Contents

1 Reminder

1.1 Bug List

- 沒開 long long
- 陣列戳出界／開不夠大/ 開太大本地 compile 噴怪 error
- 傳之前先確定選對檔案
- 寫好的函式忘記呼叫
- 變數打錯
- 0-base / 1-base
- 忘記初始化
- == 打成 =
- <= 打成 <+
- dp[i] 從 dp[i-1] 轉移時忘記特判 i > 0
- std::sort 比較運算子寫成 < 或是讓 = 的情況為 true
- 漏 case / 分 case 要好好想
- 線段樹改值懶標初始值不能設為 0
- DFS 的時候不小心覆寫到全域變數
- 浮點數誤差
- 多筆測資不能沒讀完直接 return
- 記得刪 cerr

1.2 OwO

- 可以構造複雜點的測資幫助思考
- 真的卡太久請跳題
- Enjoy The Contest!

2 Basic

2.1 Vimrc

```
set number relativenumber ai t_Co=256 tabstop=4
set mouse=a shiftwidth=4 encoding=utf8
set bs=2 ruler laststatus=2 cmdheight=2
set clipboard=unnamedplus showcmd autoread
set belloff=all
filetype indent on
"set guifont Hack:h16
":set guifont?

inoremap ( ()<Esc>i
inoremap " ""<Esc>i
inoremap [ []<Esc>i
inoremap ' ''<Esc>i
inoremap { {<CR>}<Esc>ko

vmap <C-c> "+y
inoremap <C-v> <Esc>p
nnoremap <C-v> p

nnoremap <tab> gt
nnoremap <S-tab> gT
inoremap <C-n> <Esc>:tabnew<CR>
nnoremap <C-n> :tabnew<CR>

inoremap <F9> <Esc>:w<CR>:!~/runcpp.sh %:p:t %:p:h<CR>
nnoremap <F9> :w<CR>:!~/runcpp.sh %:p:t %:p:h<CR>

syntax on
colorscheme desert
set filetype=cpp
set background=dark
hi Normal ctermfg=white ctermbg=black
```

2.2 Runcpp.sh

```
#! /bin/bash
clear
echo "Start compiling $1..."
```

```bash
 4  echo
 5  g++ -O2 -std=c++20 -Wall -Wextra -Wshadow $2/$1 -o $2/
        out
 6  if [ "$?" -ne 0 ]
 7  then
 8      exit 1
 9  fi
10  echo
11  echo "Done compiling"
12  echo "========================="
13  echo
14  echo "Input file:"
15  echo
16  cat $2/in.txt
17  echo
18  echo "========================="
19  echo
20  declare startTime=`date +%s%N`
21  $2/out < $2/in.txt > $2/out.txt
22  declare endTime=`date +%s%N`
23  delta=`expr $endTime - $startTime`
24  delta=`expr $delta / 1000000`
25  cat $2/out.txt
26  echo
27  echo "time: $delta ms"
```

2.3   Stress

```bash
 1  g++ gen.cpp -o gen.out
 2  g++ ac.cpp -o ac.out
 3  g++ wa.cpp -o wa.out
 4  for ((i=0;;i++))
 5  do
 6      echo "$i"
 7      ./gen.out > in.txt
 8      ./ac.out < in.txt > ac.txt
 9      ./wa.out < in.txt > wa.txt
10      diff ac.txt wa.txt || break
11  done
```

2.4   PBDS

```cpp
 1  #include <bits/extc++.h>
 2  using namespace __gnu_pbds;
 3
 4  // map
 5  tree<int, int, less<>, rb_tree_tag,
        tree_order_statistics_node_update> tr;
 6  tr.order_of_key(element);
 7  tr.find_by_order(rank);
 8
 9  // set
10  tree<int, null_type, less<>, rb_tree_tag,
        tree_order_statistics_node_update> tr;
11  tr.order_of_key(element);
12  tr.find_by_order(rank);
13
14  // priority queue
15  __gnu_pbds::priority_queue<int, less<int> > big_q;  //
        Big First
16  __gnu_pbds::priority_queue<int, greater<int> > small_q;
         // Small First
17  q1.join(q2);  // join
```

2.5   Random

```cpp
 1  mt19937 gen(chrono::steady_clock::now().
        time_since_epoch().count());
 2  uniform_int_distribution<int> dis(1, 100);
 3  cout << dis(gen) << endl;
 4  shuffle(v.begin(), v.end(), gen);
```

3   Python

3.1   I/O

```python
 1  import sys
 2  input = sys.stdin.readline
 3
 4  # Input
 5  def readInt():
 6      return int(input())
 7  def readList():
 8      return list(map(int,input().split()))
 9  def readStr():
10      s = input()
11      return list(s[:len(s) - 1])
12  def readVars():
13      return map(int,input().split())
14
15  # Output
16  sys.stdout.write(string)
17
18  # faster
19  def main():
20      pass
21  main()
```

3.2   Decimal

```python
 1  from decimal import *
 2  getcontext().prec = 2500000
 3  getcontext().Emax = 2500000
 4  a,b = Decimal(input()),Decimal(input())
 5  a*=b
 6  print(a)
```

4   Data Structure

4.1   BIT

```cpp
 1  struct BIT
 2  {
 3      int n;
 4      long long bit[N];
 5
 6      void init(int x, vector<long long> &a)
 7      {
 8          n = x;
 9          for(int i=1, j; i<=n; i++)
10          {
11              bit[i] += a[i-1], j = i + (i & -i);
12              if(j <= n)  bit[j] += bit[i];
13          }
14      }
15
16      void update(int x, long long dif)
17      {
18          while(x <= n)    bit[x] += dif, x += x & -x;
19      }
20
21      long long query(int l, int r)
22      {
23          if(l != 1)  return query(1, r) - query(1, l-1);
24
25          long long ret = 0;
26          while(l <= r)    ret += bit[r], r -= r & -r;
27          return ret;
28      }
29  }bm;
```

4.2   DSU

```cpp
 1  struct DSU
 2  {
 3      int h[N], s[N];
 4
 5      void init(int n)
 6      { iota(h, h+n+1, 0), fill(s, s+n+1, 1); }
 7
 8      int fh(int x)
 9      { return (h[x]==x? x: h[x]=fh(h[x])); }
10
11      bool mer(int x, int y)
```

```
12      {
13          x = fh(x), y = fh(y);
14          if(x == y)    return 0;
15          if(s[x] < s[y])    swap(x, y);
16          s[x] += s[y], s[y] = 0;
17          h[y] = x;
18          return 1;
19      }
20  }bm;
```

4.3 Segment Tree

```
1   struct segtree
2   {
3       int n, seg[1<<19];
4
5       void init(int x)
6       {
7           n = 1<<(__lg(x) + 1);
8           for(int i=1; i<2*n; i++)
9               seg[i] = inf;
10      }
11
12      void update(int x, int val)
13      {
14          x += n;
15          seg[x] = val, x /= 2;
16          while(x)
17              seg[x] = min(seg[2*x], seg[2*x+1]), x /= 2;
18      }
19
20      int query(int l, int r)
21      {
22          l += n, r += n;
23          int ret = inf;
24          while(l < r)
25          {
26              if(l & 1)
27                  ret = min(ret, seg[l++]);
28              if(r & 1)
29                  ret = min(ret, seg[--r]);
30              l /= 2, r /= 2;
31          }
32          return ret;
33      }
34  }bm;
```

4.4 Merging on Seg-Tree

```
1   struct segtree
2   {
3       struct node
4       {
5           long long sum, pre;
6           node()
7           {
8               sum = pre = 0;
9           }
10      };
11
12      node mer(node x, node y)
13      {
14          node ret;
15          ret.sum = x.sum + y.sum;
16          ret.pre = max(x.pre, x.sum + y.pre);
17          return ret;
18      }
19
20      int n;
21      node seg[1<<19];
22
23      void init(int x)
24      {
25          n = 1<<(__lg(x) + 1);
26      }
27
28      void print()
29      {
30          for(int i=1, j=1; i<=__lg(n)+1; i++)
31          {
```

```
32              for(; j<(1<<i); j++)
33                  cout << seg[j].sum << " ";
34              cout << "\n";
35          }
36          cout << "\n";
37      }
38
39      void update(int x, long long val)
40      {
41          x += n;
42          seg[x].sum = val, seg[x].pre = max(0LL, val);
43          x /= 2;
44          while(x)
45              seg[x] = mer(seg[2*x], seg[2*x+1]), x /= 2;
46      }
47
48      long long query(int l, int r)
49      {
50          l += n, r += n;
51          node retl, retr;
52          while(l < r)
53          {
54              if(l & 1)
55                  retl = mer(retl, seg[l++]);
56              if(r & 1)
57                  retr = mer(seg[--r], retr);
58              l /= 2, r /= 2;
59          }
60          return mer(retl, retr).pre;
61      }
62  }bm;
```

4.5 Heavy Light Decomposition

```
1   const int N=2e5+5;
2   int n,dfn[N],son[N],top[N],num[N],dep[N],p[N];
3   vector<int>path[N];
4   struct node
5   {
6       int mx,sum;
7   }seg[N<<2];
8   void update(int x,int l,int r,int qx,int val)
9   {
10      if(l==r)
11      {
12          seg[x].mx=seg[x].sum=val;
13          return;
14      }
15      int mid=(l+r)>>1;
16      if(qx<=mid)update(x<<1,l,mid,qx,val);
17      else update(x<<1|1,mid+1,r,qx,val);
18      seg[x].mx=max(seg[x<<1].mx,seg[x<<1|1].mx);
19      seg[x].sum=seg[x<<1].sum+seg[x<<1|1].sum;
20  }
21  int big(int x,int l,int r,int ql,int qr)
22  {
23      if(ql<=l&&r<=qr)return seg[x].mx;
24      int mid=(l+r)>>1;
25      int res=-INF;
26      if(ql<=mid)res=max(res,big(x<<1,l,mid,ql,qr));
27      if(mid<qr)res=max(res,big(x<<1|1,mid+1,r,ql,qr));
28      return res;
29  }
30  int ask(int x,int l,int r,int ql,int qr)
31  {
32      if(ql<=l&&r<=qr)return seg[x].sum;
33      int mid=(l+r)>>1;
34      int res=0;
35      if(ql<=mid)res+=ask(x<<1,l,mid,ql,qr);
36      if(mid<qr)res+=ask(x<<1|1,mid+1,r,ql,qr);
37      return res;
38  }
39  void dfs1(int now)
40  {
41      son[now]=-1;
42      num[now]=1;
43      for(auto i:path[now])
44      {
45          if(!dep[i])
46          {
47              dep[i]=dep[now]+1;
```

```
48          p[i]=now;
49          dfs1(i);
50          num[now]+=num[i];
51          if(son[now]==-1||num[i]>num[son[now]])son[
              now]=i;
52       }
53     }
54 }
55 int cnt;
56 void dfs2(int now,int t)
57 {
58     top[now]=t;
59     cnt++;
60     dfn[now]=cnt;
61     if(son[now]==-1)return;
62     dfs2(son[now],t);
63     for(auto i:path[now])
64         if(i!=p[now]&&i!=son[now])
65             dfs2(i,i);
66 }
67 int path_big(int x,int y)
68 {
69     int res=-INF;
70     while(top[x]!=top[y])
71     {
72         if(dep[top[x]]<dep[top[y]])swap(x,y);
73         res=max(res,big(1,1,n,dfn[top[x]],dfn[x]));
74         x=p[top[x]];
75     }
76     if(dfn[x]>dfn[y])swap(x,y);
77     res=max(res,big(1,1,n,dfn[x],dfn[y]));
78     return res;
79 }
80 int path_sum(int x,int y)
81 {
82     int res=0;
83     while(top[x]!=top[y])
84     {
85         if(dep[top[x]]<dep[top[y]])swap(x,y);
86         res+=ask(1,1,n,dfn[top[x]],dfn[x]);
87         x=p[top[x]];
88     }
89     if(dfn[x]>dfn[y])swap(x,y);
90     res+=ask(1,1,n,dfn[x],dfn[y]);
91     return res;
92 }
93 void buildTree()
94 {
95     FOR(i,0,n-1)
96     {
97         int a,b;cin>>a>>b;
98         path[a].pb(b);
99         path[b].pb(a);
100     }
101 }
102 void buildHLD(int root)
103 {
104     dep[root]=1;
105     dfs1(root);
106     dfs2(root,root);
107     FOR(i,1,n+1)
108     {
109         int now;cin>>now;
110         update(1,1,n,dfn[i],now);
111     }
112 }
```

4.6   Skew Heap

```
1 struct node{
2     node *l,*r;
3     int v;
4     node(int x):v(x){
5         l=r=nullptr;
6     }
7 };
8 node* merge(node* a,node* b){
9     if(!a||!b) return a?:b;
10 //  min heap
11     if(a->v>b->v) swap(a,b);
12     a->r=merge(a->r,b);
```

```
13     swap(a->l,a->r);
14     return a;
15 }
```

4.7   Leftist Heap

```
1 struct node{
2     node *l,*r;
3     int d, v;
4     node(int x):d(1),v(x){
5         l=r=nullptr;
6     }
7 };
8 static inline int d(node* x){return x?x->d:0;}
9 node* merge(node* a,node* b){
10     if(!a||!b) return a?:b;
11 //  min heap
12     if(a->v>b->v) swap(a,b);
13     a->r=merge(a->r,b);
14     if(d(a->l)<d(a->r))
15         swap(a->l,a->r);
16     a->d=d(a->r)+1;
17     return a;
18 }
```

4.8   Treap

```
1 mt19937 rng(random_device{}());
2 struct Treap
3 {
4     Treap *l,*r;
5     int val,num,pri;
6     Treap(int k)
7     {
8         l=r=NULL;
9         val=k;
10         num=1;
11         pri=rng();
12     }
13 };
14 int siz(Treap *now){return now?now->num:0;}
15 void pull(Treap *&now)
16 {
17     now->num=siz(now->l)+siz(now->r)+1;
18 }
19 Treap* merge(Treap *a,Treap *b)
20 {
21     if(!a||!b)return a?a:b;
22     else if(a->pri>b->pri)
23     {
24         a->r=merge(a->r,b);
25         pull(a);
26         return a;
27     }
28     else
29     {
30         b->l=merge(a,b->l);
31         pull(b);
32         return b;
33     }
34 }
35 void split_size(Treap *rt,Treap *&a,Treap *&b,int val)
36 {
37     if(!rt)
38     {
39         a=b=NULL;
40         return;
41     }
42     if(siz(rt->l)+1>val)
43     {
44         b=rt;
45         split_size(rt->l,a,b->l,val);
46         pull(b);
47     }
48     else
49     {
50         a=rt;
51         split_size(rt->r,a->r,b,val-siz(a->l)-1);
52         pull(a);
53     }
```

```
54  }
55  void split_val(Treap *rt,Treap *&a,Treap *&b,int val)
56  {
57      if(!rt)
58      {
59          a=b=NULL;
60          return;
61      }
62      if(rt->val<=val)
63      {
64          a=rt;
65          split_val(rt->r,a->r,b,val);
66          pull(a);
67      }
68      else
69      {
70          b=rt;
71          split_val(rt->l,a,b->l,val);
72          pull(b);
73      }
74  }
75  void treap_dfs(Treap *now)
76  {
77      if(!now)return;
78      treap_dfs(now->l);
79      cout<<now->val<<" ";
80      treap_dfs(now->r);
81  }
```

4.9　Persistent Treap

```
1   struct node {
2       node *l, *r;
3       char c; int v, sz;
4       node(char x = '$'): c(x), v(mt()), sz(1) {
5           l = r = nullptr;
6       }
7       node(node* p) {*this = *p;}
8       void pull() {
9           sz = 1;
10          for (auto i : {l, r})
11              if (i) sz += i->sz;
12      }
13  } arr[maxn], *ptr = arr;
14  inline int size(node* p) {return p ? p->sz : 0;}
15  node* merge(node* a, node* b) {
16      if (!a || !b) return a ? : b;
17      if (a->v < b->v) {
18          node* ret = new(ptr++) node(a);
19          ret->r = merge(ret->r, b), ret->pull();
20          return ret;
21      }
22      else {
23          node* ret = new(ptr++) node(b);
24          ret->l = merge(a, ret->l), ret->pull();
25          return ret;
26      }
27  }
28  P<node*> split(node* p, int k) {
29      if (!p) return {nullptr, nullptr};
30      if (k >= size(p->l) + 1) {
31          auto [a, b] = split(p->r, k - size(p->l) - 1);
32          node* ret = new(ptr++) node(p);
33          ret->r = a, ret->pull();
34          return {ret, b};
35      }
36      else {
37          auto [a, b] = split(p->l, k);
38          node* ret = new(ptr++) node(p);
39          ret->l = b, ret->pull();
40          return {a, ret};
41      }
42  }
```

4.10　Li Chao Tree

```
1   constexpr int maxn = 5e4 + 5;
2   struct line {
3       ld a, b;
4       ld operator()(ld x) {return a * x + b;}
5   } arr[(maxn + 1) << 2];
6   bool operator<(line a, line b) {return a.a < b.a;}
7   #define m ((l+r)>>1)
8   void insert(line x, int i = 1, int l = 0, int r = maxn)
        {
9       if (r - l == 1) {
10          if (x(l) > arr[i](l))
11              arr[i] = x;
12          return;
13      }
14      line a = max(arr[i], x), b = min(arr[i], x);
15      if (a(m) > b(m))
16          arr[i] = a, insert(b, i << 1, l, m);
17      else
18          arr[i] = b, insert(a, i << 1 | 1, m, r);
19  }
20  ld query(int x, int i = 1, int l = 0, int r = maxn) {
21      if (x < l || r <= x) return -numeric_limits<ld>::
            max();
22      if (r - l == 1) return arr[i](x);
23      return max({arr[i](x), query(x, i << 1, l, m),
            query(x, i << 1 | 1, m, r)});
24  }
25  #undef m
```

4.11　Time Segment Tree

```
1   constexpr int maxn = 1e5 + 5;
2   V<P<int>> arr[(maxn + 1) << 2];
3   V<int> dsu, sz;
4   V<tuple<int, int, int>> his;
5   int cnt, q;
6   int find(int x) {
7       return x == dsu[x] ? x : find(dsu[x]);
8   };
9   inline bool merge(int x, int y) {
10      int a = find(x), b = find(y);
11      if (a == b) return false;
12      if (sz[a] > sz[b]) swap(a, b);
13      his.emplace_back(a, b, sz[b]), dsu[a] = b, sz[b] +=
            sz[a];
14      return true;
15  };
16  inline void undo() {
17      auto [a, b, s] = his.back(); his.pop_back();
18      dsu[a] = a, sz[b] = s;
19  }
20  #define m ((l + r) >> 1)
21  void insert(int ql, int qr, P<int> x, int i = 1, int l
        = 0, int r = q) {
22      // debug(ql, qr, x); return;
23      if (qr <= l || r <= ql) return;
24      if (ql <= l && r <= qr) {arr[i].push_back(x);
            return;}
25      if (qr <= m)
26          insert(ql, qr, x, i << 1, l, m);
27      else if (m <= ql)
28          insert(ql, qr, x, i << 1 | 1, m, r);
29      else {
30          insert(ql, qr, x, i << 1, l, m);
31          insert(ql, qr, x, i << 1 | 1, m, r);
32      }
33  }
34  void traversal(V<int>& ans, int i = 1, int l = 0, int r
         = q) {
35      int opcnt = 0;
36      // debug(i, l, r);
37      for (auto [a, b] : arr[i])
38          if (merge(a, b))
39              opcnt++, cnt--;
40      if (r - l == 1) ans[l] = cnt;
41      else {
42          traversal(ans, i << 1, l, m);
43          traversal(ans, i << 1 | 1, m, r);
44      }
45      while (opcnt--)
46          undo(), cnt++;
47      arr[i].clear();
48  }
49  #undef m
50  inline void solve() {
```

```cpp
51    int n, m; cin>>n>>m>>q,q++;
52    dsu.resize(cnt = n), sz.assign(n, 1);
53    iota(dsu.begin(), dsu.end(), 0);
54    // a, b, time, operation
55    unordered_map<ll, V<int>> s;
56    for (int i = 0; i < m; i++) {
57        int a, b; cin>>a>>b;
58        if (a > b) swap(a, b);
59        s[((ll)a << 32) | b].emplace_back(0);
60    }
61    for (int i = 1; i < q; i++) {
62        int op,a, b;
63        cin>>op>>a>>b;
64        if (a > b) swap(a, b);
65        switch (op) {
66        case 1:
67            s[((ll)a << 32) | b].push_back(i);
68            break;
69        case 2:
70            auto tmp = s[((ll)a << 32) | b].back();
71            s[((ll)a << 32) | b].pop_back();
72            insert(tmp, i, P<int> {a, b});
73        }
74    }
75    for (auto [p, v] : s) {
76        int a = p >> 32, b = p & -1;
77        while (v.size()) {
78            insert(v.back(), q, P<int> {a, b});
79            v.pop_back();
80        }
81    }
82    V<int> ans(q);
83    traversal(ans);
84    for (auto i : ans)
85        cout<<i<<' ';
86    cout<<endl;
87 }
```

5   DP

5.1   Aliens

```cpp
1  int n; ll k;
2  vector<ll> a;
3  vector<pll> dp[2];
4  void init() {
5      cin >> n >> k;
6      Each(i, dp) i.clear(), i.resize(n);
7      a.clear(); a.resize(n);
8      Each(i, a) cin >> i;
9  }
10 pll calc(ll p) {
11     dp[0][0] = mp(0, 0);
12     dp[1][0] = mp(-a[0], 0);
13     FOR(i, 1, n, 1) {
14         if (dp[0][i-1].F > dp[1][i-1].F + a[i] - p) {
15             dp[0][i] = dp[0][i-1];
16         } else if (dp[0][i-1].F < dp[1][i-1].F + a[i] -
               p) {
17             dp[0][i] = mp(dp[1][i-1].F + a[i] - p, dp
                   [1][i-1].S+1);
18         } else {
19             dp[0][i] = mp(dp[0][i-1].F, min(dp[0][i-1].
                   S, dp[1][i-1].S+1));
20         }
21         if (dp[0][i-1].F - a[i] > dp[1][i-1].F) {
22             dp[1][i] = mp(dp[0][i-1].F - a[i], dp[0][i
                   -1].S);
23         } else if (dp[0][i-1].F - a[i] < dp[1][i-1].F)
                   {
24             dp[1][i] = dp[1][i-1];
25         } else {
26             dp[1][i] = mp(dp[1][i-1].F, min(dp[0][i-1].
                   S, dp[1][i-1].S));
27         }
28     }
29     return dp[0][n-1];
30 }
31 void solve() {
32     ll l = 0, r = 1e7;
```

```cpp
33     pll res = calc(0);
34     if (res.S <= k) return cout << res.F << endl, void
           ();
35     while (l < r) {
36         ll mid = (l+r)>>1;
37         res = calc(mid);
38         if (res.S <= k) r = mid;
39         else l = mid+1;
40     }
41     res = calc(l);
42     cout << res.F + k*l << endl;
43 }
```

6   Graph

6.1   Bellman-Ford + SPFA

```cpp
1  int n, m;
2
3  // Graph
4  vector<vector<pair<int, ll> > > g;
5  vector<ll> dis;
6  vector<bool> negCycle;
7
8  // SPFA
9  vector<int> rlx;
10 queue<int> q;
11 vector<bool> inq;
12 vector<int> pa;
13 void SPFA(vector<int>& src) {
14     dis.assign(n+1, LINF);
15     negCycle.assign(n+1, false);
16     rlx.assign(n+1, 0);
17     while (!q.empty()) q.pop();
18     inq.assign(n+1, false);
19     pa.assign(n+1, -1);
20
21     for (auto& s : src) {
22         dis[s] = 0;
23         q.push(s); inq[s] = true;
24     }
25
26     while (!q.empty()) {
27         int u = q.front();
28         q.pop(); inq[u] = false;
29         if (rlx[u] >= n) {
30             negCycle[u] = true;
31         }
32         else for (auto& e : g[u]) {
33             int v = e.first;
34             ll w = e.second;
35             if (dis[v] > dis[u] + w) {
36                 dis[v] = dis[u] + w;
37                 rlx[v] = rlx[u] + 1;
38                 pa[v] = u;
39                 if (!inq[v]) {
40                     q.push(v);
41                     inq[v] = true;
42 } } } } }
43
44
45 // Bellman-Ford
46 queue<int> q;
47 vector<int> pa;
48 void BellmanFord(vector<int>& src) {
49     dis.assign(n+1, LINF);
50     negCycle.assign(n+1, false);
51     pa.assign(n+1, -1);
52
53     for (auto& s : src) dis[s] = 0;
54
55     for (int rlx = 1; rlx <= n; rlx++) {
56         for (int u = 1; u <= n; u++) {
57             if (dis[u] == LINF) continue;  // Important
                   !!
58             for (auto& e : g[u]) {
59                 int v = e.first; ll w = e.second;
60                 if (dis[v] > dis[u] + w) {
61                     dis[v] = dis[u] + w;
62                     pa[v] = u;
```

```
63                    if (rlx == n) negCycle[v] = true;
64 } } } } }
65
66
67 // Negative Cycle Detection
68 void NegCycleDetect() {
69 /* No Neg Cycle: NO
70 Exist Any Neg Cycle:
71 YES
72 v0 v1 v2 ... vk v0 */
73
74     vector<int> src;
75     for (int i = 1; i <= n; i++)
76         src.emplace_back(i);
77
78     SPFA(src);
79     // BellmanFord(src);
80
81     int ptr = -1;
82     for (int i = 1; i <= n; i++) if (negCycle[i])
83         { ptr = i; break; }
84
85     if (ptr == -1) { return cout << "NO" << endl, void
       (); }
86
87     cout << "YES\n";
88     vector<int> ans;
89     vector<bool> vis(n+1, false);
90
91     while (true) {
92         ans.emplace_back(ptr);
93         if (vis[ptr]) break;
94         vis[ptr] = true;
95         ptr = pa[ptr];
96     }
97     reverse(ans.begin(), ans.end());
98
99     vis.assign(n+1, false);
100    for (auto& x : ans) {
101        cout << x << ' ';
102        if (vis[x]) break;
103        vis[x] = true;
104    }
105    cout << endl;
106 }
107
108 // Distance Calculation
109 void calcDis(int s) {
110    vector<int> src;
111    src.emplace_back(s);
112    SPFA(src);
113    // BellmanFord(src);
114
115    while (!q.empty()) q.pop();
116    for (int i = 1; i <= n; i++)
117        if (negCycle[i]) q.push(i);
118
119    while (!q.empty()) {
120        int u = q.front(); q.pop();
121        for (auto& e : g[u]) {
122            int v = e.first;
123            if (!negCycle[v]) {
124                q.push(v);
125                negCycle[v] = true;
126 } } } }
```

6.2   BCC - AP

```
1  int n, m;
2  int low[maxn], dfn[maxn], instp;
3  vector<int> E, g[maxn];
4  bitset<maxn> isap;
5  bitset<maxm> vis;
6  stack<int> stk;
7  int bccnt;
8  vector<int> bcc[maxn];
9  inline void popout(int u) {
10     bccnt++;
11     bcc[bccnt].emplace_back(u);
12     while (!stk.empty()) {
13         int v = stk.top();
```

```
14         if (u == v) break;
15         stk.pop();
16         bcc[bccnt].emplace_back(v);
17     }
18 }
19 void dfs(int u, bool rt = 0) {
20     stk.push(u);
21     low[u] = dfn[u] = ++instp;
22     int kid = 0;
23     Each(e, g[u]) {
24         if (vis[e]) continue;
25         vis[e] = true;
26         int v = E[e]^u;
27         if (!dfn[v]) {
28             // tree edge
29             kid++; dfs(v);
30             low[u] = min(low[u], low[v]);
31             if (!rt && low[v] >= dfn[u]) {
32                 // bcc found: u is ap
33                 isap[u] = true;
34                 popout(u);
35             }
36         } else {
37             // back edge
38             low[u] = min(low[u], dfn[v]);
39         }
40     }
41     // special case: root
42     if (rt) {
43         if (kid > 1) isap[u] = true;
44         popout(u);
45     }
46 }
47 void init() {
48     cin >> n >> m;
49     fill(low, low+maxn, INF);
50     REP(i, m) {
51         int u, v;
52         cin >> u >> v;
53         g[u].emplace_back(i);
54         g[v].emplace_back(i);
55         E.emplace_back(u^v);
56     }
57 }
58 void solve() {
59     FOR(i, 1, n+1, 1) {
60         if (!dfn[i]) dfs(i, true);
61     }
62     vector<int> ans;
63     int cnt = 0;
64     FOR(i, 1, n+1, 1) {
65         if (isap[i]) cnt++, ans.emplace_back(i);
66     }
67     cout << cnt << endl;
68     Each(i, ans) cout << i << ' ';
69     cout << endl;
70 }
```

6.3   BCC - Bridge

```
1  int n, m;
2  vector<int> g[maxn], E;
3  int low[maxn], dfn[maxn], instp;
4  int bccnt, bccid[maxn];
5  stack<int> stk;
6  bitset<maxm> vis, isbrg;
7  void init() {
8      cin >> n >> m;
9      REP(i, m) {
10         int u, v;
11         cin >> u >> v;
12         E.emplace_back(u^v);
13         g[u].emplace_back(i);
14         g[v].emplace_back(i);
15     }
16     fill(low, low+maxn, INF);
17 }
18 void popout(int u) {
19     bccnt++;
20     while (!stk.empty()) {
21         int v = stk.top();
```

```
22          if (v == u) break;
23          stk.pop();
24          bccid[v] = bccnt;
25      }
26  }
27  void dfs(int u) {
28      stk.push(u);
29      low[u] = dfn[u] = ++instp;
30
31      Each(e, g[u]) {
32          if (vis[e]) continue;
33          vis[e] = true;
34
35          int v = E[e]^u;
36          if (dfn[v]) {
37              // back edge
38              low[u] = min(low[u], dfn[v]);
39          } else {
40              // tree edge
41              dfs(v);
42              low[u] = min(low[u], low[v]);
43              if (low[v] == dfn[v]) {
44                  isbrg[e] = true;
45                  popout(u);
46              }
47          }
48      }
49  }
50  void solve() {
51      FOR(i, 1, n+1, 1) {
52          if (!dfn[i]) dfs(i);
53      }
54      vector<pii> ans;
55      vis.reset();
56      FOR(u, 1, n+1, 1) {
57          Each(e, g[u]) {
58              if (!isbrg[e] || vis[e]) continue;
59              vis[e] = true;
60              int v = E[e]^u;
61              ans.emplace_back(mp(u, v));
62          }
63      }
64      cout << (int)ans.size() << endl;
65      Each(e, ans) cout << e.F << ' ' << e.S << endl;
66  }
```

6.4 SCC - Tarjan

```
1  // 2-SAT
2  vector<int> E, g[maxn];   // 1~n, n+1~2n
3  int low[maxn], in[maxn], instp;
4  int sccnt, sccid[maxn];
5
6  stack<int> stk;
7  bitset<maxn> ins, vis;
8
9  int n, m;
10
11  void init() {
12      cin >> m >> n;
13      E.clear();
14      fill(g, g+maxn, vector<int>());
15      fill(low, low+maxn, INF);
16      memset(in, 0, sizeof(in));
17      instp = 1;
18      sccnt = 0;
19      memset(sccid, 0, sizeof(sccid));
20      ins.reset();
21      vis.reset();
22  }
23
24  inline int no(int u) {
25      return (u > n ? u-n : u+n);
26  }
27
28  int ecnt = 0;
29  inline void clause(int u, int v) {
30      E.eb(no(u)^v);
31      g[no(u)].eb(ecnt++);
32      E.eb(no(v)^u);
33      g[no(v)].eb(ecnt++);
```

```
34  }
35
36  void dfs(int u) {
37      in[u] = instp++;
38      low[u] = in[u];
39      stk.push(u);
40      ins[u] = true;
41
42      Each(e, g[u]) {
43          if (vis[e]) continue;
44          vis[e] = true;
45
46          int v = E[e]^u;
47          if (ins[v]) low[u] = min(low[u], in[v]);
48          else if (!in[v]) {
49              dfs(v);
50              low[u] = min(low[u], low[v]);
51          }
52      }
53
54      if (low[u] == in[u]) {
55          sccnt++;
56          while (!stk.empty()) {
57              int v = stk.top();
58              stk.pop();
59              ins[v] = false;
60              sccid[v] = sccnt;
61              if (u == v) break;
62          }
63      }
64  }
65
66
67  int main() {
68      WiwiHorz
69      init();
70
71      REP(i, m) {
72          char su, sv;
73          int u, v;
74          cin >> su >> u >> sv >> v;
75          if (su == '-') u = no(u);
76          if (sv == '-') v = no(v);
77          clause(u, v);
78      }
79
80      FOR(i, 1, 2*n+1, 1) {
81          if (!in[i]) dfs(i);
82      }
83
84      FOR(u, 1, n+1, 1) {
85          int du = no(u);
86          if (sccid[u] == sccid[du]) {
87              return cout << "IMPOSSIBLE\n", 0;
88          }
89      }
90
91      FOR(u, 1, n+1, 1) {
92          int du = no(u);
93          cout << (sccid[u] < sccid[du] ? '+' : '-') << '
                ';
94      }
95      cout << endl;
96
97      return 0;
98  }
```

6.5 SCC - Kosaraju

```
1  const int N = 1e5 + 10;
2  vector<int> ed[N], ed_b[N]; // 反邊
3  vector<int> SCC(N); // 最後SCC的分組
4  bitset<N> vis;
5  int SCC_cnt;
6  int n, m;
7  vector<int> pre; // 後序遍歷
8
9  void dfs(int x)
10  {
11      vis[x] = 1;
12      for(int i : ed[x]) {
```

```
13        if(vis[i]) continue;
14        dfs(i);
15    }
16    pre.push_back(x);
17 }
18
19 void dfs2(int x)
20 {
21    vis[x] = 1;
22    SCC[x] = SCC_cnt;
23    for(int i : ed_b[x]) {
24        if(vis[i]) continue;
25        dfs2(i);
26    }
27 }
28
29 void kosaraju()
30 {
31    for(int i = 1; i <= n; i++) {
32        if(!vis[i]) {
33            dfs(i);
34        }
35    }
36    SCC_cnt = 0;
37    vis = 0;
38    for(int i = n - 1; i >= 0; i--) {
39        if(!vis[pre[i]]) {
40            SCC_cnt++;
41            dfs2(pre[i]);
42        }
43    }
44 }
```

6.6   Eulerian Path - Undir

```
1  // from 1 to n
2  #define gg return cout << "IMPOSSIBLE\n", void();
3
4  int n, m;
5  vector<int> g[maxn];
6  bitset<maxn> inodd;
7
8  void init() {
9  cin >> n >> m;
10 inodd.reset();
11 for (int i = 0; i < m; i++) {
12     int u, v; cin >> u >> v;
13     inodd[u] = inodd[u] ^ true;
14     inodd[v] = inodd[v] ^ true;
15     g[u].emplace_back(v);
16     g[v].emplace_back(u);
17 } }
18 stack<int> stk;
19 void dfs(int u) {
20     while (!g[u].empty()) {
21         int v = g[u].back();
22         g[u].pop_back();
23         dfs(v);
24     }
25 stk.push(u);}
```

6.7   Eulerian Path - Dir

```
1  // from node 1 to node n
2  #define gg return cout << "IMPOSSIBLE\n", 0
3
4  int n, m;
5  vector<int> g[maxn];
6  stack<int> stk;
7  int in[maxn], out[maxn];
8
9  void init() {
10 cin >> n >> m;
11 for (int i = 0; i < m; i++) {
12     int u, v; cin >> u >> v;
13     g[u].emplace_back(v);
14     out[u]++, in[v]++;
15 }
16 for (int i = 1; i <= n; i++) {
17     if (i == 1 && out[i]-in[i] != 1) gg;
```

```
18     if (i == n && in[i]-out[i] != 1) gg;
19     if (i != 1 && i != n && in[i] != out[i]) gg;
20 } }
21 void dfs(int u) {
22     while (!g[u].empty()) {
23         int v = g[u].back();
24         g[u].pop_back();
25         dfs(v);
26     }
27     stk.push(u);
28 }
29 void solve() {
30     dfs(1)
31     for (int i = 1; i <= n; i++)
32         if ((int)g[i].size()) gg;
33     while (!stk.empty()) {
34         int u = stk.top();
35         stk.pop();
36         cout << u << ' ';
37 } }
```

6.8   Hamilton Path

```
1  // top down DP
2  // Be Aware Of Multiple Edges
3  int n, m;
4  ll dp[maxn][1<<maxn];
5  int adj[maxn][maxn];
6
7  void init() {
8      cin >> n >> m;
9      fill(dp[0], dp[maxn-1]+(1<<maxn), -1);
10 }
11
12 void DP(int i, int msk) {
13     if (dp[i][msk] != -1) return;
14     dp[i][msk] = 0;
15     REP(j, n) if (j != i && (msk & (1<<j)) && adj[j][i
            ]) {
16         int sub = msk ^ (1<<i);
17         if (dp[j][sub] == -1) DP(j, sub);
18         dp[i][msk] += dp[j][sub] * adj[j][i];
19         if (dp[i][msk] >= MOD) dp[i][msk] %= MOD;
20     }
21 }
22
23
24 int main() {
25     WiwiHorz
26     init();
27
28     REP(i, m) {
29         int u, v;
30         cin >> u >> v;
31         if (u == v) continue;
32         adj[--u][--v]++;
33     }
34
35     dp[0][1] = 1;
36     FOR(i, 1, n, 1) {
37         dp[i][1] = 0;
38         dp[i][1|(1<<i)] = adj[0][i];
39     }
40     FOR(msk, 1, (1<<n), 1) {
41         if (msk == 1) continue;
42         dp[0][msk] = 0;
43     }
44
45
46     DP(n-1, (1<<n)-1);
47     cout << dp[n-1][(1<<n)-1] << endl;
48
49     return 0;
50 }
```

6.9   Kth Shortest Path

```
1  // time: O(|E| \lg |E|+|V| \lg |V|+K)
2  // memory: O(|E| \lg |E|+|V|)
3  struct KSP{ // 1-base
```

```
4   struct nd{
5     int u,v; ll d;
6     nd(int ui=0,int vi=0,ll di=INF){ u=ui; v=vi; d=di;
              }
7   };
8   struct heap{ nd* edge; int dep; heap* chd[4]; };
9   static int cmp(heap* a,heap* b)
10  { return a->edge->d > b->edge->d; }
11  struct node{
12    int v; ll d; heap* H; nd* E;
13    node(){}
14    node(ll _d,int _v,nd* _E){ d =_d; v=_v; E=_E; }
15    node(heap* _H,ll _d){ H=_H; d=_d; }
16    friend bool operator<(node a,node b)
17    { return a.d>b.d; }
18  };
19  int n,k,s,t,dst[N]; nd *nxt[N];
20  vector<nd*> g[N],rg[N]; heap *nullNd,*head[N];
21  void init(int _n,int _k,int _s,int _t){
22    n=_n; k=_k; s=_s; t=_t;
23    for(int i=1;i<=n;i++){
24      g[i].clear(); rg[i].clear();
25      nxt[i]=NULL; head[i]=NULL; dst[i]=-1;
26    }
27  }
28  void addEdge(int ui,int vi,ll di){
29    nd* e=new nd(ui,vi,di);
30    g[ui].push_back(e); rg[vi].push_back(e);
31  }
32  queue<int> dfsQ;
33  void dijkstra(){
34    while(dfsQ.size()) dfsQ.pop();
35    priority_queue<node> Q; Q.push(node(0,t,NULL));
36    while (!Q.empty()){
37      node p=Q.top(); Q.pop(); if(dst[p.v]!=-1)continue
             ;
38      dst[p.v]=p.d; nxt[p.v]=p.E; dfsQ.push(p.v);
39      for(auto e:rg[p.v]) Q.push(node(p.d+e->d,e->u,e))
             ;
40    }
41  }
42  heap* merge(heap* curNd,heap* newNd){
43    if(curNd==nullNd) return newNd;
44    heap* root=new heap;memcpy(root,curNd,sizeof(heap))
             ;
45    if(newNd->edge->d<curNd->edge->d){
46      root->edge=newNd->edge;
47      root->chd[2]=newNd->chd[2];
48      root->chd[3]=newNd->chd[3];
49      newNd->edge=curNd->edge;
50      newNd->chd[2]=curNd->chd[2];
51      newNd->chd[3]=curNd->chd[3];
52    }
53    if(root->chd[0]->dep<root->chd[1]->dep)
54      root->chd[0]=merge(root->chd[0],newNd);
55    else root->chd[1]=merge(root->chd[1],newNd);
56    root->dep=max(root->chd[0]->dep,
57              root->chd[1]->dep)+1;
58    return root;
59  }
60  vector<heap*> V;
61  void build(){
62    nullNd=new heap; nullNd->dep=0; nullNd->edge=new nd
             ;
63    fill(nullNd->chd,nullNd->chd+4,nullNd);
64    while(not dfsQ.empty()){
65      int u=dfsQ.front(); dfsQ.pop();
66      if(!nxt[u]) head[u]=nullNd;
67      else head[u]=head[nxt[u]->v];
68      V.clear();
69      for(auto&& e:g[u]){
70        int v=e->v;
71        if(dst[v]==-1) continue;
72        e->d+=dst[v]-dst[u];
73        if(nxt[u]!=e){
74          heap* p=new heap;fill(p->chd,p->chd+4,nullNd)
                   ;
75          p->dep=1; p->edge=e; V.push_back(p);
76        }
77      }
78      if(V.empty()) continue;
79      make_heap(V.begin(),V.end(),cmp);
```

```
80  #define L(X) ((X<<1)+1)
81  #define R(X) ((X<<1)+2)
82      for(size_t i=0;i<V.size();i++){
83        if(L(i)<V.size()) V[i]->chd[2]=V[L(i)];
84        else V[i]->chd[2]=nullNd;
85        if(R(i)<V.size()) V[i]->chd[3]=V[R(i)];
86        else V[i]->chd[3]=nullNd;
87      }
88      head[u]=merge(head[u],V.front());
89    }
90  }
91  vector<ll> ans;
92  void first_K(){
93    ans.clear(); priority_queue<node> Q;
94    if(dst[s]==-1) return;
95    ans.push_back(dst[s]);
96    if(head[s]!=nullNd)
97      Q.push(node(head[s],dst[s]+head[s]->edge->d));
98    for(int _=1;_<k and not Q.empty();_++){
99      node p=Q.top(),q; Q.pop(); ans.push_back(p.d);
100     if(head[p.H->edge->v]!=nullNd){
101       q.H=head[p.H->edge->v]; q.d=p.d+q.H->edge->d;
102       Q.push(q);
103     }
104     for(int i=0;i<4;i++)
105       if(p.H->chd[i]!=nullNd){
106         q.H=p.H->chd[i];
107         q.d=p.d-p.H->edge->d+p.H->chd[i]->edge->d;
108         Q.push(q);
109  } }    }
110   void solve(){ // ans[i] stores the i-th shortest path
111     dijkstra(); build();
112     first_K(); // ans.size() might less than k
113   }
114 } solver;
```

6.10   System of Difference Constraints

```
1  vector<vector<pair<int, ll>>> G;
2  void add(int u, int v, ll w) {
3      G[u].emplace_back(make_pair(v, w));
4  }
```

- $x_u - x_v \le c \Rightarrow$ add(v, u, c)

- $x_u - x_v \ge c \Rightarrow$ add(u, v, -c)

- $x_u - x_v = c \Rightarrow$ add(v, u, c), add(u, v -c)

- $x_u \ge c \Rightarrow$ add super vertex $x_0 = 0$, then $x_u - x_0 \ge c \Rightarrow$ add(u, 0, -c)

- Don't for get non-negative constraints for every variable if specified implicitly.

- Interval sum $\Rightarrow$ Use prefix sum to transform into differential constraints. Don't for get $S_{i+1} - S_i \ge 0$ if x_i needs to be non-negative.

- $\frac{x_u}{x_v} \le c \Rightarrow \log x_u - \log x_v \le \log c$

7   String

7.1   Rolling Hash

```
1  const ll C = 27;
2  inline int id(char c) {return c-'a'+1;}
3  struct RollingHash {
4      string s; int n; ll mod;
5      vector<ll> Cexp, hs;
6      RollingHash(string& _s, ll _mod):
7          s(_s), n((int)_s.size()), mod(_mod)
8      {
9          Cexp.assign(n, 0);
10         hs.assign(n, 0);
11         Cexp[0] = 1;
12         for (int i = 1; i < n; i++) {
```

```
13        Cexp[i] = Cexp[i-1] * C;
14        if (Cexp[i] >= mod) Cexp[i] %= mod;
15      }
16      hs[0] = id(s[0]);
17      for (int i = 1; i < n; i++) {
18        hs[i] = hs[i-1] * C + id(s[i]);
19        if (hs[i] >= mod) hs[i] %= mod;
20    } }
21    inline ll query(int l, int r) {
22      ll res = hs[r] - (l ? hs[l-1] * Cexp[r-l+1] :
          0);
23      res = (res % mod + mod) % mod;
24      return res; }
25 };
```

7.2 Trie

```
1 struct node {
2     int c[26]; ll cnt;
3     node(): cnt(0) {memset(c, 0, sizeof(c));}
4     node(ll x): cnt(x) {memset(c, 0, sizeof(c));}
5 };
6 struct Trie {
7     vector<node> t;
8     void init() {
9         t.clear();
10        t.emplace_back(node());
11    }
12    void insert(string s) { int ptr = 0;
13        for (auto& i : s) {
14            if (!t[ptr].c[i-'a']) {
15                t.emplace_back(node());
16                t[ptr].c[i-'a'] = (int)t.size()-1; }
17            ptr = t[ptr].c[i-'a']; }
18        t[ptr].cnt++; }
19 } trie;
```

7.3 KMP

```
1 int n, m;
2 string s, p;
3 vector<int> f;
4 void build() {
5     f.clear(); f.resize(m, 0);
6     int ptr = 0; for (int i = 1; i < m; i++) {
7         while (ptr && p[i] != p[ptr]) ptr = f[ptr-1];
8         if (p[i] == p[ptr]) ptr++;
9         f[i] = ptr;
10 }}
11 void init() {
12    cin >> s >> p;
13    n = (int)s.size();
14    m = (int)p.size();
15    build(); }
16 void solve() {
17    int ans = 0, pi = 0;
18    for (int si = 0; si < n; si++) {
19        while (pi && s[si] != p[pi]) pi = f[pi-1];
20        if (s[si] == p[pi]) pi++;
21        if (pi == m) ans++, pi = f[pi-1];
22    }
23 cout << ans << endl; }
```

7.4 Z Value

```
1 string is, it, s;
2 int n; vector<int> z;
3 void init() {
4     cin >> is >> it;
5     s = it+'0'+is;
6     n = (int)s.size();
7     z.resize(n, 0); }
8 void solve() {
9     int ans = 0; z[0] = n;
10    for (int i = 1, l = 0, r = 0; i < n; i++) {
11        if (i <= r) z[i] = min(z[i-l], r-i+1);
12        while (i+z[i] < n && s[z[i]] == s[i+z[i]]) z[i
            ]++;
13        if (i+z[i]-1 > r) l = i, r = i+z[i]-1;
```

```
14        if (z[i] == (int)it.size()) ans++;
15    }
16    cout << ans << endl; }
```

7.5 Manacher

```
1 int n; string S, s;
2 vector<int> m;
3 void manacher() {
4 s.clear(); s.resize(2*n+1, '.');
5 for (int i = 0, j = 1; i < n; i++, j += 2) s[j] = S[i];
6 m.clear(); m.resize(2*n+1, 0);
7 // m[i] := max k such that s[i-k, i+k] is palindrome
8 int mx = 0, mxk = 0;
9 for (int i = 1; i < 2*n+1; i++) {
10    if (mx-(i-mx) >= 0) m[i] = min(m[mx-(i-mx)], mx+mxk
        -i);
11    while (0 <= i-m[i]-1 && i+m[i]+1 < 2*n+1 &&
12        s[i-m[i]-1] == s[i+m[i]+1]) m[i]++;
13    if (i+m[i] > mx+mxk) mx = i, mxk = m[i];
14 } }
15 void init() { cin >> S; n = (int)S.size(); }
16 void solve() {
17    manacher();
18    int mx = 0, ptr = 0;
19    for (int i = 0; i < 2*n+1; i++) if (mx < m[i])
20        { mx = m[i]; ptr = i; }
21    for (int i = ptr-mx; i <= ptr+mx; i++)
22        if (s[i] != '.') cout << s[i];
23 cout << endl; }
```

7.6 Suffix Array

```
1 #define F first
2 #define S second
3 struct SuffixArray {  // don't forget s += "$";
4     int n; string s;
5     vector<int> suf, lcp, rk;
6     vector<int> cnt, pos;
7     vector<pair<pii, int> > buc[2];
8     void init(string _s) {
9         s = _s; n = (int)s.size();
10 // resize(n): suf, rk, cnt, pos, lcp, buc[0~1]
11    }
12    void radix_sort() {
13        for (int t : {0, 1}) {
14            fill(cnt.begin(), cnt.end(), 0);
15            for (auto& i : buc[t]) cnt[ (t ? i.F.F : i.
                F.S) ]++;
16            for (int i = 0; i < n; i++)
17                pos[i] = (!i ? 0 : pos[i-1] + cnt[i-1])
                    ;
18            for (auto& i : buc[t])
19                buc[t^1][pos[ (t ? i.F.F : i.F.S) ]++]
                    = i;
20    }}
21    bool fill_suf() {
22        bool end = true;
23        for (int i = 0; i < n; i++) suf[i] = buc[0][i].
            S;
24        rk[suf[0]] = 0;
25        for (int i = 1; i < n; i++) {
26            int dif = (buc[0][i].F != buc[0][i-1].F);
27            end &= dif;
28            rk[suf[i]] = rk[suf[i-1]] + dif;
29        } return end;
30    }
31    void sa() {
32        for (int i = 0; i < n; i++)
33            buc[0][i] = make_pair(make_pair(s[i], s[i])
                , i);
34        sort(buc[0].begin(), buc[0].end());
35        if (fill_suf()) return;
36        for (int k = 0; (1<<k) < n; k++) {
37            for (int i = 0; i < n; i++)
38                buc[0][i] = make_pair(make_pair(rk[i],
                    rk[(i + (1<<k)) % n]), i);
39            radix_sort();
40            if (fill_suf()) return;
41    }}
```

```
42    void LCP() {   int k = 0;
43        for (int i = 0; i < n-1; i++) {
44            if (rk[i] == 0) continue;
45            int pi = rk[i];
46            int j = suf[pi-1];
47            while (i+k < n && j+k < n && s[i+k] == s[j+
                  k]) k++;
48            lcp[pi] = k;
49            k = max(k-1, 0);
50        }}
51 };
52 SuffixArray suffixarray;
```

7.7 SA-IS

```
1  const int N=300010;
2  struct SA{
3  #define REP(i,n) for(int i=0;i<int(n);i++)
4  #define REP1(i,a,b) for(int i=(a);i<=int(b);i++)
5    bool _t[N*2]; int _s[N*2],_sa[N*2];
6    int _c[N*2],x[N],_p[N],_q[N*2],hei[N],r[N];
7    int operator [](int i){ return _sa[i]; }
8    void build(int *s,int n,int m){
9      memcpy(_s,s,sizeof(int)*n);
10     sais(_s,_sa,_p,_q,_t,_c,n,m); mkhei(n);
11   }
12   void mkhei(int n){
13     REP(i,n) r[_sa[i]]=i;
14     hei[0]=0;
15     REP(i,n) if(r[i]) {
16       int ans=i>0?max(hei[r[i-1]]-1,0):0;
17       while(_s[i+ans]==_s[_sa[r[i]-1]+ans]) ans++;
18       hei[r[i]]=ans;
19     }
20   }
21   void sais(int *s,int *sa,int *p,int *q,bool *t,int *c
          ,int n,int z){
22     bool uniq=t[n-1]=true,neq;
23     int nn=0,nmxz=-1,*nsa=sa+n,*ns=s+n,lst=-1;
24 #define MS0(x,n) memset((x),0,n*sizeof(*(x)))
25 #define MAGIC(XD) MS0(sa,n);\
26 memcpy(x,c,sizeof(int)*z); XD;\
27 memcpy(x+1,c,sizeof(int)*(z-1));\
28 REP(i,n) if(sa[i]&&!t[sa[i]-1]) sa[x[s[sa[i]-1]]++]=sa[
          i]-1;\
29 memcpy(x,c,sizeof(int)*z);\
30 for(int i=n-1;i>=0;i--) if(sa[i]&&t[sa[i]-1]) sa[--x[s[
          sa[i]-1]]]=sa[i]-1;
31     MS0(c,z); REP(i,n) uniq&=++c[s[i]]<2;
32     REP(i,z-1) c[i+1]+=c[i];
33     if(uniq) { REP(i,n) sa[--c[s[i]]]=i; return; }
34     for(int i=n-2;i>=0;i--)
35       t[i]=(s[i]==s[i+1]?t[i+1]:s[i]<s[i+1]);
36     MAGIC(REP1(i,1,n-1) if(t[i]&&!t[i-1]) sa[--x[s[i
          ]]]=p[q[i]=nn++]=i);
37     REP(i,n) if(sa[i]&&t[sa[i]]&&!t[sa[i]-1]){
38       neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
39       ns[q[lst=sa[i]]]=nmxz+=neq;
40     }
41     sais(ns,nsa,p+nn,q+n,t+n,c+z,nn,nmxz+1);
42     MAGIC(for(int i=nn-1;i>=0;i--) sa[--x[s[p[nsa[i
          ]]]]]=p[nsa[i]]);
43   }
44 }sa;
45 int H[N],SA[N],RA[N];
46 void suffix_array(int* ip,int len){
47   // should padding a zero in the back
48   // ip is int array, len is array length
49   // ip[0..n-1] != 0, and ip[len]=0
50   ip[len++]=0; sa.build(ip,len,128);
51   memcpy(H,sa.hei+1,len<<2); memcpy(SA,sa._sa+1,len<<2)
          ;
52   for(int i=0;i<len;i++) RA[i]=sa.r[i]-1;
53   // resulting height, sa array \in [0,len)
54 }
```

7.8 Minimum Rotation

```
1 //rotate(begin(s), begin(s)+minRotation(s), end(s))
```

```
2  int minRotation(string s) {
3  int a = 0, n = s.size(); s += s;
4  for(int b = 0; b < n; b++) for(int k = 0; k < n; k++) {
5      if(a + k == b ||| s[a + k] < s[b + k]) {
6          b += max(0, k - 1);
7          break; }
8      if(s[a + k] > s[b + k]) {
9          a = b;
10         break;
11     } }
12 return a; }
```

7.9 Aho Corasick

```
1  struct ACautomata{
2    struct Node{
3      int cnt;
4      Node *go[26], *fail, *dic;
5      Node (){
6        cnt = 0; fail = 0; dic=0;
7        memset(go,0,sizeof(go));
8      }
9    }pool[1048576],*root;
10   int nMem;
11   Node* new_Node(){
12     pool[nMem] = Node();
13     return &pool[nMem++];
14   }
15   void init() { nMem = 0; root = new_Node(); }
16   void add(const string &str) { insert(root,str,0); }
17   void insert(Node *cur, const string &str, int pos){
18     for(int i=pos;i<str.size();i++){
19       if(!cur->go[str[i]-'a'])
20         cur->go[str[i]-'a'] = new_Node();
21       cur=cur->go[str[i]-'a'];
22     }
23     cur->cnt++;
24   }
25   void make_fail(){
26     queue<Node*> que;
27     que.push(root);
28     while (!que.empty()){
29       Node* fr=que.front(); que.pop();
30       for (int i=0; i<26; i++){
31         if (fr->go[i]){
32           Node *ptr = fr->fail;
33           while (ptr && !ptr->go[i]) ptr = ptr->fail;
34           fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
35           fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
36           que.push(fr->go[i]);
37   } } } }
38 }AC;
```

8 Geometry

8.1 Basic Operations

```
1  typedef long long T;
2  // typedef long double T;
3  const long double eps = 1e-8;
4
5  short sgn(T x) {
6      if (abs(x) < eps) return 0;
7      return x < 0 ? -1 : 1;
8  }
9
10 struct Pt {
11 T x, y;
12 Pt(T _x=0, T _y=0):x(_x), y(_y) {}
13 Pt operator+(Pt a) { return Pt(x+a.x, y+a.y); }
14 Pt operator-(Pt a) { return Pt(x-a.x, y-a.y); }
15 Pt operator*(T a)  { return Pt(x*a, y*a); }
16 Pt operator/(T a)  { return Pt(x/a, y/a); }
17 T operator*(Pt a)  { return x*a.x + y*a.y; }
18 T operator^(Pt a)  { return x*a.y - y*a.x; }
19 bool operator<(Pt a)
20     { return x < a.x || (x == a.x && y < a.y); }
21 //return sgn(x-a.x) < 0 || (sgn(x-a.x) == 0 && sgn(y-a.
          y) < 0); }
```

```cpp
bool operator==(Pt a)
    { return sgn(x-a.x) == 0 && sgn(y-a.y) == 0; }
};

Pt mv(Pt a, Pt b) { return b-a; }
T len2(Pt a) { return a*a; }
T dis2(Pt a, Pt b) { return len2(b-a); }

short ori(Pt a, Pt b) { return ((a^b)>0) - ((a^b)<0); }
bool onseg(Pt p, Pt l1, Pt l2) {
    Pt a = mv(p, l1), b = mv(p, l2);
    return ((a^b) == 0) && ((a*b) <= 0);
}
```

8.2 InPoly

```cpp
short inPoly(Pt p) {
// 0=Bound 1=In -1=Out
REP(i, n) if (onseg(p, E[i], E[(i+1)%n])) return 0;
int cnt = 0;
REP(i, n) if (banana(p, Pt(p.x+1, p.y+2e9),
                 E[i], E[(i+1)%n])) cnt ^= 1;
return (cnt ? 1 : -1);
}
```

8.3 Sort by Angle

```cpp
int ud(Pt a) {  // up or down half plane
    if (a.y > 0) return 0;
    if (a.y < 0) return 1;
    return (a.x >= 0 ? 0 : 1);
}
sort(ALL(E), [&](const Pt& a, const Pt& b){
    if (ud(a) != ud(b)) return ud(a) < ud(b);
    return (a^b) > 0;
});
```

8.4 Line Intersect Check

```cpp
inline bool banana(Pt p1, Pt p2, Pt q1, Pt q2) {
if (onseg(p1, q1, q2) || onseg(p2, q1, q2) ||
    onseg(q1, p1, p2) || onseg(q2, p1, p2)) {
    return true;
}
Pt p = mv(p1, p2), q = mv(q1, q2);
return (ori(p, mv(p1, q1)) * ori(p, mv(p1, q2)) < 0 &&
        ori(q, mv(q1, p1)) * ori(q, mv(q1, p2)) < 0);
}
```

8.5 Line Intersection

```cpp
// T: long double
Pt bananaPoint(Pt p1, Pt p2, Pt q1, Pt q2) {
if (onseg(q1, p1, p2)) return q1;
if (onseg(q2, p1, p2)) return q2;
if (onseg(p1, q1, q2)) return p1;
if (onseg(p2, q1, q2)) return p2;
double s = abs(mv(p1, p2) ^ mv(p1, q1));
double t = abs(mv(p1, p2) ^ mv(p1, q2));
return q2 * (s/(s+t)) + q1 * (t/(s+t));
}
```

8.6 Convex Hull

```cpp
vector<Pt> hull;
void convexHull() {
hull.clear(); sort(ALL(E));
REP(t, 2) {
    int b = SZ(hull);
    Each(ei, E) {
        while (SZ(hull) - b >= 2 &&
                ori(mv(hull[SZ(hull)-2], hull.back()),
                    mv(hull[SZ(hull)-2], ei)) == -1) {
            hull.pop_back();
        }
        hull.eb(ei);
    }
```

```cpp
    hull.pop_back();
    reverse(ALL(E));
} }
```

8.7 Lower Concave Hull

```cpp
struct Line {
  mutable ll m, b, p;
  bool operator<(const Line& o) const { return m < o.m;
      }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) { x->p = inf; return false; }
    if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;
    else x->p = div(y->b - x->b, x->m - y->m);
    return x->p >= y->p;
  }
  void add(ll m, ll b) {
    auto z = insert({m, b, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y =
        erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.m * x + l.b;
  }
};
```

8.8 Polygon Area

```cpp
T dbarea(vector<Pt>& e) {
ll res = 0;
REP(i, SZ(e)) res += e[i]^e[(i+1)%SZ(e)];
return abs(res);
}
```

8.9 Pick's Theorem

Consider a polygon which vertices are all lattice points.
Let i = number of points inside the polygon.
Let b = number of points on the boundary of the polygon.

Then we have the following formula:

$$Area = i + \frac{b}{2} - 1$$

8.10 Minimum Enclosing Circle

```cpp
Pt circumcenter(Pt A, Pt B, Pt C) {
// a1(x-A.x) + b1(y-A.y) = c1
// a2(x-A.x) + b2(y-A.y) = c2
// solve using Cramer's rule
T a1 = B.x-A.x, b1 = B.y-A.y, c1 = dis2(A, B)/2.0;
T a2 = C.x-A.x, b2 = C.y-A.y, c2 = dis2(A, C)/2.0;
T D  = Pt(a1, b1) ^ Pt(a2, b2);
T Dx = Pt(c1, b1) ^ Pt(c2, b2);
T Dy = Pt(a1, c1) ^ Pt(a2, c2);
if (D == 0) return Pt(-INF, -INF);
return A + Pt(Dx/D, Dy/D);
}
Pt center; T r2;
void minEncloseCircle() {
mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());
shuffle(ALL(E), gen);
center = E[0], r2 = 0;
```

```
18
19 for (int i = 0; i < n; i++) {
20     if (dis2(center, E[i]) <= r2) continue;
21     center = E[i], r2 = 0;
22     for (int j = 0; j < i; j++) {
23         if (dis2(center, E[j]) <= r2) continue;
24         center = (E[i] + E[j]) / 2.0;
25         r2 = dis2(center, E[i]);
26         for (int k = 0; k < j; k++) {
27             if (dis2(center, E[k]) <= r2) continue;
28             center = circumcenter(E[i], E[j], E[k]);
29             r2 = dis2(center, E[i]);
30         }
31     }
32 } }
```

8.11   PolyUnion

```
1  struct PY{
2    int n; Pt pt[5]; double area;
3    Pt& operator[](const int x){ return pt[x]; }
4    void init(){ //n,pt[0~n-1] must be filled
5      area=pt[n-1]^pt[0];
6      for(int i=0;i<n-1;i++) area+=pt[i]^pt[i+1];
7      if((area/=2)<0)reverse(pt,pt+n),area=-area;
8    }
9  };
10 PY py[500]; pair<double,int> c[5000];
11 inline double segP(Pt &p,Pt &p1,Pt &p2){
12   if(dcmp(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
13   return (p.x-p1.x)/(p2.x-p1.x);
14 }
15 double polyUnion(int n){ //py[0~n-1] must be filled
16   int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td;
17   for(i=0;i<n;i++) py[i][py[i].n]=py[i][0];
18   for(i=0;i<n;i++){
19     for(ii=0;ii<py[i].n;ii++){
20       r=0;
21       c[r++]=make_pair(0.0,0); c[r++]=make_pair(1.0,0);
22       for(j=0;j<n;j++){
23         if(i==j) continue;
24         for(jj=0;jj<py[j].n;jj++){
25           ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]))
              ;
26           tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj
              +1]));
27           if(ta==0 && tb==0){
28             if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
                i][ii])>0&&j<i){
29               c[r++]=make_pair(segP(py[j][jj],py[i][ii
                  ],py[i][ii+1]),1);
30               c[r++]=make_pair(segP(py[j][jj+1],py[i][
                  ii],py[i][ii+1]),-1);
31             }
32           }else if(ta>=0 && tb<0){
33             tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
34             td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
35             c[r++]=make_pair(tc/(tc-td),1);
36           }else if(ta<0 && tb>=0){
37             tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
38             td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
39             c[r++]=make_pair(tc/(tc-td),-1);
40       } } }
41       sort(c,c+r);
42       z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
          =0;
43       for(j=1;j<r;j++){
44         w=min(max(c[j].first,0.0),1.0);
45         if(!d) s+=w-z;
46         d+=c[j].second; z=w;
47       }
48       sum+=(py[i][ii]^py[i][ii+1])*s;
49     }
50   }
51   return sum/2;
52 }
```

8.12   Minkowski Sum

```
1 /* convex hull Minkowski Sum*/
```

```
2  #define INF 10000000000000LL
3  int pos( const Pt& tp ){
4    if( tp.Y == 0 ) return tp.X > 0 ? 0 : 1;
5    return tp.Y > 0 ? 0 : 1;
6  }
7  #define N 300030
8  Pt pt[ N ], qt[ N ], rt[ N ];
9  LL Lx,Rx;
10 int dn,un;
11 inline bool cmp( Pt a, Pt b ){
12     int pa=pos( a ),pb=pos( b );
13     if(pa==pb) return (a^b)>0;
14     return pa<pb;
15 }
16 int minkowskiSum(int n,int m){
17     int i,j,r,p,q,fi,fj;
18     for(i=1,p=0;i<n;i++){
19         if( pt[i].Y<pt[p].Y ||
20         (pt[i].Y==pt[p].Y && pt[i].X<pt[p].X) ) p=i; }
21     for(i=1,q=0;i<m;i++){
22         if( qt[i].Y<qt[q].Y ||
23         (qt[i].Y==qt[q].Y && qt[i].X<qt[q].X) ) q=i; }
24     rt[0]=pt[p]+qt[q];
25     r=1; i=p; j=q; fi=fj=0;
26     while(1){
27         if((fj&&j==q) ||
28         ( (!fi||i!=p) &&
29           cmp(pt[(p+1)%n]-pt[p],qt[(q+1)%m]-qt[q]) ) ){
30             rt[r]=rt[r-1]+pt[(p+1)%n]-pt[p];
31             p=(p+1)%n;
32             fi=1;
33         }else{
34             rt[r]=rt[r-1]+qt[(q+1)%m]-qt[q];
35             q=(q+1)%m;
36             fj=1;
37         }
38         if(r<=1 || ((rt[r]-rt[r-1])^(rt[r-1]-rt[r-2]))
              !=0) r++;
39         else rt[r-1]=rt[r];
40         if(i==p && j==q) break;
41     }
42     return r-1;
43 }
44 void initInConvex(int n){
45     int i,p,q;
46     LL Ly,Ry;
47     Lx=INF; Rx=-INF;
48     for(i=0;i<n;i++){
49         if(pt[i].X<Lx) Lx=pt[i].X;
50         if(pt[i].X>Rx) Rx=pt[i].X;
51     }
52     Ly=Ry=INF;
53     for(i=0;i<n;i++){
54         if(pt[i].X==Lx && pt[i].Y<Ly){ Ly=pt[i].Y; p=i;
                }
55         if(pt[i].X==Rx && pt[i].Y<Ry){ Ry=pt[i].Y; q=i;
                }
56     }
57     for(dn=0,i=p;i!=q;i=(i+1)%n){ qt[dn++]=pt[i]; }
58     qt[dn]=pt[q]; Ly=Ry=-INF;
59     for(i=0;i<n;i++){
60         if(pt[i].X==Lx && pt[i].Y>Ly){ Ly=pt[i].Y; p=i;
                }
61         if(pt[i].X==Rx && pt[i].Y>Ry){ Ry=pt[i].Y; q=i;
                }
62     }
63     for(un=0,i=p;i!=q;i=(i+n-1)%n){ rt[un++]=pt[i]; }
64     rt[un]=pt[q];
65 }
66 inline int inConvex(Pt p){
67     int L,R,M;
68     if(p.X<Lx || p.X>Rx) return 0;
69     L=0;R=dn;
70     while(L<R-1){ M=(L+R)/2;
71         if(p.X<qt[M].X) R=M; else L=M; }
72     if(tri(qt[L],qt[R],p)<0) return 0;
73     L=0;R=un;
74     while(L<R-1){ M=(L+R)/2;
75         if(p.X<rt[M].X) R=M; else L=M; }
76     if(tri(rt[L],rt[R],p)>0) return 0;
77     return 1;
78 }
```

```
79  int main(){
80      int n,m,i;
81      Pt p;
82      scanf("%d",&n);
83      for(i=0;i<n;i++) scanf("%lld%lld",&pt[i].X,&pt[i].Y
            );
84      scanf("%d",&m);
85      for(i=0;i<m;i++) scanf("%lld%lld",&qt[i].X,&qt[i].Y
            );
86      n=minkowskiSum(n,m);
87      for(i=0;i<n;i++) pt[i]=rt[i];
88      scanf("%d",&m);
89      for(i=0;i<m;i++) scanf("%lld%lld",&qt[i].X,&qt[i].Y
            );
90      n=minkowskiSum(n,m);
91      for(i=0;i<n;i++) pt[i]=rt[i];
92      initInConvex(n);
93      scanf("%d",&m);
94      for(i=0;i<m;i++){
95          scanf("%lld %lld",&p.X,&p.Y);
96          p.X*=3; p.Y*=3;
97          puts(inConvex(p)?"YES":"NO");
98      }
99  }
```

9 Number Theory

9.1 FFT

```
1   typedef complex<double> cp;
2
3   const double pi = acos(-1);
4   const int NN = 131072;
5
6   struct FastFourierTransform{
7       /*
8           Iterative Fast Fourier Transform
9           How this works? Look at this
10          0th recursion 0(000)   1(001)   2(010)   3(011)
                4(100)   5(101)   6(110)   7(111)
11          1th recursion 0(000)   2(010)   4(100)   6(110)
                | 1(011)   3(011)   5(101)   7(111)
12          2th recursion 0(000)   4(100) | 2(010)   6(110)
                | 1(011)   5(101) | 3(011)   7(111)
13          3th recursion 0(000) | 4(100) | 2(010) | 6(110)
                | 1(011) | 5(101) | 3(011) | 7(111)
14          All the bits are reversed => We can save the
                reverse of the numbers in an array!
15      */
16      int n, rev[NN];
17      cp omega[NN], iomega[NN];
18      void init(int n_){
19          n = n_;
20          for(int i = 0;i < n_;i++){
21              //Calculate the nth roots of unity
22              omega[i] = cp(cos(2*pi*i/n_),sin(2*pi*i/n_)
                    );
23              iomega[i] = conj(omega[i]);
24          }
25          int k = __lg(n_);
26          for(int i = 0;i < n_;i++){
27              int t = 0;
28              for(int j = 0;j < k;j++){
29                  if(i & (1<<j)) t |= (1<<(k-j-1));
30              }
31              rev[i] = t;
32          }
33      }
34
35      void transform(vector<cp> &a, cp* xomega){
36          for(int i = 0;i < n;i++)
37              if(i < rev[i]) swap(a[i],a[rev[i]]);
38          for(int len = 2; len <= n; len <<= 1){
39              int mid = len >> 1;
40              int r = n/len;
41              for(int j = 0;j < n;j += len)
42                  for(int i = 0;i < mid;i++){
43                      cp tmp = xomega[r*i] * a[j+mid+i];
44                      a[j+mid+i] = a[j+i] - tmp;
45                      a[j+i] = a[j+i] + tmp;
46                  }
47              }
48          }
49
50      void fft(vector<cp> &a){ transform(a,omega); }
51      void ifft(vector<cp> &a){ transform(a,iomega); for(
            int i = 0;i < n;i++) a[i] /= n;}
52  } FFT;
53
54
55  const int MAXN = 262144;
56  // (must be 2^k)
57  // 262144, 524288, 1048576, 2097152, 4194304
58  // before any usage, run pre_fft() first
59  typedef long double ld;
60  typedef complex<ld> cplx; //real() ,imag()
61  const ld PI = acosl(-1);
62  const cplx I(0, 1);
63  cplx omega[MAXN+1];
64  void pre_fft(){
65      for(int i=0; i<=MAXN; i++) {
66          omega[i] = exp(i * 2 * PI / MAXN * I);
67      }
68  }
69  // n must be 2^k
70  void fft(int n, cplx a[], bool inv=false){
71      int basic = MAXN / n;
72      int theta = basic;
73      for (int m = n; m >= 2; m >>= 1) {
74          int mh = m >> 1;
75          for (int i = 0; i < mh; i++) {
76              cplx w = omega[inv ? MAXN - (i * theta %
                    MAXN) : i * theta % MAXN];
77              for (int j = i; j < n; j += m) {
78                  int k = j + mh;
79                  cplx x = a[j] - a[k];
80                  a[j] += a[k];
81                  a[k] = w * x;
82              }
83          }
84          theta = (theta * 2) % MAXN;
85      }
86      int i = 0;
87      for (int j = 1; j < n - 1; j++) {
88          for (int k = n >> 1; k > (i ^= k); k >>= 1);
89          if (j < i) swap(a[i], a[j]);
90      }
91      if(inv) {
92          for (i = 0; i < n; i++) a[i] /= n;
93      }
94  }
95  cplx arr[MAXN + 1];
96  inline void mul(int _n,long long a[],int _m,long long b
        [],long long ans[]){
97      int n=1, sum = _n + _m - 1;
98      while(n < sum) n <<= 1;
99      for(int i = 0; i < n; i++) {
100         double x= (i < _n ? a[i] : 0), y=(i < _m ? b[i]
                : 0);
101         arr[i] = complex<double>(x + y, x - y);
102     }
103     fft(n, arr);
104     for(int i = 0; i < n; i++) arr[i]=arr[i]*arr[i];
105     fft(n,arr,true);
106     for(int i=0;i<sum;i++) ans[i]=(long long int)(arr[i
            ].real() / 4 + 0.5);
107 }
108
109 long long a[MAXN];
110 long long b[MAXN];
111 long long ans[MAXN];
112 int a_length;
113 int b_length;
```

9.2 Pollard's rho

```
1   ll add(ll x, ll y, ll p) {
2       return (x + y) % p;
3   }
4   ll qMul(ll x,ll y,ll mod){
```

```
 5      ll ret = x * y - (ll)((long double)x / mod * y) *
            mod;
 6      return ret<0?ret+mod:ret;
 7  }
 8  ll f(ll x, ll mod) { return add(qMul(x,x,mod),1,mod); }
 9  ll pollard_rho(ll n) {
10      if(!(n & 1)) return 2;
11      while(true) {
12          ll y = 2, x = rand() % (n - 1) + 1, res = 1;
13          for(int sz = 2; res == 1; sz *= 2) {
14              for(int i = 0; i < sz && res <= 1; i++) {
15                  x = f(x, n);
16                  res = __gcd(llabs(x - y), n);
17              }
18              y = x;
19          }
20          if (res != 0 && res != n) return res;
21      }
22  }
23  vector<ll> ret;
24  void fact(ll x) {
25      if(miller_rabin(x)) {
26          ret.push_back(x);
27          return;
28      }
29      ll f = pollard_rho(x);
30      fact(f); fact(x / f);
31  }
```

9.3   Miller Rabin

```
 1  // n < 4,759,123,141        3 :   2, 7, 61
 2  // n < 1,122,004,669,633    4 :   2, 13, 23, 1662803
 3  // n < 3,474,749,660,383      6 :   pirmes <= 13
 4  // n < 2^64                    7 :
 5  // 2, 325, 9375, 28178, 450775, 9780504, 1795265022
 6  bool witness(ll a,ll n,ll u,int t){
 7      if(!(a%=n)) return 0;
 8      ll x=mypow(a,u,n);
 9      for(int i=0;i<t;i++) {
10          ll nx=mul(x,x,n);
11          if(nx==1&&x!=1&&x!=n-1) return 1;
12          x=nx;
13      }
14      return x!=1;
15  }
16  bool miller_rabin(ll n,int s=100) {
17      // iterate s times of witness on n
18      // return 1 if prime, 0 otherwise
19      if(n<2) return 0;
20      if(!(n&1)) return n == 2;
21      ll u=n-1; int t=0;
22      while(!(u&1)) u>>=1, t++;
23      while(s--){
24          ll a=randll()%(n-1)+1;
25          if(witness(a,n,u,t)) return 0;
26      }
27      return 1;
28  }
```

9.4   Fast Power

Note: $a^n \equiv a^{(n \bmod (p-1))} (mod \ p)$

9.5   Extend GCD

```
 1  ll GCD;
 2  pll extgcd(ll a, ll b) {
 3      if (b == 0) {
 4          GCD = a;
 5          return pll{1, 0};
 6      }
 7      pll ans = extgcd(b, a % b);
 8      return pll{ans.S, ans.F - a/b * ans.S};
 9  }
10  pll bezout(ll a, ll b, ll c) {
11      bool negx = (a < 0), negy = (b < 0);
12      pll ans = extgcd(abs(a), abs(b));
13      if (c % GCD != 0) return pll{-LLINF, -LLINF};
14      return pll{ans.F * c/GCD * (negx ? -1 : 1),
```

```
15          ans.S * c/GCD * (negy ? -1 : 1)};
16  }
17  ll inv(ll a, ll p) {
18      if (p == 1) return -1;
19      pll ans = bezout(a % p, -p, 1);
20      if (ans == pll{-LLINF, -LLINF}) return -1;
21      return (ans.F % p + p) % p;
22  }
```

9.6   Mu + Phi

```
 1  const int maxn = 1e6 + 5;
 2  ll f[maxn];
 3  vector<int> lpf, prime;
 4  void build() {
 5  lpf.clear(); lpf.resize(maxn, 1);
 6  prime.clear();
 7  f[1] = ...;  /* mu[1] = 1, phi[1] = 1 */
 8  for (int i = 2; i < maxn; i++) {
 9      if (lpf[i] == 1) {
10          lpf[i] = i; prime.emplace_back(i);
11          f[i] = ...;  /* mu[i] = 1, phi[i] = i-1 */
12      }
13      for (auto& j : prime) {
14          if (i*j >= maxn) break;
15          lpf[i*j] = j;
16          if (i % j == 0) f[i*j] = ...;  /* 0, phi[i]*j
                */
17          else f[i*j] = ...;  /* -mu[i], phi[i]*phi[j] */
18          if (j >= lpf[i]) break;
19  } } }
```

9.7   Other Formulas

- Inversion:
  $aa^{-1} \equiv 1 \pmod{m}$. a^{-1} exists iff $\gcd(a,m) = 1$.

- Linear inversion:
  $a^{-1} \equiv (m - \lfloor \frac{m}{a} \rfloor) \times (m \bmod a)^{-1} \pmod{m}$

- Fermat's little theorem:
  $a^p \equiv a \pmod{p}$ if p is prime.

- Euler function:
  $\phi(n) = n \prod_{p|n} \frac{p-1}{p}$

- Euler theorem:
  $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a,n) = 1$.

- Extended Euclidean algorithm:
  $ax + by = \gcd(a,b) = \gcd(b, a \bmod b) = \gcd(b, a - \lfloor \frac{a}{b} \rfloor b) = bx_1 + (a - \lfloor \frac{a}{b} \rfloor b)y_1 = ay_1 + b(x_1 - \lfloor \frac{a}{b} \rfloor y_1)$

- Divisor function:
  $\sigma_x(n) = \sum_{d|n} d^x$. $n = \prod_{i=1}^{r} p_i^{a_i}$.
  $\sigma_x(n) = \prod_{i=1}^{r} \frac{p_i^{(a_i+1)x}-1}{p_i^x-1}$ if $x \neq 0$. $\sigma_0(n) = \prod_{i=1}^{r}(a_i + 1)$.

- Chinese remainder theorem (Coprime Moduli):
  $x \equiv a_i \pmod{m_i}$.
  $M = \prod m_i$. $M_i = M/m_i$. $t_i = M_i^{-1}$.
  $x = kM + \sum a_i t_i M_i, k \in \mathbb{Z}$.

- Chinese remainder theorem:
  $x \equiv a_1 \pmod{m_1}, x \equiv a_2 \pmod{m_2} \Rightarrow x = m_1 p + a_1 = m_2 q + a_2 \Rightarrow m_1 p - m_2 q = a_2 - a_1$
  Solve for (p, q) using ExtGCD.
  $x \equiv m_1 p + a_1 \equiv m_2 q + a_2 \pmod{lcm(m_1,m_2)}$

- Avoiding Overflow: $ca \bmod cb = c(a \bmod b)$

- Dirichlet Convolution: $(f * g)(n) = \sum_{d|n} f(n)g(n/d)$

- Important Multiplicative Functions + Proterties:
  1. $\epsilon(n) = [n = 1]$

2. $1(n) = 1$
3. $id(n) = n$
4. $\mu(n) = 0$ if n has squared prime factor
5. $\mu(n) = (-1)^k$ if $n = p_1 p_2 \cdots p_k$
6. $\epsilon = \mu * 1$
7. $\phi = \mu * id$
8. $[n = 1] = \sum_{d|n} \mu(d)$
9. $[gcd = 1] = \sum_{d|gcd} \mu(d)$

- Möbius inversion: $f = g * 1 \Leftrightarrow g = f * \mu$

9.8 Polynomial

```cpp
const int maxk = 20;
const int maxn = 1<<maxk;
const ll LINF = 1e18;

/* P = r*2^k + 1
P                    r    k    g
998244353            119  23   3
1004535809           479  21   3

P                    r    k    g
3                    1    1    2
5                    1    2    2
17                   1    4    3
97                   3    5    5
193                  3    6    5
257                  1    8    3
7681                 15   9    17
12289                3    12   11
40961                5    13   3
65537                1    16   3
786433               3    18   10
5767169              11   19   3
7340033              7    20   3
23068673             11   21   3
104857601            25   22   3
167772161            5    25   3
469762049            7    26   3
1004535809           479  21   3
2013265921           15   27   31
2281701377           17   27   3
3221225473           3    30   5
75161927681          35   31   3
77309411329          9    33   7
206158430209         3    36   22
2061584302081        15   37   7
2748779069441        5    39   3
6597069766657        3    41   5
39582418599937       9    42   5
79164837199873       9    43   5
263882790666241      15   44   7
1231453023109121     35   45   3
1337006139375617     19   46   3
3799912185593857     27   47   5
4222124650659841     15   48   19
7881299347898369     7    50   6
31525197391593473    7    52   3
180143985094819841   5    55   6
1945555039024054273  27   56   5
4179340454199820289  29   57   3
9097271247288401921  505  54   6 */

const int g = 3;
const ll MOD = 998244353;

ll pw(ll a, ll n) { /* fast pow */ }

#define siz(x) (int)x.size()

template<typename T>
vector<T>& operator+=(vector<T>& a, const vector<T>& b)
    {
    if (siz(a) < siz(b)) a.resize(siz(b));
    for (int i = 0; i < min(siz(a), siz(b)); i++) {
        a[i] += b[i];
        a[i] -= a[i] >= MOD ? MOD : 0;
    }
```

```cpp
    return a;
}

template<typename T>
vector<T>& operator-=(vector<T>& a, const vector<T>& b)
    {
    if (siz(a) < siz(b)) a.resize(siz(b));
    for (int i = 0; i < min(siz(a), siz(b)); i++) {
        a[i] -= b[i];
        a[i] += a[i] < 0 ? MOD : 0;
    }
    return a;
}

template<typename T>
vector<T> operator-(const vector<T>& a) {
    vector<T> ret(siz(a));
    for (int i = 0; i < siz(a); i++) {
        ret[i] = -a[i] < 0 ? -a[i] + MOD : -a[i];
    }
    return ret;
}

vector<ll> X, iX;
vector<int> rev;

void init_ntt() {
    X.clear(); X.resize(maxn, 1);  // x1 = g^((p-1)/n)
    iX.clear(); iX.resize(maxn, 1);

    ll u = pw(g, (MOD-1)/maxn);
    ll iu = pw(u, MOD-2);

    for (int i = 1; i < maxn; i++) {
        X[i] = X[i-1] * u;
        iX[i] = iX[i-1] * iu;
        if (X[i] >= MOD) X[i] %= MOD;
        if (iX[i] >= MOD) iX[i] %= MOD;
    }

    rev.clear(); rev.resize(maxn, 0);
    for (int i = 1, hb = -1; i < maxn; i++) {
        if (!(i & (i-1))) hb++;
        rev[i] = rev[i ^ (1<<hb)] | (1<<(maxk-hb-1));
} }

template<typename T>
void NTT(vector<T>& a, bool inv=false) {

    int _n = (int)a.size();
    int k = __lg(_n) + ((1<<__lg(_n)) != _n);
    int n = 1<<k;
    a.resize(n, 0);

    short shift = maxk-k;
    for (int i = 0; i < n; i++)
        if (i > (rev[i]>>shift))
            swap(a[i], a[rev[i]>>shift]);

    for (int len = 2, half = 1, div = maxn>>1; len <= n
        ; len<<=1, half<<=1, div>>=1) {
        for (int i = 0; i < n; i += len) {
            for (int j = 0; j < half; j++) {
                T u = a[i+j];
                T v = a[i+j+half] * (inv ? iX[j*div] :
                    X[j*div]) % MOD;
                a[i+j] = (u+v >= MOD ? u+v-MOD : u+v);
                a[i+j+half] = (u-v < 0 ? u-v+MOD : u-v)
                    ;
    } } }

    if (inv) {
        T dn = pw(n, MOD-2);
        for (auto& x : a) {
            x *= dn;
            if (x >= MOD) x %= MOD;
} } }

template<typename T>
inline void resize(vector<T>& a) {
    int cnt = (int)a.size();
    for (; cnt > 0; cnt--) if (a[cnt-1]) break;
```

```
144      a.resize(max(cnt, 1));
145 }
146
147 template<typename T>
148 vector<T>& operator*=(vector<T>& a, vector<T> b) {
149     int na = (int)a.size();
150     int nb = (int)b.size();
151     a.resize(na + nb - 1, 0);
152     b.resize(na + nb - 1, 0);
153
154     NTT(a); NTT(b);
155     for (int i = 0; i < (int)a.size(); i++) {
156         a[i] *= b[i];
157         if (a[i] >= MOD) a[i] %= MOD;
158     }
159     NTT(a, true);
160
161     resize(a);
162     return a;
163 }
164
165 template<typename T>
166 void inv(vector<T>& ia, int N) {
167     vector<T> _a(move(ia));
168     ia.resize(1, pw(_a[0], MOD-2));
169     vector<T> a(1, -_a[0] + (-_a[0] < 0 ? MOD : 0));
170
171     for (int n = 1; n < N; n<<=1) {
172         // n -> 2*n
173         // ia' = ia(2-a*ia);
174
175         for (int i = n; i < min(siz(_a), (n<<1)); i++)
176             a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD :
                    0));
177
178         vector<T> tmp = ia;
179         ia *= a;
180         ia.resize(n<<1);
181         ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia
                [0] + 2;
182         ia *= tmp;
183         ia.resize(n<<1);
184     }
185     ia.resize(N);
186 }
187
188 template<typename T>
189 void mod(vector<T>& a, vector<T>& b) {
190     int n = (int)a.size()-1, m = (int)b.size()-1;
191     if (n < m) return;
192
193     vector<T> ra = a, rb = b;
194     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n
            -m+1));
195     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n
            -m+1));
196
197     inv(rb, n-m+1);
198
199     vector<T> q = move(ra);
200     q *= rb;
201     q.resize(n-m+1);
202     reverse(q.begin(), q.end());
203
204     q *= b;
205     a -= q;
206     resize(a);
207 }
208
209 /* Kitamasa Method (Fast Linear Recurrence):
210 Find a[K] (Given a[j] = c[0]a[j-N] + ... + c[N-1]a[j
    -1])
211 Let B(x) = x^N - c[N-1]x^(N-1) - ... - c[1]x^1 - c[0]
212 Let R(x) = x^K mod B(x)   (get x^K using fast pow and
    use poly mod to get R(x))
213 Let r[i] = the coefficient of x^i in R(x)
214 => a[K] = a[0]r[0] + a[1]r[1] + ... + a[N-1]r[N-1] */
```

10 Linear Algebra

10.1 Gaussian-Jordan Elimination

```
1  int n; vector<vector<ll> > v;
2  void gauss(vector<vector<ll>>& v) {
3  int r = 0;
4  for (int i = 0; i < n; i++) {
5      bool ok = false;
6      for (int j = r; j < n; j++) {
7          if (v[j][i] == 0) continue;
8          swap(v[j], v[r]);
9          ok = true; break;
10     }
11     if (!ok) continue;
12     ll div = inv(v[r][i]);
13     for (int j = 0; j < n+1; j++) {
14         v[r][j] *= div;
15         if (v[r][j] >= MOD) v[r][j] %= MOD;
16     }
17     for (int j = 0; j < n; j++) {
18         if (j == r) continue;
19         ll t = v[j][i];
20         for (int k = 0; k < n+1; k++) {
21             v[j][k] -= v[r][k] * t % MOD;
22             if (v[j][k] < 0) v[j][k] += MOD;
23     } }
24     r++;
25 } }
```

10.2 Determinant

1. Use GJ Elimination, if there's any row consists of only 0, then det = 0, otherwise det = product of diagonal elements.

2. Properties of det:

   - Transpose: Unchanged
   - Row Operation 1 - Swap 2 rows: $-det$
   - Row Operation 2 - $k\overrightarrow{r_i}$: $k \times det$
   - Row Operation 3 - $k\overrightarrow{r_i}$ add to $\overrightarrow{r_j}$: Unchaged

11 Flow / Matching

11.1 Dinic

```
1  struct Dinic
2  {
3      int n,s,t,level[N],iter[N];
4      struct edge{int to,cap,rev;};
5      vector<edge>path[N];
6      void init(int _n,int _s,int _t)
7      {
8          n=_n,s=_s,t=_t;
9          FOR(i,0,n+1)path[i].clear();
10     }
11     void add(int a,int b,int c)
12     {
13         edge now;
14         now.to=b,now.cap=c,now.rev=sz(path[b]);
15         path[a].pb(now);
16         now.to=a,now.cap=0,now.rev=sz(path[a])-1;
17         path[b].pb(now);
18     }
19     void bfs()
20     {
21         memset(level,-1,sizeof(level));
22         level[s]=0;
23         queue<int>q;q.push(s);
24         while(q.size())
25         {
26             int now=q.front();q.pop();
27             for(edge e:path[now])
28             {
29                 if(e.cap>0&&level[e.to]==-1)
```

```
30                {
31                        level[e.to]=level[now]+1;
32                        q.push(e.to);
33                }
34            }
35        }
36    }
37    int dfs(int now,int flow)
38    {
39        if(now==t)return flow;
40        for(int &i=iter[now];i<sz(path[now]);i++)
41        {
42            edge &e=path[now][i];
43            if(e.cap>0&&level[e.to]==level[now]+1)
44            {
45                int res=dfs(e.to,min(flow,e.cap));
46                if(res>0)
47                {
48                    e.cap-=res;
49                    path[e.to][e.rev].cap+=res;
50                    return res;
51                }
52            }
53        }
54        return 0;
55    }
56    int dinic()
57    {
58        int res=0;
59        while(true)
60        {
61            bfs();
62            if(level[t]==-1)break;
63            memset(iter,0,sizeof(iter));
64            int now=0;
65            while((now=dfs(s,INF))>0)res+=now;
66        }
67        return res;
68    }
69 };
```

11.2 ISAP

```
1  #define SZ(c) ((int)(c).size())
2  struct Maxflow{
3    static const int MAXV=50010;
4    static const int INF =1000000;
5    struct Edge{
6      int v,c,r;
7      Edge(int _v,int _c,int _r):v(_v),c(_c),r(_r){}
8    };
9    int s,t; vector<Edge> G[MAXV];
10   int iter[MAXV],d[MAXV],gap[MAXV],tot;
11   void init(int n,int _s,int _t){
12     tot=n,s=_s,t=_t;
13     for(int i=0;i<=tot;i++){
14       G[i].clear(); iter[i]=d[i]=gap[i]=0;
15     }
16   }
17   void addEdge(int u,int v,int c){
18     G[u].push_back(Edge(v,c,SZ(G[v])));
19     G[v].push_back(Edge(u,0,SZ(G[u])-1));
20   }
21   int DFS(int p,int flow){
22     if(p==t) return flow;
23     for(int &i=iter[p];i<SZ(G[p]);i++){
24       Edge &e=G[p][i];
25       if(e.c>0&&d[p]==d[e.v]+1){
26         int f=DFS(e.v,min(flow,e.c));
27         if(f){ e.c-=f; G[e.v][e.r].c+=f; return f; }
28       }
29     }
30     if((--gap[d[p]])==0) d[s]=tot;
31     else{ d[p]++; iter[p]=0; ++gap[d[p]]; }
32     return 0;
33   }
34   int flow(){
35     int res=0;
36     for(res=0,gap[0]=tot;d[s]<tot;res+=DFS(s,INF));
37     return res;
38   } // reset: set iter,d,gap to 0
```

```
39 } flow;
```

11.3 MCMF

```
1  struct MCMF
2  {
3      int n,s,t,par[N+5],p_i[N+5],dis[N+5],vis[N+5];
4      struct edge{int to,cap,rev,cost;};
5      vector<edge>path[N];
6      void init(int _n,int _s,int _t)
7      {
8          n=_n,s=_s,t=_t;
9          FOR(i,0,2*n+5)par[i]=p_i[i]=vis[i]=0;
10     }
11     void add(int a,int b,int c,int d)
12     {
13         path[a].pb({b,c,sz(path[b]),d});
14         path[b].pb({a,0,sz(path[a])-1,-d});
15     }
16     void spfa()
17     {
18         FOR(i,0,n*2+5)dis[i]=INF,vis[i]=0;
19         dis[s]=0;
20         queue<int>q;q.push(s);
21         while(!q.empty())
22         {
23             int now=q.front();
24             q.pop();
25             vis[now]=0;
26             for(int i=0;i<sz(path[now]);i++)
27             {
28                 edge e=path[now][i];
29                 if(e.cap>0&&dis[e.to]>dis[now]+e.cost)
30                 {
31                     dis[e.to]=dis[now]+e.cost;
32                     par[e.to]=now;
33                     p_i[e.to]=i;
34                     if(vis[e.to]==0)
35                     {
36                         vis[e.to]=1;
37                         q.push(e.to);
38                     }
39                 }
40             }
41         }
42     }
43     pii flow()
44     {
45         int flow=0,cost=0;
46         while(true)
47         {
48             spfa();
49             if(dis[t]==INF)break;
50             int mn=INF;
51             for(int i=t;i!=s;i=par[i])
52                 mn=min(mn,path[par[i]][p_i[i]].cap);
53             flow+=mn;cost+=dis[t]*mn;
54             for(int i=t;i!=s;i=par[i])
55             {
56                 edge &now=path[par[i]][p_i[i]];
57                 now.cap-=mn;
58                 path[i][now.rev].cap+=mn;
59             }
60         }
61         return mp(flow,cost);
62     }
63 };
```

11.4 Hopcroft-Karp

```
1  struct HopcroftKarp {
2      // id: X = [1, nx], Y = [nx+1, nx+ny]
3      int n, nx, ny, m, MXCNT;
4      vector<vector<int> > g;
5      vector<int> mx, my, dis, vis;
6      void init(int nnx, int nny, int mm) {
7          nx = nnx, ny = nny, m = mm;
8          n = nx + ny + 1;
9          g.clear(); g.resize(n);
10     }
```

```
11      void add(int x, int y) {
12          g[x].emplace_back(y);
13          g[y].emplace_back(x);
14      }
15      bool dfs(int x) {
16          vis[x] = true;
17          Each(y, g[x]) {
18              int px = my[y];
19              if (px == -1 ||
20                  (dis[px] == dis[x]+1 &&
21                  !vis[px] && dfs(px))) {
22                  mx[x] = y;
23                  my[y] = x;
24                  return true;
25              }
26          }
27          return false;
28      }
29      void get() {
30          mx.clear(); mx.resize(n, -1);
31          my.clear(); my.resize(n, -1);
32
33          while (true) {
34              queue<int> q;
35              dis.clear(); dis.resize(n, -1);
36              for (int x = 1; x <= nx; x++){
37                  if (mx[x] == -1) {
38                      dis[x] = 0;
39                      q.push(x);
40                  }
41              }
42              while (!q.empty()) {
43                  int x = q.front(); q.pop();
44                  Each(y, g[x]) {
45                      if (my[y] != -1 && dis[my[y]] ==
                               -1) {
46                          dis[my[y]] = dis[x] + 1;
47                          q.push(my[y]);
48                      }
49                  }
50              }
51
52              bool brk = true;
53              vis.clear(); vis.resize(n, 0);
54              for (int x = 1; x <= nx; x++)
55                  if (mx[x] == -1 && dfs(x))
56                      brk = false;
57
58              if (brk) break;
59          }
60          MXCNT = 0;
61          for (int x = 1; x <= nx; x++) if (mx[x] != -1)
62              MXCNT++;
63      }
} hk;
```

11.5   Cover / Independent Set

```
1  V(E) Cover: choose some V(E) to cover all E(V)
2  V(E) Independ: set of V(E) not adj to each other
3
4  M = Max Matching
5  Cv = Min V Cover
6  Ce = Min E Cover
7  Iv = Max V Ind
8  Ie = Max E Ind (equiv to M)
9
10 M = Cv (Konig Theorem)
11 Iv = V \ Cv
12 Ce = V - M
13
14 Construct Cv:
15 1. Run Dinic
16 2. Find s-t min cut
17 3. Cv = {X in T} + {Y in S}
```

11.6   KM

```
1 struct KM
2 {
```

```
3      int n,mx[1005],my[1005],pa[1005];
4      int g[1005][1005],lx[1005],ly[1005],sy[1005];
5      bool vx[1005],vy[1005];
6      void init(int _n)
7      {
8          n=_n;
9          FOR(i,1,n+1)fill(g[i],g[i]+1+n,0);
10     }
11     void add(int a,int b,int c){g[a][b]=c;}
12     void augment(int y)
13     {
14         for(int x,z;y;y=z)
15             x=pa[y],z=mx[x],my[y]=x,mx[x]=y;
16     }
17     void bfs(int st)
18     {
19         FOR(i,1,n+1)sy[i]=INF,vx[i]=vy[i]=0;
20         queue<int>q;q.push(st);
21         for(;;)
22         {
23             while(!q.empty())
24             {
25                 int x=q.front();q.pop();
26                 vx[x]=1;
27                 FOR(y,1,n+1)if(!vy[y])
28                 {
29                     int t=lx[x]+ly[y]-g[x][y];
30                     if(t==0)
31                     {
32                         pa[y]=x;
33                         if(!my[y]){augment(y);return;}
34                         vy[y]=1, q.push(my[y]);
35                     }
36                     else if(sy[y]>t)pa[y]=x,sy[y]=t;
37                 }
38             }
39             int cut=INF;
40             FOR(y,1,n+1)if(!vy[y]&&cut>sy[y])cut=sy[y];
41             FOR(j,1,n+1)
42             {
43                 if(vx[j])lx[j]-=cut;
44                 if(vy[j])ly[j]+=cut;
45                 else sy[j]-=cut;
46             }
47             FOR(y,1,n+1)
48             {
49                 if(!vy[y]&&sy[y]==0)
50                 {
51                     if(!my[y]){augment(y);return;}
52                     vy[y]=1;q.push(my[y]);
53                 }
54             }
55         }
56     }
57     int solve()
58     {
59         fill(mx,mx+n+1,0);fill(my,my+n+1,0);
60         fill(ly,ly+n+1,0);fill(lx,lx+n+1,0);
61         FOR(x,1,n+1)FOR(y,1,n+1)
62             lx[x]=max(lx[x],g[x][y]);
63         FOR(x,1,n+1)bfs(x);
64         int ans=0;
65         FOR(y,1,n+1)ans+=g[my[y]][y];
66         return ans;
67     }
};
```

12   Combinatorics

12.1   Catalan Number

$$C_0 = 1, C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}, C_n = C_n^{2n} - C_{n-1}^{2n}$$

0	1	1	2	5
4	14	42	132	429
8	1430	4862	16796	58786
12	208012	742900	2674440	9694845

12.2   Burnside's Lemma

Let X be the original set.
Let G be the group of operations acting on X.
Let X^g be the set of x not affected by g.
Let X/G be the set of orbits.
Then the following equation holds:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

13   Special Numbers

13.1   Fibonacci Series

1	1	1	2	3
5	5	8	13	21
9	34	55	89	144
13	233	377	610	987
17	1597	2584	4181	6765
21	10946	17711	28657	46368
25	75025	121393	196418	317811
29	514229	832040	1346269	2178309
33	3524578	5702887	9227465	14930352

$f(45) \approx 10^9, f(88) \approx 10^{18}$

13.2   Prime Numbers

- First 50 prime numbers:

1	2	3	5	7	11
6	13	17	19	23	29
11	31	37	41	43	47
16	53	59	61	67	71
21	73	79	83	89	97
26	101	103	107	109	113
31	127	131	137	139	149
36	151	157	163	167	173
41	179	181	191	193	197
46	199	211	223	227	229

- Very large prime numbers:
  1000001333   1000500889   2500001909
  2000000659   900004151   850001359

- $\pi(n) \equiv$ Number of primes $\leq n \approx n/((\ln n) - 1)$
  $\pi(100) = 25, \pi(200) = 46$
  $\pi(500) = 95, \pi(1000) = 168$
  $\pi(2000) = 303, \pi(4000) = 550$
  $\pi(10^4) = 1229, \pi(10^5) = 9592$
  $\pi(10^6) = 78498, \pi(10^7) = 664579$