point &p) const { return x*p.x + y*p.y; }
```

```
  double operator%(const point &p) const { return x*p.y - y*p.x; }
  double dist() const { return x*x + y*y; }
};

point O, W;

struct event {
    point p;
    int z;
    event(const point &p, int z): p(p), z(z) {}
    event() {}
};

struct segment {
    point u, v;
    segment(const point &u, const point &v): u(u), v(v) {}
    segment() {}
    bool operator<(const segment &s) const { return dist() < s.dist(); }
    double dist() const {
    double den = (u-v)%(W-O);
    if (cmp(den)==0) return -1e-20;
    double t = (u-O)%(W-O) / den; // t = u+(v-u)t-O paralelo a (W-O)
    if (cmp(t,0)<0 || cmp(t,1) > 0) return -1e-20;
    point p = (u-O) + (v-u)*t;
    if ( p*(W-O) < 0) return -1e-20;
    return p.dist();
  }
};

int above(point p) {
  if (p.y == O.y) return p.x > O.x;
  return p.y > O.y;
}

bool circular_order(point p, point q) {
  int tmp = above(q) - above(p);
  if (tmp!=0) return tmp > 0;
  return (p-O)%(q-O) > 0;
}

bool event_order(event P, event Q) {
  return circular_order(P.p, Q.p);
}

int sc, ks, wc;
point k[10010];
segment w[10010];

int count(int sk) {
    vector<event> ev;
    set&lt;segment&gt; sg;
    O = k[sk];
    int ans = 0;
    for (int i=0; i<ks; i++) if (sk != i) ev.pb(event(k[i], -1));
    for (int i=0; i<wc; i++) {
        double cr = (w[i].u - O)%(w[i].v - O); // cross vai dar quem "vem antes"
        if (cr < 0) swap(w[i].u, w[i].v); // troca ptos se necessario
        ev.pb(event(w[i].u, (i<<1)));
        ev.pb(event(w[i].v, (i<<1)+1));
    }
    sort(ev.begin(), ev.end(), event_order);
    W = ev[0].p;
    for (int i=0; i<wc; i++) if (w[i].dist() > 0) {
        sg.insert(w[i]); // coloca muros ativos
    }
    for (int i=0; i<ev.size(); i++) {
        W = ev[i].p;
        int z = ev[i].z;
        if (z<0) {
            if (sg.empty()) { ans++; continue; } // sem muros ativos nesse angulo
            segment fs = *sg.begin(); // muro mais perto de sk para o evento atual
```

```
90              if (fs.dist() > (ev[i].p-O).dist()) ans++; // kid (w-o) esta mais
    perto que o 1o muro
91          } else if (z&1) sg.erase(w[z>>1]);
92          else sg.insert(w[z>>1]);
93      }
94      return ans;
95  }
96
97  int main() {
98      while (scanf("%d %d %d", &sc, &ks, &wc) != EOF) {
99          for (int i=0; i<ks; i++) scanf("%lf %lf", &(k[i].x), &(k[i].y));
100         for (int i=0; i<wc; i++) {
101             point u, v;
102             scanf("%lf %lf %lf %lf", &(u.x), &(u.y), &(v.x), &(v.y));
103             w[i] = segment(u, v);
104         }
105         for (int i=0; i<sc; i++) {
106             printf("%d\n", count(i));
107         }
108     }
109     return 0;
110 }
```

4.8 Polygon Area $< \mathcal{O}(N) >$ - Ingress

```
1  #include <cstdio>
2  #include <cstdlib>
3
4  using namespace std;
5
6  #define ABS(x) (x < 0)?(-x):(x)
7
8  typedef struct {
9    int x, y;
10 } point;
11
12 point vec[100001], temp;
13
14 int ccw (point p1, point p2, point p3) {
15   return (p2.x - p1.x) * (p3.y - p1.y) - (p2.y - p1.y) * (p3.x - p1.x);
16 }
17
18 int dist (point p1, point p2) {
19   return (p1.x - p2.x)*(p1.x - p2.x) + (p1.y - p2.y)*(p1.y - p2.y);
20 }
21
22 int comp (const void *vp1, const void *vp2) {
23   point *p1 = (point *)vp1;
24   point *p2 = (point *)vp2;
25
26   int o = ccw(vec[1], *p1, *p2);
27   if (o == 0) return (dist(vec[1], *p2) >= dist(vec[1], *p1))? -1 : 1;
28   return (o > 0)? -1: 1;
29 }
30
31 double area(int m) {
32   int sum = 0, sub = 0;
33   for (int i = 1; i <= m; i++) {
34     sum += (vec[i-1].x * vec[i].y);
35     sub += (vec[i-1].y * vec[i].x);
36   }
37   return ABS((sum-sub)/2.0);
38 }
39
40 int main () {
41   int n, miny = 0;
42
43   vec[0] = (point){100000,100000};
44
45   scanf("%d", &n);
46   for (int i = 1; i <= n; i++) {
```

```
47        scanf("%d %d", &vec[i].x, &vec[i].y);
48        if (vec[miny].y > vec[i].y) miny = i;
49      }
50
51      temp = vec[1];
52      vec[1] = vec[miny];
53      vec[miny] = temp;
54      //printf("1: %d %d\nminy: %d %d\n",vec[1].x,vec[1].y,vec[miny].x,vec[miny].y);
55
56      // qsort pior caso mto grande. USAR SORT
57      qsort(&vec[2], n-1, sizeof(point), comp);
58
59      vec[0] = vec[n];
60      //for(int i=0; i <= n; i++)
61       // printf("P%d (%d, %d)\n", i, vec[i].x, vec[i].y);
62
63      int m = 1;
64
65      for (int i = 2; i <= n; i++) {
66        while (ccw(vec[m-1], vec[m], vec[i]) < 0) {
67          if (m > 1) m -= 1;
68          else if ( i == n) break;
69          else i += 1;
70        }
71        m++;
72        temp = vec[i];
73        vec[i] = vec[m];
74        vec[m] = temp;
75      }
76
77    // printf("\n\n");
78    // for(int i=0; i <= n; i++)
79    //    printf("P%d (%d, %d)\n", i, vec[i].x, vec[i].y);
80    // printf("M: %d\n", m);
81      printf("%.2lf\n", area(m));
82
83      return 0;
84  }
```

4.9 Smallest Enclosing Circle $< \mathcal{O}(N^2) >$ - Torres de Telefonia Celular

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cmath>
4  #include <algorithm>
5  #include <set>
6  using namespace std;
7
8  #define EPS 1e-12
9  #define dist(a, b) ((a[0]-b[0])*(a[0]-b[0])+(a[1]-b[1])*(a[1]-b[1]))
10 #define min(a, b) ((a) - (b) < EPS ? (a) : (b))
11 #define max(a, b) (EPS > (b) - (a) ? (a) : (b))
12
13 int n;
14 int p[40][2], pi[40];
15 char in[40];
16
17 int f[40], tf;
18
19 bool cmp(int a, int b){
20     if(p[a][0] < p[b][0]) return true;
21     if(p[a][0] > p[b][0]) return false;
22     return (p[a][1] < p[b][1]);
23 }
24
25 inline int ccw(int a, int b, int c){
26     return (p[b][0] - p[a][0])*(p[c][1]-p[a][1]) - (p[b][1] - p[a][1])*(p[c][0] -
       p[a][0]);
27 }
28
29 inline double cosang(int o, int a, int b){
```

```
30      return ((p[a][0]-p[o][0])*(p[b][0]-p[o][0]) + (p[a][1]-p[o][1])*(p[b][1]-p[o
        ][1])) / (sqrt(dist(p[o], p[a]))*sqrt(dist(p[o], p[b])));
31  }
32
33  void fc(){
34      int tl = 0, tu = 0;
35      for(int i=0; i<n; i++){
36          if(in[pi[i]]) continue;
37          while(tl >= 2 && !(ccw(f[tl-2], f[tl-1], pi[i]) > 0)) tl--;
38          f[tl++] = pi[i];
39      }
40      if(tl) tl--;
41
42      for(int i=n-1; i>=0; i--){
43          if(in[pi[i]]) continue;
44          while(tu >= 2 && !(ccw(f[tl+tu-2], f[tl+tu-1], pi[i]) > 0)) tu--;
45          f[tl+tu++] = pi[i];
46      }
47      if(tu) tu--;
48
49      tf = tl+tu;
50
51      return;
52  }
53
54  bool circun(int i, int j, int k, double *o){
55      double ij[2], ki[2];
56      ij[0] = p[i][0]-p[j][0];
57      ij[1] = p[i][1]-p[j][1];
58      ki[0] = p[k][0]-p[i][0];
59      ki[1] = p[k][1]-p[i][1];
60
61      double den = 2*(ij[0]*ki[1] - ij[1]*ki[0]);
62      if(!den) return false;
63
64      double aux1, aux2;
65      aux1 = ij[1]*(p[j][1]+p[i][1]) + ij[0]*(p[j][0]+p[i][0]);
66      aux2 = ki[1]*(p[k][1]+p[i][1]) + ki[0]*(p[k][0]+p[i][0]);
67
68      o[0] = (ki[1]*aux1 - ij[1]*aux2)/den;
69      o[1] = (ij[0]*aux2 - ki[0]*aux1)/den;
70
71      return true;
72  }
73
74  double applet(){
75      fc();
76
77      if(tf <= 1) return 0;
78      if(tf == 2) return dist(p[f[0]], p[f[1]])/4.0;
79
80      int s0 = 0, s1 = 1;
81
82      while(1){
83          int v;
84          double a = -2, tc; // i.e. a < cosseno do maior angulo possivel
85          for(int i=0; i<tf; i++){
86              if(i == s0 || i == s1) continue;
87              tc = cosang(f[i], f[s0], f[s1]);
88              if(tc > a){
89                  a = tc;
90                  v = i;
91              }
92          }
93          if(a < EPS){
94              return dist(p[f[s0]], p[f[s1]])/4.0;
95          }
96
97          double c1 = cosang(f[s0], f[s1], f[v]), c2 = cosang(f[s1], f[v], f[s0]);
98          if(c1 > EPS && c2 > EPS){
99              double o[2], r;
100             circun(f[s0], f[s1], f[v], o);
101             return dist(o, p[f[s0]]);
```

```
102             }
103
104             s0 = (c1 < EPS ? s1 : s0);
105             s1 = v;
106         }
107     }
108     // */
109
110     int main(){
111         while(scanf("%d", &n) && n){
112             for(int i=0; i<n; i++){
113                 scanf("%d %d", p[i], p[i]+1);
114                 pi[i] = i;
115             }
116             sort(pi, pi+n, cmp);
117
118             double m = 1e12;
119             for(int i=0; i<n; i++){
120                 for(int j=i+1; j<n; j++){
121                     double o[2], r1, r2;
122
123                     o[0] = (p[i][0] + p[j][0])/2.0;
124                     o[1] = (p[i][1] + p[j][1])/2.0;
125                     r1 = dist(o, p[i]);
126
127                     memset(in, 0, sizeof(in));
128                     for(int k=0; k<n; k++)
129                         in[k] = (dist(p[k], o) < r1);
130                     in[i] = in[j] = !(0);
131
132                     r2 = applet();
133
134                     m = min(max(r1, r2), m);
135                 }
136             }
137
138             for(int i=0; i<n; i++){
139                 for(int j=i+1; j<n; j++){
140                     for(int k=j+1; k<n; k++){
141                         double o[2], r1, r2;
142
143                         if(!circun(i, j, k, o)) continue;
144                         r1 = dist(o, p[i]);
145
146                         memset(in, 0, sizeof(in));
147                         for(int l=0; l<n; l++)
148                             in[l] = (dist(p[l], o) < r1);
149                         in[i] = in[j] = in[k] = !(0);
150
151                         r2 = applet();
152
153                         m = min(max(r1, r2), m);
154                     }
155                 }
156             }
157
158             printf("%.2lf\n", sqrt(m));
159         }
160         return 0;
161     }
```

5 Data Structures

5.1 BIT $< \mathcal{O}(\log MaxVal), \mathcal{O}(MaxVal) >$ - Balé

```
#include <cstdio>
#include <cstring>

int tree[100010];
int maxval=100001;

int n, np, v[100010];

int read(int idx){
  int sum=0;
  while(idx > 0){
    sum += tree[idx];
    idx -= (idx & -idx);
  }
  return sum;
}

void update(int idx, int val){
  while(idx <= maxval){
    tree[idx] += val;
    idx += (idx & -idx);
  }
}

int main(){
  memset(tree, 0, sizeof(tree));

  scanf("%d", &n);
  for(int i=1; i<=n; i++)
    scanf("%d", v+i);

  np=0;
  for(int i=n; i>=1; i--){
    np += read(v[i]);
    update(v[i],1);
  }
  printf("%d\n", np);
  return 0;
}
```

5.2 Segment Tree $< \mathcal{O}(N \log N), \mathcal{O}(\log N), \mathcal{O}(\log N) >$ - To Poland

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

int ntc, n, m, q, a, b;
char cmd[2];

const int N = 1e5+1;  // limit for array size
int t[2 * N];

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

void modify(int p, int value) {  // set value at position p
    for (t[p += n] = value; p > 1; p >>= 1) t[p>>1] = max(t[p], t[p^1]);
}

int query(int l, int r) {  // sum on interval [l, r)
    int res = 0;
    for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
        if (l&1) res = max(res, t[l++]);
        if (r&1) res = max(res, t[--r]);
    }
    return res;
```

```
}


int main() {
    scanf("%d", &ntc);
    for (int tc = 0; tc < ntc; tc++) {
        memset(t, 0, sizeof(t));
        printf("Testcase %d:\n", tc);
        scanf("%d %d", &n, &m);
        for (int i = 0; i < n; i++) {
            // read to segtree (start at index n)
            scanf("%d", t + n + i);
        }
        build();
        scanf("%d", &q);
        while (q--) {
            scanf("%s", cmd);
            switch (cmd[0]) {
                case 'A':
                    scanf("%d", &a);
                    m += a;
                    break;
                case 'B':
                    scanf("%d %d", &a, &b);
                    modify(a, b);
                    break;
                case 'C':
                    scanf("%d %d", &a, &b);
                    // query interval is open at right
                    printf("%d\n", abs(m - query(a, b+1)));
                    break;
            }
        }
        printf("\n");
    }
}
```

5.3 Sum Matrix (DP) $< \mathcal{O}(N * M) >$ - Colheita de Caju

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

int mat[1001][1001], ac[1001][1001];
int l, c, m, n, ans;

int main() {
    scanf("%d %d %d %d", &l, &c, &m, &n);
    for (int i = 1; i <= l; i++)
        for (int j = 1; j <= c; j++)
            scanf("%d", &mat[i][j]);
    ans = 0;
    // sentinel to avoid segfault (dp base)
    for (int i = 0; i < l; i++)
        ac[i][0] = ac[0][i] = 0;

    // dp
    for (int i = 1; i <= l; i++)
        for (int j = 1; j <= c; j++)
            ac[i][j] = ac[i-1][j] + ac[i][j-1] - ac[i-1][j-1] + mat[i][j];

    // compute max sum based on sum for
    for (int i = m; i <= l; i++)
        for (int j = n; j <= c; j++)
            ans = max(ans, ac[i][j] - ac[i-m][j] - ac[i][j-n] + ac[i-m][j-n]);
    printf("%d\n", ans);
}
```

5.4 Lazy Segment Tree $< \mathcal{O}(N \log N), \mathcal{O}(\log N), \mathcal{O}(\log N) >$ - Homem, Elefante, Rato

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

typedef struct { int h, e, r; } node;

const int N = 100010;  // limit for array size
int n;  // array size
node t[2 * N];
int h;
char d[N];

void calc(int p, int k) {
    if (d[p] == 0) {
        t[p].h = t[p<<1].h + t[p<<1|1].h;
        t[p].e = t[p<<1].e + t[p<<1|1].e;
        t[p].r = t[p<<1].r + t[p<<1|1].r;
    } else {
        //t[p].h = t[p].e = t[p].r = 0;
        for (int i = 0; i < d[p]; i++) {
            int h = t[p].h, e = t[p].e, r = t[p].r;
            t[p].e = h;
            t[p].r = e;
            t[p].h = r;
        }
    }
}

void apply(int p, int value, int k) {
    //t[p].h = t[p].e = t[p].r = 0;
    for (int i = 0; i < value%3; i++) {
        int h = t[p].h, e = t[p].e, r = t[p].r;
        t[p].e = h;
        t[p].r = e;
        t[p].h = r;
    }
    if (p < n) d[p] = (d[p]+value)%3;
}

void build(int l, int r) {
    int k = 2;
    for (l += n, r += n-1; l > 1; k <<= 1) {
        l >>= 1, r >>= 1;
        for (int i = r; i >= l; --i) calc(i, k);
    }
}

void push(int l, int r) {
    int s = h, k = 1 << (h-1);
    for (l += n, r += n-1; s > 0; --s, k >>= 1)
    for (int i = l >> s; i <= r >> s; ++i) if (d[i] != 0) {
        apply(i<<1, d[i], k);
        apply(i<<1|1, d[i], k);
        d[i] = 0;
    }
}

void modify(int l, int r, int value) {
    if (value == 0) return;
    push(l, l + 1);
    push(r - 1, r);
    bool cl = false, cr = false;
    int k = 1;
    for (l += n, r += n; l < r; l >>= 1, r >>= 1, k <<= 1) {
        if (cl) calc(l - 1, k);
        if (cr) calc(r, k);
        if (l&1) apply(l++, value, k), cl = true;
        if (r&1) apply(--r, value, k), cr = true;
    }
    for (--l; r > 0; l >>= 1, r >>= 1, k <<= 1) {
```

```
70          if (cl) calc(l, k);
71          if (cr && (!cl || l != r)) calc(r, k);
72      }
73  }
74
75  int query(int l, int r, node &res) {
76      push(l, l + 1);
77      push(r - 1, r);
78      res.h = res.e = res.r = 0;
79      for (l += n, r += n; l < r; l >>= 1, r >>= 1) {
80          if (l&1) {
81              res.h += t[l].h;
82              res.e += t[l].e;
83              res.r += t[l++].r;
84          }
85          if (r&1) {
86              res.h += t[--r].h;
87              res.e += t[r].e;
88              res.r += t[r].r;
89          }
90      }
91  }
92
93  int m;
94
95  int main() {
96      int a, b;
97      node ans;
98      char cmd[2];
99      while (scanf("%d %d", &n, &m) != EOF) {
100         h = sizeof(int) * 8 - __builtin_clz(n);
101         memset(d, 0, sizeof(d));
102         for (int i = 0; i < n; i++) {
103             t[i].h = t[i].e = t[i].r = 0;
104             t[i+n].h = 1, t[i+n].e = 0, t[i+n].r = 0;
105         }
106         build(0, n);
107         for (int i = 0; i < m; i++) {
108             scanf("%s %d %d", cmd, &a, &b);
109             if (cmd[0] == 'C') {
110                 node ans;
111                 query(a-1, b, ans); // open right interval
112                 printf("%d %d %d\n", ans.h, ans.e, ans.r);
113             } else if (cmd[0] == 'M') {
114                 node ans;
115                 modify(a-1, b, 1); // open right interval
116             }
117         }
118         printf("\n");
119     }
120     return 0;
121 }
```

5.5 Min Range DP - Keep it Energized

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define pb push_back
5  #define mp make_pair
6
7  #define MAXN 100010
8  const long long INF = 0x3f3f3f3f;
9
10 typedef pair<long long, pair<long long, long long> > piii;
11
12 long long e[MAXN];
13 piii s[MAXN];
14 long long n, m, x, y, z, ans, res;
15
16 const long long N = MAXN;  // limit for array size
```

```
17  long long t[2 * N];
18
19  void build() {  // build the tree
20      for (long long i = m - 1; i > 0; --i) t[i] = min(t[i<<1], t[i<<1|1]);
21  }
22
23  void modify(long long p, long long value) {  // set value at position p
24      for (t[p += m] = value; p > 1; p >>= 1) t[p>>1] = min(t[p], t[p^1]);
25  }
26
27  long long query(long long l, long long r) {  // sum on interval [l, r)
28      long long res = INF;
29      for (l += m, r += m; l < r; l >>= 1, r >>= 1) {
30          if (l&1) res = min(res, t[l++]);
31          if (r&1) res = min(res, t[--r]);
32      }
33      return res;
34  }
35
36  bool cmp(piii i, piii j) {
37      long long a = e[i.first] + i.second.first;
38      long long b = e[j.first];
39      return a < b;
40  }
41
42  int main() {
43      while (scanf("%lld %lld", &n, &m) != EOF) {
44          e[1] = 0;
45          for (long long i=1; i<=n; i++) {
46              scanf("%lld", &x);
47              e[i+1] = e[i] + x;
48          }
49          for (long long i=0; i<m; i++) {
50              long long a, c;
51              long long b;
52              scanf("%lld %lld %lld", &a, &b, &c);
53              s[i] = mp(a, mp(b, c));
54          }
55          sort(s, s+m);
56      for (long long i=0; i<m; i++) t[i+m] = INF;
57      build();
58          res = INF;
59          for (long long i=m-1; i>=0; i--) {
60        long long costi = INF;
61        if (e[s[i].first] + s[i].second.first >= e[n+1]) costi = s[i].second.second
      ;
62              long long j = upper_bound(s+i, s+m, s[i], cmp) - s;
63              long long costj = INF;
64              if (j <= m) costj = query(i, j) + s[i].second.second;
65              ans = min(costi, costj);
66              if (s[i].first == 1) res = min(res, ans);
67        modify(i, ans);
68          }
69          printf("%lld\n", (res == INF)?-1:res);
70      }
71      return 0;
72  }
```

6 Algorithms

6.1 Knapsack - Pedido de Desculpas

```cpp
#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;

int c, f, n, d;
int vec[2][50];
int mem[10001][51];

int pd( int i, int cap) {
  if (i >= f) return 0;
  if (cap < 0) return 0;
  if (mem[cap][i] != -1) return mem[cap][i];
  int opc1 = pd(i+1, cap);
  int opc2 = 0;
  if (cap - vec[0][i] >= 0)
    opc2 = vec[1][i] + pd(i+1, cap - vec[0][i]);
  mem[cap][i] = max(opc1, opc2);
  return mem[cap][i];
}

int main () {
  int t = 1;
  while (1) {
    scanf("%d %d", &c, &f);
    memset(mem, -1, sizeof(mem));
    if (!c && !f) return 0;
    for (int i = 0; i < f; i++) {
      scanf("%d %d", &n, &d);
      vec[0][i] = n;
      vec[1][i] = d;
    }
    printf("Teste %d\n%d\n\n", t, pd(0, c));
    t++;
  }
  return 0;
}
```

6.2 Linear Transformation + Fast Pow - Quantas Chamadas Recursivas

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

typedef long long ll;

ll sol(ll n, int m) {
    ll a, b, c, d, r;
    a = 1; b = 0; c = 0; d = 1;
    while (n) {
        if (n&1) {
            r = ((d*b) + (c*a)) % m;
            b = (d*(b+a) + (c*b)) % m;
            a = r;
        }
        r = (c*c + d*d) % m;
        d = (d * (2*c + d)) % m;
        c = r;
        n >>= 1;
    }
    return (a+b) % m;
}

int main() {
    ll n;
```

```
25      int b, r, c = 0;
26      while (scanf("%lld %d", &n, &b) && (n || b)) {
27          //r = ((sol(n, b) % b) + b) % b;
28          r = (((2*sol(n, b) - 1) % b) + b) % b;
29          printf("Case %d: %lld %d %d\n", ++c, n, b, r);
30      }
31      return 0;
32  }
```

6.3 LT + FP + Search skipping interval - Registrador de Deslocamento

```
1   #include <bits/stdc++.h>
2
3   using namespace std;
4
5   #define STEP 65536
6
7   typedef unsigned int uint;
8
9   uint trans[32][32], aux[32][32];
10
11  inline uint skip(uint state, uint n) {
12      uint st[32], nst[32], nstate = 0;
13      memset(st, 0, sizeof(st));
14      memset(nst, 0, sizeof(nst));
15      // convert state to array
16      for (int i = 0; i < n; i++) {
17          st[i] = (state>>(n-i-1))&1ll;
18      }
19      // multiply
20      for(int i = 0; i < n; i++) {
21          for(int j = 0; j < n; j++) {
22              nst[i] = (nst[i] + trans[i][j]*st[j])%2;
23          }
24      }
25      // convert array to state
26      for (int i = 0; i < n; i++) {
27          nstate |= (nst[n-1-i]%2)<<i;
28      }
29      return nstate;
30  }
31
32  inline uint next(uint state, uint t, uint qt_bits) {
33      uint bit = (__builtin_popcount(state & t) % 2);
34      return ((bit << (qt_bits - 1ll)) | (state >> 1ll)) & ((1ll << qt_bits) - 1);
35  }
36
37  int main() {
38      uint n, t, ini, end, mask, state;
39      uint bit;
40      long long s = 0;
41
42      while (scanf ("%d %d", &n, &t) && (n || t)) {
43          map<uint, uint> iini, iend;
44          memset(trans, 0, sizeof(trans));
45          s = STEP;
46          mask = 0; // 1 on xor position
47          for (int i = 0; i < t; ++i) {
48              scanf ("%d", &bit);
49              mask |= (1ll << bit);
50              trans[0][n-bit-1] = 1;
51          }
52
53          scanf ("%x %x", &ini, &end);
54
55          // init interval
56          state = ini;
57          for (int i = 0; i < STEP ; ++i) {
58              if (state == end) {
59                  printf("%d\n", i);
```

```
                goto fim;
            }
            if (iini.find(state) != iini.end()) break;
            iini[state] = 1;
            state = next(state, mask, n);
        }
        // end interval
        state = end;
        for (int i = 0; i < STEP ; ++i) {
            if (iend.find(state) != iend.end()) break;
            iend[state] = i;
            state = next(state, mask, n);
        }
        // init trans
        for (int i = 1; i < n; i++) {
            trans[i][i-1] = 1;
        }
        // print trans
        /*for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                printf("%lld", trans[i][j]);
            }
            printf("\n");
        }*/
        // precompute trans^step
        for (int x = 0; x < 16; x++) {
            // trans = trans*trans
            memset(aux, 0, sizeof(aux));
            for(int i = 0; i < n; i++) {
                for(int j = 0; j < n; j++) {
                    for(int k=0; k < n; k++) {
                        aux[i][j] += trans[i][k]*trans[k][j];
                    }
                }
            }
            for(int i = 0; i < n; i++) {
                for(int j = 0; j < n; j++) {
                    trans[i][j] = aux[i][j]%2;
                }
            }
        }

        // start search k-steps at a time
        state = skip(ini, n);
        while (1) {
            if (iend.find(state) != iend.end()) {
                printf("%lld\n", s - iend[state]);
                break;
            } else if (iini.find(state) != iini.end()) {
                printf("*\n");
                break;
            }
            state = skip(state, n);
            s += STEP;
        }
        fim: s++;
    }

    return 0;
}
```

6.4 Big Integer - Krakóvia

```
from sys import stdin, stdout
c = 0
while (1):
    n, f = stdin.readline().split()
    n = int(n)
    f = int(f)
    c += 1

```

```
    if (n == 0) and (f == 0):
        break

    ac = 0
    for i in range(0, n):
        ac += int(stdin.readline())
    stdout.write("Bill #" + str(c) + " costs " + str(ac) + ": each friend should
    pay " + str(int(ac/f)) + "\n\n")
```

6.5 Iterative DP - Bolsa de Valores

```
#include <cstdio>

using namespace std;

#define MAX(a, b) (a>b)?(a):(b)

int main () {
  int n, c;
  int vec[200001];
  scanf("%d %d", &n, &c);
  for ( int i = 0; i < n; i++)
    scanf("%d", &vec[i]);

  int max0=0, max1=0;

  for ( int i=n-1; i >=0; i--){
    max0 = MAX(max0, max1-vec[i]-c);
    max1 = MAX(max1, max0+vec[i]);
  }

  printf("%d\n", max0);
  return 0;
}
```

6.6 Find m-th element in sequence DP - Enumerating Brackets

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

typedef unsigned long long ull;

ull m, tot = 0, mem[1010][1010];
int n, a = 0, f = 0;

ull pd(int a, int f) {
    int pos = a+f;
    if (pos >= n) return 1;
    ull &res = mem[a][f];
    if (res > 0) return res;
    ull opc1 = 0, opc2 = 0;
    if (a < n/2) opc1 = pd(a+1, f);
    if (f < a) opc2 = pd(a, f+1);
    res = min(opc1 + opc2, (ull)1e18);
    return res;
}

int main() {
    scanf("%d %llu", &n, &m);
    memset(mem, 0, sizeof(mem));
    pd(0, 0);
    for (int i=0; i<n; i++) {
        if (m <= tot+mem[a+1][f]) printf("("), a++;
        else printf(")"), tot+=mem[a+1][f], f++;
    }
    printf("\n");
    //printf("%llu\n", pd(0, 0));
    return 0;
}
```

6.7 Max Gain with Probability DP - Quiz Universitário

```
#include <cstdio>
#include <cstring>

#define MAX(a, b) (((a)<(b))?(b):(a))

int N, K;
int P[1001], C[1001];

double memo[1001][1001];
bool calc[1001][1001];

double pd (int qst, int pul) {
    if (qst >= N) return 0.0;
    if (!calc[qst][pul]) {
        double n_pula = (C[qst]/100.0)*(P[qst] + pd(qst+1, pul));
        double pula = 0.0;
        if (pul > 0) pula = (P[qst] + pd(qst+1, pul-1));
        memo[qst][pul] = MAX(pula, n_pula);
        calc[qst][pul] = 1;
        return memo[qst][pul];
    }
    return memo[qst][pul];
}

int main () {
  scanf("%d %d", &N, &K);
  for (int i = 0; i < N; i++) {
    scanf("%d", &P[i]);
  }
  for (int i = 0; i < N; i++) {
    scanf("%d", &C[i]);
  }
    memset(calc, 0, sizeof(calc));
  printf("%.2lf\n", pd(0, K));
  return 0;
}
```

6.8 Max Increasing Subsequence DP - Trainsorting

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

int w[2020], c[2020], d[2020];
int n;

int main() {
    scanf("%d", &n);
    for (int i=0; i<n; i++) scanf("%d", w+i);
    c[n] = d[n] = 0;
    for (int i=n-1; i>=0; i--) {
        for (int j=i+1; j<=n; j++) {
            if ((j==n || w[j]>w[i]) && d[j]+1>d[i]) d[i] = d[j]+1;
            if ((j==n || w[j]<w[i]) && c[j]+1>c[i]) c[i] = c[j]+1;
        }
    }
    int ans = 0;
    for (int i=0; i<n; i++) ans = max(ans, c[i]+d[i]-1);
    printf("%d\n", ans);
    return 0;
}
```

6.9 Binary Search - Pie!

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

```

```
int t, n, f, r, q;
double mv, ia, ib, ih, vol[10010];

int main() {
    scanf("%d", &t);
    while (t--) {
        scanf("%d %d", &n, &f);
        for (int i = 0; i < n; i++) {
            scanf("%d", &r);
            vol[i] = M_PI*r*r;
            if (!i) mv = vol[i];
            mv = max(mv, vol[i]);
        }
        ia = 0.0;
        ib = mv;
        for (int i = 0; i < 80; i++) {
            q = 0;
            ih = (ia+ib)/2.0;
            for (int j = 0; j < n; j++) q += floor(vol[j]/ih);
            if (q <= f) ib = ih;
            else ia = ih;
        }
        printf("%.4lf\n", (ia+ib)/2);
    }
}
```

6.10 Ternary Search - A Caminhada da Vergonha de Cersei

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

#define dist(p, x, y) (sqrt(((x)-(p))*((x)-(p))+(y)*(y)))

int n, p, x[50050], y[50050];
double a, b, m1, m2;

double mdist(double p) {
    double x_ = dist(p, x[0], y[0]);
    for (int i = 1; i < n; i++) {
        x_ = max(x_, dist(p, x[i], y[i]));
    }
    return x_;
}

int main() {
    scanf("%d %d", &n, &p);
    for (int i = 0; i < n; i++)
        scanf("%d %d", x+i, y+i);
    a = 0; b = p;
    for (int i = 0; i < 100; i++) {
        m1 = (a+((b-a)/3));
        m2 = (b-((b-a)/3));
        if (mdist(m1) < mdist(m2)) b = m2;
        else a = m1;
    }
    printf("%.2lf %.2lf\n", (a+b)/2, mdist((a+b)/2));
}
```

6.11 Simpson Integration - Environment Protection

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

int w, d, a, k;
int p[4][9];

double vp(double x, double d){
  double a[4] = {0, 0, 0, 0};
  for(int i=0; i<4; i++)
```

```
10      for(int j=k; j>=0; j--)
11        a[i] = a[i]*x + p[i][j];
12
13      if(a[0]/a[1] < -d) return 0;
14      return (a[0]/a[1])-max((a[2]/a[3]), -d);
15    }
16
17    double itgl(double d){
18      double inc = w/1e5;
19      double ar = 0;
20      for(double x=0; x<w; x+=inc)
21        ar += (vp(x, d)+vp(x+inc, d))*inc/2.0;
22      return ar;
23    }
24
25
26    #define N 1500
27    inline double simpson(double a, double b, double d){
28      double h = (b-a)/N;
29      double h2 = h/2;
30      double h6 = h/6;
31      double x = a;
32      double nx = x + h;
33      double res = 0;
34
35      for(int i = 0; i < N; i++){
36        nx = x + h;
37        res += h6*(vp(x, d)+4*vp(x+h2, d)+vp(nx, d));
38        x = nx;
39      }
40      return res;
41    }
42
43    int main(){
44      while(scanf("%d %d %d %d", &w, &d, &a, &k) != EOF){
45        for(int i=0; i<4; i++)
46        for(int j=0; j<=k; j++)
47          scanf("%d", p[i]+j);
48
49        double i = 0, j = d;
50        for(int k=0; k<30; k++){
51          double m = (i+j)/2;
52
53          if(simpson(0, w, m) < a) i = m;
54          else j = m;
55        }
56
57        printf("%.5lf\n", (i+j)/2);
58      }
59
60      return 0;
61    }
```