Template for XCPC

SUNCHAOYI

2025-2026

Contents

1 Compile

1.1 Fast I/O

```cpp
#include <bits/stdc++.h>
#define init(x) memset (x,0,sizeof (x))
#define ll long long
#define ull unsigned long long
#define INF 0x3f3f3f3f
#define pii pair <int,int>
using namespace std;
const int MAX = 1e5 + 5;
const int MOD = 1e9 + 7;
char s[200];
inline int read ();
inline void output ();
int main ()
{
  //freopen (".in","r",stdin);
  //freopen (".out","w",stdout);
  return 0;
}
inline int read ()
{
    int s = 0;int f = 1;
    char ch = getchar ();
    while ((ch < '0' || ch > '9') && ch != EOF)
    {
        if (ch == '-') f = -1;
        ch = getchar ();
    }
    while (ch >= '0' && ch <= '9')
    {
        s = s * 10 + ch - '0';
        ch = getchar ();
    }
    return s * f;
}
inline void output (int x)
{
    if (x < 0) putchar ('-');
    x = (x > 0) ? x : -x;
    int cnt = 0;
    while (x)
    {
        s[cnt++] = x % 10 + '0';
        x /= 10;
    }
    while (cnt) putchar (s[--cnt]);
}
```

1.2 Run.bash

```bash
#!/bin/bash
g++ -std=c++17 -O2 -Wall "$1" -o main
./main < in.txt > out.txt
```

1.3 Run.ps1

```powershell
# new file : `type nul > filename.cpp`
# run : `.\run.ps1 filename.cpp`
g++ -std=c++17 -O2 -Wall $args[0] -o main
cat in.txt | .\main | Out-File -FilePath out.txt
```

2 Graph

2.1 Diameter of a Tree

```
template <typename T>
class Diameter
{
    int n,p;
    vector <T> dis;
    vector <vector <pair <int,T>>> ve;
    void dfs (int u,int fa)
    {
        for (const auto& [v,w] : ve[u])
        {
            if (v == fa) continue;
            dis[v] = dis[u] + w;
            if (dis[v] > dis[p]) p = v;
            dfs (v,u);
        }
    }

    public:
    Diameter (int n) : n(n),ve(n + 1) {}
    void add (int u,int v,T w = 1) {ve[u].push_back ({v,w});ve[v].push_back ({u,w
       });}
    T calc ()
    {
        dis.assign (n + 1,0);
        p = 1;dfs (1,0);
        dis[p] = 0;dfs (p,0);
        return dis[p];
    }
};
```

2.2 Centroid of a Tree

```
class Centroid
{
    int n;
    vector <int> sz,w,cen;
    vector <vector <int>> ve;
    void dfs (int u,int fa)
    {
        sz[u] = 1;w[u] = 0;
        for (auto v : ve[u])
        {
            if (v == fa) continue;
            dfs (v,u);
            sz[u] += sz[v];
            w[u] = max (w[u],sz[v]);
        }
        w[u] = max (w[u],n - sz[u]);
        if (w[u] <= n / 2) cen.push_back (u);
    }

    public:
    Centroid (int n) : n(n),ve (n + 1),sz (n + 1),w (n + 1) {}
    void add (int u,int v) {ve[u].push_back (v);ve[v].push_back (u);}
    vector <int> calc ()
    {
        cen.clear ();
        dfs (1, 0);
        sort (cen.begin (),cen.end ());
```

```
        return cen;
    }
};
```

2.3 Minimum Spanning Tree

2.3.1 Prim

```
template <typename T>
class MST
{
    int n;T ans;
    vector <int> vis;
    vector <vector <int>> g;
    vector <T> dis;

    public :
    MST (int n) : n (n),vis (n + 1,0),g (n + 1,vector <int> (n + 1,INF)),dis (n +
         1,INF) {dis[1] = ans = 0;vis[1] = 1;}
    void add (int u,int v,T w) {g[u][v] = g[v][u] = w;}
    T calc ()
    {
        for (int i = 2;i <= n;++i) dis[i] = g[1][i];
        for (int i = 1;i < n;++i)
        {
            int k = 0;
            for (int j = 1;j <= n;++j)
                if (!vis[j] && dis[j] < dis[k]) k = j;
            vis[k] = 1;
            ans += dis[k];
            for (int j = 1;j <= n;++j)
                if (!vis[j] && g[k][j] < dis[j]) dis[j] = g[k][j];
        }
        return ans;
    }
};
```

2.3.2 Kruskal

```
template <typename T>
class MST
{
    int n,m,e_cnt,cnt;T ans;
    struct node
    {
        int u,v;T w;
    };
    vector <int> fa;
    vector <node> g;
    public:
    MST (int n,int m) : n (n),m(m),fa (n + 1,0),g (m + 1) {cnt = e_cnt = ans =
        0;}
    void add (int u,int v,int w) {g[++e_cnt].u = u,g[e_cnt].v = v,g[e_cnt].w = w
        ;}
    int getfa (int u) {return fa[u] == u ? u : fa[u] = getfa (fa[u]);}
    T calc ()
    {
        sort (g.begin (),g.end (),[] (auto &x,auto &y) {return x.w < y.w;});
        for (int i = 1;i <= n;++i) fa[i] = i;
        for (int i = 1;cnt < n && i <= m;++i)
        {
            int dx = getfa (g[i].u),dy = getfa (g[i].v);
            if (dx == dy) continue;
            ans += g[i].w;fa[dx] = dy;++cnt;
```

```
        }
        return ans;
    }
};
```

2.4 LCA

```
class LCA
{
    static constexpr int lg = 20;
    int n;
    vector <int> dep;
    vector <vector <int>> f,ve;

    public:
    LCA (int n) : n (n),ve (n + 1),dep (n + 1),f (n + 1,vector <int> (lg + 1,0))
        {}
    void add (int u,int v) {ve[u].push_back (v);ve[v].push_back (u);}
    void pre (int u,int fa)
    {
        f[u][0] = fa;dep[u] = dep[fa] + 1;
        for (int i = 0;i < lg;++i) f[u][i + 1] = f[f[u][i]][i];
        for (auto v : ve[u])
            if (v != fa) pre (v,u);
    }
    int query (int u,int v)
    {
        if (dep[u] < dep[v]) swap (u,v);
        for (int i = lg;~i;--i)
        {
            if (dep[f[u][i]] >= dep[v]) u = f[u][i];
            if (u == v) return u;
        }
        for (int i = lg;~i;--i)
            if (f[u][i] != f[v][i]) u = f[u][i],v = f[v][i];
        return f[u][0];
    }
};
```

2.5 Topological Sorting

```
class Topo
{
    int n;
    vector <int> deg;
    vector <vector <int>> ve;

    public:
    Topo (int n) : n (n),ve (n + 1),deg (n + 1,0) {}
    void add (int u,int v)
    {
        ve[u].push_back (v);
        ++deg[v];
    }
    vector <int> calc ()
    {
        queue <int> q;
        vector <int> lst;
        for (int i = 1;i <= n;++i)
            if (!deg[i]) q.push (i);
        while (!q.empty ())
        {
            int u = q.front ();q.pop ();
            lst.push_back (u);
```

```
            for (auto v : ve[u])
            {
                --deg[v];
                if (!deg[v]) q.push (v);
            }
        }
        return lst;
    }
};
```

2.6 Shortest Path

2.6.1 Dijstrka

```
class dijkstra
{
    int n,m,cnt;
    vector <int> head,to,nxt,val,vis;
    vector <ll> dis;

    public:
    dijkstra (int n,int m) :
        n (n),m (m),vis (n + 1,0),head (n + 1,0),dis (n + 1,INF),
        to (2 * m + 1,0),nxt (2 * m + 1,0),val (2 * m + 1,0) {cnt = 0;}
    void add (int u,int v,int w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = w;nxt[cnt] = head[v];head[v] = cnt;
    }
    vector <ll> calc (int s)
    {
        priority_queue <pii> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,dis[i] = INF;
        q.push ({0,s});
        dis[s] = 0;
        while (!q.empty ())
        {
            int u = q.top ().second;q.pop ();
            if (vis[u]) continue;
            vis[u] = 1;
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
                if (dis[v] > dis[u] + val[i])
                {
                    dis[v] = dis[u] + val[i];
                    q.push ({-dis[v],v});
                }
            }
        }
        return vector <ll> (dis.begin () + 1,dis.end ());
    }
};
```

2.6.2 SPFA

```
class SPFA
{
    int n,m,cnt;
    vector <int> head,to,nxt,val,vis,times;
    vector <ll> dis;

    public:
    SPFA (int n,int m) :
```

```
        n (n),m (m),times (n + 1,0),vis (n + 1,0),head (n + 1,0),dis (n + 1,INF),
        to (2 * m + 1,0),nxt (2 * m + 1,0),val (2 * m + 1,0) {cnt = 0;}
    void add (int u,int v,int w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = w;nxt[cnt] = head[v];head[v] = cnt;
    }
    vector <ll> calc (int s)
    {
        queue <int> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,dis[i] = INF;
        dis[s] = 0,vis[s] = 1;q.push (s);
        while (!q.empty())
        {
            int u = q.front ();
            q.pop (),vis[u] = 0;
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
                if (dis[v] > dis[u] + val[i])
                {
                    dis[v] = dis[u] + val[i];
                    times[v] = times[u] + 1;
                    if (times[v] >= n) return {-1};//Negative Cycle
                    if (!vis[v]) q.push (v),vis[v] = 1;
                }
            }
        }
        return vector <ll> (dis.begin () + 1,dis.end ());
    }
};
```

2.7 Tarjan

```
class Tarjan
{
    int n,m,cnt,times,scc_cnt;
    vector <int> head,to,nxt,val,vis,low,scc,dfn;
    stack <int> s;
    void tarjan (int u)
    {
        low[u] = dfn[u] = ++times;
        s.push (u);
        for (int i = head[u];i;i = nxt[i])
        {
            int v = to[i];
            if (!dfn[v])
            {
                tarjan (v);
                low[u] = min (low[u],low[v]);
            }
            else if (!vis[v]) low[u] = min (low[u],dfn[v]);
        }
        if (low[u] == dfn[u])
        {
            ++scc_cnt;
            while (1)
            {
                int x = s.top ();s.pop ();
                scc[x] = scc_cnt;
                if (x == u) break;
            }
        }
    }
```

```
    public:
    Tarjan (int n,int m) :
        n (n),m (m),vis (n + 1,0),head (n + 1,0),low (n + 1,0),dfn (n + 1,0),scc
            (n + 1,0),
        to (2 * m + 1,0),nxt (2 * m + 1,0) {cnt = times = scc_cnt = 0;}
    void add (int u,int v) // Note that the bidirectional edges
    {
        to[++cnt] = v;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;nxt[cnt] = head[v];head[v] = cnt;
    }
    vector <int> calc ()
    {
        for (int i = 1;i <= n;++i)
            if (!dfn[i]) tarjan (i);
        return vector <int> (scc.begin () + 1,scc.end ());
    }
};
```

2.8 Bipartite Graph Matchings

```
class Matching
{
    int l,r;//the number of left/right side points
    vector <vector <int>> ve;
    vector <int> vis,op;
    bool dfs (int u)
    {
        for (auto v : ve[u])
        {
            if (vis[v]) continue;
            vis[v] = 1;
            if (!op[v] || dfs (op[v])) {op[v] = u;return true;}
        }
        return false;
    }

    public:
    Matching (int l,int r) : l (l),r (r),vis (r + 1,0),op (r + 1,0),ve (l + 1) {}
    void add (int u,int v) {ve[u].push_back (v);}
    int calc ()
    {
        int ans = 0;
        for (int i = 1;i <= l;++i)
        {
            vis.assign (r + 1,0);
            if (dfs (i)) ++ans;
        }
        return ans;
    }
};
```

2.9 Flow

2.9.1 Edmonds–Karp

```
template <typename T>
class EK
{
    int n,m,s,t,cnt;
    vector <int> head,to,nxt,vis,pre,edge;
    vector <T> val;
    bool bfs ()
    {
```

```
        queue <T> q;
        for (int i = 1;i <= n;++i) vis[i] = 0,pre[i] = edge[i] = -1;
        vis[s] = 1;q.push (s);
        while (!q.empty ())
        {
            int u = q.front ();q.pop ();
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
                if (!vis[v] && val[i])
                {
                    pre[v] = u;edge[v] = i;vis[v] = 1;
                    q.push (v);
                    if (v == t) return 1;
                }
            }
        }
        return 0;
    }

    public :
    EK (int n,int m,int s,int t) :
        n (n),m (m),s (s),t (t),
        vis (n + 1,0),head (n + 1,0),pre (n + 1,-1),edge (n + 1,-1),
        to (m + 1,0),nxt (m + 1,0),val (m + 1,0) {cnt = 1;}
    void add (int u,int v,T w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = 0;nxt[cnt] = head[v];head[v] = cnt;
    }
    T calc ()
    {
        T ans = 0;
        while (bfs ())
        {
            T mn = INF;
            for (int i = t;i != s;i = pre[i]) mn = min (mn,val[edge[i]]);
            for (int i = t;i != s;i = pre[i]) val[edge[i]] -= mn,val[edge[i] ^ 1]
                 += mn;
            ans += mn;
        }
        return ans;
    }
};
```

2.9.2 Dinic

```
template <typename T>
class Dinic
{
    int n,m,s,t,cnt;
    vector <int> head,to,nxt,cur,dep;
    vector <T> val;
    int bfs ()
    {
        for (int i = 0;i <= n;++i) dep[i] = 0,cur[i] = head[i];
        queue <int> q;
        q.push (s),dep[s] = 1;
        while (!q.empty ())
        {
            int u = q.front ();q.pop ();
            for (int i = head[u];i;i = nxt[i])
            {
                int v = to[i];
```

```
                if (val[i] && !dep[v]) q.push (v),dep[v] = dep[u] + 1;
            }
        }
        return dep[t];
    }
    T dfs (int u,int t,T flow)
    {
        if (u == t) return flow;
        T ans = 0;
        for (int &i = cur[u];i && ans < flow;i = nxt[i])
        {
            int v = to[i];
            if (val[i] && dep[v] == dep[u] + 1)
            {
                int x = dfs (v,t,min (val[i], flow - ans));
                if (x) val[i] -= x,val[i ^ 1] += x,ans += x;
            }
        }
        if (ans < flow) dep[u] = -1;
        return ans;
    }

    public :
    Dinic (int n,int m,int s,int t) :
        n (n),m (m),s (s),t (t),
        head (n + 1,0),cur (n + 1,0),dep (n + 1,0),
        to (m + 1,0),nxt (m + 1,0),val (m + 1,0) {cnt = 1;}
    void add (int u,int v,T w)
    {
        to[++cnt] = v;val[cnt] = w;nxt[cnt] = head[u];head[u] = cnt;
        to[++cnt] = u;val[cnt] = 0;nxt[cnt] = head[v];head[v] = cnt;
    }
    T calc ()
    {
        T ans = 0;
        while (bfs ())
        {
            T x;
            while ((x = dfs (s,t,INF))) ans += x;
        }
        return ans;
    }
};
```

3 Data Structure

3.1 Segment Tree

3.2 Fenwick Tree

3.3 Heavy-Light Decomposition

3.4 Splay Tree

3.5 Sparse Table

3.6 Persistent Data Structure