**Problem Statement**

**问题描述**

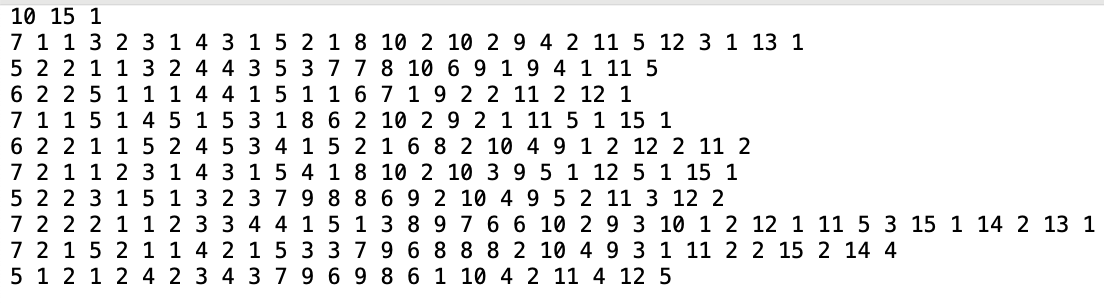
Flexible Job Shop是生产调度中一个经典的NP-hard问题：假设我们有n个待加工的任务J={J1,J2,…,Jn}及m台加工机器M={M1,M2,…,Mm}，每个任务i都有自己的加工工序{Oi1,Oi2,…,Oik}（k为任务i的工序数，不同任务的工序数可能不同），每道工序o都可以在机器集M的一个子集Mo中选择任意一台机器完成，且在不同机器上加工时间可能不同。每个机器同一时间仅能加工一个任务，且当一个任务在一台机器上开始加工后不能中断，所有机器在时刻0都是空闲状态。我们的优化目标是最小化最大完工时间（Makespan），最大完工时间即为最后一个被完成的任务的完工时刻。

以上为经典的FJS问题，然而现在我们在此基础上还需考虑一个插单问题：在根据初始任务信息完成调度方案的设计后，我们又得到了一个新的任务，该任务同样有多道加工工序且每道工序都可以在一个机器集子集中选择任意一台机器完成，我们需要将该任务插入先前的调度方案（即原先调度方案中各工序相对顺序不改变，仅将新任务的各工序插入合适的位置进行调整），并仍然希望最大完工时间可以尽可能小。

因此，需设计一个调度算法并提供两个接口（函数）：接口1能够读入一个FJS问题并返回一个调度方案；接口2能够读入一个新订单，将该订单插入原先调度方案，并返回新的调度方案。可以将这两个问题分别独立考虑，也可以在设计FJS算法时考虑调度方案的稳健性（即提高应对插单的能力，注意在设计第一个调度方案时我们不知道新订单的具体信息，但可以假设它和初始订单集有一定的相似性）。

**数据格式**

**输入数据通过以下形式给出（txt文件）：**



对于每组数据，第一行有三个整数，第一个表示总订单数n（包括插单订单），第二个表示机器数m，第三个表示插入订单数p。

从第二行开始每行对应一个订单，第一个整数为该订单的工序数，从第二个整数开始依次为每道工序，对于每道工序首先给出一个整数表示它可以选择的机器数m，之后紧跟m对数字，每对包含两个整数，第一个整数为机器编号，第二个整数为该工序在该机器上的加工时间。

以上图中第二行为例，该订单有**7**道工序，第一道工序可以选择的机器为**1**台：（机器**1**，加工时间**3**），第二道工序可以选择的机器为**2**台：（机器**3**，加工时间**1**）、（机器**4**，加工时间**3**），第三道工序可以选择的机器为**1**台：（机器**5**，加工时间**2**），第四道工序可以选择的机器为**1**台（机器**8**，加工时间**10**），第五道工序可以选择的机器为**2**台（机器**10**，加工时间**2**）、（机器**9**，加工时间**4**），第六道工序可以选择的机器为**2**台（机器**11**，加工时间**5**）、（机器**12**，加工时间**3**），第七道工序可以选择的机器为**1**台：（机器**13**，加工时间**1**）。

第2行～第n-p+1行为初始已知订单，最后p行表示后续插入的订单，虽然插入订单在输入数据中一起给出，但在设计初始方案时请当作插入订单未知。

**输出数据格式要求：**

输出分为两个txt文件，第一个文件为不包含最后插入订单的调度方案，包含n-p行，每行第一个整数表示该订单的工序数k（与输入保持一致），后面紧跟k对整数，每对整数分别表示当前工序选择的机器编号及开始时间。输出中所有订单的先后顺序与输入保持一致。

第二个文件为包含插入订单的调度方案，包含n行，前n-p行与文件1一致，最后p行表示插入订单，格式同上。

两个文件分别命名为result1.txt、result2.txt。

**数据集说明**

我们使用的数据集有如下性质：

1. 每个订单都有5～7道工序，第三道和第七道工序不是每个订单都有
2. 机器共15台，其中

机器1、2对应第一道工序

机器3、4对应第二道工序

机器5对应第三道工序

机器6、7、8对应第四道工序

机器9、10对应第五道工序

机器11、12对应第六道工序

机器13、14、15对应第七道工序

1. 假设耗时最多的订单总加工时间为T，则插入订单的插入时间服从[0,T]的均匀分布。若插入时间为t，则原先调度方案开始时间小于t的各工序已经处于完成或开工状态，只能在未开始的工序间插入新订单的各工序。