# 2022秋-算法设计与分析

# 贪心算法分析实验报告

|  |  |
| --- | --- |
| 姓名： |  |
| 学号： |  |

|  |
| --- |
| 2022年12月06日 |

目录

[2022秋-算法设计与分析 1](#_Toc121250569)

[贪心算法分析实验报告 1](#_Toc121250570)

[1. 实验要求 2](#_Toc121250571)

[2. 实验报告 2](#_Toc121250572)

[2.1 基于两种贪心策略的贪心算法实现 2](#_Toc121250573)

[2.1.1 基于负载均衡贪心策略的贪心算法实现 2](#_Toc121250574)

[2.1.2 基于逆降贪心策略的贪心算法实现 3](#_Toc121250575)

[2.1.3 基于两种贪心策略的贪心算法执行 4](#_Toc121250576)

[2.2 基于回溯法的最优解求解算法实现 5](#_Toc121250577)

[2.3 三种算法对随机产生的大量测试样本求解 6](#_Toc121250578)

[2.4 处理机数量和任务数量对三种算法求解的影响 6](#_Toc121250579)

[2.4.1 处理机数量对三种算法求解的影响 6](#_Toc121250580)

[2.4.2 任务数量对三种算法求解的影响 7](#_Toc121250581)

[2.5 构造问题输入使贪心算法结果接近最差 8](#_Toc121250582)

[3. 实验总结 8](#_Toc121250583)

# 1. 实验要求

对多机调度算法进行分析，理解算法的工作流程。具体要求如下：

1. 针对多机调度问题，实现基于两种贪心策略的贪心算法；

2. 针对多机调度问题，实现遍历的最优解求解算法（也可以用回溯法）；

3. 以处理机数量作业数量为输入规模，固定, ，随机产生大量测试样本，用两种贪心算法分别求解，并计算最优解（无法在合理时间内完成最优解计算则记录为“最优解求解失败”），并进行记录；

4. 改变, ，对不同组合的结果进行对比分析，通过统计画图，与理论值进行对照分析，并撰写实验报告；

5. 针对两种贪心策略，构造问题输入，使得贪心算法结果接近最差。结合证明过程展开讨论。

6. 附加：模拟一个GPU集群在线调度问题，该集群有块GPU，共享开放给全校师生。用户提交任务的时间点符合泊松分布，单个任务使用单块GPU所需的时间符合均匀分布。假设提交的任务均具有高度并行性，可以无损失地拆分到任意多块GPU并行执行。用户要求的完成时间为任务提交时间加上单块GPU执行任务所需时间。请针对该场景：

- 模拟生成多组任务集（注意考虑轻负载、中等负载、重负载等不同情况）；

- 设计两种以上调度策略；

- 考虑两个指标：集群利用率，用户平均延迟，使用调度策略进行模拟，对结果进行分析对比，撰写实验报告。

# 2. 实验报告

多机调度问题：给定台相同的机器：，及个作业：，需时间，每台机器一次只能执行一个作业，如何分配任务才能尽快完成所有作业？

## 2.1 基于两种贪心策略的贪心算法实现

### **2.1.1 基于负载均衡贪心策略的贪心算法实现**

负载均衡：按输入顺序分配作业，把每项作业分配给当前负载最轻的机器。

定义函数greedy1\_solve()来实现基于负载均衡贪心策略的贪心算法。函数返回int类型值，表示该算法计算出多机调度问题所需时间，即每台机器完成所分配的任务所需时间的最大值。函数使用4个参数，分别为二维数组类型的machines，使用传引用方式传参，以及一位数组类型的tasks，int类型的m和n，均使用传值方式传参。函数执行时，修改machines数组，使数组machines[i]表示第i+1台机器所执行的任务时间，用于函数返回后输出展示。tasks表示所需执行的任务时间，m为机器数量，n为任务数量。

函数内部首先定义一维数组len，用于记录各机器负载，即当前各机器所需执行的所有任务时间之和。遍历tasks数组给各机器分配任务，首先遍历len数组，找出当前负载最小的机器，将任务分配给该机器，更新len和machines数组。遍历tasks完毕后，遍历一遍len数组找出负载最大机器，该负载数即为函数返回值。

具体实现如下图所示：

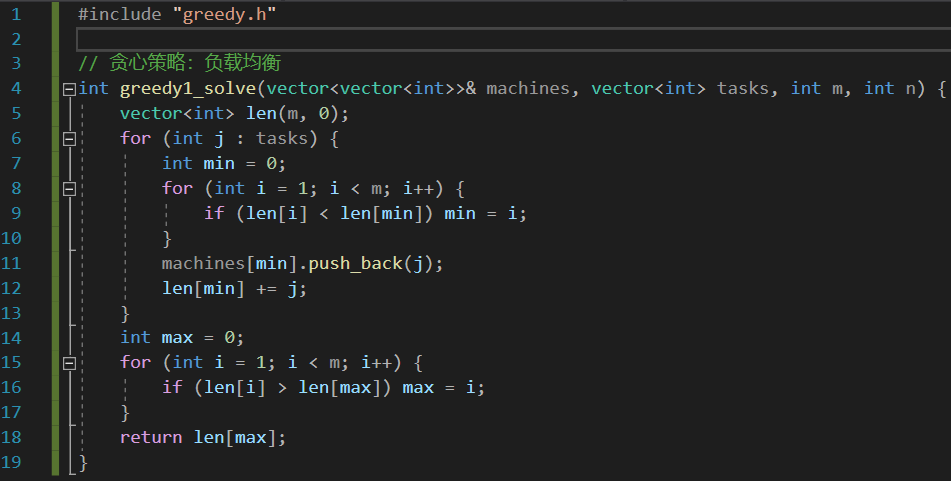


图1：greedy1\_solve()实现

### **2.1.2 基于逆降贪心策略的贪心算法实现**

逆降贪心策略：首先按处理时长从大到小对作业重新排序，然后运用贪心策略分配任务。

和基于负载均衡贪心策略的贪心算法实现一样，定义greedy2\_solve()来实现基于逆降贪心策略的贪心算法。由于两种贪心策略尽算法逻辑不同，所以采用与greedy1\_solve()相同的参数和返回值。

函数内部首先对task数组按从大到小顺序排序，可以使用STL标准sort()函数和greater<int>()函数实现排序。然后调用greedy1\_solve()即可。

具体实现如下图所示：

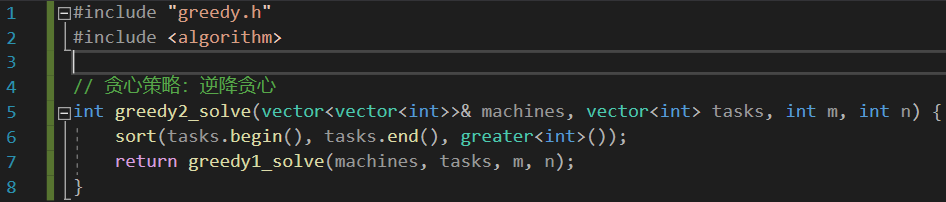


图2：greedy2\_solve()实现

### **2.1.3 基于两种贪心策略的贪心算法执行**

在main()函数中定义m、n、machines和tasks，分别调用greedy1\_solve()函数与greedy2\_solve()函数执行，并将结果输出，与标准结果对比发现完全相同，表明算法实现正确。

main()具体实现如下图所示：

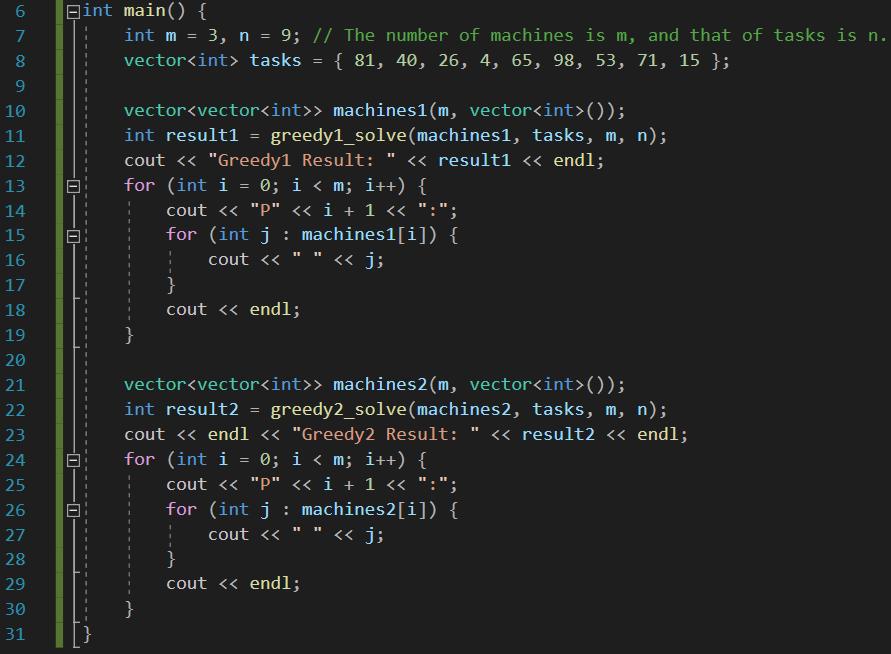


图3：main()实现

两种算法运行结果与标准结果比较如下图：

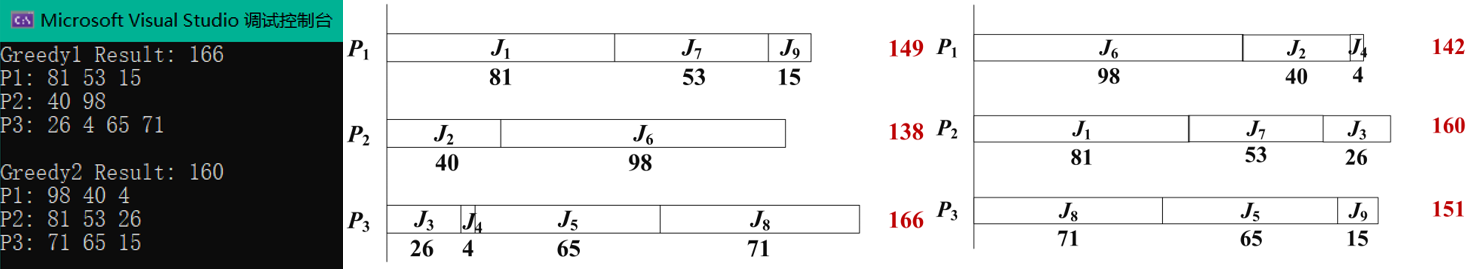


图4：运行结果与标准结果比较

## 2.2 基于回溯法的最优解求解算法实现

定义函数backTracing()来使用回溯法求解多机调度问题的最优解。采用深度优先搜索进行回溯，首先在backTracing()函数内将传入参数赋值给全局变量，然后调用search()函数进行深度优先搜素。使用dep变量记录搜索深度，最大深度为任务数量n。当达到任务数量n时，如果当前解优于最优解，则更新最优解和最优解时各机器的任务分配情况；当未达到任务数量n时，遍历所有机器，将当前任务dep分配给每个机器一次，分配后进行search()递归，递归后取消分配，以实现回溯。

具体实现如下图所示：

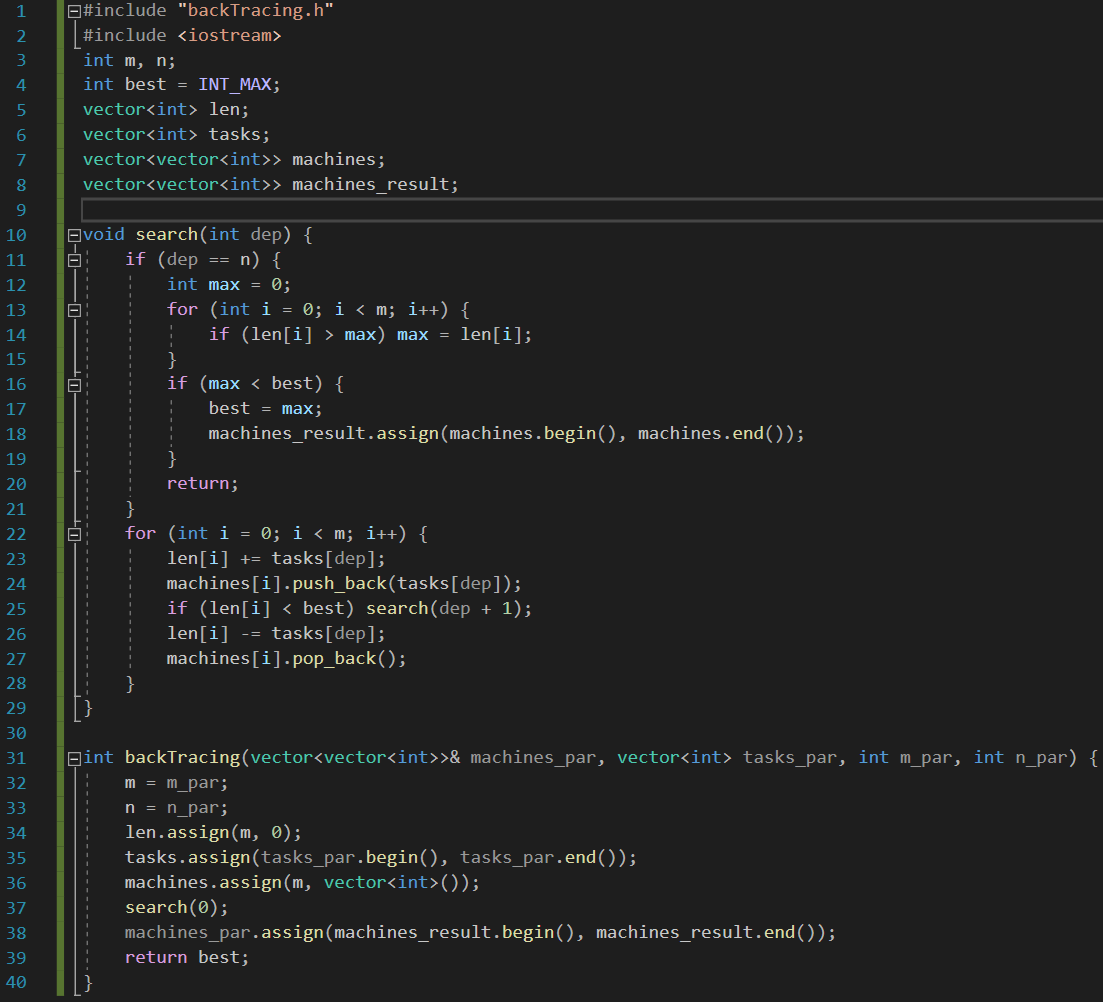


图5：backTracing()实现

运行回溯法，得到结果与标准结果进行比较，发现运行结果是该情况下的一种最优解，完成所有任务所需时间最短。

回溯法运行结果与标准结果比较如下图所示：

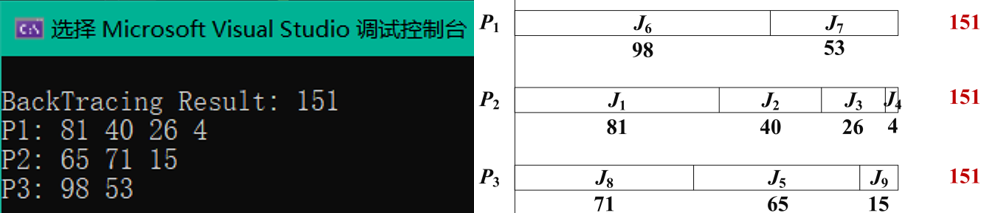


图6：运行结果与标准结果比较

## 2.3 三种算法对随机产生的大量测试样本求解

固定处理机数量m为3，任务数量n为9，随机产生10个任务列表，交由两种贪心算法进行处理，使用回溯算法计算出最优解，进行记录。记录结果如下：

表1：三种算法的求解结果统计

|  |  |  |  |
| --- | --- | --- | --- |
| 任务列表序号 | 负载均衡贪心法 | 逆降贪心法 | 回溯法 |
| 1 | 169 | 156 | 148 |
| 2 | 184 | 185 | 148 |
| 3 | 173 | 143 | 142 |
| 4 | 166 | 136 | 135 |
| 5 | 181 | 154 | 135 |
| 6 | 174 | 162 | 135 |
| 7 | 174 | 144 | 135 |
| 8 | 136 | 137 | 132 |
| 9 | 174 | 152 | 132 |
| 10 | 162 | 156 | 132 |

可以看出，使用回溯法求出的最优解总是优于使用贪心法求出的近似解。而两种贪心法中，逆降贪心法求出的解相比于负载均衡贪心法求出的解，有8次更优，2次更劣，可见，总体而言，逆降贪心法优于负载均衡贪心法。原因是，逆降贪心法先对随机产生的任务进行了排序，然后依次按照负载均衡贪心法分配任务，使得分配出的任务更加的“平均”，更加的“负载均衡”。

## 2.4 处理机数量和任务数量对三种算法求解的影响

### **2.4.1 处理机数量对三种算法求解的影响**

固定任务数量n为9，设置处理机数量为1-5，随机产生大量任务，求出三种算法在不同处理机数量下，完成n个任务所需的平均时间，得到结果如下图所示：

图7：不同处理机数量下三种算法执行任务的平均时间

可以看到：

1. 在处理机数量为1时，三种算法的求解出任务执行时间是相同的，因为只有1个处理机，所以按任何顺序执行任务得到的结果都应该是相同的，符合理论值。

2. 随着处理机数量的增加，三种算法执行任务时间都在下降，因为处理机越多，并行处理任务的能力越强，所以执行任务的时间越短，也符合理论值。

3. 在处理机数量大于1时，回溯法处理任务时间最短，因为回溯法求出的是最优解，也符合理论值。

4. 负载均衡贪心法总体上慢于逆降贪心法，因为逆降贪心法先对任务进行排序，相当于更加的“负载均衡”，也符合理论值。

### **2.4.2 任务数量对三种算法求解的影响**

固定处理机数量为3，设置任务数量为1-9，各重复10次，求出在不同任务数量下，各算法求解出的最优处理时间，得到结果如下图所示：

图8：不同任务数量下三种算法执行任务的平均时间

可以看到：

1. 在任务数量不大于处理机数量，即任务数量小于等于3时，三种算法求解出的结果是一样的，因为这种情况下，解决任务所需的最短时间取决于各任务所需时间的最长值，符合理论值。

2. 随着任务数量的增加，三种算法求解出所需时间都会增长，因为任务越多，处理时间越多，也符合理论值；

3. 在任务数量大于处理机数量时，回溯法处理任务时间更短，因为回溯法可以求出最优解，也符合理论值；

4. 负载均衡贪心法总体上慢于逆降贪心法，因为逆降贪心法先对任务进行排序，相当于更加的“负载均衡”，也符合理论值。

## 2.5 构造问题输入使贪心算法结果接近最差

m台机器的多机调度问题，对任意输入I，设G(I)，DG(I)，O(I)分别为负载均衡贪心调度、逆降贪心调度和最优调度的完成时间，则有：

显然，m越小，贪心算法表现越好，越接近与最优解；m越大，贪心算法表现越差。因此，问题输入可以将机器数量m设置为较大数，进行求解。

经过多次尝试，考虑到运行时间较长，最终设置m为5，n为15，此时，G(I)应接近于1.8O(I)，DG(I)应接近于1.4O(I)。实际运行结果如下图所示：

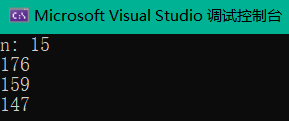


图9：运行结果

可以看到，实际运行结果并不符合倍数关系，猜测原因是n设置太小导致随机性太大。由于在m为5，n为15的情况下，运行时间超过了5分钟，所以无法将n设置过大。但可以看出，随着m的增大，倍数会越来越接近2和1.5，即，机器数量m越大，贪心算法结果越差。

# 3. 实验总结

1. 通过此次实验，我对贪心算法在多机调度问题上的应用有了更深的认识，也学会了使用回溯法求解多机调度问题的最优解。