姓名：蒋志远
班级：软工1506
学号：U201517149

背景问题

组装某产品有六道工序，由一条装配线完成。装配线由一系列工作站组成，被装配的产品在装配线上流动，每个工作站都要完成一道或几道工序，这些工序按先后次序在各工作站上完成。关于这些工序有如下的数据：

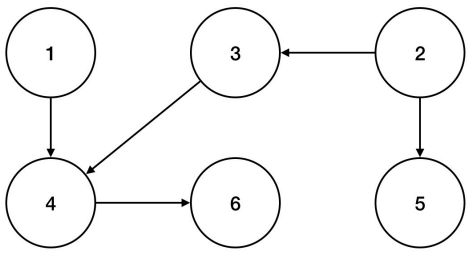
工序	所需时间（分）	前驱工序
1	3	无
2	5	无
3	2	2
4	6	1，3
5	8	2
6	3	4

另外工艺流程特别要求，在任一给定的工作站上，不管完成哪些工序，可用的总时间不能超过10分钟。如何将这些工序分配给各工作站，以使所需的工作站数为最少?

数学模型

问题分析

根据问题的描述，可以在画出如下的工序依赖关系图：



圈中的编号表示各个工序的序号，箭头的指向表示之间的依赖关系

可以考虑使用枚举法来寻找最优的解，可以找出所有符合要求的拓扑排序序列，而后利用贪心算法，在序列中按顺序将工序分组，一组的总工序时间不超过10小时即可。

如一种可行的排序：

$$1, 2, 3, 5, 4, 6$$

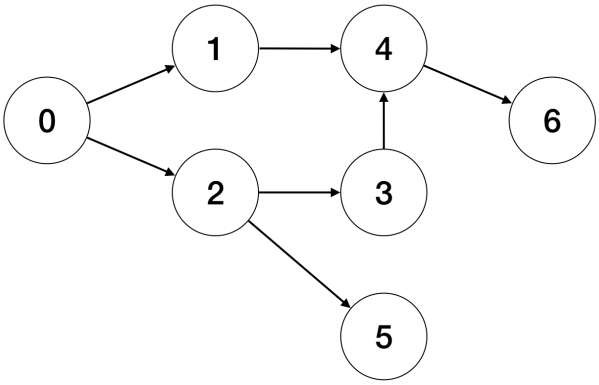
我们可以以一下的方式进行分组：

$$[1, 2, 3], [5], [4, 6]$$

目标就是找到所用分组数最少的分组方式

模型求解

我们先用关系矩阵的方式将关系图表示出来，这里添加了一个0节点，方便算法的进行，关系图如下



Python

```python
import numpy as np
from itertools import permutations

graph = np.zeros((7, 7), dtype=np.int8)
graph[0, 1] = 1
graph[0, 2] = 1
graph[1, 4] = 1
graph[2, 3] = 1
graph[2, 5] = 1
graph[3, 4] = 1
graph[4, 6] = 1

time_table = [0, 3, 5, 2, 6, 8, 3]
best_station = [[0 for i in range(10)]]
```

用贪心的算法对拓扑排序后的序列进行分组

```python
def find_station_count(seq):
    count, time = 0, 0
    stations = [[]]
    idx = 0
    seq = seq[1:] #去掉序列中的0
    for i in seq:
        time += time_table[i]
        stations[idx].append(i)
        if time > 10:
            time = time_table[i]
            stations[idx].pop()
            stations.append([i])
            idx += 1
    return stations
```

用递归的方式将所有的序列找出

```python
def find_seq(graph, sequence, unused_set):
    if len(unused_set) == 0:   #递归终止条件
        # print('seq:' + str(sequence))
        global best_station
        stat = find_station_count(sequence)
        if len(stat) < len(best_station[0]): #寻找分组数最少的解
            best_station = [stat]
        elif len(stat) == len(best_station[0]):
            best_station.append(stat)
        return
    node_without_parent = []
    for i in unused_set:
        if np.sum(graph[:, i]) == 0:
            node_without_parent.append(i)

    for nop in node_without_parent:
        updated_set = unused_set - set([nop])
        other_graph = graph.copy()
        for col in range(7):
            other_graph[nop, col] = 0
        temp_seq = list(sequence)
        temp_seq.append(nop)
        find_seq(other_graph, temp_seq, updated_set)
```

调用查找结果并打印

```python
def main():
    find_seq(graph, [], set([i for i in range(7)]))
    print('the best stations:')
    for i, stat in zip(range(len(best_station)), best_station):
        print('solution ' + str(i + 1))
        for sub_stat in stat:
            print(sub_stat, end='')
        print()


if __name__ == '__main__':
    main()
```

求解结果

```
the best stations:
solution 1, station_count:3
[1, 2, 3][4, 6][5]
solution 2, station_count:3
[1, 2, 3][5][4, 6]
solution 3, station_count:3
[1, 2][5, 3][4, 6]
solution 4, station_count:3
[2, 1, 3][4, 6][5]
solution 5, station_count:3
[2, 1, 3][5][4, 6]
solution 6, station_count:3
[2, 1][5, 3][4, 6]
solution 7, station_count:3
[2, 3, 1][4, 6][5]
solution 8, station_count:3
[2, 3, 1][5][4, 6]
```

结果分析

从结果可以看出，可以使用的最少工作站数为3，流水线工序有多种排列方法

模型检验

试取Solution 4的结果 `[2, 1, 3][4, 6][5]` 进行分析检验：

第一个工作站时间：$5+3+2=10 \leq 10$
第二个工作站时间：$6+3=9 \leq 10$
第三个工作站时间：$8 \leq 10$
观察也能发现此序列符合依赖关系要求