

**实验报告封面**

**课程名称： 算法课程设计 课程代码： GF2001**

**任课老师： 实验指导老师:**

# 实验报告名称：

学生姓名：

学号： 教学班：

递交日期：

签收人：

我申明，本报告内的实验已按要求完成，报告完全是由我个人完成，并没有抄袭行为。我已经保留了这份实验报告的副本。

申明人(签名):

**实验报告评语与评分：**

**评阅老师签名：**

**内页写作格式**

**一、实验名称：**要用最简练的语言反映实验的内容。

**二、实验日期：**写明做实验的具体年、月、日及组別。

**三、实验目的：**使用简洁的文字或关键字来敘述，是以怎样的目的作此实验的。

**四、实验环境：**实验的操作系统和软件等。

**五、实验的步骤和方法：**这是实验报告极其重要的内容。这部分要写明经过哪几个步骤。

**六、数据记录和计算：**指从实验中测到的数据以及计算结果。

**七、实验结果或结论：**即根据实验过程中所见到的现象和测得的数据，得出结论。

**八、备注或说明**：可写上实验成功或失败的原因，实验后的心得体会、建议等。

|  |  |
| --- | --- |
| **1、实验名称** | **实验3 贪心法** |
| **2、实验日期** |  |
| **3、实验目的** | |
| 1. 掌握算法设计概念及流程 2. 掌握算法评价的方法 | |
| **4、实验环境** | |
| 1、硬件环境：i386兼容机，2G剩余硬盘空间；  2、软件环境：Windows XP以上兼容系统；  3、使用资源：开发环境VC++2017。 | |
| **5、实验的步骤和方法** | |
| 1. **假设有面值为5元、2元、1元、5角、2角、1角的货币，需要找给顾客4元6角现金，使付出的货币的数量最少。** 2. **编程求解0-1背包问题：有7个物品{"A",35,10}, {"B",30,49}, {"C",60,30}, {"D",50,50},{"E",40,35}, {"F",10,40}, {"G",25,30}，背包容量是150，求装入背包的物品和获得的价值。** 3. **多机调度问题：设有*n*个独立的作业{1, 2, …, *n*}，由*m*台相同的机器{*M*1, *M*2, …, *Mm*}进行加工处理，作业*i*所需的处理时间为*ti*（1≤*i*≤*n*），每个作业均可在任何一台机器上加工处理，但不可间断、拆分。多机调度问题要求给出一种作业调度方案，使所给的*n*个作业在尽可能短的时间内由*m*台机器加工处理完成。如：设7个独立作业{1, 2, 3, 4, 5, 6, 7}由3台机器{*M*1, *M*2, *M*3}加工处理，各作业所需的处理时间分别为{2, 14, 4, 16, 6, 5, 3}，求作业调度安排。**   完成上面的练习后，填写实验报告，将**源代码文件、实验报告、运行结果截图，打包“学号姓名小班”**上交到智慧树。 | |
| **6、数据记录和计算** | |
| （算法的流程图请贴在此处，还有关键算法，产生的结果图）  第一题流程图：  01-liuchengtu  第一题关键算法代码：  void payBack(int yuan, int jiao)  {  int count = 0; //定义整数变量count，表示找现金的货币数量  while (yuan >= 5) //当需要找的现金的元部分大于等于5元时  {  yuan -= 5; //减去5元  count++; //货币数量加1  cout << "找了1张5元的货币。" << endl; //输出  }  while (yuan >= 2) //当需要找的现金的元部分大于等于2元时  {  yuan -= 2; //减去2元  count++; //货币数量加1  cout << "找了1张2元的货币。" << endl; //输出  }  while (yuan >= 1) //当需要找的现金的元部分大于等于1元时  {  yuan -= 1; //减去1元  count++; //货币数量加1  cout << "找了1张1元的货币。" << endl; //输出  }  while (jiao >= 5) //当需要找的现金的角部分大于等于5角时  {  jiao -= 5; //减去5角  count++; //货币数量加1  cout << "找了1张5角的货币。" << endl; //输出  }  while (jiao >= 2) //当需要找的现金的角部分大于等于2角时  {  jiao -= 2; //减去2角  count++; //货币数量加1  cout << "找了1张2角的货币。" << endl; //输出  }  while (jiao >= 1) //当需要找的现金的角部分大于等于1角时  {  jiao -= 1; //减去1角  count++; //货币数量加1  cout << "找了1张1角的货币。" << endl; //输出  }  }  第一题结果图：    第二题流程图：  02-liuchengtu  第二题关键算法代码：  int knapsack() {  memset(dp, 0, sizeof(dp));  for (int i = 1; i <= n; i++) {  for (int j = 1; j <= w; j++) {  if (wt[i-1] <= j) {  dp[i][j] = max(val[i-1] + dp[i-1][j-wt[i-1]], dp[i-1][j]);  } else {  dp[i][j] = dp[i-1][j];  }  }  }  int j = w;  vector<string> selected;  for (int i = n; i > 0 && j > 0; i--) {  if (dp[i][j] != dp[i-1][j]) {  selected.push\_back(items[i-1]);  j -= wt[i-1];  }  }  cout << "Selected items: ";  for (int i = selected.size()-1; i >= 0; i--) {  cout << selected[i] << " ";  }  cout << endl;  return dp[n][w];  }  第二题结果图：    第三题流程图：  第三题关键算法代码：  void greedy() {  // 对作业的处理时间进行排序  sort(t, t + n);  // 定义机器的处理时间  int machine[m] = {0};  // 遍历每个作业  for (int i = 0; i < n; i++) {  // 寻找处理时间最小的机器  int min\_machine = 0;  for (int j = 1; j < m; j++) {  if (machine[j] < machine[min\_machine]) {  min\_machine = j;  }  }  // 将作业分配给该机器进行处理  machine[min\_machine] += t[i];  }  // 输出机器的处理时间  for (int i = 0; i < m; i++) {  cout << "机器M" << i + 1 << "的处理时间为：" << machine[i] << endl;  }  }  第三题结果图： | |
| **7、实验思考(从算法特点，使用场景，实现思路方面进行思考)** | |
|  | |
| **8、实验结果或结论（**实验结果怎么样？你从这个实验你学会了什么？得出了什么结论？）  **）** | |