# B题：动车组检修

## 问题一

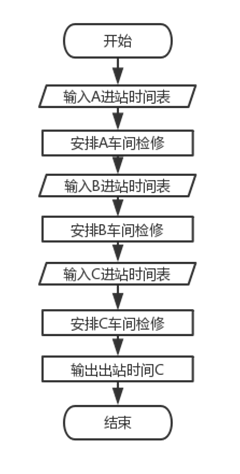
如图1所示，动车组的一次检修包括a，b，c三个工序。每个工序拥有的作业车间和需要花费的时间如表1所示，相同工序不同车间的耗费时间相同。动车组按a→b→c顺序进行检修，完成一个检修工序后驶入下一个有空闲位置的车间进行下一个检修工序，若下一个工序所有车间都处于占用状态，则动车组需要在上一个车间中等待。动车运用所某12小时内每十五分钟来1辆待检修的动车，按照目前的车间设置，维修完所有这些动车组总共需要多长时间？请给出安排检修的最佳方案。假设第一辆动车组抵达动车运用所时，所有检修车间都是空闲的，且车间之间的转换时间忽略不计。

表1 检修基本数据

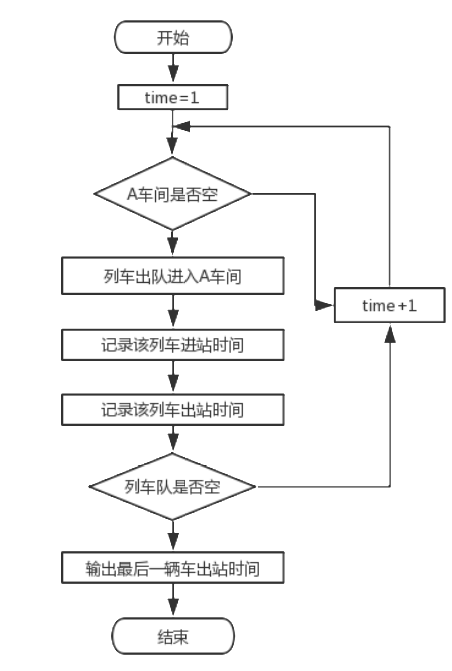
|  |  |  |  |
| --- | --- | --- | --- |
| 工序类别 | a | b | c |
| 车间数量（个） | 3 | 8 | 5 |
| 耗费时间（小时） | 1 | 2 | 1.5 |

## 问题解答

根据问题一，列车每隔15分钟发车一趟，12个小时总共发48趟。发车后，列车会寻找A车间的是否有空位，没有空位的时候就会继续等待，直到有空位的时候驶入车间。B、C车间也会如此。由于问题一的条件特殊，车次是连续进入。故问题可以转换成一个时序表，给定一个进站时间，然后安排排队，最后输出出站时间，该出站时间也为下一车间的进站时间，最后一辆车的出站时间就是所需要的总时间。整体流程图如下：



## 问题一A车间的程序流程图：



## 第一问代码：

1. #创建列车类
2. **class** CRH:
3. starttime =0
4. endtime=0
5. #创建A车间类
6. **class** A:
7. **def** \_\_init\_\_(self):    #定义车间的一些属性
8. self.name = ' '
9. self.state = 0
10. self.maintenancetime =60
11. #创建B车间类
12. **class** B:
13. **def** \_\_init\_\_(self):
14. self.name =' '
15. self.state = 0
16. self.maintenancetime =120
17. #创建C车间类
18. **class** C:
19. **def** \_\_init\_\_(self):
20. self.name = ' '
21. self.state = 0
22. self.maintenancetime =90
23. CRHqueue =[] #创建列车等待队列
24. time = 0    #初始化时间序列
25. Aworkshop = Bworkshoop = Cworkshoop=[] #初始化车间状态队列
27. **for** i **in** range(48):   #创建48个列车实例
28. locals()['CRH%d'%i] =CRH()
29. CRHqueue.append(locals()['CRH%d'%i])
31. CRHqueue.reverse()   #队列反置，使得第一辆车被队列弹出
32. **for** i **in** range(3):    #创建3个A车间实例
33. locals()['A%d'%i] = A()
34. locals()['A%d'%i].name = 'A%d'%i
35. **for** i **in** range(8):    #创建8个B车间实例
36. locals()['B%d'%i] = B()
37. locals()['B%d'%i].name = 'B%d'%i
38. **for** i **in** range(5):      #创建5个C车间实例
39. locals()['C%d'%i] = C()
40. locals()['C%d'%i].name = 'C%d'%i
42. CRHname = 0     #列车名字序号
43. **while** True:
44. **if**(len(Aworkshop))<3): #车间是否为空的判定
45. Aworkshop.append(CRHqueue.pop())
46. locals()['CRH%d'%CRHname].starttime+= time
47. locals()['CRH%d'%CRHname].endtime+=locals()['CRH%d'%CRHname].starttime+60
48. CRHname+=1
49. time+=1
50. index =[]
51. **for** i **in** range(len(Aworkshop)):  #检验A车间内列车出车间条件
52. **if**(Aworkshop[i].endtime == time):
53. index.append(Aworkshop[i])
54. **for** i **in** range(len(index)):
55. Aworkshop.remove(index[i])   #移除A车间维修好的列车
57. **if**(len(Aworkshop)==0 **and** len(CRHqueue)==0):  #A车间为空时结束循环
58. **break**
60. **print**('%d分钟后最后一辆列出开出A站'%CRH48.endtime)#打印最后一辆车出A车间时间

## 问题：

1. 只能完成第一问车子在A车间的进站和出站情况。
2. 利用队列的思想解决的一问A车间的问题，代码不能普适性解决问题。
3. 代码冗余不够精简，定义了很多没有必要的变量。
4. 其它问题还未想到解决方法，类似于甘特图是不是更佳的解决方法？