广东金融学院实验报告

课程名称：算法分析与设计

装订线

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验编号  及实验名称 | 算法分析与设计实验12 | | | 系 别 | 互金学院 |
| 姓 名 | 林旋华 | 学 号 | 181543306 | 班 级 | 1815433 |
| 实验地点 | 电教503 | 实验日期 | 2020.12.18 | 实验时数 | 2 |
| 指导教师 | 郭艺辉 | 同组其他成员 | 无 | 成 绩 |  |
| 1. 实验目的及要求 2. 掌握优先队列式分支限界法的基本思想以及基本原理。 3. 掌握使用优先队列式分支限界法求解问题的一般特征以及步骤。 4. 掌握优先队列式分支限界法算法设计方法以及复杂性分析方法。 5. 掌握使用优先队列式分支限界法求解0-1背包问题的算法设计思想、算法计算过程以及程序编码实现。 | | | | | |
| 1. 实验环境及相关情况（包含使用软件、实验设备、主要仪器及材料等）   1) 操作系统：Windows操作系统  2) 开发工具：Eclipse、JDK  3) 开发语言：Java | | | | | |
| 1. 实验内容及步骤（包含简要的实验步骤流程）   0-1背包问题的问题提出是，有*n*个物品，其中物品*i*的重量是，价值是，有一容量为*C*的背包，要求选择若干物品装入背包，使装入背包的物品总价值达到最大。0-1背包问题中，物品*i*在考虑是否装入背包时只有两种选择，即要么全部装入背包，要么全部不装入背包，不能只装入物品*i*的一部分，也不能将物品*i*装入背包多次。此问题的形式化描述是：给定，要求找出*n*元0-1向量,, ，使得目标函数达到最大，且要满足约束条件。设，使用优先队列式分支限界法求解此问题。   1. 写出算法实现代码并拷屏程序的运行结果。 2. 对算法做复杂性分析。 | | | | | |
| 1. 实验结果（包括程序或图表、结论陈述、数据记录及分析等，可附页）   1）import java.util.Stack;  class HeapNode {  double upbound; // 结点的价值上界  double value; // 结点所对应的价值  double weight; // 结点所相应的重量  int level; // 活节点在子集树中所处的层序号  public HeapNode() {}  }  // 分支限界法实现01背包问题  public class BBKnapsack {  int[] weight;  int[] value;  int max; // 背包的最大承重量  int n;  double c\_weight; // 当前背包重量  double c\_value; // 当前背包价值  double bestv; // 最优的背包价值  Stack<HeapNode> heap;  public BBKnapsack() {  weight = new int[] { 14, 7, 15, 9, 20, 0 };  value = new int[] { 6, 6, 8, 15, 18, 0 };  max = 40;  n = weight.length - 1;  c\_weight = 0;  c\_value = 0;  bestv = 0;  heap = new Stack<HeapNode>();  }  // 求子树的最大上界  private double maxBound(int t) {  double left = max - c\_weight;  double bound = c\_value;  // 剩余容量和价值上界  while ((t < n) && (weight[t] <= left)) {  left -= weight[t];  bound += value[t];  t++;  }  if (t < n) {  bound += ((value[t] / weight[t]) \* left); // 装填剩余容量装满背包  }  return bound;  }  // 将一个新的活结点插入到子集树和最大堆heap中  private void addLiveNode(double upper, double cvalue, double cweight,  int level) {  HeapNode node = new HeapNode();  node.upbound = upper;  node.value = cvalue;  node.weight = cweight;  node.level = level;  if (level <= n) {  heap.push(node);  }  }  // 利用分支限界法，返回最大价值bestv  private double knapsack() {  int i = 0;  double upbound = maxBound(i);  // 调用maxBound求出价值上界，bestv为最优值  while (true) // 非叶子结点  {  double wt = c\_weight + weight[i];  if (wt <= max) // 左儿子结点为可行结点  {  if ((c\_value + value[i]) > bestv) {  bestv = c\_value + value[i];  }  addLiveNode(upbound, c\_value + value[i], c\_weight + weight[i], i + 1);  }  upbound = maxBound(i + 1);  if (upbound >= bestv) { // 右子树可能含最优解  addLiveNode(upbound, c\_value, c\_weight, i + 1);  }  if (heap.empty()) {  return bestv;  }  HeapNode node = heap.peek();  // 取下一扩展结点  heap.pop();  //System.out.println(node.value + " ");  c\_weight = node.weight;  c\_value = node.value;  upbound = node.upbound;  i = node.level;  }  }  public static void main(String[] args) {  BBKnapsack knap = new BBKnapsack();  double opt\_value = knap.knapsack();  System.out.println("最大价值是:" + opt\_value);  }  }     1. 复杂度：O（2n） | | | | | |
| 1. 实验总结（包括心得体会、问题回答及实验改进意见，可附页）   在做本次实验之前，我对分支限界法的原理并不是很理解，经过查看课件及网上查找资料，同时结合自己对回溯法等的理解，我对分支限界法有了一个较好的理解，知道了两种主要的分支限界法及分支限界法如何应用于解0-1背包问题。在查找资料的过程中，我查看了许多网上的别人的代码实现，结合课本上的代码完成了该实验。通过本次试验，我基本上掌握了优先队列分支限界法解0-1背包问题的原理，同时锻炼了自己动手编写及调试代码的能力，分支限界法还是比较抽象，课后我会多加练习，争取尽快掌握。 | | | | | |
| 六、教师评语  1、完成所有规定的实验内容，实验步骤正确，结果正确；  2、完成绝大部分规定的实验内容，实验步骤正确，结果正确；  3、完成大部分规定的实验内容，实验步骤正确，结果正确；  4、基本完成规定的实验内容，实验步骤基本正确，所完成的结果基本正确；  5、未能很好地完成规定的实验内容或实验步骤不正确或结果不正确。  6、其它：  评定等级：优秀 良好 中等 及格 不及格  教师签名：郭艺辉 | | | | | |