2SAT

int T, n, m, depth, scc;

int h[N], e[M], ne[M], idx;

int dfn[N], low[N], bel[N];

bool st[N];stack<int> stk;

void add(int a, int b) {e[idx] = b; ne[idx] = h[a]; h[a] = idx++;}

void tarjan(int u) {

dfn[u] = low[u] = ++depth;

st[u] = true;

stk.push(u);

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j);

low[u] = min(low[u], low[j]);

} else {

if (st[j]) low[u] = min(low[u], dfn[j]);

}

}

if (dfn[u] == low[u]) {

++scc;

while (true) {

int t = stk.top();

stk.pop();

st[t] = false;

bel[t] = scc;

if (t == u) break;

}

}

}

void solve() {

cin >> n >> m;

idx = scc = depth = 0;

memset(h, -1, sizeof h);

for (int i = 0; i < 2 \* n; i++) {

dfn[i] = low[i] = 0;

}

for (int i = 0; i < m; i++) {

char o1, o2;

int x1, x2;

cin >> o1 >> x1 >> o2 >> x2;

x1--; x2--;

int a = x1 \* 2 + (o1 == 'h'), b = x2 \* 2 + (o2 == 'h');

/\*

a | b == 1;

a == 0 -> b == 1

b == 0 -> a == 1

(a ^ 1) == 0 -> b == 1

(b ^ 1) == 0 -> a == 1

\*/

add(a ^ 1, b);

add(b ^ 1, a);

}

for (int i = 0; i < 2 \* n; i++) {

if (dfn[i] == 0) {

tarjan(i);

}

}

for (int i = 0; i < n; i++) {

if (bel[2 \* i] == bel[2 \* i + 1]) {

cout << "BAD" << endl;

return;

}

}

cout << "GOOD" << endl;

}

点分治

int n, k, rt, rts;

int h[N], e[M], ne[M], idx;

int sz[N], f[T], t[T];

bool vis[N]; ll ans;

void dfs\_root(int u, int fa) {

sz[u] = 1;

int maxs = 0;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || vis[j]) continue;

dfs\_root(j, u);

sz[u] += sz[j];

maxs = max(maxs, sz[j]);

}

maxs = max(maxs, sz[0] - sz[u]);

if (rts > maxs) {

rts = maxs;

rt = u;

}

}

void get\_dist(int u, int fa, int d) {

if (d > k) return;

t[d]++;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || vis[j]) continue;

get\_dist(j, u, d + 1);

}

}

void solove(int u) {

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (vis[j]) continue;

get\_dist(j, u, 1);

for (int p = 1; p < k; p++) {

int q = k - p;

ans = ans + f[p] \* t[q];

}

ans += t[k];

for (int i = 1; i <= k; i++) {

f[i] += t[i];

t[i] = 0;

}

}

memset(f, 0, sizeof f);

}

void divide(int u) {

vis[u] = true;

solove(u);

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (vis[j]) continue;

rts = sz[0] = sz[j];

dfs\_root(j, u);

divide(rt);

}

}

int main() {

rts = sz[0] = n;

dfs\_root(1, -1);

divide(rt); cout << ans << endl; }

树上启发式合并

int n, m, idx, tot;

int h[N], e[M], ne[M];

int col[N];

int l[N], r[N], id[N], hs[N], sz[N];

int cnt[N];

ll ans[N];

ll maxcnt, maxsum;

void add(int a, int b) { e[idx] = b; ne[idx] = h[a]; h[a] = idx++;}

void dfs1(int u, int fa) {

l[u] = ++tot;

id[tot] = u;

hs[u] = -1;

sz[u] = 1;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa) continue;

dfs1(j, u);

sz[u] += sz[j];

if (hs[u] == -1 || sz[j] > sz[hs[u]]) {

hs[u] = j;

}

}

r[u] = tot;

}

void add(int u) {

u = col[u];

cnt[u]++;

if (cnt[u] > maxcnt) {

maxcnt = cnt[u];

maxsum = 0;

}

if (maxcnt == cnt[u]) maxsum += u;

}

void del(int u) {

u = col[u];

cnt[u]--;

}

void dfs2(int u, int fa, bool flag) {

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || j == hs[u]) continue;

dfs2(j, u, false);

}

if (hs[u] != -1) {

dfs2(hs[u], u, true);

}

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || j == hs[u]) continue;

for (int t = l[j]; t <= r[j]; t++) {

add(id[t]);

}

}

add(u);

ans[u] = maxsum;

if (!flag) {

maxsum = maxcnt = 0;

for (int i = l[u]; i <= r[u]; i++)

del(id[i]);

}

}

int main() {

cin >> n;

for (int i = 1; i <= n; i++) {

cin >> col[i];

}

memset(h, -1, sizeof h);

for (int i = 0; i < n - 1; i++) {

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

dfs1(1, 0);

dfs2(1, 0, false);

for (int i = 1; i <= n; i++) {

cout << ans[i] << " ";

}

cout << endl;

}

费用流

struct MinCostGraph {

int s, t, tot;

int h[N], e[M], ne[M], idx;

ll w1[M], w2[M], flow, cost;

ll dis[N], pf[N];

bool vis[N];

int pre[N];

void add(int a, int b, ll v1, ll v2, ll c) {

e[idx] = b; w1[idx] = v1; w2[idx] = c; ne[idx] = h[a]; h[a] = idx++;

e[idx] = a; w1[idx] = v2; w2[idx] = -c; ne[idx] = h[b]; h[b] = idx++;

}

bool spfa() {

ll INF = 1ll << 60;

for (int i = 1; i <= tot; i++) {

dis[i] = INF;

vis[i] = false;

pre[i] = -1;

pf[i] = INF;

}

dis[s] = 0;

vis[s] = true;

queue<int> q;

q.push(s);

while (!q.empty()) {

int u = q.front();

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (w1[i] && dis[j] > dis[u] + w2[i]) {

dis[j] = dis[u] + w2[i];

pre[j] = i;

pf[j] = min(pf[u], w1[i]);

if (!vis[j]) {

vis[j] = true;

q.push(j);

}

}

}

q.pop();

vis[u] = false;

}

return dis[t] != INF;

}

void augment() {

ll f = pf[t];

flow += f;

cost += f \* dis[t];

int u = t;

while (~pre[u]) {

w1[pre[u]] -= f;

w1[pre[u] ^ 1] += f;

u = e[pre[u] ^ 1];

}

}

PII solve() {

flow = 0, cost = 0;

while (spfa()) {

augment();

}

return {flow, cost};

}

void init(int \_s, int \_t, int \_tot) {

s = \_s; t = \_t; tot = \_tot;

idx = 0;

for (int i = 1; i <= tot; i++) h[i] = -1;

}

}g;

最大流

struct FlowGraph {

int s, t, tot;

int h[N], e[M], ne[M], idx;

ll w[M];

int dis[N], cur[N];

void add(int a, int b, ll v1, ll v2) {

e[idx] = b; w[idx] = v1; ne[idx] = h[a]; h[a] = idx++;

e[idx] = a; w[idx] = v2; ne[idx] = h[b]; h[b] = idx++;

}

bool bfs() {

for (int i = 1; i <= tot; i++) {

dis[i] = 0;

cur[i] = h[i];

}

queue<int> q;

q.push(s);

dis[s] = 1;

while (!q.empty()) {

int u = q.front(); q.pop();

for (int i = h[u]; ~i; i = ne[i]) {

int j = e[i];

if (w[i] && !dis[j]) {

dis[j] = dis[u] + 1;

if (j == t) return true;

q.push(j);

}

}

}

return false;

}

ll dfs(int u, ll m) {

if (u == t) return m;

ll ans = 0;

for (int i = cur[u]; ~i; cur[u] = i = ne[i]) {

int j = e[i];

if (w[i] && dis[j] == dis[u] + 1) {

ll f = dfs(j, min(w[i], m));

w[i] -= f;

w[i ^ 1] += f;

m -= f;

ans += f;

if (!m) break;

}

}

if (!ans) {

dis[u] = -1;

}

return ans;

}

ll dinic() {

ll ans = 0;

while (bfs()) {

ans += dfs(s, 1e18);

}

return ans;

}

void init(int \_s, int \_t, int \_tot) {

s = \_s; t = \_t; tot = \_tot;

idx = 0;

for (int i = 1; i <= tot; i++) h[i] = -1;

}

}g;

最小树形图

typedef long long ll; typedef pair<int, ll> PII;

const int N = 110, M = N \* 2, MOD = 1000000007, INF = 0x3f3f3f3f;

int T, n, m, root;int g[N][N], d[N][N];

int dfn[N], low[N], id[N], stk[N], top, cnt, dep; int pre[N];

bool st[N];

void dfs(int u) {

st[u] = true;

for (int i = 1; i <= n; i++)

if (!st[i] && g[u][i] != INF) {

dfs(i);

}

}

bool check(int root) {

dfs(root);

for (int i = 1; i <= n; i++) {

if (!st[i]) {

return false;

}

}

return true;

}

void tarjan(int u) {

dfn[u] = low[u] = ++dep;

stk[++top] = u;

st[u] = true;

int j = pre[u];

if (!dfn[j]) {

tarjan(j);

low[u] = min(low[u], low[j]);

} else if (st[j]) {

low[u] = min(low[u], dfn[j]);

}

if (dfn[u] == low[u]) {

++cnt;

while (true) {

int t = stk[top--];

st[t] = false;

id[t] = cnt;

if (t == u) break;

}

}

}

int work(int root) {

memset(st, false, sizeof st);

int ans = 0;

while (true) {

for (int i = 1; i <= n; i++) {

pre[i] = i;

for (int j = 1; j <= n; j++) {

if (g[pre[i]][i] > g[j][i]) {

pre[i] = j;

}

}

}

dep = cnt = 0;

memset(dfn, 0, sizeof dfn);

for (int i = 1; i <= n; i++)

if (!dfn[i]) {

tarjan(i);

}

if (cnt == n) {

for (int i = 1; i <= n; i++)

if (i != root)

ans += g[pre[i]][i];

break;

}

for (int i = 1; i <= n; i++)

if (i != root && id[pre[i]] == id[i]) {

ans += g[pre[i]][i];

}

for (int i = 1; i <= cnt; i++)

for (int j = 1; j <= cnt; j++)

d[i][j] = INF;

for (int i = 1; i <= n; i++)

for (int j = 1; j <= n; j++)

if (g[i][j] < INF && id[i] != id[j]) {

int a = id[i], b = id[j];

if (id[pre[j]] == id[j]) {

d[a][b] = min(d[a][b], g[i][j] - g[pre[j]][j]);

} else {

d[a][b] = min(d[a][b], g[i][j]);

}

}

n = cnt;

root = id[root];

memcpy(g, d, sizeof g);

}

return ans;

}

void solve() {

cin >> n >> m >> root;

memset(g, 0x3f, sizeof g);

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

int a, b, c;

cin >> a >> b >> c;

if (b == root) continue;

g[a][b] = min(g[a][b], c);

}

if (!check(root)) {

puts("-1");

} else {

printf("%d\n", work(root));

}

}

强连通分量

int h[N], e[M], ne[M], idx; int dfn[N], low[N], id[N], cnt[N], depth, scc;

int stk[N], top; bool st[N];

void add(int a, int b) { e[idx] = b; ne[idx] = h[a]; h[a] = idx++;}

void tarjan(int u) {

dfn[u] = low[u] = ++depth;

st[u] = true;

stk[++top] = u;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j);

low[u] = min(low[u], low[j]);

} else {

if (st[j]) low[u] = min(low[u], dfn[j]);

}

}

if (dfn[u] == low[u]) {

++scc;

while (true) {

int t = stk[top--];

st[t] = false;

id[t] = scc;

cnt[scc]++;

if (t == u) break;

}

}

}

割点

int n, m, depth, root;

int h[N], e[M], ne[M], idx; int dfn[N], low[N], cut[N];

void add(int a, int b) { e[idx] = b; ne[idx] = h[a]; h[a] = idx++;}

void tarjan(int u, int fa) {

dfn[u] = low[u] = ++depth;

int s = 0;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j, u);

s++;

low[u] = min(low[u], low[j]);

if (low[j] >= dfn[u]) {

cut[u] = 1;

}

} else {

if (j != fa) low[u] = min(low[u], dfn[j]);

}

}

if (root == u && s <= 1) cut[u] = 0;

}

割边

int n, m, depth; int dfn[N], low[N]; int h[N], e[M], ne[M], idx;

vector<int> ans;

void add(int a, int b) { e[idx] = b; ne[idx] = h[a]; h[a] = idx++;}

void dfs(int u, int id) {

dfn[u] = low[u] = ++depth;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (!dfn[j]) {

dfs(j, i);

low[u] = min(low[u], low[j]);

if (dfn[j] == low[j]) ans.push\_back(id + 1);

} else if ((id ^ 1) != i) low[u] = min(low[u], dfn[j]);

}

}