# 20221201 数据结构与算法 解题报告

## 最大生成森林

使用 Kruskal 算法进行生成森林操作即可，记录所能记录到的最大边权。

*Largest-Spanning-Forest.cpp*

#include <bits/stdc++.h>  
using namespace std;  
  
struct Edge  
{  
 int u, v, w;  
 Edge(int \_u = 0, int \_v = 0, int \_w = 0) : u(\_u), v(\_v), w(\_w) {}  
 friend bool operator<(const Edge &a, const Edge &b) { return a.w < b.w; }  
};  
  
int getFather(int cur, vector<int> &fah)  
{  
 if (fah[cur] == -1)  
 return cur;  
 else  
 return fah[cur] = getFather(fah[cur], fah);  
}  
  
bool merge(int u, int v, vector<int> &fah)  
{  
 if (getFather(u, fah) != getFather(v, fah))  
 {  
 fah[getFather(u, fah)] = getFather(v, fah);  
 return true;  
 }  
 else  
 return false;  
}  
  
int main()  
{  
 freopen("init.in", "r", stdin);  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n, m;  
 cin >> n >> m;  
 vector<Edge> edges;  
 for (; m--;)  
 {  
 int u, v, w;  
 cin >> u >> v >> w;  
 edges.push\_back(Edge(u, v, w));  
 }  
  
 sort(edges.rbegin(), edges.rend());  
  
 vector<int> fah(n, -1);  
 int ans = 0;  
 for (auto &edge : edges)  
 if (merge(edge.u, edge.v, fah))  
 ans += edge.w;  
  
 cout << ans << endl;  
 }  
 return 0;  
}

## Highways

使用邻接矩阵记录图，使用 Kruskal 或 Prim 算法进行生成树操作即可。

*Highways-Kruskal.cpp*

#include <bits/stdc++.h>  
using namespace std;  
  
struct Edge  
{  
 int u, v, w;  
 Edge(int \_u = 0, int \_v = 0, int \_w = 0) : u(\_u), v(\_v), w(\_w) {}  
 friend bool operator<(const Edge &a, const Edge &b) { return a.w < b.w; }  
};  
  
int getFather(int cur, vector<int> &fah)  
{  
 if (fah[cur] == -1)  
 return cur;  
 else  
 return fah[cur] = getFather(fah[cur], fah);  
}  
  
bool merge(int u, int v, vector<int> &fah)  
{  
 if (getFather(u, fah) != getFather(v, fah))  
 {  
 fah[getFather(u, fah)] = getFather(v, fah);  
 return true;  
 }  
 else  
 return false;  
}  
  
int main()  
{  
 freopen("init.in", "r", stdin);  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n;  
 cin >> n;  
 vector<Edge> edges;  
 for (int u = 0; u < n; u++)  
 for (int v = 0; v < n; v++)  
 {  
 int w;  
 cin >> w;  
 if (u != v)  
 edges.push\_back(Edge(u, v, w));  
 }  
  
 sort(edges.begin(), edges.end());  
  
 vector<int> fah(n, -1);  
 int ans = 0;  
 for (auto &edge : edges)  
 if (merge(edge.u, edge.v, fah))  
 ans = edge.w;  
  
 cout << ans << endl;  
 }  
 return 0;  
}

*Highways-Prim.cpp*

#include <bits/stdc++.h>  
using namespace std;  
  
struct Node  
{  
 int num, preW;  
 Node(int \_num = 0, int \_preW = 0) : num(\_num), preW(\_preW) {}  
 friend bool operator>(const Node &a, const Node &b) { return a.preW > b.preW; }  
};  
  
int main(int argc, char const \*argv[])  
{  
 freopen("init.in", "r", stdin);  
 int T;  
 cin >> T;  
 for (; T--;)  
 {  
 int n;  
 cin >> n;  
 vector<vector<int>> w(n, vector<int>(n, 0));  
 for (int i = 0; i < n; i++)  
 for (int j = 0; j < n; j++)  
 cin >> w[i][j];  
  
 vector<bool> sgn(n, false);  
 priority\_queue<Node, vector<Node>, greater<Node>> hep;  
 hep.push(Node(0, 0));  
 int ans = 0;  
 for (int count = 0; count < n - 1; count++)  
 {  
 Node cur;  
 for (; sgn[(cur = hep.top()).num]; hep.pop())  
 ;  
 ans = cur.preW;  
 sgn[cur.num] = true;  
 hep.pop();  
 for (int i = 0; i < n; i++)  
 if (w[cur.num][i] && !sgn[i])  
 hep.push(Node(i, w[cur.num][i]));  
 }  
  
 cout << ans << endl;  
 }  
 return 0;  
}