**TỔNG HỢP CODE LÝ THUYẾT ĐỒ THỊ**

1. **Cây khung nhỏ nhất**
2. Kruskal

Thuật toán gồm 2 phần:

* Sắp xếp mảng m cạnh theo trọng số tăng dần.
* Duyệt m cạnh, mỗi cạnh dùng Disjoint Set.

struct DSU\_rollback {

vector<int> lab;

vector<pii> his;

int n;

int root(int x) {

if(lab[x] < 0) return x;

his.pb({x, lab[x]});

return lab[x] = root(lab[x]);

}

bool join(int u,int v) {

u = root(u);

v = root(v);

if(u == v) return 0;

his.pb({u, lab[u]});

his.pb({v, lab[v]});

if(lab[u] > lab[v]) swap(u,v);

lab[u] += lab[v];

lab[v] = u;

return 1;

}

void rollback(bool ok) { /// ok = có rollback không?

if(ok) {

while(his.size()) {

lab[his.back().st] = his.back().nd;

his.pop\_back();

}

}

else his.clear();

}

DSU\_rollback(int n) : n(n), lab(n+1, -1), his(0) {}

};

1. Prim

Thuật toán **Prim** kết nạp từng **đỉnh** vào đồ thị theo tiêu chí: đỉnh được nạp vào tiếp theo phải **chưa được nạp** và **gần nhất** với các đỉnh đã được nạp vào đồ thị.

int mst() {

priority\_queue<pii, vector<pii>, greater<pii> > heap;

vector<int> d(n+1, inf);

vector<bool> mst(n+1, 0);

heap.push({d[1] = 0, 1});

while(!heap.empty()) {

while(!heap.empty() && mst[heap.top().nd]) heap.pop();

if(heap.empty()) break;

int dmin = heap.top().st;

int u = heap.top().nd;

mst[u] = 1; weight += dmin;

heap.pop();

for(pii i:g[u]) {

int v = i.st, c = i.nd;

if(d[v] > c) heap.push({d[v] = c, v});

}

}

return weight;

}

1. **DFS, topo sort**

int cau = 0;

forinc(u, 1, n) {

for(int v : g[u]) if(p[v] == u) {

if(low[v]>num[u]) ++cau;

}

}

int khop = 0;

forinc(u, 1, n) {

int cnt = 0, socon = 0;

for(int v : g[u]) if(!nho[v] && p[v]==u) {

nho[v] = 1;

socon++;

if(low[v]>=num[u]) ++cnt;

}

int bac = (p[u]==0) ? socon-1 : cnt;

if(bac > 0) ++khop;

}

void DFS(int u) // DFS cơ bản

{

cl[u] = 1;

num[u] = low[u] = ++id;

s[++sn] = u;

for(auto &v : g[u]) if(p[u]!=v) {

if (!cl[v]) {

p[v] = u;

DFS(v);

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

}

else

low[u] = min(low[u],num[v]);

}

}

// tìm liên thông mạnh

int cl[maxn], s[maxn], sn;

void DFS(int u) {

cl[u] = 1;

num[u] = low[u] = ++id;

s[++sn] = u;

for(auto &v : g[u]) {

if (!cl[v]) {

DFS(v);

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

} else if (cl[v] == 1)

low[u]=min(low[u],num[v]);

}

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

++sLTM;

int v;

do {

v=s[sn--];

x[++slx]=v;

LTM[v]=sLTM;

cl[v]=2;

} while (v!=u);

}

}

// Xây dựng DAG với mỗi đỉnh là một thành phần LTM

vector<int> gLTM[maxn

for(int i=1; i<=sLTM; ++i) gLTM[i].clear();

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

for(auto &v : g[u]) if (LTM[u]!=LTM[v]) {

int i=LTM[u], j=LTM[v];

gLTM[i].push\_back(j);

}