暨南大学本科实验报告专用纸

课程名称 算法分析与设计实验 成绩评定

实验项目名称 工作分配问题 指导教师 李展

实验项目编号 实验十一 实验项目类型 综合性 实验地点

学生姓名 张印祺 学号 201801948

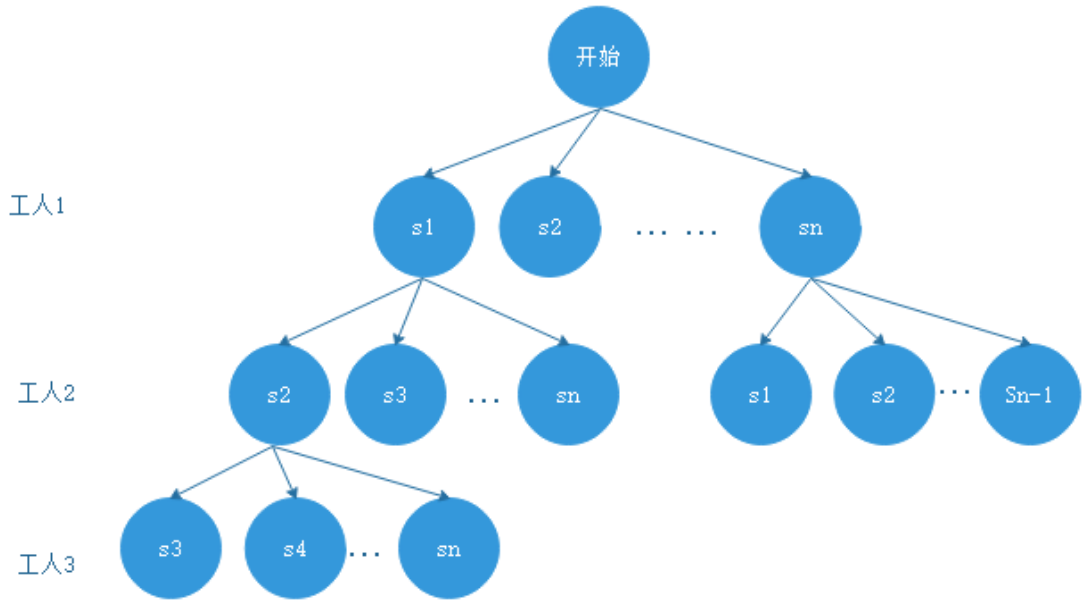
学院 信息科学技术 系 计算机科学 专业 网络工程

实验时间 2020 年 5 月 27 日

一、问题描述

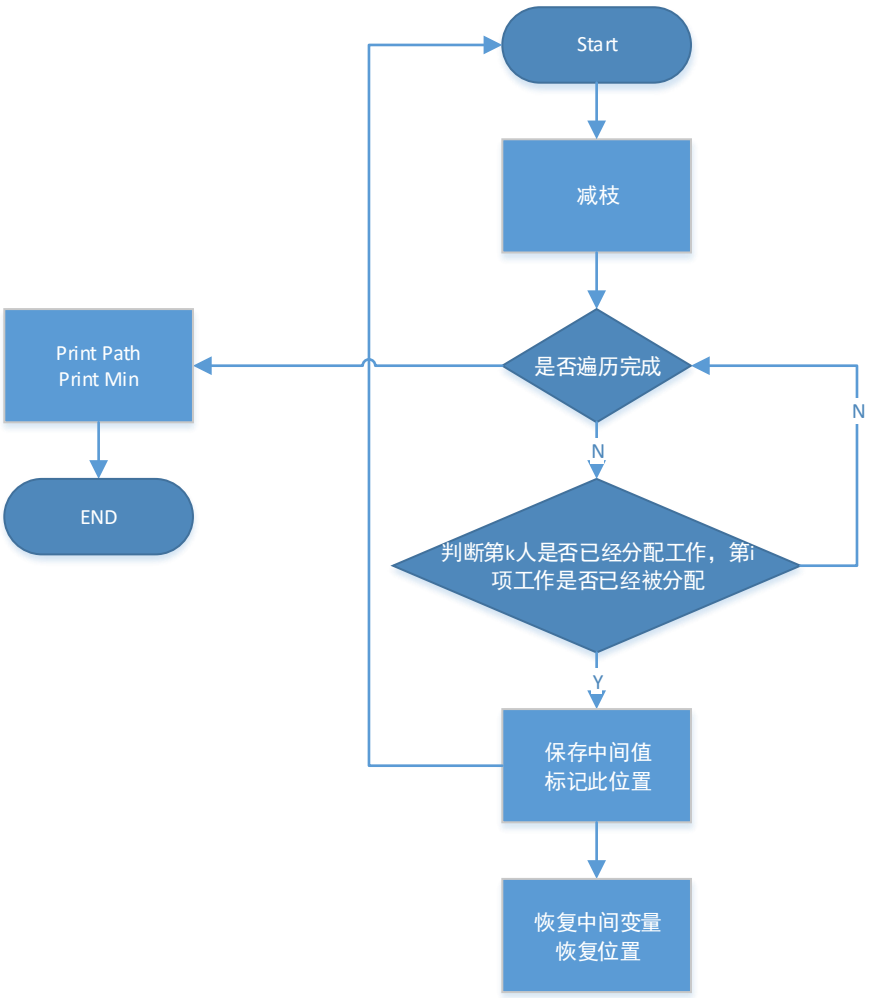
设有 n 件工作分配给 n 个人，将工作 i 分配给第 j 个人所需的费用为 cij。试设计一个算法，为每个人都分配 1 件不同的工作，并使总费用最低。

二、算法思路



这是一个排列组合问题。我们假设工人都排队接受工作，那么我们只用考虑工分配作的排序问题，因此，一共有 n!种分配情况。可以用回溯法遍历每一种情况选出最优解。这里的界限是总费用最低，可以一边遍历一边记录已经已经花费的费用，如果新添加的结点费用大于已存在的解的费用，则可以删除该结点对应的子树。

三、流程图



四、测试结果

```
a = [[10,2,3], [2,3,4], [3,4,5]]  # 9
b = [[50,43,1,58,60], [87,22,5,62,71], [62,98,97,27,38], [56,57,96,73,71], [92,36,43,27,95]]
c = [[50,43,1,58], [87,22,5,62], [62,98,97,27], [56,57,96,73]]
d = [[8,6,13,4], [9,5,7,15], [5,2,7,11], [14,12,16,13]]
e = [[85,45,12,16], [71,59,86,41], [88,45,23,76], [86,78,39,51]]
result=[a,b,c,d,e]
for i in result:
    solu = Solution(i)
    solu.getRul()
```

```
人员依次选择的工作为:  [2, 1, 3]
最短时间为:  9
人员依次选择的工作为:  [3, 2, 5, 1, 4]
最短时间为:  144
人员依次选择的工作为:  [3, 2, 4, 1]
最短时间为:  106
人员依次选择的工作为:  [4, 3, 2, 1]
最短时间为:  27
人员依次选择的工作为:  [4, 1, 2, 3]
最短时间为:  171
```

五、实验总结

算法在遍历解空间树时，采用了深度优先算法，利用栈来存放结点，当栈满时保存工作安排序列并且记录最低费用。

每访问一个结点都要用一个数组记录该结点已入栈，因此子树中的结点才不会再次出现同样的工作。但是每次查找数组都要耗时 o(n)，解空间一共有 n!种解，每遍历耗时 o(n!)，因此 T(n) = o(n*n!)。

六、源代码

```
import numpy as np

class Solution:
    def __init__(self, p):
        self.n = len(p)  # 获取 n 大小
        self.minP = float("inf")  # 记录最小的 花费
        self.tempMinP = 0  # 保存中间值
        self.p = p      # 人员花费时间表
        self.t = np.zeros((self.n, self.n))  # 标记数组全为零，
        self.b = 0      # 记录是否已经走完一趟（0 无，1 走完一次）
        self.path = []  # 负责记录路径
    def getMinP(self, k=0):
        if self.b == 1 and self.tempMinP >= self.minP: return  # 进行减
枝
        if k == self.n:
            self.b = 1
            if self.tempMinP < self.minP:
                self.minP = self.tempMinP
                self.path = [np.argmax(v)+1 for v in self.t]
            return

        for i in range(self.n):
            if (1 not in self.t[k]) and (1 not in self.t[:,i]):  # 判断
第 k 人是否已经分配工作，第 i 项工作是否已经被分配
                self.tempMinP += self.p[k][i]  # 添加一个值
                self.t[k, i] = 1               # 标记此位置已经走过
                self.getMinP(k + 1)            # 添加下一个人员
                self.tempMinP -= self.p[k][i]  # 恢复中间变量
                self.t[k, i] = 0               # 恢复位置
    def getRul(self):
        self.getMinP()
        print('人员依次选择的工作为：',self.path)
print('最短时间为：',self.minP)
```