

专业课程实验报告

课程名称：算法分析与设计

开课学期： 2021 至 2022 学年 第 1 学期

专业：软件工程 年级班级：19级3班

学生姓名：冯春霖 学号：222019321062074

实验教师：曹严元

计算机与信息科学学院 软件学院

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验项目名称 | | 分枝—限界算法 | | | |
| 实验时间 | | 2021 年 12 月 22 日 | 实验类型 | | □验证性 □设计性 □综合性 |
| 一、实验目的   1. 掌握分枝—限界的基本思想方法； 2. 了解适用于用分枝—限界方法求解的问题类型，并能设计相应动态规划算法； 3. 掌握分枝—限界算法复杂性分析方法，分析问题复杂性。   二、实验要求   1. 预习实验指导书及教材的有关内容，掌握分枝—限界的基本思想； 2. 严格按照实验内容进行实验，培养良好的算法设计和编程的习惯； 3. 认真听讲，服从安排，独立思考并完成实验。 | | | | | |
| 三、实验内容与设计（主要内容，操作步骤、算法描述或程序代码）  **对分支限界法的理解：**  分支限界法与回溯算法都是在解空间树上搜索问题的解，但不同的是，分支限界法的目标是求出满足约束条件的一个解或最优解，因而在解空间树的剪枝和扩展上与回溯法有所区别。在分支限界法中，使用限界函数进行合理剪枝，限界函数可在程序运行前进行预估，也可在程序运行过程中进行动态更新。分支限界法的每个节点仅有一次机会成文活节点，因而在限界函数的约束下进行扩展结点的选择是算法的关键。由于分支限界法逐层进行的特性，分支限界法一般采用BFS的方法进行扩展解空间树。  **选做问题2. 实现货郎担问题的分枝—限界算法**  **1. 选择合适的数据结构来表示问题**  货郎担问题(旅行售货员问题)对于使用邻接矩阵来表示间城市距离的图，需要一个二维数组进行存储。对于图中的节点，使用优先级队列来存储当前可用的活结点，并重载 < 运算符便于优先级队列的自动维护      **3. 根据分枝—限界法的基本原理，写出求解货郎担问题的伪码算法**    **4. 编码解决问题（C++）**      priority\_queue<MinHeapNode<Type>> pq;      MinHeapNode<Type> E, N;      Type bestc, cc, rcost, MinSum, \*MinOut, b;      int i, j;      a = G;      n = tn;      NoEdge = tNoEdge;      MinSum = 0;               //最小出边费用和      MinOut = new Type[n + 1]; //计算MinOut[i]=顶点i的最小出边费用      for (i = 1; i <= n; i++)      {          MinOut[i] = NoEdge;          for (j = 1; j <= n; j++)              if (a[i][j] != NoEdge && (a[i][j] < MinOut[i] || MinOut[i] == NoEdge))                  MinOut[i] = a[i][j];          if (MinOut[i] == NoEdge) //无回路              return NoEdge;          MinSum += MinOut[i];      }      //初始化      E.s = 0;      E.cc = 0;      E.rcost = MinSum;      E.x = new int[n];      for (i = 0; i < n; i++)          E.x[i] = i + 1;      bestc = NoEdge;      //搜索排列空间树      while (E.s < n - 1) //非叶结点      {          if (E.s == n - 2) //当前扩展结点是叶结点的父结点 再加2条边构成回路          {                 //所构成回路是否优于当前最优解              if (a[E.x[n - 2]][E.x[n - 1]] != NoEdge && a[E.x[n - 1]][1] != NoEdge &&                  (E.cc + a[E.x[n - 2]][E.x[n - 1]] + a[E.x[n - 1]][1] < bestc || bestc == NoEdge))              {                  //费用更小的路                  bestc = E.cc + a[E.x[n - 2]][E.x[n - 1]] + a[E.x[n - 1]][1];                  E.cc = bestc;                  E.lcost = bestc;                  E.s++;                  pq.push(E);              }              else                  delete[] E.x; //舍弃扩展结点          }          else //产生当前扩展结点儿子结点          {              for (i = E.s + 1; i < n; i++)                  if (a[E.x[E.s]][E.x[i]] != NoEdge)                  {                      //可行儿子结点                      cc = E.cc + a[E.x[E.s]][E.x[i]];    //当前费用                      rcost = E.rcost - MinOut[E.x[E.s]]; //更新最小出边费用和                      b = cc + rcost;                     //下界                      if (b < bestc || bestc == NoEdge)   //子树可能含最优解 结点插入最小堆                      {                          N.s = E.s + 1;                          N.cc = cc;                          N.lcost = b;                          N.rcost = rcost;                          N.x = new int[n];                          for (j = 0; j < n; j++)                              N.x[j] = E.x[j];                          N.x[E.s + 1] = E.x[i]; //得到新的路径                          N.x[i] = E.x[E.s + 1];                          pq.push(N); //加入优先队列                      }                  }              delete[] E.x; //完成结点扩展          }          if (pq.empty()) //堆已空              break;          E = pq.top(); //取下一扩展结点          pq.pop();      }      if (bestc == NoEdge) //无回路          return NoEdge;      for (i = 0; i < n; i++) //将最优解复制到v[1:n]          v[i + 1] = E.x[i];      while (pq.size()) //释放最小堆中全部结点      {          E = pq.top();          pq.pop();          delete[] E.x;      }      return bestc;    **5. 时空复杂度分析**  时间复杂度：对于最朴素的暴力解法，旅行售货员问题共有 n! 种可能的排列方式，需构造 n 层的解空间树，时间复杂度约为 O(N!) 。通过限界函数来进行剪枝，每层仅需扩展一个节点，时间复杂度的下届为 O(N) .  空间复杂度：通过邻接矩阵表示的图需要一个二维数组来存储节点间的位置关系，空间复杂度为 O(N ^ 2) ，同时需要一个长度最长为N的优先队列存储活节点，综合得空间复杂度为 O(N ^ 2) .  **6. 与动态规划法解题比较**  分支限界法是一种迭代式的解法，每次扩展解空间树后继续向下一层进行，而动态规划法是一种暴力的迭代解法，需要遍历整个解空间树才能得到答案。相比于动态规划法，分支限界法进行了大量的剪枝操作，极大的优化了效率 | | | | | |
| 四、测试数据和执行结果 （在给定数据下，执行操作、算法和程序的结果，可使用数据、图表、截图等给出）  测试环境：VSCode + MSVC C/C++(20)  输入测试样例进行测试，输入输出如图所示  输入第一行为节点数量n和边数e  随后的e行是从节点i到节点j的距离dist[i, j]        可以看到程序能正确计算出最短路径6 + 5 + 10 + 4 = 25，路径为1 -> 3 -> 2 -> 4 -> 1 | | | | | |
| 五、实验结果分析及总结（对实验的结果是否达到预期进行分析，总结实验的收获和存在的问题等）  通过本次实验，我对分支限界法有了一定的的了解，包括：  1. 掌握了了分支限界法的概念，对分支限界法的基础知识有了较好的了解，对回溯算法的解空间树、限界函数等概念有了清晰的认识，能够对使用分支限界法的算法进行复杂度分析，对分支限界法与回溯、动态规划等算法思想进行比较并了解其关系  2. 能够画出分支限界法的解空间树，对抽象的算法问题进行实际分析  3. 对C++的优先级队列等数据结构有了更好的了解  4. 能够较好的运用回溯算法解决学习生活中遇到的实际问题  5. 对所学知识加以运用，使用C++实现了使用分支界限法实现的解旅行售货员问题的算法 | | | | | |
| 教  师  评  阅 | 实验内容和设计（A-E）： | | |  | |
| 操作过程、算法或代码（A-E）： | | |  | |
| 实验结果（A-E）： | | |  | |
| 实验分析和总结（A-E）： | | |  | |
| 实验成绩（A-E）：  反馈评语： | | | | |