Conteúdo

1 Template

1.1 template

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

#define DEBUG if(1)
#define MAXN 100
#define MAX INT_MAX
#define MAXLL INT_MAX
#define MAXU ULLONG_MAX
#define MIN -2000000
#define endl "\n"
#define INF INT_MAX
#define MOD 1000000007
#define s(n) scanf("%d", &n)
#define ss(a,b) scanf("%d %d",&a,&b)
#define pb push_back
#define mp make_pair
#define M_PI 3.14159265358979323846
#define sz(a) int(a.size())
#define lli unsigned long long int
#define rep(i,a,n) for (int i=a;i<n;i++)
#define ler(a,n,vec) for(int i=0;i<n;i++)s(a), vec.pb(a)
typedef vector<lli> vi;
typedef vector<vi> vvi;
typedef pair<int,int> ii;
#define DEBUG if(1)
#define F first
#define S second
int dx[] = {0, 1, 0, -1};
int dy[] = {1, 0, -1, 0};
int ddx[] = {1, 0};
int ddy[] = {1, 1};
```

2 Data Structures

2.1 Binary Indexed Tree

```
int tree[N];
void atualiza(int x, int v){
   for(;x<=n;x+=(x&-x)) tree[x] += v;
}

int sum(int x){
   int s = 0;
   for(;x>0;x-=(x&-x)) s += tree[x];
   return s;
}
```

2.2 Bit 2D

```
int bit[MAX][MAX], n, m;

int sum(int x, int y){
   int ans = 0;
   for(int i=x;i>0;i-=i&-i)
      for(int j=y;j>0;j-=j&-j) ans += bit[i][j];

   return ans;
}

void update(int x, int y, int d){
   for(int i=x;i<=n;i+=i&-i)
      for(int j=y;j<=m;j+=j&-j) bit[i][j] += d;
}

int querySq(int x, int xx, int y, int yy){
    return query(xx, yy)+query(x-1, y-1)-query(x-1, yy)-query(xx, y-1);;
}
```

2.3 Xor Segtree

```
int N, M, v[MAXN], st[5*MAXN], lazy[5*MAXN];
int left(int p){
    return (p << 1);
```

```
}
int right(int p){
    return (p << 1) + 1;
}
void build_tree(int p, int L, int R){
    lazy[p] = -1;
    if(L == R){
        st[p] = v[L];
        return;
    }
    int mid = (L + R) >> 1;
    build_tree(left(p), L, mid);
    build_tree(right(p), mid+1, R);
    st[p] = st[left(p)] ^ st[right(p)];
}

void propagate(int p, int L, int R){
    if(lazy[p] != -1){
        if((R-L+1) & 1)st[p] = lazy[p];
        else st[p] = 0;
        if(L != R)lazy[left(p)] = lazy[right(p)] = lazy[p];
    }
    lazy[p] = -1;
}
void update(int p, int L, int R, int i, int j, int val){
    propagate(p, L, R);
    if(L > j or R < i)return;
    if(L >= i and R <= j){
        lazy[p] = val;
        propagate(p, L, R);
        return;
    }
    int mid = (L + R) >> 1;
    update(left(p), L, mid, i, j, val);
    update(right(p), mid+1, R, i, j, val);
    st[p] = st[left(p)] ^ st[right(p)];
}

void update(int i, int j, int val){
    update(1, 0, N-1, i, j, val);
}
int main(){
```

```
    while(scanf("%d%d", &N, &M) > 0){
        for(int i=0; i<N; i++){
            scanf("%d", &v[i]);
        }
        build_tree(1, 0, N-1);
        while(M--){
            int X, Y, V;
            scanf("%d%d%d", &X, &Y, &V);
            update(X-1, Y-1, V);
            printf("%s\n", st[1] ? "Possivel" : "Impossivel");
        }
    }
}
```

2.4 SegT + Lazy

```
#define MAX 1000000
int tree[MAX];
int A[MAX];
int n;

void build(int node=1, int start=1, int end=n){
   if(start == end) tree[node] = A[start];
   else {
      int mid = (start + end) >> 1;
      build(2*node, start, mid);
      build(2*node+1, mid+1, end);
      tree[node] = tree[2*node] + tree[2*node+1];
   }
}

void update(int idx, int val, int node=1, int start=1, int end=n){
   if(start == end){
      A[idx] += val;
      tree[node] += val;
   } else {
      int mid = (start + end) >> 1;
      if(start <= idx and idx <= mid) update(idx, val, 2*node, start, mid);
      else update(idx, val, 2*node+1, mid+1, end);
      tree[node] = tree[2*node] + tree[2*node+1];
   }
}
```

```
int query(int l, int r, int node=1, int start=1, int end = n){
   if(r < start or end < l) return 0;
   if(l <= start and end <= r) return tree[node];
   int mid = (start + end) >> 1;
   int p1 = query(l, r, 2*node, start, mid);
   int p2 = query(l, r, 2*node+1, mid+1, end);
   return (p1+p2);
}


void updateRange(int l, int r, int val, int node = 1, int start = 1, int end = n){
   if(start > end or start > r or end < l) return ;
   if(start == end){
      tree[node] += val;
      return ;
   }

   int mid = (start + end) >> 1;
   updateRange(l, r, val, 2*node, start, mid);
   updateRange(l, r, val, 2*node+1, mid+1, end);

   tree[node] = tree[2*node+1] + tree[2*node];
}

void lazyUpdate(int l, int r, int val, int node = 1, int start = 1, int end = n){
   if(lazy[node]!=0){
      tree[node] += (end - start + 1) * lazy[node];
      if(start != end){
         lazy[2*node] += lazy[node];
         lazy[2*node+1] += lazy[node]; //Update nos filhos que precisam ser atualizados
      }
      lazy[node] = 0;
   }

   if(start > end or start > r or end < l) return ;
   if(start >= l and end <= r){
      tree[node] += (end - start + 1) * val;
      if(start != end){
         lazy[2*node] += val;
         lazy[2*node+1] += val;
      return ;
```

```
      }
   }

   int mid = (start + end) >> 1;
   lazyUpdate(l,r,val,2*node, start,mid);
   lazyUpdate(l,r,val,2*node+1, mid+1, end);
   tree[node] = tree[2*node] + tree[2*node+1];
}


int lazyQuery(int l, int r, int node = 1 , int start = 1, int end = n){
   if(start > end or start > r or end < l) return 0;
   if(lazy[node]!=0){
      tree[node] += (end - start + 1) * lazy[node];
      if(start != end){
         lazy[2*node] += lazy[node];
         lazy[2*node+1] += lazy[node];
      }
      lazy[node] = 0;
   }
   if(start >= l and end <= r) return tree[node];
   int mid = (start + end) >> 1;
   int p1 = lazyQuery(l, r, 2*node, start, mid);
   int p2 = lazyQuery(l, r, 2*node+1, mid+1, end);
   return (p1+p2);
}
```

2.5 Sqrt Decomp

```
int a[MAX];
int sum[MAX];
int k; // sqrt(n)
void update(int id, int vl){ // update index id with value vl
   sum[id / k] = sum[id / k] - a[id] + vl;
   a[id] = vl;
}
int query(int lf, int rg) { // query in range lf to rg
   int res = 0;
   int now = lf;
   while((now + 1) % k != 0 and now <= rg)res += a[now++];
   while(now + k <= rg)res += sum[now / k], now+=k;
   while(now <= rg)res += a[now++];
```

```
   return res;
}
```

2.6 Misof Tree (kth Element)

```
int tree[17][65536];
void insert(int x) { for (int i=0; i<17; i++) { tree[i][x]++; x/=2; } }
void erase(int x) { for (int i=0; i<17; i++) { tree[i][x]--; x/=2; } }
int kThElement(int k) { int a=0, b=16; while (b--) { a*=2; if (tree[b][a]<k) k-=tree[b][a++]; }
    return a; }
```

2.7 Merge Sort Tree

```
int n,m;

vi tree[1000000];
int a[250000];

void build(int node=1, int l=1, int r=n){
   if(l == r){
      tree[node].pb(a[l]);
      return ;
   }
   int mid = (r + l) / 2;
   build(2*node, l, mid);
   build(2*node+1, mid+1, r);

   tree[node].resize(sz(tree[2*node]) + sz(tree[2*node+1]));
   merge(tree[2*node].begin(), tree[2*node].end(), tree[2*node+1].begin(),
       tree[2*node+1].end(), tree[node].begin());
}

int query(int l, int r, int k, int node = 1, int start = 1, int end = n){
   if(end < l or start > r) return 0;
   if(l <= start and end <= r){
      return upper_bound(tree[node].begin(), tree[node].end(), k) - tree[node].begin();
   }
   int mid = (start + end) / 2;
   return query(l, r, k, 2*node, start, mid) + query(l, r, k, 2*node+1, mid+1, end);
}
```

2.8 Sparse Table

```
/*
   Sparse:
   Query O(1)
   Upd (NLogn)
   Build (NLogn)

   SetTree and Bit:
   Query O(Log N)
   Upd O(Log N)
   Build O(N)
 */

void build(){
   for(int j=0;(1<<j)<=n;j++){
      for(int i=0;(i + (1 << j) - 1) < n; i++){
         if(j){
            int pos1 = spt[i][j-1];
            int pos2 = spt[i + (1 << (j-1))][j-1];
            spt[i][j] = a[pos1] < a[pos2] ? pos1 : pos2;
         } else {
            spt[i][j] = i;
         }
      }
   }
}

int rmq(int i, int j){
   int len = j - i + 1;
   int k = (int) floor(log((double) len) / log(2.0));
   if(a[spt[i][k]] <= a[spt[j-(1<<k)+1][k]]){
      return spt[i][k];
   } else {
      return spt[j - (1 << k) + 1][k];
   }
}
```

2.9 Union Find Qt

```
int n,m;
```

```
int pai[100010];
int peso[100010];
int vec[100010];
int qt[100010];
int ans = 0;

int find(int x){
   if(pai[x] == x) return x;
   return pai[x] = find(pai[x]);
}

int test(int x, int y){
   int aa = vec[find(pai[x])];
   int bb = vec[find(pai[y])];
   // cout << aa << " - " << bb << endl;

   // cout << "X : " << x << " - " << "Y : " << y << endl;
   if(aa > bb){
      if(find(1) == find(x)) return 1;
   } else if(aa < bb){
      if((find(1)) == find(y)) return 1;
   }
   return 0;
}

void join(int x, int y){
   x = find(x);
   y = find(y);
   if(x == y) return ;

   if(peso[x] < peso[y]){
      pai[x] = y;
      vec[y] += vec[x];
   } else if(peso[x] > peso[y]){
      pai[y] = x;
      vec[x] += vec[y];
   } else {
      pai[x] = y;
      peso[y]++;
      vec[y] += vec[x];
   }
}
```

```
int main(){
   ios::sync_with_stdio(false);
   cin.tie(0);
   while(1){
      cin >> n >> m;
      ans = 0;
      if(n == 0 and m == 0) break;
      for(int i=1;i<=n;i++) cin >> vec[i];
      memset(peso, 0, sizeof peso);
      for(int i=1;i<=n;i++) pai[i] = i;
      for(int i=0;i<m;i++){
         int a,b,c;
         cin >> a >> b >> c;
         if(a == 1){
            join(b,c);
         } else {
            ans += test(b,c);
         }
      }
      cout << ans << endl;
   }


   return 0;
}
```

3 Dynamic Programming

3.1 Coin Change

```
int pd[MAX];

int solve(int x, vi &c){
   if(x==0) return 1;
   if(x<0) return 0;
   if(pd[x]>=0) return pd[x];

   for(int i=0;i<sz(c);i++){
      if(solve(x - c[i] , c)) return pd[x-c[i]] = 1;
   }

   return 0;
}

int main(){
   int n;
   vi vec;
   int tot = 0;
   for(int i=0;i<n;i++){
      int a;
         cin >> a;
            vec.pb(a);
      tot+=a;
   }
   for(int i=0;i<=tot;i++)pd[i] = -1;

   if(solve(tot/2, vec)) cout << "YES" << endl;
   else cout << "NO" << endl;

   return 0;
}
```

3.2 Distinct Substrings

```
int v[MAX], dp[MAX], l[500];
char sub[MAX], s[MAX], ss[MAX];;
const int mod = 1e9+7;
void solve(int tc){
```

```
   printf("Caso #%d:\n", tc);
   int m;
   scanf("%s %d", s+1, &m);
   for(int i = 0; i < m; i++){
      scanf("%s", ss+1);
      int jj = 1;
      int ans;
      int equal = 0;
      for(int j =1 ; s[j]; j++){
         if(s[j] == ss[jj])jj++;
         if(ss[jj] == '\0'){
            ans = j;
            if(s[j+1] == '\0')equal = 1;
            break;
         }
      }
      if(equal){
         printf("1\n");
         continue;
      }
      if(ss[jj] != '\0')printf("0\n");
      else {
         memset(l, 0, sizeof l);
         int k;
         dp[ans] = 1;
           for(int j = ans+1; s[j]; j++){
               dp[j] = dp[j-1]*2;
               if(l[s[j]])dp[j]-=dp[l[s[j]]-1];
               l[s[j]] = j;
               k = j;
           }
           printf("%d\n", dp[k]);
      }
   }

}
```

3.3 LIS

```
int lis(vector<int> *Ar, int N){
   for(int i = N - 1; i >= 0; i--) {
        LIS[i] = 1;
```

```
        for(int j = i + 1; j < N; j++) {
            if(Ar[i] < Ar[j]) LIS[i] = max(LIS[j] + 1, LIS[i]);
        }
    }
}

 for (i = 0; i < n; i++ )
        lis[i] = 1;

 for (i = 1; i < n; i++ )
        for (j = 0; j < i; j++ )
            if ( arr[i] > arr[j] && lis[i] < lis[j] + 1)
                lis[i] = lis[j] + 1;

 for (i = 0; i < n; i++ )
        if (max < lis[i])
            max = lis[i];
```

3.4 Logest Common Subsequence

```
string a,b;
int pd[1500][1500];

void lcs(){

   for(int i=0;i<=sz(a);i++){
      for(int j=0;j<=sz(b);j++){
         if(i == 0 or j == 0) pd[i][j] = 0;
         else if(a[i-1] == b[j-1]) pd[i][j] = pd[i-1][j-1] + 1;
         else pd[i][j] = max(pd[i-1][j], pd[i][j-1]);
      }
   }

   cout << pd[sz(a)][sz(b)] << endl;
}

void pickString(){
      for(int i=0;i<=sz(a);i++){
         for(int j=0;j<=sz(b);j++) cout << pd[i][j] << " ";
            cout << endl;
      }
      cout << " ---- " << endl;
```

```
   int n = sz(a), m = sz(b);
   string ans = "";
   while(pd[n][m]!=0){
      if(pd[n][m] > pd[n-1][m] and pd[n][m] > pd[n][m-1]) ans += a[n-1],n--, m--;
      else if(pd[n-1][m] >= pd[n][m] and pd[n-1][m] > pd[n][m-1]) n--;
      else if(pd[n][m-1] > pd[n-1][m] and pd[n][m-1] >= pd[n][m]) m--;
      else n--;
      cout << n << " - " << m << endl;
   }
   reverse(ans.begin(), ans.end());
   cout << "Ans : " << ans << endl;
}

int main(){

   cin >> a >> b;
   lcs();

   pickString();

   return 0;
}
```

3.5 Longest Palindrome

```
string a, b;
int DP[MAX][MAX];
int dp(int l, int r){
   if(DP[l][r] != -1) return DP[l][r];
   if(l == r) return DP[l][r] = 1;
   if(l+1 == r){
      return DP[l][r] = a[l] == a[r] ? 2 : 1;
   }
   if(a[l] == a[r]) return DP[l][r] = 2 + dp(l+1, r-1);
   return DP[l][r] = max(dp(l, r-1), dp(l+1, r));
}
int main(){
   int n;
   cin >> n;
   getchar();
   for(int i = 0; i < n; i++){
```

```
      getline(cin, a);
      if(a.size() == 0){printf("0\n"); continue;}
      memset(DP, -1, sizeof DP);
      printf("%d\n", dp(0, a.size()-1));
   }
}
```

3.6 Max 2D Range Sum

```
for(int i=0; i<n; i++) {
       for(int j=0; j<n; j++) {
           scanf("%d", &A[i][j]);
           if(i > 0) A[i][j] += A[i-1][j];
           if(j > 0) A[i][j] += A[i][j-1];
           if(i > 0 && j > 0) A[i][j]-=A[i-1][j-1];
       }
    }
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        for(int k=i; k<n; k++){
            for(int l=j; l<n; l++){
                int c = A[k][l];
                if(i > 0) c-= A[i-1][l];
                if(j > 0) c-= A[k][j-1];
                if(i > 0 && j > 0) c+= A[i-1][j-1];
                ans = max(ans, c);
            }
        }
    }
}
```

3.7 Min Cost Path Matrix

```
int cost[101][101];

int minCost(int m, int n){
   int i,j;
   int tc[101][101];
   tc[0][0] = cost[0][0];

   for(int i=1;i<=n;i++) tc[i][0] = tc[i-1][0] + cost[i][0];
```

```
   for(int j=1;j<=m;j++) tc[0][j] = tc[0][j-1] + cost[0][j];

   for(int i=1;i<=m;i++)
      for(int j=1;j<=n;j++) tc[i][j] = min(tc[i-1][j-1], min(tc[i-1][j], tc[i][j-1])) +
          cost[i][j];

   return tc[m][n];
}


int main(){
   int n,m;
   cin >> n >> m;
   for(int i=0;i<n;i++)
      for(int j=0;j<m;j++) s(cost[i][j]);

   cout << minCost(2, 2) << endl;

   return 0;
}
```

3.8 Knap Pick

```
int n,s;
int peso[501];
int valor[501];
int pd[501][501];
int ans = 0;
int picks[501][501];

int knap(int obj, int aguenta, int cc){
   int take, dontTake;

   take = dontTake = 0;
   cc++;

   if(obj==0){
      if(peso[0]<=aguenta){
         picks[obj][aguenta]=1;
         pd[obj][aguenta] = valor[0];
         return valor[0];
      } else {
```

```
            picks[obj][aguenta] = -1;
            pd[obj][aguenta] = 0;
            return 0;
        }
    }

    if(peso[obj] <= aguenta){
        take = valor[obj] + knap(obj-1, aguenta - peso[obj],cc);
        dontTake = knap(obj-1, aguenta,cc);
        pd[obj][aguenta] = max(take, dontTake);
        ans = max(ans,cc);
        if(take > dontTake) picks[obj][aguenta]=1;
        else picks[obj][aguenta]=-1;

        return pd[obj][aguenta];
    }
}

void printPicks(){
    int aguenta = s;
    for(int i = n-1; i>=0;i--)
        if(picks[i][aguenta]==1){
            cout << valor[i] << " - " << peso[i] << endl;
            aguenta -= peso[i];
        }
}

int main(){
    ss(n,s);
    memset(pd,-1,sizeof pd);

    for(int i=0;i<n;i++) ss(valor[i], peso[i]);

    cout << knap(n-1,s, 0) << endl;



    cout << ans << endl;
    cout << " ------ " << endl;
    printPicks();
```

```
   return 0;
}
```

3.9 Knap Bottom Up

```
int k, m;
lli pd[2000][2000];
ii vec[2001];

int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   cin >> k >> m;
   for(int i=1;i<=m;i++){
      int a,b;
      cin >> a >> b;
      vec[i] = {a, b};
   }

   for(int i=1;i<=m;i++){
      for(int j=1;j<=k;j++){
         int v = vec[i].S, w = vec[i].F;
         if(j >= w) pd[i][j] = max(pd[i-1][j], pd[i-1][j - w] + v);
         else pd[i][j] = pd[i-1][j];
      }
   }

   cout << pd[m][k] << endl;
```

3.10 Palindrome Partition

```
//Fazer toda substring ser um palindromo
int pd[2001][2001];
int c[2001];
string s;

int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);
```

```
  int t = 1;
  while(1){
     int n;
      cin >> n;
      if(n == 0) break;
      cin >> s;
      if(t!=1) cout << endl;
      memset(pd, -1, sizeof pd);
      memset(c, 0, sizeof c);
      for(int i=0;i<n;i++) pd[i][i] = 1;

      for(int i=1;i<n;i++){
        for(int j=0;j< n - i;j++){
           if(s[j] == s[i + j] and pd[j+1][i+j-1] == 1) pd[j][i+j] = 1;
           else if(i == 1 and s[j] == s[j+i]) pd[j][i+j] = 1;
           else pd[j][i+j] = 0;
        }
      }

      cout << "Teste " << t++ << endl;

      for(int i=0;i<n;i++){
        if(pd[0][i]){
           c[i] = 0;
        } else {
           c[i] = INT_MAX;
           for(int j=0;j<i;j++){
              if(pd[j+1][i] and 1 + c[j] < c[i]) c[i] = 1 + c[j];
           }
        }
      }
     cout << c[n-1] + 1 << endl;
  }


  return 0;
}
```

3.11 Digit Dp

```
lli pd[20][2][200];
```

```
lli solve(int pos, int can, lli sum, vi b){
   if(pos == sz(b))
      return sum;


   if(pd[pos][can][sum] != -1) return pd[pos][can][sum];

   int atual = can ? (b[pos]) : 9;

   lli aux = 0;

   for(int i=0;i<=atual;i++){
      if(i == b[pos]) aux += solve(pos+1, can, sum + i, b);
      else aux += solve(pos+1, 0, sum+i, b);
   }

   return pd[pos][can][sum] = aux;
}
```

3.12 LIS Cute

```
//SEM REPETIR NUMEROS : 1 4 4 5, : 3

set<int> st;
set<int>::iterator it;

int main(){
   st.clear();
   int n;
   int vec[50];
   cin >> n;
   for(int i=0;i<n;i++) cin >> vec[i];

   for(int i=0;i<n;i++){
      st.insert(vec[i]);
      it = st.find(vec[i]);
      it++; if(it!=st.end()) st.erase(it);
   }

   cout << st.size() << endl;
```

```
   return 0;
}

//PARA NUMEROS REPETIDOS : 1 4 4 5. : 4

multiset<int> s;
multiset<int> ::iterator it;

int main(){
   int n;
   cin >> n;
   vi vec;
   for(int i=0;i<n;i++){
      int a;
      cin >> a;
      vec.pb(a);
   }

   for(int i=0;i<sz(vec);i++){
      s.insert(vec[i]);
      it = s.upper_bound(vec[i]);

      for(auto j : s) cout << j << " ";
         cout << endl;

      if(it != s.end()) s.erase(it);
   }

   for(auto j : s) cout << j << " ";
      cout << endl;

   cout << sz(s) << endl;


   return 0;
}
```

4 Graph

4.1 Belman

```
bool ans = false;
for(int i = 1; i <=n; i++)dist[i] = INF;
dist[1] = 0;
for(int i = 1; i <=n; i++){
   for(int k = 1; k <=n; k++){
      for(auto f : adj[k]){
         if(dist[k] == INF)continue;
         if(dist[f.first] > dist[k] + f.second){
            dist[f.first] = dist[k] + f.second;
            if(i == n)ans = true;// negative cycle
         }
      }
   }
}
```

4.2 Bipartite Checking

```
vector<int>adj[MAX];
int vstd[MAX];
int q[MAX];
bool dfs(int u){ // return true if is bipartite
    bool bic = true;
    for(int x : adj[u]){
        if(!vstd[x]){
            vstd[x] = (vstd[u] == 1) ? 2 : 1;
            bic &= dfs(x);
        }else if(vstd[x] == vstd[u])return false;
    }
    return bic;
}
```

4.3 Cut Vertex + Cut Edge

```
int dfsCnt;
int low[MAX], p[MAX], d[MAX], child[MAX];
vector<int>adj[MAX];
int vstd[MAX];
```

```
int root;
void dfs(int v){ // first start root and d[root] = 0/ (child/vstd) = 0 / p = -1
   vstd[v] = 1;
   low[v] = d[v];
   for(auto u : adj[v]){
      if(!vstd[u]){
         child[v]++;
         p[u] = v;
         d[u] = d[v]+1;
         dfs(u);
         low[v] = min(low[v], low[u]);
         if(low[u] > d[v])printf("%d - - %d (cut edge) \n", v, u);
         if(low[u] >= d[v] && (v != root || child[v] > 1))printf("%d - > cut vertex\n", v);
      }
      else if(u != p[v])low[v] = min(low[v], d[u]);
   }
   vstd[v] = 2;
}
```

4.4 Cycles Finding

```
void dfs(int u){
    vstd[u] = 1;
    for(int i = 0; i <sz(adj[u]); i++){
        int v = adj[u][i].fi;
        if(!vstd[v]){
            p[v] = u;
            dfs(v);
        }
        else if(vstd[v] == 1){
            if(v != p[u]){
                printf("------Cycle-----\n");
                printf("%d -> %d\n", u, v);
                int k = u;
                while(p[k] != v){
                    printf("%d -> %d\n", p[k], k);
                    k = p[k];
                }
                printf("%d -> %d\n", p[k], k);
                printf("-----EndCycle----\n");
            }
        }
```

```
    }
    vstd[u] = 2;
}
```

4.5 Dijkstra

```
int dist[MAX];
vector<ii>adj[MAX];
void dijk(int v){
    dist[v] = 0;
    int u;
    priority_queue< ii, vector<ii>, greater<ii> >pq;
    pq.push(mk(dist[v], v));
    while(!pq.empty()){
        u = pq.top().se;
        pq.pop();
        for(ii p: adj[u])
            if(dist[p.fi] > dist[u] + p.se){
                dist[p.fi] = dist[u] + p.se;
                pq.push(mk(dist[p.fi], p.fi));
            }
    }
}
```

4.6 Dinics

```
const int INF = 1e9+10;
struct edge{
   int v1, v2, cap, flow;
   edge(int _v1, int _v2, int _cap, int _flow) : v1(_v1), v2(_v2), cap(_cap), flow(_flow) {}
};
vector<edge> ed;
vector<int> adj[N];
int dist[N], que[N], nxt[N];
int src, sink, cntE;
int n;
void add_edge(int u, int v, int cap){ // antes de adicionar qualquer aresta lembrar de zerar o
     contador(cntE)
   adj[u].push_back(cntE++);
   adj[v].push_back(cntE++);
   ed.push_back(edge(u, v, cap, 0));
```

```
   ed.push_back(edge(v, u, 0, 0));
}
int bfs(){
   memset(dist, -1, sizeof dist);
   dist[src] = 0;
   int head = 0, tail = 0;
   que[tail++] = src;
   while(head < tail && dist[sink] == -1){
      int v = que[head++];
      for(int i = 0; i < (int)adj[v].size(); i++){
         int id = adj[v][i];
         int to = ed[id].v2;
         if(dist[to] == -1 && ed[id].flow < ed[id].cap){
            que[tail++] = to;
            dist[to] = dist[v] + 1;
         }
      }
   }
   return dist[sink] != -1;
}
int dfs(int v, int flow) {
   if(!flow || v == sink)return flow;
   while(nxt[v] < (int)adj[v].size()) {
      int id = adj[v][nxt[v]];
      int to = ed[id].v2;
      if(dist[to] != dist[v] + 1){
         ++nxt[v];
         continue;
      }
      int curFlow = dfs(to, min(flow, ed[id].cap - ed[id].flow));
      if(curFlow) {
         ed[id].flow += curFlow;
         ed[id^1].flow -= curFlow;
         return curFlow;
      }
      ++nxt[v];
   }
   return 0;
}
int dinic() {
   int flow = 0;
   while(true){
```

```
      if(!bfs())break; // nao encontrou caminho ate o destino
      memset(nxt, 0, sizeof nxt);
      while(int curFlow = dfs(src, INF))flow+=curFlow;
   }
   return flow;
}
void startFlow(int tam){
   for(int i = 0; i < tam; i++)adj[i].clear();
   ed.clear();
   cntE = 0;
}
```

4.7 Edmonds Karp

```
/* Edmonds Karp Max flow
   Time complexity O(VE^2)
   nao esquecer de adicionar os back edges na lista de adjascencias
*/
int res[MAX][MAX], mf, flow, src, sink;
vector<int> p, adj[MAX];
void augment(int v, int minEdge) {
   if(v == src) flow = minEdge;
   else if(p[v] != -1){
      augment(p[v], min(minEdge, res[p[v]][v]));
      res[p[v]][v] -= flow;
      res[v][p[v]] += flow;
   }
}
int maxflow(){
   mf = 0;
   while(1){
      flow = 0;
      bitset<MAX> vis; vis[src] = true;
      queue<int>q; q.push(src);
      p.assign(MAX, -1);
      while(!q.empty()){
         int u = q.front(); q.pop();
         if(u == sink) break;
         for(int j = 0; j < adj[u].size(); j++){
            int v = adj[u][j];
            if(res[u][v] > 0 && !vis[v]){
               vis[v] = true, q.push(v), p[v] = u;
```

```
            }
         }
      }
      augment(sink, INF);
      if(flow == 0)break;
      mf+=flow;
   }
   return mf;
}
```

4.8 Floyd

```
/* Floyd Warshal
   N MAX = 400
   Adj[i][j] contains the weight of edge (i, j)
   or INF (1B) if there is no such edge
 */
// remember loop is k->i-> j
for(int k = 0; k< N; k++)
   for(int i = 0; i<N; i++)
      for(int j = 0; j<N; j++)
         adj[i][j] = min(adj[i][j], adj[i][k]+adj[k][j]);
/* parent matrix
   let p be a 2D parent matrix , where p[i][j] is the last vertex before j
   on shortest path from i to j
 */
// initialize the parent matrix
for(int i = 0; i<N; i++)
   for(int j = 0; j<N; j++)
      p[i][j] = i;

// floyd + update parent
for(int k = 0; k< N; k++)
   for(int i = 0; i<N; i++)
      for(int j = 0; j<N; j++)
         if(adj[i][k]+adj[k][j] < adj[i][j]){
            adj[i][j] = adj[i][k]+adj[k][j];
            p[i][j] = p[k][j];
         }
// print the shortest path
void printPath(int i, int j){
   if(i != j)printPath(i, p[i][j]);
```

```
   printf(" %d", j);
}
/* Minimax and Maximin
   adj[i][j] is the weigth from i to j
*/
for(int k = 0; k < N; k++)
   for(int i = 0; i< N; i++)
      for(int j = 0 ; j< N; j++)
         adj[i][j] = min(adj[i][j], max(adj[i][k], adj[k][j]));

/* Transitive Closure
Check if two vertex i j are directly or indirectly connected by checking adj[i][j]
where matrix adj contains true if i is directly connected to j or false otherwise
*/
for(int k = 0; k<N; k++)
   for(int i = 0; i<N; i++)
      for(int j = 0; j<N; k++)
         adj[i][j] |= (adj[i][k] & adj[k][j]);
```

4.9 Hungarian

```
#define hungType double
hungType mcost[MAX][MAX]; // cost matrix
hungType hung(const int &n, const int &m) { // idx start 1 ( n <= m)
    vector<hungType> u(n+1), v(m+1), minv;
    vector<int> par(m+1), next(m+1);
    vector<char>vstd;
    repn(i, 1, n) {
        par[0] = i;
        int idj = 0;
        minv.assign(m+1, linf);
        vstd.assign(m+1, false);
        do {
            vstd[idj] = true;
            int idi = par[idj], temp;
            hungType det = linf;
            repn(j, 1, m) if (!vstd[j]) {
                hungType cur = mcost[idi][j]-u[idi]-v[j];
                if (cur < minv[j])minv[j] = cur, next[j] = idj;
                if (minv[j] < det)det = minv[j], temp = j;
            }
            repn(j, 0, m){
```

```
            if (vstd[j])u[par[j]]+=det, v[j]-=det;
            else minv[j]-=det;
        }
        idj = temp;
    } while (par[idj]);
    do {
        int temp = next[idj];
        par[idj] = par[temp];
        idj = temp;
    } while (idj);
  }
  return abs(v[0])+eps;
}
```

4.10 LCA

```
// LCA pre-pro O(n log(n)) // query O (log(n))
//LCA CodCad
#define MAXN 50500
#define MAXL 20

int n;
int nivel[MAXN];
int ancestral[MAXN][MAXL];
int pai[MAXN];

vi adj[MAXN];


void dfs(int u){

  for(int i=0;i<sz(adj[u]);i++){
    int v = adj[u][i];
    if(nivel[v] == -1){
      pai[v] = u;
      nivel[v] = nivel[u] + 1;
      dfs(v);
    }
  }
}

void build(){
```

```
    memset(pai, -1, sizeof pai);
    memset(nivel, -1, sizeof nivel);
    memset(ancestral, -1, sizeof ancestral);

    nivel[1] = 0;
    dfs(1);

    for(int i=1;i<=n;i++) ancestral[i][0] = pai[i];

    for(int j=1;j<MAXL;j++){
        for(int i=1;i<=n;i++){
            ancestral[i][j] = ancestral[ancestral[i][j-1]][j-1];
        }
    }
}

int LCA(int u, int v){
    if(nivel[u] < nivel[v]) swap(u, v);
    //Sobe o nivel pra deixar igual em no max logn passos
    for(int i = MAXL - 1; i>=0;i--){
        if(nivel[u] - (1 << i) >= nivel[v]) u = ancestral[u][i];
    }

    if(u == v) return u;

    for(int i=MAXL-1;i>=0;i--){
        if(ancestral[u][i] != -1 and ancestral[u][i] != ancestral[v][i]){
            u = ancestral[u][i];
            v = ancestral[v][i];
        }
    }
    return pai[u];
}


int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> n;
    for(int i=0;i<n-1;i++){
        int a,b;
```

```
      cin >> a >> b;
      adj[a].pb(b);
      adj[b].pb(a);
   }

   build();

   //To pick path:
   // cout << nivel[u] + nivel[v] - 2*nivel[LCA(u,v)];

   return 0;
}
```

4.11 Min Cut

```
#include <bits/stdc++.h>
using namespace std;
const int N = 60;
const int INF = 1e9+10;
struct edge{
   int v1, v2, cap, flow;
   edge(int _v1, int _v2, int _cap, int _flow) : v1(_v1), v2(_v2), cap(_cap), flow(_flow) {}
};
vector<edge> ed;
vector<int> adj[N];
int dist[N], que[N], nxt[N], vstd[N];
int src, sink, cntE;
int group[N];
int n;
void add_edge(int u, int v, int cap){ // antes de adicionar qualquer aresta lembrar de zerar o
    contador(cntE)
   adj[u].push_back(cntE++);
   adj[v].push_back(cntE++);
   ed.push_back(edge(u, v, cap, 0));
   ed.push_back(edge(v, u, 0, 0));
}
int bfs(){
   memset(dist, -1, sizeof dist);
   dist[src] = 0;
   int head = 0, tail = 0;
   que[tail++] = src;
   while(head < tail && dist[sink] == -1){
```

```
      int v = que[head++];
      for(int i = 0; i < (int)adj[v].size(); i++){
         int id = adj[v][i];
         int to = ed[id].v2;
         if(dist[to] == -1 && ed[id].flow < ed[id].cap){
            que[tail++] = to;
            dist[to] = dist[v] + 1;
         }
      }
   }
   return dist[sink] != -1;
}
int dfs(int v, int flow) {
   if(!flow || v == sink)return flow;
   while(nxt[v] < (int)adj[v].size()) {
      int id = adj[v][nxt[v]];
      int to = ed[id].v2;
      if(dist[to] != dist[v] + 1){
         ++nxt[v];
         continue;
      }
      int curFlow = dfs(to, min(flow, ed[id].cap - ed[id].flow));
      if(curFlow) {
         ed[id].flow += curFlow;
         ed[id^1].flow -= curFlow;
         return curFlow;
      }
      ++nxt[v];
   }
   return 0;
}
int dinic() {
   int flow = 0;
   while(true){
      if(!bfs())break; // nao encontrou caminho ate o destino
      memset(nxt, 0, sizeof nxt);
      while(int curFlow = dfs(src, INF))flow+=curFlow;
   }
   return flow;
}
void startFlow(int tam){
   for(int i = 0; i < tam; i++)adj[i].clear();
```

```
   ed.clear();
   cntE = 0;
}
void minCut(int u){ /* todos as arestas que ligarem um vertice que foi visitado com um vertice
   nao visitado fazem parte do min cut*/
   vstd[u] = 1;
   for(int i = 0; i < (int)adj[u].size(); i++){
      int v = adj[u][i];
      int to = ed[v].v2;
      if(vstd[to])continue;
      if(ed[v].cap - ed[v].flow == 0)continue;
      else minCut(to);
   }
}
int main(){
   int m;
   src = 0;
   sink = 1;
   int u, v, cap;
   while(scanf("%d %d", &n, &m), n + m){
      startFlow(n);
      for(int i = 0; i < m; i++){
         scanf("%d %d %d", &u, &v, &cap);
         u--;v--;
         add_edge(u, v, cap);
         add_edge(v, u, cap);
      }
      dinic();
      memset(vstd, 0, sizeof vstd);
      memset(group, 0, sizeof group);
      minCut(src);
      for(int i = 0; i < (int)ed.size(); i+=4){ // lembrar que para cada edge adicionado tem o
          edge de volta
         u = ed[i].v1;
         v = ed[i].v2;
         if(vstd[u] != vstd[v])printf("%d %d\n",u + 1, v + 1);
      }
      printf("\n");
   }

   return 0;
}
```

4.12 SSC

```
#include <bits/stdc++.h>
using namespace std;
const int MAX = 100;
#define mset0(x) memset(x, 0, sizeof x)
#define mset1(x) memset(x, -1, sizeof x)
int dfs_l[MAX];
int dfs_n[MAX];
int vstd[MAX];
vector<int>S;
vector<int> adj[MAX];
int dfsCount, numSCC;

void dfs(int at){
   dfs_n[at] = dfs_l[at] = dfsCount++;
   S.push_back(at);
   vstd[at] = 1;

   for(int i = 0; i<adj[at].size(); i++){
      int to = adj[at][i];
      if(dfs_n[to] == -1){
         dfs(to);
      }
      if(vstd[to]){
         dfs_l[at] = min(dfs_l[at], dfs_l[to]);
      }
   }

   if(dfs_l[at] == dfs_n[at]){ // this is a root(start) of an SCC
      printf("SSC %d:", ++numSCC);
      while(true){
         int v = S.back();
         S.pop_back();
         vstd[v] = 0;
         printf(" %d", v);
         if(at == v)break;
      }
      printf("\n");

   }
}
int main(){
```

```
   int N, M, to;
   scanf("%d", &N);
   for(int i = 0; i<N; i++){
      scanf("%d", &M);
      adj[i].clear();
      for(int j = 0; j<M; j++){
         scanf("%d", &to);
         adj[i].push_back(to);
      }
   }
   mset0(dfs_l);
   mset1(dfs_n);
   mset0(vstd);
   dfsCount = numSCC = 0;
   S.clear();


   for(int i = 0; i<N; i++){
      if(dfs_n[i] == -1){
         dfs(i);
      }
   }

   return 0;
}
```

4.13 HLD

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5+10;
const int maxlg = 17;
int topch[maxn], sgtpos[maxn], idch[maxn], subsz[maxn], curch, par[maxn];
int dep[maxn];
int sgt[4*maxn], a[maxn], next[maxn];
int pos, n;
vector<pair<int, int> > edges;
vector<int>adj[maxn], costs[maxn], costs2[maxn];
void build(int op ,int id = 1, int lf = 0, int rg = n){
   if(lf == rg)sgt[id] = a[lf]; // change
   else{
```

```
      int md = (lf+rg)>>1;
      build(op, id<<1, lf, md);
      build(op, (id<<1)+1, md+1, rg);
      if(op)sgt[id] = max(sgt[id<<1],sgt[(id<<1)+1]); // change
      else sgt[id] = sgt[id<<1] + sgt[(id<<1)+1]; // change
   }
}
void update(int op ,int idx, int val, int lf = 0, int rg = n, int id = 1){
   if(lf == rg){ // change
      a[idx] = sgt[id] = val;
   }
   else{
      int md = (lf+rg)>>1;
      if(lf<=idx && idx <=md)update(idx, val, lf, md, id << 1);
      else update(op, idx, val, md+1, rg, 1+(id<<1));
      if(op)sgt[id] = max(sgt[id<<1],sgt[(id<<1)+1]); // change
      else sgt[id] = sgt[id<<1] + sgt[(id<<1)+1] ; // change
   }
}
int query(int op, int esq, int dir, int lf = 0, int rg = n, int id = 1){
   if(dir < lf || rg < esq)return 0;
   if(esq <= lf && rg<=dir)return sgt[id];
   int md = (lf+rg)>>1;
   int l = query(op, esq, dir, lf, md, id<<1);
   int r = query(op, esq, dir, md+1, rg, 1 + (id<<1));
   if(op) return max(l, r); // change
   else return l+r;
}
void dfs(int v, int p, int h = 0){ // dfs(root, root)
   dep[v] = h, subsz[v] = 1;
   for(int i = 0; i < adj[v].size(); i++)if(adj[v][i] != p){
      par[adj[v][i]] = v;
      dfs(adj[v][i], v, h+1);
      subsz[v]+=subsz[adj[v][i]];
   }
}
void hld(int cur, int cst = -1, int prev = -1){
   if(topch[curch] == -1)topch[curch] = cur;
   idch[cur] = curch, sgtpos[cur] = pos, a[pos++] = cst;
   int hvy = -1, ncost;
   for(int i = 0; i < adj[cur].size(); i++)if(adj[cur][i] != prev){
      if(hvy == -1 || subsz[hvy] < subsz[adj[cur][i]]){
```

```
            hvy = adj[cur][i];
            ncost = costs[cur][i];
        }
    }
    if(hvy != -1)hld(hvy,ncost,cur);
    for(int i = 0; i < adj[cur].size(); i++)if(adj[cur][i] != prev && adj[cur][i] != hvy){
        curch++;
        hld(adj[cur][i], costs[cur][i], cur);
    }
}
int query_hld(int op, int u, int v){
    int res = -1, k = -1;
    int res2 = 0;
    int topu, topv, chu, chv;
    chu = idch[u], chv = idch[v];
    while(chu != chv){
        topu = topch[chu];
        topv = topch[chv];
        if(dep[topu] < dep[topv])swap(u, v), swap(topu, topv), swap(chu, chv);
        if(op)res = max(res, query(op, sgtpos[topu], sgtpos[u]));
        else res2 +=query(op, sgtpos[topu], sgtpos[u]);
        u = par[topu];
        chu = idch[u];
    }
    if(u != v){
        if(dep[u] > dep[v])swap(u, v);
        if(op)res = max(res, query(op, sgtpos[u]+1, sgtpos[v]));
        else res2+= query(op, sgtpos[u]+1, sgtpos[v]);
    }
    return op?res:res2;
}
int mn;
vector<int>moch, mosz;
int pd[5*1000+1][1001];
int run(int idx, int cap){
    if(idx == mn)return 0;
    int &res = pd[idx][cap];
    if(res != -1)return res;
    else{
        res = run(idx+1, cap);
        if(cap >= mosz[idx])res = max(res, run(idx+1, cap-mosz[idx])+moch[idx]);
    }
```

```
    return res;
}
void solve(){
    int nn, k;
    scanf("%d %d", &nn, &k);
    for(int i = 1; i <=nn; i++){
        adj[i].clear();
        costs[i].clear();
        topch[i] = -1;
    }
    edges.clear();
    edges.push_back(make_pair(0, 0));
    curch = pos = 1;
    int u, v, w;
    for(int i = 1; i <nn; i++){
        scanf("%d%d%d", &u, &v, &w);
        adj[u].push_back(v);
        adj[v].push_back(u);
        costs[u].push_back(w);
        costs[v].push_back(w);
        costs2[u].push_back(1);
        costs2[v].push_back(1);
        edges.push_back(make_pair(u, v));
    }
    int root = 1;
    dfs(root, root);
    hld(root);
    n = pos;
    build(1);
    int q;
    scanf("%d", &q);
    mn = q;
    memset(pd, -1, sizeof pd);
    vector<pair<int, int> >qr;
    while(q--){
        int a, b;
        scanf("%d %d", &a, &b);
        qr.push_back(make_pair(a, b));
        moch.push_back(query_hld(1, a, b));
    }
    for(int i = 1; i <= nn; i++){
        for(int j = 0; j < costs[i].size(); j++)
```

```
        costs[i][j] = costs2[i][j];
   }
   for(int i = 1; i <=nn; ++i)topch[i] = -1;
   curch = pos = 1;
   root = 1;
   dfs(root, root);
   hld(root);
   n = pos;
   build(0);
   for(int i = 0; i < qr.size(); i++){
      int a = qr[i].first;
      int b = qr[i].second;
      mosz.push_back(query_hld(0, a, b)+1);
   }

   int ans = run(0, k);
   if(ans == 0)printf("-1\n");
   else printf("%d\n", ans);
}
int main(){
   solve();
   return 0;
}
```

4.14 MaxFlow

```
int adjMatrix[MAXN][MAXN], max_flow, flow, src, sink;
vi parent, adjList[MAXN];
int n,m;

void augment(int v, int minEdge){
   if(v == src){
      flow = minEdge;
   } else if(parent[v] != -1){
      augment(parent[v], min(minEdge, adjMatrix[parent[v]][v]));
      adjMatrix[parent[v]][v] -= flow;
      adjMatrix[v][parent[v]] += flow;
   }
}

int maxFlow(){
   max_flow = 0;
```

```
    while(1){
        flow = 0;
        bitset<MAXN> vis;
        vis[src] = true;
        queue<int> q;
        q.push(src);
        parent.assign(MAXN, -1);
        while(!q.empty()){
            int u = q.front();
            q.pop();
            if(u == sink) break;
            for(int j=0;j<sz(adjList[u]);j++){
                int v = adjList[u][j];

                if(adjMatrix[u][v] > 0 and !vis[v]){
                    vis[v] = true;
                    q.push(v);
                    parent[v] = u;
                }
            }
        }
        augment(sink, INF);
        if(flow == 0) break;
        max_flow += flow;
    }
    return max_flow;
}

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> n >> m;
    for(int i=0;i<m;i++){
        int a,b,c;
        cin >> a >> b >> c;
        adjList[a].pb(b);
        adjList[b].pb(a);
        adjMatrix[a][b] = c;
        adjMatrix[b][a] = c;
    }
```

```
   src = 1, sink = n;
   cout << maxFlow() << endl;

   return 0;
}
```

4.15 Detect and Take Cycle

```
void dfs(int u){ // memset pai -1
   // if(sz(cic)) return ;
   vis[u] = 1;
   for(int i=0;i<sz(vec[u]);i++){
      int v = vec[u][i];
      // if(sz(cic)) return ;
      if(!vis[v]) pai[v] = u, dfs(v);
      if(vis[v] == 1){
         int k = u;
         if(u != pai[v]){
         cout << "--- Cycle ---" << endl;
            cout << u << " -- " << v << endl;
            cic.pb({u,v});
            while(v != pai[k]){
               cic.pb({pai[k], k});
               cout << pai[k] << " - " << k << endl;
               k = pai[k];
            }
            cout << pai[k] << " - " << k << endl;
            cic.pb({pai[k], k});
            // return ;
         }
      }
   }
   vis[u] = 2;
}

bool TopologicalSort(int a, int b){ //If aa != n tem ciclo
   int gg[501];
   queue<int> q;
   for(int i=1;i<=n;i++) gg[i] = grau[i];
   for(int i=1;i<=n;i++){
      if(i == b) gg[i]--;
      if(!gg[i]) q.push(i);
```

```
   }
   int aa = 0;
   while(!q.empty()){
      int u = q.front();
      q.pop();
      for(int i=0;i<sz(vec[u]);i++){
         int v = vec[u][i];
         if(a != u or v != b){
            gg[v]--;
            if(!gg[v]) q.push(v);
         }
      }
      aa++;
   }
   if(aa == n) return true;
   return false;
}
```

4.16 Kruskal

```
#include <bits/stdc++.h>
#define DEBUG if(0)
using namespace std;

struct t_aresta{
    double dis;
    double x, y;
};

bool comp(t_aresta a, t_aresta b){ return a.dis < b.dis; }

#define MAXN 200000
#define MAXM 200000

double n, m;
t_aresta aresta[MAXM];

double pai[MAXN];
double peso[MAXN];
```

```
t_aresta mst[MAXM];

double procurar(int x){
    if(pai[x] == x) return x;
    return pai[x] = procurar(pai[x]);
}

void join(int a, int b){

    a = procurar(a);
    b = procurar(b);
     if(peso[a] < peso[b]) pai[a] = b;
    else if(peso[b] < peso[a]) pai[b] = a;
    else{
        pai[a] = b;
        peso[b]++;
    }
}

void krusInit(){
    for(int i=0;i<20000;i++){
                    pai[i]=i;
                    peso[i]=0;
                    mst[i].x = 0;
                    mst[i].y = 0;
                    mst[i].dis = 0;
                    aresta[i].x = 0;
                    aresta[i].y = 0;
                    aresta[i].dis = 0;
    }
}

int main(){

    int n,m;

        cin >> n >> m;
            krusInit();

        for(int i=1;i<=m;i++){
            cin >> aresta[i].x >> aresta[i].y >> aresta[i].dis;
        }
```

```
            sort(aresta+1, aresta+m+1, comp);

            for(int i=1;i<=m;i++){
            }

            int contador=0;
            int tam2=0;

            for(int i = 1; i <= m ; i++){
                if( procurar(aresta[i].x) != procurar(aresta[i].y)){

                        join(aresta[i].x,aresta[i].y);
                        mst[tam2] = aresta[i];
                        contador+=mst[tam2].dis;

                        tam2++;
                }
            }
            cout << contador << endl;

    return 0;
}
```

4.17 Pontes Conexas

```
#include <bits/stdc++.h>
using namespace std;
#define DEBUG if(0)
#define MAX 2000
#define D cout << "DEBUG" << endl

int n, time_s, visit[MAX];
vector<int> ADJ[MAX];

int dfs(int u, int pai, vector<pair<int,int> >& ans){
   //cout << "U : " << u << endl;
   int menor = visit[u] = time_s++;
   int filhos = 0;

   for(int i=0;i<ADJ[u].size();i++){
      //cout << "Adj u i : " << ADJ[u][i] << endl;
```

```
      if(visit[ADJ[u][i]]==0){
      filhos++;
      int m = dfs(ADJ[u][i], u , ans);
      menor = min(menor,m);
         if(visit[u]<m){
            ans.push_back(make_pair(u,ADJ[u][i]));
         }
      }else if (ADJ[u][i]!=pai){
         menor = min(menor,visit[ADJ[u][i]]);
      }
   }
   return menor;
}

vector<pair<int,int> > get_articulacoes(){
   vector<pair<int,int> > ans;
   time_s = 1;
   memset(visit, 0 , n*sizeof(int));
   for(int i=0;i<n;i++){
      dfs(i,-1,ans);
   }
   return ans;
}

int main (){
   //int n;
   while(cin >> n){
      for(int i=0;i<MAX;i++){
         ADJ[i].clear();
      }
      cout << endl;
      for(int i=0;i<n;i++){
         int element,tam;
         char a;
            cin >> element >> a >> tam >> a;
               for(int j=0;j<tam;j++){
                  int t;
                     cin >> t;
                     ADJ[element].push_back(t);
               }
      }
```

```
    /*for(int i=0;i<n;i++){
       for(int j=0;j<ADJ[i].size();j++){
          cout << i << " - " << ADJ[i][j] << " ";
       } cout << endl;
    }*/

    vector<pair<int,int> > vec = get_articulacoes();

    /*for(int i=0;i<vec.size();i++){
       cout << "Vec first : " << vec[i].first << endl;
       cout << "Vec second : " << vec[i].second << endl;
    }*/

    sort(vec.begin(),vec.end());

    cout << vec.size() << " critical links" << endl;
    for(int i=0;i<vec.size();i++){
       if(vec[i].second<vec[i].first) swap(vec[i].second,vec[i].first);
       cout << vec[i].first << " - " << vec[i].second << endl;
    }
    if(!n) break;
  }


  return 0;
}
```

4.18 SCC

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

#define DEBUG if(1)
#define MAXN 50500
#define MAX 160005
#define MAXL 20
#define MIN -2000000
#define endl "\n"
#define INF 99999999
#define MOD 1000000007
#define s(n) scanf("%d", &n)
#define ss(a,b) scanf("%d %d",&a,&b)
```

```
#define pb push_back
#define mp make_pair
#define M_PI 3.14159265358979323846
#define sz(a) int(a.size())
#define lli long long int
#define rep(i,a,n) for (int i=a;i<n;i++)
#define ler(a,n,vec) for(int i=0;i<n;i++)s(a), vec.pb(a)
typedef vector<lli> vi;
typedef vector<vi> vvi;
typedef pair<int,int> ii;
typedef pair<string, pair<int, char> > ps;
#define F first
#define S second
int dx[] = {0, 1, 0, -1};
int dy[] = {1, 0, -1, 0};
int ddx[] = {1, 0};
int ddy[] = {1, 1};

int n, m;
vi vec[100001], vec2[100001];

lli c[100001];
int vis[100001];
// vi tempo;
stack<int> tempo;
vector<lli> aa;

void dfs(int x){
   vis[x] = 1;
   for(int i=0;i<sz(vec[x]);i++){
      int u = vec[x][i];
      if(!vis[u]) dfs(u);
   }
   // tempo.pb(x);
   tempo.push(x);
}

void dfs2(int x){
   vis[x] = 1;
   aa.pb(c[x]);
   for(int i=0;i<sz(vec2[x]);i++){
      int u = vec2[x][i];
```

```
      if(!vis[u]) dfs2(u);
   }
}


int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   cin >> n;
   // for(int i=1;i<=n;i++) scanf("%lld", &c[i]);
   for(int i=1;i<=n;i++) cin >> c[i];
   cin >> m;
   for(int i=0;i<m;i++){
      int a,b;
      cin >> a >> b;
      // scanf("%d %d", &a, &b);
      vec[a].pb(b);
      vec2[b].pb(a);
   }

   for(int i=1;i<=n;i++) if(!vis[i]) dfs(i);

   memset(vis, 0, sizeof vis);
   lli ans = 0, ans2 = 1;
   while(sz(tempo)){
      int u = tempo.top();
      // cout << "U : " << u << endl;
      tempo.pop();
      if(vis[u]) continue;
      dfs2(u);
      // sort(aa.begin(), aa.end());
      // bb.pb(aa);
      int aux2 = 0;
      lli cc = 99999999999999999;
      for(int i=0;i<sz(aa);i++) if(aa[i] < cc) cc = aa[i];
      for(int i=0;i<sz(aa);i++) if(aa[i] == cc) aux2++;
      ans += cc;
      ans2 = (ans2 * aux2)%MOD;

      aa.clear();
   }
```

```
    // printf("%lld %lld\n", ans, ans2);
    cout << ans << " " << ans2 << endl;

    return 0;
}
```

5 Strings

5.1 Kmp

```
void table(string p){
   v[0] = v[1] = 0;
   int cur = 0;
   for(int j = 2; j < sz(p); j++){
      while(cur != 0 && p[cur] != p[j-1]) cur=v[cur];
      if(p[cur] == p[j-1]) cur++;
      v[j] = cur;
   }
}
bool kmp(string p, string text){
   table(p);
   int cur = 0;
   for(int j = 0; j <sz(text); j++){
      while(cur > 0 && p[cur] != text[j]) cur = v[cur];

      if(p[cur] == text[j])
         if(++cur == sz(p)) return true;

   }
   return false;
}
```

5.2 Rabin Karp

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9+7;
int base = 257;
vector<int>ans;
unsigned hash(const string& s){
    long long ret = 0;
    for (int i = 0; i < s.size(); i++)ret=(ret*base+ s[i])%mod;
    return ret;
}
int rabin_karp(const string& needle, const string& haystack){
    long long hash1 = hash(needle);
    long long hash2 = 0;
    long long power = 1;
```

```
    for (int i = 0; i < needle.size(); i++)power=(power*base)%mod;
    for (int i = 0; i < haystack.size(); i++){
        hash2 = hash2*base + haystack[i];
        hash2%=mod;
        if (i >= needle.size()){
            hash2 -= power*haystack[i-needle.size()]%mod;
            while(hash2 < 0)hash2+=mod;
        }
        if (i >= needle.size()-1 && hash1 == hash2)ans.push_back(i - (needle.size()-1));
    }
    if(ans.size())return 1;
    return 0;
}
int main(){
    string s = "auhas";
    string s2 = "asdauhasaeauhasuh";
    string s3 = "kkkkkkkkk";
    if(rabin_karp(s, s2)){
        printf("found\n");
        for(int i = 0; i < ans.size(); i++){
            printf("%d ", ans[i]);
        }
        printf("\n");
    }
    ans.clear();
    if(!rabin_karp(s, s3)){
        printf("not found\n");
    }
}
```

5.3 Suffix Array

```
#include <vector>
#include <iostream>
#include <cstdio>
#include <cstring>
#include <string>
#include <algorithm>
using namespace std;
// Begins Suffix Arrays implementation
// O(n log n) - Manber and Myers algorithm
// Refer to "Suffix arrays: A new method for on-line string searches",
```

```
// by Udi Manber and Gene Myers
//Usage:
// Fill str with the characters of the string.
// Call SuffixSort(n), where n is the length of the string stored in str.
// That's it!
//Output:
// pos = The suffix array. Contains the n suffixes of str sorted in lexicographical order.
//       Each suffix is represented as a single integer (the position of str where it starts).
// rank = The inverse of the suffix array. rank[i] = the index of the suffix str[i..n)
//        in the pos array. (In other words, pos[i] = k <==> rank[k] = i)
//        With this array, you can compare two suffixes in O(1): Suffix str[i..n) is smaller
//        than str[j..n) if and only if rank[i] < rank[j]

#define N 1000005
char str[N]; //input
int rank[N], pos[N]; //output
int cnt[N], next[N]; //internal
bool bh[N], b2h[N];
// Compares two suffixes according to their first characters
bool smaller_first_char(int a, int b) {
   return str[a] < str[b];
}
void suffixSort(int n) {
   //sort suffixes according to their first characters
   for (int i = 0; i < n; ++i) {
      pos[i] = i;
   }
   sort(pos, pos + n, smaller_first_char);
   //{pos contains the list of suffixes sorted by their first character}
   for (int i = 0; i < n; ++i) {
      bh[i] = i == 0 || str[pos[i]] != str[pos[i - 1]];
      b2h[i] = false;
   }
   for (int h = 1; h < n; h <<= 1) {
      //{bh[i] == false if the first h characters of pos[i-1] == the first h characters of
          pos[i]}
      int buckets = 0;
      for (int i = 0, j; i < n; i = j) {
         j = i + 1;
         while (j < n && !bh[j]) j++;
         next[i] = j;
         buckets++;
```

```
      }
      if (buckets == n)break; // We are done! Lucky bastards!
      //{suffixes are separted in buckets containing strings starting with the same h
          characters}

      for (int i = 0; i < n; i = next[i]) {
         cnt[i] = 0;
         for (int j = i; j < next[i]; ++j) {
            rank[pos[j]] = i;
         }
      }
      cnt[rank[n - h]]++;
      b2h[rank[n - h]] = true;
      for (int i = 0; i < n; i = next[i]) {
         for (int j = i; j < next[i]; ++j) {
            int s = pos[j] - h;
            if (s >= 0) {
               int head = rank[s];
               rank[s] = head + cnt[head]++;
               b2h[rank[s]] = true;
            }
         }
         for (int j = i; j < next[i]; ++j) {
            int s = pos[j] - h;
            if (s >= 0 && b2h[rank[s]]) {
               for (int k = rank[s] + 1; !bh[k] && b2h[k]; k++)
                  b2h[k] = false;
            }
         }
      }
      for (int i = 0; i < n; ++i) {
         pos[rank[i]] = i;
         bh[i] |= b2h[i];
      }
   }
   for (int i = 0; i < n; ++i) rank[pos[i]] = i;
}
// End of suffix array algorithm
// Begin of the O(n) longest common prefix algorithm
// Refer to "Linear-Time Longest-Common-Prefix Computation in Suffix
// Arrays and Its Applications" by Toru Kasai, Gunho Lee, Hiroki
// Arimura, Setsuo Arikawa, and Kunsoo Park.
```

```
int height[N];
// height[i] = length of the longest common prefix of suffix pos[i] and suffix pos[i-1]
// height[0] = 0
void getHeight(int n) {
   for (int i = 0; i < n; ++i) rank[pos[i]] = i;
   height[0] = 0;
   for (int i = 0, h = 0; i < n; ++i) {
      if (rank[i] > 0) {
         int j = pos[rank[i] - 1];
         while (i + h < n && j + h < n && str[i + h] == str[j + h])h++;
         height[rank[i]] = h;
         if (h > 0) h--;
      }
   }
}
// End of longest common prefixes algorithm
int find(char * p) {
   int l = 0, sz = strlen(p), h = strlen(str) - 1;
   while (l <= h) {
      int mid = l + (h - l) / 2;
      char * tmp = (char*)malloc(sz+1);
      strncpy(tmp, str + pos[mid], sz);
      tmp[sz] = 0;
      int cmp = strcmp(p, tmp);
      if (cmp < 0)h = mid - 1;
      if (cmp > 0)l = mid + 1;
      if (!cmp)return mid;
   }
   return -1;
}

int main() {
   string s;
   int T, q;
   char p[2000], tmp[200];
   scanf("%d", &T);
   gets(str);
   while (T--) {
      gets(str);
      scanf("%d", &q);
      gets(tmp);
      long long len = 0LL + strlen(str);
```

```
      suffixSort(len);
      while (q--) {
         gets(p);
         if (find(p) != -1)
            puts("y");
         else
            puts("n");
      }

   }
}
```

5.4 Rabin Karp Tiago

```
#define d 256
vector<int> ans;

void rabinKarp(string txt, string pat, int q){
   int M = sz(pat);
   int N = sz(txt);
   int p = 0, t = 0, h = 1, j;

   if(M > N) return ;

   for(int i=0;i<M-1;i++){
      h = (h*d)%q;
   }

   for(int i=0;i<M;i++){
      p = (d*p + pat[i])%q;
      t = (d*t + txt[i])%q;
   }

   for(int i=0;i<=N-M;i++){
      if(p == t){
         for(j=0;j<M;j++){
            if(txt[i+j] != pat[j]) break;
         }

         if(j == M)
            ans.pb(i);
```

```
      }

      if(i < (N - M)){
         t = (d*(t - txt[i]*h) + txt[i+M])%q;
         if(t < 0) t += q;
      }
   }
}

int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   int n;
   while(cin >> n){
      string a,b;
      cin >> a >> b;

      rabinKarp(b, a, 541);

      if(!sz(ans)) cout << endl;
      else {
         for(int i=0;i<sz(ans);i++) cout << ans[i] << endl;
      }
      ans.clear();
   }


   return 0;
}

// _--- -- -- --- -- -- -- --- -

/* Following program is a C implementation of Rabin Karp
Algorithm given in the CLRS book */
#include<stdio.h>
#include<string.h>

// d is the number of characters in the input alphabet
#define d 256

/* pat -> pattern
```

```
    txt -> text
    q -> A prime number
*/
void search(char pat[], char txt[], int q)
{
    int M = strlen(pat);
    int N = strlen(txt);
    int i, j;
    int p = 0; // hash value for pattern
    int t = 0; // hash value for txt
    int h = 1;

    // The value of h would be "pow(d, M-1)%q"
    for (i = 0; i < M-1; i++)
        h = (h*d)%q;

    // Calculate the hash value of pattern and first
    // window of text
    for (i = 0; i < M; i++)
    {
        p = (d*p + pat[i])%q;
        t = (d*t + txt[i])%q;
    }

    // Slide the pattern over text one by one
    for (i = 0; i <= N - M; i++)
    {

        // Check the hash values of current window of text
        // and pattern. If the hash values match then only
        // check for characters on by one
        if ( p == t )
        {
            /* Check for characters one by one */
            for (j = 0; j < M; j++)
            {
                if (txt[i+j] != pat[j])
                    break;
            }

            // if p == t and pat[0...M-1] = txt[i, i+1, ...i+M-1]
            if (j == M)
```

```
                printf("Pattern found at index %d \n", i);
        }

        // Calculate hash value for next window of text: Remove
        // leading digit, add trailing digit
        if ( i < N-M )
        {
            t = (d*(t - txt[i]*h) + txt[i+M])%q;

            // We might get negative value of t, converting it
            // to positive
            if (t < 0)
            t = (t + q);
        }
    }
}

/* Driver program to test above function */
int main()
{
    char txt[] = "GEEKS FOR GEEKS";
    char pat[] = "GEEK";
    int q = 101; // A prime number
    search(pat, txt, q);
    return 0;
}
```

5.5 Z Function

```
/*
Z algorithm (Linear time pattern searching Algorithm)
This algorithm finds all occurrences of a pattern in a text in linear time. Let length of text
    be n and of pattern be m, then total time taken is O(m + n) with linear space complexity.
    Now we can see that both time and space complexity is same as KMP algorithm but this
    algorithm is Simpler to understand.

In this algorithm, we construct a Z array.

What is Z Array?
For a string str[0..n-1], Z array is of same length as string. An element Z[i] of Z array
    stores length of the longest substring starting from str[i] which is also a prefix of
    str[0..n-1]. The first entry of Z array is meaning less as complete string is always prefix
```

```
    of itself.
*/
string s;
int z[1000001];
int n;

void zAlgo(){
  int l = 0, r = 0;
  for(int i=1;i<n;i++){
    if(l > r){
      l = r = i;
      while(r < n and s[r - l] == s[r]) r++;
      z[i] = r - l, r--;
    } else {
      int k = i - l;
      if(z[k] < r - i + 1) z[i] = z[k];
      else {
        l = i;
        while(r < n and s[r - l] == s[r]) r++;
        z[i] = r - l;
        r--;
      }
    }
  }
}

int main(){
  ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n;
  cin >> s;

  s += s;
  n = sz(s);

  zAlgo();

  for(int i=1;i<n;i++){
    int atual = z[i];

    if(i + atual >= n) continue;
```

```
    if(s[atual + i] < s[i]){
      cout << "No" << endl;
      return 0;
    }
  }

  cout << "Yes" << endl;

  return 0;
}
```

5.6 Double Hash

```
void solve(){
     int n = s.size();

    const lli p = 31;
    const lli m = MOD, mm = MOD2;
    vector<long long> p_pow(n), p_pow2(n);
    p_pow[0] = 1;
    p_pow2[0] = 1;
    for (lli i = 1; i < n; i++)
        p_pow[i] = (p_pow[i-1] * p) % m, p_pow2[i] = (p_pow2[i-1]*p)%mm;

    vector<long long> h(n + 1, 0), hh(n + 1, 0);
    for (lli i = 0; i < n; i++){
        h[i+1] = (h[i] + (s[i] - 'a' + 1) * p_pow[i]) % m;
      hh[i+1] = (hh[i] + (s[i] - 'a' + 1) * p_pow2[i]) %mm;
    }

    lli cnt = 0;
    for (lli l = k; l <= k; l++) {
      set<pair<lli,lli>>vec;
      set<lli> ans;
        for (lli i = 0; i <= n - l; i++) {
            lli cur_h = (h[i + l] + m - h[i]) % m;
            lli cur_h2 = (hh[i+l] + mm - hh[i])%mm;
            cur_h = (cur_h * p_pow[n-i-1]) % m;
            cur_h2 = (cur_h2 * p_pow2[n-i-1])%mm;
            vec.insert({cur_h, cur_h2});
        }
```

```
        for(auto o : vec) cout << o.F << " - " << o.S << endl;
        cout << sz(vec) << endl;
    }
}

int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   int t;
   cin >> t;
   while(t--){
      cin >> n >> k;
      cin >> s;

      // cout << compHash() << endl;
      solve();
   }

   return 0;
}
```

6 Math

6.1 Combinatorics

```
long long fac[N], ifac[N];
long long PowerMod(long long a, long long n){
    long long ret = 1;
    while (n){
        if (n & 1){
            ret *= a;
            ret %= MOD;
        }
        a *= a;
        a %= MOD;
        n /= 2;
    }
    return ret;
}
inline void pre(){
    int i;
    fac[0] = 1;
    for (i = 1; i < N; i++){
        fac[i] = (i * fac[i - 1]) % MOD;
    }
    ifac[N - 1] = PowerMod(fac[N - 1], MOD - 2);
    for (i = N - 2; i >= 0; i--){
        ifac[i] = ((i + 1) * ifac[i + 1]) % MOD;
    }
}
long long com(int n, int r){
    long long ret = fac[n];
    ret *= ifac[r];
    ret %= MOD;
    ret *= ifac[n - r];
    ret %= MOD;
    return ret;
}
```

6.2 Digits Count

```
int digitsCount(int n, int b){
    return (int)floor(1 + log10((double)n) / log10((double)b));
```

```
}
int digitsCount(int n){
    return (int)floor(1 + log10((double)n));
}
```

6.3 Fast Exp

```
int fastExp(int a, int b, int c) {
        int ans=1;
        while(b != 0) {
                if(b%2 == 1)ans=(ans*a)%c;
                a=(a*a)%c;
                b /= 2;
        }
        return ans;
}
```

6.4 Hex Base

```
string hexbase(long long num, long long b){
   string res;
   string d = "0123456789ABCDEF";
    while(num > 0){
        res = d[num % b] + res;
        num /= b;
    }
        return res;
}
```

6.5 Leap

```
bool leap(int yr){
   if(((yr % 4 == 0) && !(yr % 100 == 0))|| (yr % 400 == 0)) return true;
   else return false;
}
```

6.6 Matrix Exp

```
typedef vector<vector<long long> >matrix;
```

```
matrix mul(const matrix &A, const matrix &B){ // A*B
   matrix C(n, vector<long long>(n));
   for(int i = 0; i < n; i++){
      for(int j = 0; j < n; j++){
         C[i][j] = 0;
         for(int k = 0; k < n; k++){
            C[i][j]+=A[i][k]*B[k][j];
 //if overflow C[i][j]+= ((A[i][k]%mod)*(B[k][j]%mod))%mod
         }
         C[i][j]%=mod;
      }
   }
   return C;
}
matrix mpow(matrix &A, long long power){ // A ^ power
   int K = n;
   matrix iden(n, vector<long long>(n));
   for(int i = 0; i < K; i++)
      for(int j = 0; j < K; j++)iden[i][j] = (i == j);
   while(power){
      if(power&1) iden = mul(iden, A);
      A = mul(A, A);
      power >>= 1;
   }
   return iden;
}
////////////////////////////////////////////////////////////
```

6.7 Moves

```
Cavalo
int cvX[]={-2,-1,1,2,2,1,-1,-2};
int cvY[]={1,2,2,1,-1,-2,-2,-1};
Cavalo 3D
int cvX[]={-2, -1, 1, 2, 2, 1, -1, -2, 0, 0, 1, -1, 0,  0,  1, -1, 0, 0, 2, -2, 0, 0,  2, -2};
int cvY[]={ 1, 2, 2, 1, -1, -2, -2, -1, 1, -1, 0, 0, 1, -1, 0, 0, 2, -2, 0, 0, 2, -2, 0,  0};
int cvZ[]={ 0, 0, 0, 0, 0, 0,  0,  0,  2,  2,  2,  2, -2, -2, -2, -2, 1, 1, 1, 1, -1, -1, -1, -1};
Rei Com Diagonal
int kx[]={1,-1,0,0,1,-1,1,-1};
int ky[]={0,0,1,-1,1,-1,-1,1};
mapa com andar sem Diagonal
int dx[]={1, -1, 0, 0, 0, 0}
```

```
int dy[]={0, 0, 1, -1, 0, 0}
int dz[]={0, 0, 0, 0, 1, -1}
```

6.8 Pascal

```
int c[MAX][MAX];
void pascalTriangle(){
   for(int i=0;i<MAX;i++) c[i][0] = 1, c[i][i] = 1;
   for(int i=1;i<MAX;i++) for(int j=1;j<i;j++) c[i][j] = c[i-1][j-1] + c[i-1][j];
}
```

6.9 Power of Two

```
bool isPowerOfTwo(int x){
    return (x && !(x & (x - 1)));
}
```

6.10 Ret in Ret

```
bool retInRet(int a, int b, int p, int q){
   if(a < b)swap(a, b);
   if(p < q)swap(p, q);
   if((p<=a && q<=b) || (p >a && b>= (2*p*q*a+((p*p -
       q*q))*sqrt(p*p+q*q-a*a))/(p*p+q*q)))return true;
   else{
      swap(a, p);
      swap(b, q);
      if((p<=a && q<=b) || (p >a && b>= (2*p*q*a+((p*p -
          q*q))*sqrt(p*p+q*q-a*a))/(p*p+q*q)))return true;
      return false;
   }
}
```

6.11 Sieve

```
ll _sieve_size;
bitset<10000010>bs;
vector<int>primes;
void sieve(ll upperbound){
```

```
   _sieve_size = upperbound+1;
   bs.set();
   bs[0] = bs[1] = 0;
   for(ll i = 2; i<= _sieve_size; i++)if(bs[i]){
      for(ll j = sqr(i); j<=_sieve_size; j+=i)bs[j] = 0;
      primes.push_back((int)i);
   }
}
bool isPrime(ll N){
   if( N <= _sieve_size) return bs[N];
   for(int i = 0; i<size(primes); i++)
      if(N%primes[i] == 0)return false;
   return true;
}
```

6.12 Bin Search

```
/*LE*/ while(l <= h){ m = (l+h) / 2; if(works(m)) l=m+1; else h=m-1; } return l-1;
/*GE*/ while(l <= h){ m = (l+h) / 2; if(works(m)) h=m-1; else l=m+1; } return h+1;
```

6.13 Aritmetica Modular

```
/*

    3.24 Aritmetica Modular

    Funcoes

        invMod(a,m)

            Devolve o inverso modular (a^(-1))%n do numero a

        powMod(x,k,m)

            Devolve a exponenciacao modular (x^k)%m

        sqrtMod(n,p)

            Devolve a raiz modular (sqrt(n))%p do numero n

        addMod(a,b,m)
```

```
            Devolve a soma modular (a+b)%m

        subMod(a,b,m)

            Devolve a subtracao modular (a+b)%m

        mulMod(a,b,m)

            Devolve a multiplicacao modular (a+b)%m

        divMod(a,b,m)

            Devolve a divisao modular (a+b)%m

            Numero%MOD + MOD)%MOD , para no dar problema com o negativo

*/

int extgcd(int a, int b, int &x, int &y)
{
    int g = a; x = 1; y = 0;
    if(b!=0) g = extgcd(b,a%b,y,x), y -= (a/b)*x;
    return g;
}



long long int invMod(long long int a, long long int m)
{
    int x, y;
    if (extgcd(a, m, x, y) == 1) return (x + m) % m;
    else return 0; //Nao Resolvivel
}

long long int powMod(long long int x,long long int k,long long int m)
{
    if(k==0) return 1;
    if(k%2==0) return powMod(x*x%m,k/2,m);
    else return x*powMod(x,k-1,m)%m;
}
```

```
int sqrtMod(int n, int p)
{
    int S, Q, W, i, m = invMod(n, p);

    for(Q=p-1,S=0;Q%2==0;Q/=2,++S);

    do{ W=rand()%p; } while(W== 0||jacobi(W,p)!=-1);

    for(int R = powMod(n,(Q+1)/2,p), V=powMod(W,Q,p);;)
    {
        int z=R*R*m%p;
        for(i=0;i<S&&z%p!=1;z*=z,++i);
        if(i==0) return R;
        R=(R*powMod(V,1<<(S-i-1),p))%p;
    }
}

long long int addMod(long long int x, long long int y,long long int n)
{
    return ((x % n) + (y % n)) % n;
}

int subMod(int x, int y, int n)
{
    return ((x % n) - (y % n)) % n;
}

long long int mulMod(long long int x,long long int y,long long int n)
{
    return ((x % n)*(y % n)) % n;
}

long long int divMod(long long int x,long long int y,long long int n)
{
    return mulMod(x,invMod(y,n),n);
}
```

6.14 Binomial e Modular

```
int a, b, n;

/*
```

```
   Coeficiente Binomial,
    s  serve para N at 20.
   Para calcularmos nCr mod p, onde p e primo
   Sendo nCr:
   n! / (r! * (n - r)!)
    Ns  calculamos o fatorial mod P e usamos
   modular inverse pra achar nCr mod p. Se ns
   temos que achar nCr mod M (onde m  no primo)
   pode-se fatorizar M em primos e usar teorema chines
   do resto
   e achar nCr mod m

   Isso acontece porque calcular:
   n! / (r! ( n - r)! ) % p d overflow mt facil
   Pela propriedade
   n/k = n* k^-1
   Usando o pequeno teorema de fermat:
   a^p = a (mod p) para P primo e a > 0, p > a
    ento  a ^ (p-2) = a ^ -1 (mod p), portanto, conseguimos
   achar a^(p-2)%p , facilmente com fast Expo

*/

lli C(lli n, lli r){
   lli f[n+1];
   f[0] = 1;

   for(int i=1;i<=n;i++){
      f[i] = i*f[i-1];
   }
   return f[n]/f[r]/f[n-r];
}

lli powMod(lli a, lli b, lli MOD2){
   lli x=1, y = a;
   while(b > 0){
      if(b%2==1){
         x = (x*y);
         if(x > MOD2) x%=MOD2;
      }
      y = (y*y);
      if(y > MOD2) y%=MOD2;
```

```
      b/=2;
   }
   return x;
}

/*
   Modular Multiplicative INverse
   Using Euler's Theorem
   a^(phi(m)) = 1 (mod m)
   a ^ (-1) = a^(m-2) (mod m)

*/

lli inverseEuler(lli n, lli MOD2){
   return powMod(n, MOD2-2, MOD2);
}

/*
   Coeficiente Binomial com Modulo
*/

bool check(int aa, int bb){
   lli res = (a*aa) + (b*bb);

   while(res > 0){
      int atual = res%10;
      res /= 10;
      if(atual != a and atual != b) return false;
   }
   return true;
}

lli C(lli n, lli r, lli MOD2){
   vector<lli> f(n+1, 1);
   //  D  pra otimizar o calculo do fatorial
   // Deixando todos pre calculados
   for(lli i = 2;i<=n;i++){
      f[i] = (f[i-1]*i)%MOD2;
   }
   return (f[n]*((inverseEuler(f[r], MOD2) * inverseEuler(f[n-r], MOD2))%MOD2))%MOD2;
}
```

```
int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

   cin >> a >> b >> n;
   int aa = n, bb = 0;
   lli ans = 0;
   while(aa >= 0){
      if(check(aa, bb)){
         ans += C(n, min(aa, bb), MOD);
         if(ans > MOD) ans%= MOD;
      }
      aa--;
      bb++;
   }

   cout << ans%MOD << endl;

   return 0;
}
```

6.15 Crivo

```
int at[MAX];
   int n;

void crivo(){
   at[1] = 1;

   for(int i=2;i<=n;i++){
      if(!at[i])
      for(int j=2;i*j<=n;j++){
         at[i*j] = 1;
      }
   }

}

int main(){
   cin >> n;

   crivo();
```

```
   // for(int i=1;i<=10;i++) cout << at[i] << endl;

   for(int i=2;i<=n;i++){
      if(at[i]==0) cout << i << " ";
   }
   cout << endl;

   return 0;
}
```

6.16 Divisores Sqrt N

```
int fun(int x){
   if(pd[x] != -1) return pd[x];
   int ans = 0;
   vi v;
   for(int i=1;i<=sqrt(x);i++){
      if(x%i==0){
         if(x/i == i) ans++;
         else ans++, v.pb(x/i);
      }
   }
   for(int i = sz(v)-1;i>=0;i--) ans++;
   // for(int i=1;i<=x;i++) if(x%i==0) ans++;
   // cout << x << " - " << ans << endl;
   return pd[x] = ans;
}
```

6.17 Divisores Sqrt$_3 N$

```
void crivo(int x, bool prime[], bool primesquare[], int aa[]){
   // bool primo[x+1];
   // bool primesquare[x+1];
   // int crivo[x];
   for(int i=2;i<=x;i++) prime[i] = true;
   for(int i=0;i<=(x*x+1);i++) primesquare[i] = false;

   prime[1] = false;
   for(int p = 2; p*p <= x;p++){
      if(prime[p] == true){
          for(int i = p*2; i<= x; i += p) prime[i] = false;
```

```
      }
   }
   int j = 0;
   for(int p=2;p<=x;p++){
      if(prime[p]){
         aa[j] = p;
         primesquare[p*p] = true;
         j++;
      }
   }
}

int countDivisors(int x){
   if(pd[x]!=-1) return pd[x];
   if(x == 1) return 1;
   bool prime[x+1], primesquare[x*x+1];
   int aa[x];
   crivo(x, prime, primesquare, aa);
   int ans = 1;
   for(int i=0;;i++){
      if(aa[i]*aa[i]*aa[i] > x) break;
      int cnt = 1;
      while(x%aa[i]==0){
         x = x/aa[i];
         cnt = cnt+1;
      }
      ans = ans*cnt;
   }
   if(prime[x]) ans = ans*2;
   else if(primesquare[x]) ans = ans*3;
   else if(x != 1) ans = ans*4;
   return pd[x] = ans;
}
```

6.18 Fast Exp

```
/*
   Modular Exponentiation
   x,y and p
   x^y) % p , ex : 2 3 5, 2^3%5 = 8%5 = 3
   Calculate in O(Log Y)
*/
```

```
int fexp(int x, int y){ //Calcula x^y em Log Y
   int ans = 1;
   while(y>0){
      if(y&1) ans*=x; //y&1 verifica se  impar
      y = y >> 1;
      x*=x;
   }
   return ans;
}

int modFexp(int x, int y, int p){ //Calcula x^y%p em Log Y
   int ans = 1;
   while(y > 0){
      if(y&1) ans = (ans*x) % p;
      y = y >> 1;
      x = (x*x) % p;
   }
   return ans;
}



int main(){

   int a,b,c;
   cin >> a >> b >> c;

   cout << fexp(a,b) << endl;
   cout << modFexp(a,b,c) << endl;

   return 0;
}
```

6.19 Gcd e Lcm

```
lli gcd(lli a, lli b){
   while(1){
      if(a == 0) return b;
      b%=a;
      if(b == 0) return a;
      a %= b;
```

```
   }
}

lli lcm(lli a, lli b){
   lli temp = gcd(a,b);
   return temp ? (a / temp*b) : 0;
}
```

6.20 Phi Function

```
int phi(int a){
   int res = a;
   for(int p=2;p*p<=a;p++){
      if(a%p == 0){
         while(a%p == 0) a/=p;
         res -= res/p;
      }
   }
   if(a > 1) res -= res/a;
   return res;
}

/*
   Quantidade de coprimos que existem em 'a'
   Coprimos   gcd(a, x) == 1
*/
```

6.21 Primality Test

```
bool isPrime(lli n){
    if (n <= 1) return false;
    if (n <= 3) return true;

    if (n%2 == 0 || n%3 == 0) return false;

    for (lli i=5; i*i<=n; i=i+6)
        if (n%i == 0 || n%(i+2) == 0)
           return false;

    return true;
}
```

7 D&C

7.1 closestPair

```
struct pt {
   int x, y, id;
};
inline bool cmp_x (const pt & a, const pt & b) {
   return a.x < b.x || a.x == b.x && a.y < b.y;
}
inline bool cmp_y (const pt & a, const pt & b) {
   return a.y < b.y;
}
pt a[MAXN];
double mindist;
int ansa, ansb;
inline void upd_ans (const pt & a, const pt & b) {
   double dist = sqrt ((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y) + .0);
   if (dist < mindist)
      mindist = dist, ansa = a.id, ansb = b.id;
}
void rec (int l, int r) {
   if (r - l <= 3) {
      for (int i=l; i<=r; ++i)
         for (int j=i+1; j<=r; ++j)
            upd_ans (a[i], a[j]);
      sort (a+l, a+r+1, &cmp_y);
      return;
   }
   int m = (l + r) >> 1;
   int midx = a[m].x;
   rec (l, m), rec (m+1, r);
   static pt t[MAXN];
   merge (a+l, a+m+1, a+m+1, a+r+1, t, &cmp_y);
   copy (t, t+r-l+1, a+l);

   int tsz = 0;
   for (int i=l; i<=r; ++i)
      if (abs (a[i].x - midx) < mindist) {
         for (int j=tsz-1; j>=0 && a[i].y - t[j].y < mindist; --j)
            upd_ans (a[i], t[j]);
         t[tsz++] = a[i];
      }
```

```
}
int main(){
   sort (a, a+n, &cmp_x);
   mindist = 1E20;
   rec (0, n-1);
}
```

8 Misc

8.1 Erase Digits, Lower Number

```
void insertInNonDecOrder(deque<char> &dq, char ch){
    if (dq.empty())
        dq.push_back(ch);

    else{
        char temp = dq.back();
        while( temp > ch && !dq.empty()){
            dq.pop_back();
            if (!dq.empty())
                temp = dq.back();
        }
        dq.push_back(ch);
    }
    return;
}

string buildLowestNumber(string str, int n){
    int len = str.length();
    int k = len - n;

    deque<char> dq;
    string res = "";

    int i;
    for (i=0; i<=len-k; i++)

        insertInNonDecOrder(dq, str[i]);

    while (i < len){
        res += dq.front();

        dq.pop_front();

        insertInNonDecOrder(dq, str[i]);
        i++;
    }

    res += dq.front();
    dq.pop_front();
```

```
    return res;
}

int main(){
   string s;
   int k;
   while(cin >> s >> k){
      cout << buildLowestNumber(s, k) << endl;
   }
    return 0;
}
```

8.2 Checa se a é substring de b

```
bool check(string a, string b){
   if(b.find(a) != std::string::npos) return true;
   return false;
}
```

8.3 String to long long, long long to string

```
lli n,m;
string::size_type sz = 0;

lli solve(lli x){
   std::string aux = to_string(x);
   // cout << "Aux : " << aux << endl;
   for(int i=aux.size()-1;i>=0;i--) aux+=aux[i];

   long long ans = stoll(aux, &sz, 0);
   // cout << ans << endl;
   return ans;
}

int main(){
   cin >> n >> m;
   vector<lli> ans;

   for(lli i=1;i<=n;i++){
      ans.pb(solve(i));
   }
```

```
    lli sum = 0;
    for(int i=0;i<sz(ans);i++) sum+=(ans[i]);

    cout << sum%m << endl;

    return 0;
}
```

8.4 String operations, "big"num

```
//SUB ENTRE STRINGS

bool isSmaller(string str1, string str2)
{
    // Calculate lengths of both string
    int n1 = str1.length(), n2 = str2.length();

    if (n1 < n2)
    return true;
    if (n2 < n1)
    return false;

    for (int i=0; i<n1; i++)
    if (str1[i] < str2[i])
        return true;
    else if (str1[i] > str2[i])
        return false;

    return false;
}

// Function for find difference of larger numbers
string findDiff(string str1, string str2)
{
    // Before proceeding further, make sure str1
    // is not smaller
    if (isSmaller(str1, str2))
        swap(str1, str2);

    // Take an empty string for storing result
    string str = "";
```

```
// Calculate length of both string
int n1 = str1.length(), n2 = str2.length();

// Reverse both of strings
reverse(str1.begin(), str1.end());
reverse(str2.begin(), str2.end());

int carry = 0;

// Run loop till small string length
// and subtract digit of str1 to str2
for (int i=0; i<n2; i++)
{
    // Do school mathematics, compute difference of
    // current digits

    int sub = ((str1[i]-'0')-(str2[i]-'0')-carry);

    // If subtraction is less then zero
    // we add then we add 10 into sub and
    // take carry as 1 for calculating next step
    if (sub < 0)
    {
        sub = sub + 10;
        carry = 1;
    }
    else
        carry = 0;

    str.push_back(sub + '0');
}

// subtract remaining digits of larger number
for (int i=n2; i<n1; i++)
{
    int sub = ((str1[i]-'0') - carry);

    // if the sub value is -ve, then make it positive
    if (sub < 0)
    {
        sub = sub + 10;
```

```
            carry = 1;
        }
        else
            carry = 0;

        str.push_back(sub + '0');
    }

    // reverse resultant string
    reverse(str.begin(), str.end());

    return str;
}

//SUM ENTRE STRINGS
string doSum(string a, string b)
{
    if(a.size() < b.size())
        swap(a, b);

    int j = a.size()-1;
    for(int i=b.size()-1; i>=0; i--, j--)
        a[j]+=(b[i]-'0');

    for(int i=a.size()-1; i>0; i--)
    {
        if(a[i] > '9')
        {
            int d = a[i]-'0';
            a[i-1] = ((a[i-1]-'0') + d/10) + '0';
            a[i] = (d%10)+'0';
        }
    }
    if(a[0] > '9')
    {
        string k;
        k+=a[0];
        a[0] = ((a[0]-'0')%10)+'0';
        k[0] = ((k[0]-'0')/10)+'0';
        a = k+a;
    }
    return a;
```

```
}
```

8.5 Enigma Final Br 2017

```
string s;
int k;
int n;

string ans;
int pd[1001][1000][9];

/*
   Engima (Final Brasileira 2017)
   dado uma string de at 1000 digitos, fazer um nmero que seja divisivel por k

   Primeiro:
      Notar que todo nmero 110 por exemplo,  feito por:
         10^2 * 1 + 10^1 * 1 + 10^0 * 0
      Com isso, d pra usar aritmetica modular pra pegar resto de nmeros grandes
   Usa digit dp e tenta formar o nmero gulosamente, a primeira resposta  a certa

*/

lli modFexp(lli x, lli y, lli p){ //Calcula x^y%p em Log Y
   int ans2 = 1;
   while(y > 0){
      if(y&1) ans2 = (ans2*x) % p;
      y = y >> 1;
      x = (x*x) % p;
   }
   return ans2;
}


int solve(int pos, int resto, int can){
   if(pos == sz(s)){
      if(!resto){
         cout << ans << endl;
         exit(0);
      }
      return !resto;
   }
```

```
   int &anss = pd[pos][resto][can];
   if(anss != -1) return anss;

   lli xx = modFexp(10LL, n - pos - 1, k);

   int a = 0;
   if(s[pos] == '?'){
      int atual = !can ? 1 : 0;
      for(int i=atual;i<=9;i++){
         ans[pos] = (i + '0');
         a += solve(pos+1, (resto + (xx * i)%k)%k, i >= 1 ? 1 : can);
      }
   } else {
      a += solve(pos+1, (resto + (xx * (s[pos] - '0'))%k)%k, s[pos] - '0' >= 1 ? 1 : can);
   }
   return anss = a;
}


int main(){
   ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> s >> k;
  n = sz(s);
  ans = s;
  memset(pd, -1, sizeof pd);
  solve(0,0,0);

  cout << "*" << endl;

    return 0;
}
```

8.6 Imperial Roads (LCA Query Path, Max Edge)

```
#define MAXN 200001
#define MAXL 20

/*
   Problema: Dada uma MST, falar a nova MST tal que
```

```
   tenha obrigatoriamente a aresta da query

   Caso a aresta j esteja na MST, a resposta  a MST
   Caso contrrio, pega a maior aresta no caminho do vrtice
   x  at  y, tira ela e inclui a nova aresta (x, y) Pois garante
   que  no  ter  ciclo,  j  que vc  s  pode ter 1 caminho entre cada
   par de  ns .

*/

int n, m, q;
vector<ii> vec[MAXN];
int pai[MAXN];
int peso[MAXN];
int vis[MAXN];

int ancestral[MAXN][MAXL];
int distancia[MAXN][MAXL];
int pai2[MAXN];
int nivel[MAXN];
int pw[MAXN];

map<ii, int> cam;
multiset<ii> have;

int proc(int x){
   if(pai[x] == x) return x;
   return pai[x] = proc(pai[x]);
}

void junta(int a, int b){
   a = proc(a);
   b = proc(b);

   if(peso[a] < peso[b]){
      pai[a] = b;
   } else if(peso[a] > peso[b]){
      pai[b] = a;
   } else {
      pai[a] = b;
      peso[b]++;
   }
```

```
}

void dfs(int x){
   vis[x] = 1;
   for(auto i : vec[x]){
      int y = i.S;
      if(vis[y]) continue;
      int xx = x, yy = y;
      if(xx > yy) swap(xx, yy);
      if(have.find({xx, yy}) == have.end()) continue;
      pai2[y] = x;
      nivel[y] = nivel[x] + 1;
      pw[y] = i.F;
      dfs(y);
   }
}

void build(){
   memset(ancestral, -1, sizeof ancestral);
   memset(pai2, -1, sizeof pai2);
   dfs(1);

   for(int i=1;i<=n;i++) ancestral[i][0] = pai2[i];
   for(int i=1;i<=n;i++) distancia[i][0] = pw[i];

   for(int j=1;j<MAXL;j++){
      for(int i=1;i<=n;i++){
         ancestral[i][j] = ancestral[ancestral[i][j-1]][j-1];
      }
   }

   for(int j=1;j<MAXL;j++){
      for(int i=1;i<=n;i++){
         distancia[i][j] = max(distancia[i][j-1], distancia[ancestral[i][j-1]][j-1]);
      }
   }

}

int LCA(int u, int v){
   if(nivel[u] < nivel[v]) swap(u, v);
```

```
   for(int i=MAXL-1;i>=0;i--){
      if(nivel[u] - (1 << i) >= nivel[v]){
         u = ancestral[u][i];
      }
   }

   if(u == v) return u;

   for(int i=MAXL-1;i>=0;i--){
      if(ancestral[u][i] != -1 and ancestral[u][i] != ancestral[v][i]){
         u = ancestral[u][i];
         v = ancestral[v][i];
      }
   }

   return pai2[u];
}

int LCAPath(int u, int v){
   if(nivel[u] < nivel[v]) swap(u, v);
   // int path = max(pw[u], pw[v]);
   int path = 0;
   // cout << "U : " << u << " ; " << pw[u] << endl;
   // cout << "V : " << v << " ; " << pw[v] << endl;
   for(int i=MAXL-1;i>=0;i--){
      if(nivel[u] - (1 << i) >= nivel[v]){
         path = max(path, distancia[u][i]);
         // path = max(path, pw[u]);
         u = ancestral[u][i];
         // path = max(path, pw[u]);
         // path = max(path, distancia[u][i]);
      }
   }
   // cout << u << " -- " << v << " : " << path << endl;
   if(u == v) return path;

   for(int i=MAXL-1;i>=0;i--){
      if(ancestral[u][i] != -1 and ancestral[u][i] != ancestral[v][i]){
         path = max({path, distancia[u][i], distancia[v][i]});
         // path = max(path, pw[u]);
         // path = max(path, pw[v]);
         path = max({path, distancia[u][i], distancia[v][i]});
```

```
        u = ancestral[u][i];
        v = ancestral[v][i];
        // path = max(path, pw[u]);
        // path = max(path, pw[v]);
      }
   }
   path = max({path, pw[u], pw[v]});
   // cout << u << " --- " << v << endl;
   // cout << "Pai : " << pai2[u] << " - " << pai2[v] << endl;
   return path;
}

int main(){
   ios_base::sync_with_stdio(false);
  cin.tie(NULL);

  cin >> n >> m;
  for(int i=1;i<=n;i++) pai[i] = i;

  vector< pair<int, ii> > edges;
  for(int i=0;i<m;i++){
   int a, b, c;
   cin >> a >> b >> c;
   vec[a].pb({c, b});
   vec[b].pb({c, a});
   cam[{a,b}] = c;
   cam[{b,a}] = c;
   edges.pb({c, {a, b}});
  }

  sort(edges.begin(), edges.end());

  lli ans = 0;
  for(int i=0;i<sz(edges);i++){
   int x = edges[i].S.F, y = edges[i].S.S;
   int w = edges[i].F;
   if(proc(x) != proc(y)){
      ans += w;
      if(x > y) swap(x, y);
      have.insert({x, y});
      junta(x, y);
   }
```

```
  }

  build();

  // for(auto i : have) cout << i.F << " - " << i.S << " - " << cam[{i.F, i.S}] << endl;

  // for(int i=1;i<=n;i++) cout << i << " - " << pai2[i] << " : " << pw[i] << endl;

  cin >> q;
  for(int i=0;i<q;i++){
   int a, b;
   cin >> a >> b;
   if(a > b) swap(a, b);
   if(have.find({a, b}) != have.end()) cout << ans << endl;
   else {
      // if(a > b) swap(a, b);
      // cout << "Query : " << a << " " << b << endl;
      // cout << "LCA : " << LCA(a, b) << endl;
      // cout << a << " - " << b << " : " << LCAPath(a, b) << endl;
      // cout << "Cam : " << cam[{a, b}] << endl;
      cout << ans - LCAPath(a,b) + cam[{a, b}] << endl;
   }
  }

   return 0;
}
```

8.7 Seg Tree Kadane

```
/*
   Longest sum subsequence array
   Querys
*/

struct Node{
   lli maxSum;
   lli preSum;
   lli sufSum;
   lli totalSum;
   Node(){
      preSum = sufSum = maxSum = totalSum = -INT_MAX;
   }
```

```
};

int n,m;
lli a[51000];
Node tree[150000];

//Merge left and right child
Node merge(Node a, Node b){
   Node aux;

   aux.totalSum = a.totalSum + b.totalSum;
   aux.maxSum = max({a.maxSum, a.sufSum + b.preSum, b.maxSum});
   aux.preSum = max(a.preSum, a.totalSum + b.preSum);
   aux.sufSum = max(a.sufSum + b.totalSum, b.sufSum);

   return aux;
}

void build(int node = 1, int start = 1 ,int end = n){
   if(start == end){
      tree[node].preSum = a[start];
      tree[node].sufSum = a[start];
      tree[node].totalSum = a[start];
      tree[node].maxSum = a[start];
   }  else {
      int mid = (start + end) >> 1;
      build(2*node, start, mid);
      build(2*node+1, mid+1, end);
      tree[node] = merge(tree[2*node], tree[2*node+1]);
   }
}

Node query(int l, int r, int node = 1, int start = 1, int end = n){
   if(start > r or end < l){
      Node aux;
      return aux;
   }
   if(start >= l and end <= r) return tree[node];

   int mid = (start + end) >> 1;
   Node p1 = query(l, r, 2*node, start, mid);
   Node p2 = query(l ,r, 2*node+1, mid+1, end);
```

```
   Node ans = merge(p1, p2);
   return ans;
}

int main(){
   ios_base::sync_with_stdio(false);
   cin.tie(NULL);

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

   cin >> m;

   build();

   for(int i=0;i<m;i++){
      int a, b;
      cin >> a >> b;
      Node aux = query(a, b);
      cout << aux.maxSum << endl;
   }

   return 0;
}
```

8.8 Line Intersection

```
#define pdd pair<double, double>

int n;
pdd x1, x2;
vector<int> vec[150];
pdd graph[150];
int vis[150];

int dp[150];

pdd lineIntersection(pdd A, pdd B, pdd C, pdd D){
   double a1 = B.S - A.S;
   double b1 = A.F - B.F;
   double c1 = a1*(A.F) + b1*A.S;
```

```
    double a2 = D.S - C.S;
    double b2 = C.F - D.F;
    double c2 = a2*C.F + b2*C.S;

    double det = a1*b2 - a2*b1;
    if(det == 0){
        return {FLT_MAX, FLT_MAX};
    } else {
        double x = (b2*c1 - b1*c2)/det;
        double y = (a1*c2 - a2*c1)/det;
        return {x,y};
    }
}

bool check(pdd a, pdd b){
    pdd aa = lineIntersection(a, x1, b, x2);
    pdd bb = lineIntersection(a, x2, b, x1);
    if(a.S <= b.S){
        if(aa.S > a.S and bb.S > a.S) return true;
    } else {
        if(aa.S > b.S and bb.S > b.S) return true;
    }
    return false;
}

int dfs(int x, int a){
    if(dp[x] != -1) return dp[x];

    int aa = 1;
    vis[x] = 1;
    for(int i=0;i<sz(vec[x]);i++){
        int u = vec[x][i];
        aa = max(aa, dfs(u, a) + 1);
    }
    return dp[x] = aa;
}

bool comp(pdd a, pdd b){
    return a.S > b.S;
}

int main(){
```

```
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    double aa, bb;
    cin >> n;
    cin >> aa >> bb;

    memset(dp, -1, sizeof dp);

    x1 = {aa, 0}, x2 = {bb, 0};

    for(int i=0;i<n;i++){
        double a,b;
        cin >> a >> b;
        graph[i] = {a,b};
    }

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(i == j) continue;
            if(graph[i].S > graph[j].S){
                if(check(graph[i], graph[j])) vec[i].pb(j);
            }
        }
    }

    int ans = 0;
    for(int i=0;i<n;i++){
        ans = max(ans, dfs(i, 1));
    }

    cout << ans << endl;

    return 0;
}
```

8.9 Polar Sort and Cover Angle

```
vector<ii> vec, aux;
int xa, xb, xc, xd, ya, yb, yc, yd;
long double ans = 360;
```

```
/*
   DADO N PARES DE COORDENADAS
   Achar o angulo minimo que cubra todas as coordenadas
   comp:  ordenao  clockwise coordenadas
   https://stackoverflow.com/questions/6989100/sort-points-in-clockwise-order?utm_medium=organic&utm_source=g

*/

bool comp(ii &a, ii &b){
   if(a.F >= 0 and b.F < 0) return true;
   if(a.F < 0 and b.F >= 0) return false;
   if(a.F == 0 and b.F == 0){
      if(a.S >= 0 or b.S >= 0) return a.S > b.S;
      return b.S > a.S;
   }
   int det = (a.F * b.S) - (b.F * a.S);
   if(det < 0) return true;
   if(det > 0) return false;
   int d1 = (a.F * a.F) + (a.S*a.S);
   int d2 = (b.F * b.F) + (b.S*b.S);
   return d1 > d2;
}

long double takeAngle(int x, int y, int xx, int yy){

   long double aa = x*xx + y*yy;
   long double bb = x*yy - y*xx;
   long double angle = atan2(bb, aa);
   return angle*180/M_PI;
}

int main(){
   ios_base::sync_with_stdio(false);
   int n;
   cin >> n;
   if(n == 1){
      cout << "0.0000000" << endl;
      return 0;
   }
   for(int i=0;i<n;i++){
      int a,b;
      cin >> a >> b;
```

```
      aux.pb({a,b});
   }

   sort(aux.begin(), aux.end(), comp);

   ans = 0;
   long double aux2 = takeAngle(aux[sz(aux) - 1].F, aux[sz(aux) - 1].S, aux[0].F, aux[0].S);
   aux2 = abs(aux2);
   long double maxi = aux2;

   for(int i=0;i<sz(aux)-1;i++){
      long double aa = takeAngle(aux[i].F, aux[i].S, aux[i+1].F, aux[i+1].S);
      aa = abs(aa);

      maxi = max(maxi, aa);
      aux2 += aa;
   }
   ans = aux2 - maxi;
   printf("%.10Lf\n", abs(ans));

   return 0;
}
```

8.10 Line Intersection

```
#define pdd pair<double, double>

int n;
pdd x1, x2;
vector<int> vec[150];
pdd graph[150];
int vis[150];

int dp[150];

pdd lineIntersection(pdd A, pdd B, pdd C, pdd D){
   double a1 = B.S - A.S;
   double b1 = A.F - B.F;
   double c1 = a1*(A.F) + b1*A.S;

   double a2 = D.S - C.S;
   double b2 = C.F - D.F;
```

```
    double c2 = a2*C.F + b2*C.S;

    double det = a1*b2 - a2*b1;
    if(det == 0){
        return {FLT_MAX, FLT_MAX};
    } else {
        double x = (b2*c1 - b1*c2)/det;
        double y = (a1*c2 - a2*c1)/det;
        return {x,y};
    }
}

bool check(pdd a, pdd b){
    pdd aa = lineIntersection(a, x1, b, x2);
    pdd bb = lineIntersection(a, x2, b, x1);
    if(a.S <= b.S){
        if(aa.S > a.S and bb.S > a.S) return true;
    } else {
        if(aa.S > b.S and bb.S > b.S) return true;
    }
    return false;
}

int dfs(int x, int a){
    if(dp[x] != -1) return dp[x];

    int aa = 1;
    vis[x] = 1;
    for(int i=0;i<sz(vec[x]);i++){
        int u = vec[x][i];
        aa = max(aa, dfs(u, a) + 1);
    }
    return dp[x] = aa;
}

bool comp(pdd a, pdd b){
    return a.S > b.S;
}

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
```

```
    double aa, bb;
    cin >> n;
    cin >> aa >> bb;

    memset(dp, -1, sizeof dp);

    x1 = {aa, 0}, x2 = {bb, 0};

    for(int i=0;i<n;i++){
        double a,b;
        cin >> a >> b;
        graph[i] = {a,b};
    }

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            if(i == j) continue;
            if(graph[i].S > graph[j].S){
                if(check(graph[i], graph[j])) vec[i].pb(j);
            }
        }
    }

    int ans = 0;
    for(int i=0;i<n;i++){
        ans = max(ans, dfs(i, 1));
    }

    cout << ans << endl;

    return 0;
}
```