**《算法设计与分析》实验报告**

实验名称 ： 实验9 分支限界法实验2

实验日期 ： 　 2025/5/27

姓 名 ： 　　　高心阳

学 号 ： 　 084623237

班 级 ： 　 计算机232

成 绩 ：

**人工智能与信息技术学院**

**南京中医药大学**

|  |
| --- |
| **实验目的：** |
| 1. 熟悉优先分支限界的概念 2. 掌握优先分支限界法的使用 |
| **实验内容和要求** |
| 1、最小重量机器设计问题  最小重量机器设计问题  题目描述  某工厂需要设计一台机器，该机器由若干个部件组成。每个部件有多种型号可供选择，不同型号的部件具有不同的重量和成本。工厂希望在满足总成本不超过给定预算的前提下，设计出重量最小的机器。  输入：  部件数量n（n>=1）。  每个部件的型号数量mi （i=1,2,…,n）。  每个部件的型号的重量和成本矩阵W和C。其中W[i][j]表示第i个部件的第j种型号的重量，C[i][j] 表示其成本。  总预算B。  输出  最小的机器总重量。  每个部件选择的型号（编号从 1 开始）。  示例    ，每个部件选择的型号是（\*，\*，\*）。  要求使用分支限界法设计算法，求解上述问题。  2、加权连通子图问题  题目描述：  给定一个无向图G=（V，E），其中V是顶点集合，E是边集合，每条边都有一个非负的权重。现在需要从图中选择一些边，使得所有顶点都连通，同时选择的边的总权重尽可能小。但是，与传统的最小生成树不同，这次还需要满足一个额外的约束条件：选择的边的数量不能超过k条。  输入  顶点数量n（n>=2）。  边的数量m（m>=n-1）。  边的权重矩阵W，其中W[i][j]表示顶点i和顶点j之间的边的权重。若顶点i和顶点j 之间没有直接的边，则W[i][j]=∞ 。  边的数量限制k（n-1<=k<=m）。  输出  满足条件的最小总权重。  选择的边的列表（边的编号从 1 开始）。    \*3、自行在竞赛网站上找相关题目练习。 |
| **运行结果（写清题号）** |
| |  | | --- | | Task1 | | // encoding:utf-8  #include <bits/stdc++.h>  #define makep std::make\_pair  #define get std::cin  #define put std::cout  #define endl '\n'  using ll = long long;  using paii = std::pair<int, int>;  using pall = std::pair<long long , long long>;  using vpii = std::vector<std::pair<int, int> >;  using vint = std::vector<int>;  using vll = std::vector<long long>;  using vll2 = std::vector<std::vector<long long> >;  using vbool = std::vector<bool>;  using vbool2 = std::vector<std::vector<bool> >;  using qll = std::queue<long long>;  using mll = std::map<long long, long long>;  using sll = std::set<ll>;  template <class T>  T qpow(T a, ll b) {  T ans = 1;  while (b > 0) {  if (b & 1)  ans \*= a;  a \*= a;  b >>= 1;  }  return ans;  }  template <class T>  T tmin(T a, T b) {  return a < b ? a : b;  }  template <class T>  T tmax(T a, T b) {  return a > b ? a : b;  }  struct Node  {  ll w;  ll c;  };  const int N = 1000 + 3;  ll n, m, w, c, maxc, minw = LLONG\_MAX;  Node node[N][N];  vll best;  vll min;  void bfs(int curr, ll currc, ll currw, vll prew) {  if (curr == n + 1) {  if (currw < minw && currc <= maxc) {  minw = currw;  best = prew;  }  return;  }  ll indexminw = currw;  for (int i = curr; i <= n; i++)  indexminw += min[i];  if (indexminw > minw)  return;  if (currc > maxc)  return;  for (ll i = 1; i <= node[curr][0].w; i++) {  prew.push\_back(i);  bfs(curr + 1, currc + node[curr][i].c, currw + node[curr][i].w, prew);  prew.pop\_back();  }  }  void solve() {  get >> n >> maxc;  min.assign(n+3, LLONG\_MAX);  for (int i = 1; i <= n; i++) {  get >> m;  node[i][0].w = m;  node[i][0].c = -1;  for (int j = 1; j <= m; j++) {  get >> w >> c;  node[i][j].w = w;  node[i][j].c = c;  if (min[i] > w)  min[i] = w;  }  }  vll prew;  prew.push\_back(-1);  bfs(1, 0, 0, prew);  put << minw << endl;  for (ll i = 1; i < best.size(); i++)  put << best[i] << "\t";  put << endl;  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  int T;  T = 1;  // std::cin >> T;  while (T--) {  solve();  }  return 0;  } | |  |  |  | | --- | | Task2 | | // encoding:utf-8  #include <bits/stdc++.h>  #define makep std::make\_pair  #define get std::cin  #define put std::cout  #define endl '\n'  using ll = long long;  using paii = std::pair<int, int>;  using pall = std::pair<long long , long long>;  using vpii = std::vector<std::pair<int, int> >;  using vint = std::vector<int>;  using vll = std::vector<long long>;  using vll2 = std::vector<std::vector<long long> >;  using vbool = std::vector<bool>;  using vbool2 = std::vector<std::vector<bool> >;  using qll = std::queue<long long>;  using mll = std::map<long long, long long>;  using sll = std::set<ll>;  template <class T>  T qpow(T a, ll b) {  T ans = 1;  while (b > 0) {  if (b & 1)  ans \*= a;  a \*= a;  b >>= 1;  }  return ans;  }  template <class T>  T tmin(T a, T b) {  return a < b ? a : b;  }  template <class T>  T tmax(T a, T b) {  return a > b ? a : b;  }  ll n, m, k, minc = LLONG\_MAX;  std::vector<paii> best;  struct Edge {  int u, v;  ll w;  };  std::vector<Edge> all\_edges;  int findx(std::vector<int>& parent, int a) {  if (parent[a] == a)  return a;  return parent[a] = findx(parent, parent[a]);  }  void merge(std::vector<int>& parent, int a, int b) {  int root\_a = findx(parent, a);  int root\_b = findx(parent, b);  if (root\_a != root\_b)  parent[root\_a] = root\_b;  }  void dfs(int edge\_idx, ll curr\_cost, int curr\_edges\_count,  std::vector<int> curr\_parent, int curr\_components,  std::vector<paii>& curr\_path) {  if (curr\_cost >= minc)  return;  if (curr\_edges\_count > k)  return;  if (curr\_components == 1) {  if (curr\_cost < minc) {  minc = curr\_cost;  best = curr\_path;  }  return;  }  if (edge\_idx == all\_edges.size()) {  return;  }  if (curr\_edges\_count + (curr\_components - 1) > k) {  return;  }  dfs(edge\_idx + 1, curr\_cost, curr\_edges\_count,  curr\_parent, curr\_components, curr\_path);  Edge& edge = all\_edges[edge\_idx];  int root\_u = findx(curr\_parent, edge.u);  int root\_v = findx(curr\_parent, edge.v);  if (root\_u != root\_v) {  std::vector<int> next\_parent = curr\_parent;  merge(next\_parent, root\_u, root\_v);  curr\_path.push\_back({edge.u, edge.v});  dfs(edge\_idx + 1, curr\_cost + edge.w, curr\_edges\_count + 1,  next\_parent, curr\_components - 1, curr\_path);  curr\_path.pop\_back();  }  }  void solve() {  std::cin >> n >> m >> k;  for (int i = 0; i < m; i++) {  ll s, e, v;  std::cin >> s >> e >> v;  all\_edges.push\_back({(int)s, (int)e, v});  }  std::sort(all\_edges.begin(), all\_edges.end(), [](const Edge& a, const Edge& b) {  return a.w < b.w;  });  std::vector<int> initial\_parent(n + 1);  std::iota(initial\_parent.begin(), initial\_parent.end(), 0);  std::vector<paii> curr\_path;  dfs(0, 0, 0, initial\_parent, n, curr\_path);  std::cout << minc << '\n';  for (size\_t i = 0; i < best.size(); i++) {  std::cout << "(" << best[i].first << "," << best[i].second << ")";  }  std::cout << '\n';  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  int T;  T = 1;  // std::cin >> T;  while (T--) {  solve();  }  return 0;  } | |  | |
| **实验的体会与建议** |
| 通过最后一题的逐步debug，深度学习了分支限界法的运用技巧，掌握了循环法和迭代法两种方式进行分支限界的技巧，收获颇丰。 |