ACM模版

by PandaGhost

写在前面

基础模版

```cpp
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  #define OPFI(x) freopen(#x".in", "r", stdin);\
5                  freopen(#x".out", "w", stdout)
6  #define REP(i, a, b) for(int i=(a); i<=(b); ++i)
```

```
 7  #define REPd(i, a, b) for(int i=(a); i>=(b); --i)
 8  inline ll rd(){
 9      ll r=0, k=1; char c;
10      while(!isdigit(c=getchar())) if(c=='-') k=-k;
11      while(isdigit(c)) r=r*10+c-'0', c=getchar();
12      return r*k;
13  }
14  int main(){
15      return 0;
16  }
```

vimrc

```
 1  syntax on
 2  set ts=4
 3  set expandtab
 4  set autoindent
 5  set cindent
 6  set shiftwidth=4
 7  set nu
 8  set softtabstop=4
 9  set smartindent
10  set showmatch
11  set ruler
12  set mouse=a
13  inoremap <F1> <esc>:w<CR>
14  inoremap <F5> <esc>:below term<CR>
15  nmap <F1> :w<CR>
16  nmap <F5> :below term<CR>
17  colo habamax
18  set title
19  set shell=powershell
20  set wim=list
21  set backspace=indent,eol,start
22  set nocompatible
```

数据结构

zkw 线段树

单点修 区间查

```
1   ll s[N<<2], a[N];
2   int M;
3
4   ll f(ll x, ll y){
5       return x+y; // 改这
6   }
7
8   void build(){
9       for(M=1; M<=n+1; M<<=1);
10      REP(i, 1, n) s[i+M]=a[i];
11      REPd(i, M-1, 1) s[i]=f(s[2*i], s[2*i+1]);
12  }
13
14  ll qrange(int l, int r, ll init){ // 根据 f 传 init
15      ll res=init;
16      for(l=l+M-1, r=r+M+1; l^r^1; l>>=1, r>>=1){
17          if(~l&1) res=f(res, s[l^1]);
18          if(r&1) res=f(res, s[r^1]);
19      }
20      return res;
21  }
22
23  void edit(int x, ll v){
24      for(s[x+=M]=v, x>>=1; x; x>>=1){
25          s[x]=f(s[2*x], s[2*x+1]);
26      }
27  }
28
29  ll qpoint(int x){
30      return s[x+M];
31  }
```

珂朵莉树

```
1   struct node{
2       int l, r;
3       mutable int v;
4       bool operator<(const node& rhs) const { return l<rhs.l;
    }
5   };
6
7   set<node> odt;
8   typedef set<node>::iterator iter;
```

```
 9
10   iter split(ll p){
11       iter tmp=odt.lower_bound((node){p, 0, 0});
12       if(tmp!=odt.end()&&tmp->l==p) return tmp;
13       --tmp;
14       int tl=tmp->l, tr=tmp->r, tv=tmp->v;
15       odt.erase(tmp);
16       odt.insert((node){tl, p-1, tv});
17       return odt.insert((node){p, tr, tv}).first;
18   }
19
20   // 【修改 & 查询】注意 split 顺序
21   // iter itr=split(r+1), itl=split(l);
```

FHQ-Treap

以模版文艺平衡树为例

```
 1   int n, m, clk, rt;
 2   struct node{
 3       int key, val, sz, tag, ls, rs;
 4   }t[N];
 5   int newnode(int k){ return t[++clk]=(node){k, rand(), 1, 0},
     clk; }
 6   void down(int o){
 7       if(t[o].tag){
 8           t[t[o].ls].tag=1-t[t[o].ls].tag;
 9           t[t[o].rs].tag=1-t[t[o].rs].tag;
10           swap(t[t[o].ls].ls, t[t[o].ls].rs);
11           swap(t[t[o].rs].ls, t[t[o].rs].rs);
12           t[o].tag=0;
13       }
14   }
15   void up(int o){ t[o].sz=t[t[o].ls].sz+t[t[o].rs].sz+1; }
16   void split(int o, int x, int &L, int &R){
17       if(o==0) return L=R=0, void(); down(o);
18       if(t[t[o].ls].sz+1>=x) R=o, split(t[o].ls, x, L,
     t[o].ls);
19       else L=o, split(t[o].rs, x-t[t[o].ls].sz-1, t[o].rs, R);
20       up(o);
21   }
22   int merge(int L, int R){
23       if(L==0||R==0) return L+R;
```

```
24      if(t[L].val>t[R].val) return down(L),
    t[L].rs=merge(t[L].rs, R), up(L), L;
25      else return down(R), t[R].ls=merge(L, t[R].ls), up(R),
    R;
26  }
```

数学

快速幂

```
1   const ll MOD=998244353; // 改模数
2
3   ll qpow(ll a, ll x){
4       ll res=1;
5       a%=MOD;
6       while(x){
7           if(x&1) res=res*a%MOD;
8           a=a*a%MOD, x>>=1;
9       }
10      return res;
11  }
12
13  ll inv(ll x){ return qpow(x, MOD-2); } // 模数为质数时
```

高斯消元

```
1   const int N=110;
2   ll n;
3   double a[N][N], b[N];
4   void work(){
5       n=rd();
6       REP(i, 1, n){
7           REP(j, 1, n) a[i][j]=rd();
8           b[i]=rd();
9       }
10      REP(i, 1, n){
11          int t=i;
12          REP(j, i+1, n) if(abs(a[j][i])>1e-7&&(abs(a[t]
    [i])>abs(a[j][i])||abs(a[t][i])<1e-7)) t=j;
13          REP(j, i, n) swap(a[t][j], a[i][j]);
14          if(abs(a[i][i])<1e-7){
15              puts("No Solution");
```

```
16          return 0;
17      }
18      swap(b[t], b[i]);
19      double e=a[i][i];
20      REP(j, i, n) a[i][j]/=e;
21      b[i]/=e;
22      REP(j, i+1, n){
23          double d=a[j][i];
24          REP(k, i, n) a[j][k]-=d*a[i][k];
25          b[j]-=d*b[i];
26      }
27  }
28  REPd(i, n, 1) REP(j, 1, i-1) b[j]-=a[j][i]*b[i], a[j]
    [i]=0;
29  // REP(i, 1, n) printf("%.2f\n", b[i]);
30  // b[1...n] 保存 Ax=b 的解
31  }
```

图论

倍增

```
1  void dfs(int x, int fa){
2      pa[x][0]=fa; dep[x]=dep[fa]+1;
3      REP(i, 1, SP) pa[x][i]=pa[pa[x][i-1]][i-1];
4      for(int& v:g[x]) if(v!=fa){
5          dfs(v, x);
6      }
7  }
8
9  int lca(int x, int y){
10      if (dep[x]<dep[y]) swap(x, y);
11      int t=dep[x]-dep[y];
12      REP(i, 0, SP) if(t&(1<<i)) x=pa[x][i];
13      REPd(i, SP-1, -1){
14          int xx=pa[x][i], yy=pa[y][i];
15          if (xx!=yy) x=xx, y=yy;
16      }
17      return x==y?x:pa[x][0];
18  }
```

网络流

不是我写的，但是看着还好

其中 ll 是我改的，不敢保证有没有漏改，但是过了洛谷模版题

最大流

```
1    constexpr ll INF = LLONG_MAX / 2;
2
3    struct E {
4        int to; ll cp;
5        E(int to, ll cp): to(to), cp(cp) {}
6    };
7
8    struct Dinic {
9        static const int M = 1E5 * 5;
10       int m, s, t;
11       vector<E> edges;
12       vector<int> G[M];
13       int d[M];
14       int cur[M];
15
16       void init(int n, int s, int t) {
17           this->s = s; this->t = t;
18           for (int i = 0; i <= n; i++) G[i].clear();
19           edges.clear(); m = 0;
20       }
21
22       void addedge(int u, int v, ll cap) {
23           edges.emplace_back(v, cap);
24           edges.emplace_back(u, 0);
25           G[u].push_back(m++);
26           G[v].push_back(m++);
27       }
28
29       bool BFS() {
30           memset(d, 0, sizeof d);
31           queue<int> Q;
32           Q.push(s); d[s] = 1;
33           while (!Q.empty()) {
34               int x = Q.front(); Q.pop();
35               for (int& i: G[x]) {
36                   E &e = edges[i];
37                   if (!d[e.to] && e.cp > 0) {
```

```
38                    d[e.to] = d[x] + 1;
39                    Q.push(e.to);
40                }
41            }
42        }
43        return d[t];
44    }
45
46    ll DFS(int u, ll cp) {
47        if (u == t || !cp) return cp;
48        ll tmp = cp, f;
49        for (int& i = cur[u]; i < G[u].size(); i++) {
50            E& e = edges[G[u][i]];
51            if (d[u] + 1 == d[e.to]) {
52                f = DFS(e.to, min(cp, e.cp));
53                e.cp -= f;
54                edges[G[u][i] ^ 1].cp += f;
55                cp -= f;
56                if (!cp) break;
57            }
58        }
59        return tmp - cp;
60    }
61
62    ll go() {
63        ll flow = 0;
64        while (BFS()) {
65            memset(cur, 0, sizeof cur);
66            flow += DFS(s, INF);
67        }
68        return flow;
69    }
70 } DC;
```

费用流

```
1  constexpr ll INF = LLONG_MAX / 2;
2
3  struct E {
4      int from, to; ll cp, v;
5      E() {}
6      E(int f, int t, ll cp, ll v) : from(f), to(t), cp(cp),
   v(v) {}
```

```
7    };
8
9    struct MCMF {
10       static const int M = 1E5 * 5;
11       int n, m, s, t;
12       vector<E> edges;
13       vector<int> G[M];
14       bool inq[M];
15       ll d[M], a[M];
16       int p[M];
17
18       void init(int _n, int _s, int _t) {
19           n = _n; s = _s; t = _t;
20           REP (i, 0, n + 1) G[i].clear();
21           edges.clear(); m = 0;
22       }
23
24       void addedge(int from, int to, ll cap, ll cost) {
25           edges.emplace_back(from, to, cap, cost);
26           edges.emplace_back(to, from, 0, -cost);
27           G[from].push_back(m++);
28           G[to].push_back(m++);
29       }
30
31       bool BellmanFord(ll &flow, ll &cost) {
32           REP (i, 0, n + 1) d[i] = INF;
33           memset(inq, 0, sizeof inq);
34           d[s] = 0, a[s] = INF, inq[s] = true;
35           queue<int> Q; Q.push(s);
36           while (!Q.empty()) {
37               int u = Q.front(); Q.pop();
38               inq[u] = false;
39               for (int& idx: G[u]) {
40                   E &e = edges[idx];
41                   if (e.cp && d[e.to] > d[u] + e.v) {
42                       d[e.to] = d[u] + e.v;
43                       p[e.to] = idx;
44                       a[e.to] = min(a[u], e.cp);
45                       if (!inq[e.to]) {
46                           Q.push(e.to);
47                           inq[e.to] = true;
48                       }
49                   }
50               }
```

```
51              }
52              if (d[t] == INF) return false;
53              flow += a[t];
54              cost += a[t] * d[t];
55              int u = t;
56              while (u != s) {
57                  edges[p[u]].cp -= a[t];
58                  edges[p[u] ^ 1].cp += a[t];
59                  u = edges[p[u]].from;
60              }
61              return true;
62          }
63
64      pair<ll, ll> go() {
65          ll flow = 0, cost = 0;
66          while (BellmanFord(flow, cost));
67          return make_pair(flow, cost);
68      }
69  } MM;
```

二分图最大匹配

ps. 建单向图（即只有左部指向右部的边）

```
1   struct MaxMatch {
2       int n;
3       vector<int> G[N];
4       int vis[N], left[N], clk;
5
6       void init(int n) {
7           this->n = n;
8           REP (i, 0, n + 1) G[i].clear();
9           memset(left, -1, sizeof left);
10          memset(vis, -1, sizeof vis);
11      }
12
13      bool dfs(int u) {
14          for (int v: G[u])
15              if (vis[v] != clk) {
16                  vis[v] = clk;
17                  if (left[v] == -1 || dfs(left[v])) {
18                      left[v] = u;
19                      return true;
```

```
20                    }
21                }
22            return false;
23        }
24
25        int match() {
26            int ret = 0;
27            for (clk = 0; clk <= n; ++clk)
28                if (dfs(clk)) ++ret;
29            return ret;
30        }
31  } MM;
```

Tarjan 强连通分量缩点

```
1   int low[N], dfn[N], clk, B, bl[N];
2   vector<int> bcc[N];
3   void init() { B = clk = 0; memset(dfn, 0, sizeof dfn); }
4   void tarjan(int u) {
5       static int st[N], p;
6       static bool in[N];
7       dfn[u] = low[u] = ++clk;
8       st[p++] = u; in[u] = true;
9       for (int& v: G[u]) {
10          if (!dfn[v]) {
11              tarjan(v);
12              low[u] = min(low[u], low[v]);
13          } else if (in[v]) low[u] = min(low[u], dfn[v]);
14      }
15      if (dfn[u] == low[u]) {
16          ++B;
17          while (1) {
18              int x = st[--p]; in[x] = false;
19              bl[x] = B; bcc[B].push_back(x);
20              if (x == u) break;
21          }
22      }
23  }
```