**《算法设计与分析》实验报告**

实验名称 ： 实验8 分支限界法 1

实验日期 ： 　 2025.5.13

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. 熟悉分支限界法的使用 |
| **实验内容和要求** |
| **1、最大团问题**  **问题描述：**  最大团问题（Maximum Clique Problem, MCP）是图论中一个经典的组合优化问题，也是一类NP完全问题。给定无向图G=(V，E)。如果U是V的子集，且对U中任意两个顶点u和v有(u，v)∈ E，则称U是G的完全子图。G的完全子图U是G的团当且仅当U不包含在G的更大的完全子图中。G的最大团是指G中所含顶点数最多的团。最大团就是在一个无向图中找出一个点数最多的完全图。  **输入示例：（第一行为顶点数量n，随后若干行为边，以0 0结束）**  5  1 2  1 4  1 5  2 3  2 5  3 5  4 5  0 0  **输出示例（任一最大团）**  1 2 5（或1 4 5 或2 3 5）  **2、最优工程布线问题**  **问题描述：**  布线问题就是在m\*n的方格阵列中，指定一个方格的中点a,另一个方格的中点b,问题要求找出a到b的最短布线方案。布线时只能沿直线或直角，不能走斜线。为了避免线路相交，已布过线的方格做了封锁标记(灰色)，其他线路不允许穿过被封锁的方格。  **输入：**    **输出：**  a到b的最短布线方案。（含路径长度和该路径上的坐标点）  \*3、**平方数问题（选做题，HDU1426）**  **问题描述：**  给定一个正整数N，请找到一个最小的非负整数M，满足M2 % 10x = N (x=0,1,2,3….)。其中，0≤N≤109。  **输入示例：（第一行为测试实例数k，随后k行每行为一个N）**  3  3  21  25  **输出示例：（k行，每行为一个实例结果，如M无解则输出None）**  None  11  5  **备注：这是一个竞赛中比较简单的分析限界算法题，但容易超时，请选做的同学在HDU网站上提交一下试试看是否能够AC。**  **不会做的同学可以网上搜索解析，百度HDU4394即可。** |
| **运行结果（写清题号）** |
| |  | | --- | | Task1.cpp | | #include <bits/stdc++.h>  #define endl '\n'  #define paii std::pair<int,int>  #define pall std::pair<ll ,ll>  #define vpii std::vector<paii>  #define vll std::vector<ll>  #define vbool std::vector<bool>  #define qll std::queue<ll>  #define makep std::make\_pair  #define get std::cin  #define put std::cout  typedef long long ll;  const ll N = 1e3 + 3;  ll graph[N][N];  ll n, a, b;  vll ans;  vbool visited;  void bfs(vll curr, qll nodes) {  if (nodes.empty() && curr.size() > ans.size()) {  ans = curr;  return;  }  while (!nodes.empty()) {  ll curr\_node = nodes.front();  nodes.pop();  bool insert = true;  for (ll i = 0; i < curr.size(); i++) {  if (graph[curr\_node][curr[i]] != 1) {  insert = false;  break;  }  }  if (insert) {  curr.push\_back(curr\_node);  visited[curr\_node] = true;  qll new\_nodes;  for (ll i = 1; i <= n; i++)  if (graph[curr\_node][i] == 1 && !visited[i])  new\_nodes.push(i);  bfs(curr, new\_nodes);  visited[curr\_node] = false;  curr.pop\_back();  }  }  }  void solve() {  get >> n;  visited.assign(n + 3, false);  for (ll i = 1; i <= n; i++)  for (ll j = 1; j <= n; j++)  graph[i][j] = 0;  while (get >> a >> b) {  if (a == 0 && b == 0)  break;  graph[a][b] = 1;  graph[b][a] = 1;  }  vll curr;  qll nodes;  for (ll i = 1; i <= n; ++i)  nodes.push(i);  bfs(curr, nodes);  for (ll i = 0; i < ans.size(); i++)  put << ans[i] << "\t";  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  std::cout.tie(0);  int T;  T = 1;  // cin >> T;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task2.cpp | | #include <bits/stdc++.h>  #define paii std::pair<int,int>  #define pall std::pair<long long ,long long>  #define vpii std::vector<std::pair<int,int> >  #define vll std::vector<long long>  #define vll2 std::vector<std::vector<long long> >  #define vbool std::vector<bool>  #define vbool2 std::vector<std::vector<bool> >  #define qll std::queue<long long>  #define makep std::make\_pair  #define get std::cin  #define put std::cout  typedef long long ll;  vll2 graph;  vbool2 visited;  ll r, c, n, x, y, sr, sc;  vpii ans;  const int dr[] = {1, -1, 0, 0};  const int dc[] = {0, 0, 1, -1};  void bfs(vpii curr\_path) {  pall curr\_node = curr\_path.back();  ll curr\_r = curr\_node.first, curr\_c = curr\_node.second;  if (graph[curr\_r][curr\_c] == 2 && (ans.empty() || ans.size() > curr\_path.size())) {  ans = curr\_path;  return;  }  for (int i = 0; i < 4; i++) {  ll nxtr = curr\_r + dr[i], nxtc = curr\_c + dc[i];  if (nxtr > 0 && nxtr <= r && nxtc > 0 && nxtc <= c && graph[nxtr][nxtc] != -1 && !visited[nxtr][nxtc]) {  curr\_path.push\_back(makep(nxtr, nxtc));  visited[nxtr][nxtc] = true;  bfs(curr\_path);  visited[nxtr][nxtc] = false;  curr\_path.pop\_back();  }  }  if (!ans.empty())  return;  }  void solve() {  get >> r >> c >> n >> sr >> sc >> x >> y;  graph.assign(r + 3, vll(c + 3, 0));  visited.assign(r + 3, vbool(c + 3, false));  graph[x][y] = 2; // end node  while (n--) {  get >> x >> y;  graph[x][y] = -1;  }  vpii curr\_path;  curr\_path.push\_back(makep(sr, sc));  bfs(curr\_path);  put << ans.size() << endl;  for (ll i = 0; i < ans.size(); i++)  put << ans[i].first << "\t" << ans[i].second << endl;  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  std::cout.tie(0);  int T;  T = 1;  // cin >> T;  while (T--) {  solve();  }  return 0;  } |  |  | | --- | | Task3.cpp | | #include <bits/stdc++.h>  #define get std::cin  #define put std::cout  using ll = 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 min(T a, T b) {  return a < b ? a : b;  }  template <class T>  T max(T a, T b) {  return a > b ? a : b;  }  ll goal;  mll t;  void solve() {  get >> goal;  if (goal == 0) {  put << "None" << endl;  return;  }  ll goal\_p = goal, zero\_num = 0, ans = -1;  while (goal\_p > 0) {  if (goal\_p % 10 == 0)  zero\_num++;  else  break;  goal\_p /= 10;  }  if (zero\_num & 1) {  if (goal != 0)  ans = -1;  else  ans = 0;  }  else {  if (t.count(goal)) {  ans = t[goal];  } else {  ll num = goal\_p, MOD = 1;  if (num == 0)  MOD = 10;  else {  while (num) {  num /= 10;  MOD \*= 10;  }  }  ll curr\_mod = 1;  sll curr\_solutions;  curr\_solutions.insert(0);  while (curr\_mod < MOD) {  ll next\_mod = curr\_mod \* 10;  sll next\_solutions;  for (auto &it : curr\_solutions) {  for (int i = 0; i < 10; i++) {  ll new\_m = it + i \* curr\_mod;  if ((new\_m \* new\_m) % next\_mod == goal\_p % next\_mod) {  next\_solutions.insert(new\_m);  }  }  }  curr\_solutions = next\_solutions;  curr\_mod = next\_mod;  if (next\_solutions.empty())  break;  }  if (curr\_solutions.empty())  t[goal] = -1;  else  t[goal] = (\*curr\_solutions.begin()) \* qpow(10, zero\_num / 2);  ans = t[goal];  }  }  if (ans != -1)  put << ans << endl;  else  put << "None" << endl;  }  int main() {  std::ios::sync\_with\_stdio(0);  std::cin.tie(0);  std::cout.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。 |