题目要求：<22-P1065-作业调度方案-题目要求.docx>

## 一、约束条件：

1. 每个工件的下一个工序必须在上一个工序之后
2. 同一台机器同一时刻只能加工一个工件
3. 按题目顺序安排下一个工件

## 二、分析

题目已经给出了安排好的工序，而每个工序需要在几号机上完成以及每个工序的时间也给了出来，我们要做的就是合理安排机器的工作，让总的加工时间最短。

按照题意的约定，最短方案有且只有一种，而且不必判断输入的合法性。

我们可以把机器想成若干个「时间线」，在这条时间线上去安排工作。

那么明显的，每个时间段对应的机器就只有俩状态：

1.我在干活

2.我闲着呢

而每一个工件也有自己的加工要求，对于每个工件的工序，总应该先完成小号工序再完成大号工序，也就是必须顺着编号来。

每台机器只能在某时刻进行一种工作，并且后面的安排不能把前面的安排改动掉。

模拟的思想便是从左到右无限扫描整个时间线，然后去尝试插空。

## 三、主要算法

1.按照安排顺序模拟

2.每个工件记录最后时间

考虑到n,m<20，无需考虑时间复杂度，只需细心即可。

由于条件复杂，需注意每个变量的含义

## 四、我的代码：

附上文件路径：<22-P1065-作业调度方案.cpp>

1. #include<stdio.h>
2. #include<stdlib.h>
3. #define N 10000
4. struct work{
5. int machine;
6. int time;
7. }work[25][25];
8. int main(void)
9. {
10. int m,n,res=0;
11. int now[25]={0},bf[25]={0},schedule[405],machine[25][N]={0};
12. *//now记录当前每个工件已完成的工序数，bf标记此次工序完成后的时间点*
13. *//schedule记录初始安排顺序，machine记录某机器已经被占据的时间段，值为1时表示占用*
14. scanf("%d%d",&m,&n);
15. for(int i=0;i<m\*n;i++) scanf("%d",&schedule[i]);
16. *//读入给定的工序安排顺序*
17. for(int i=0;i<n;i++){
18. for(int j=0;j<m;j++)
19. scanf("%d",&work[i][j].machine);}
20. *//读入n个工件的各个（共m个工序）工序所用机器号*
21. for(int i=0;i<n;i++){
22. for(int j=0;j<m;j++)
23. scanf("%d",&work[i][j].time);}
24. *//读入n个工件的各个（共m个工序）工序所用时间*
25. for (int i=0;i<m\*n;i++) {
26. int current=schedule[i];*//定位当前所安排的操作*
27. now[current-1]++;*//now记录每个工件所完成的工序数*
28. int mac=work[current-1][now[current-1]-1].machine,time=work[current-1][now[current-1]-1].time;
29. *//记录当前工件所需的时间和机器号*
30. *//mac指代当前工件的当前工序对应的机器号*
31. *//time指代当前工件的当前工序对应的时间*
32. *//    printf("%d:cur=%d,machine=%d,time=%d\a\n",i,current,mac,time);*
33. *//打印当前工件占用的机器与占用时长*
34. int s=0;
35. for (int j=bf[current-1]+1;;j++) {
36. *//变量j从上次工序完成后的时间后开始*
37. *//j指示当前的时间游标，s指示当前的连续空当的时间单位*
38. if (machine[mac][j]==0)s++;
39. else s=0;*//如果空当不满足条件则赋空当时间为0，跳过下列代码段将游标右移*
40. if(s==time)
41. {*//如果空当时间可以满足当前工件的需求则进入*
42. for(int k=j-time+1;k<=j;k++)machine[mac][k]=1;
43. *//将已经用来处理工序的时间段标记为1，该机器的该段时间线已被占据*
44. *//    printf("%d:%d~%d已被占据\n",mac,j-time+1,j);*
45. *//打印已被占据的时间线*
46. if (j>res) res=j;*//如果先前总共所需时间要小于处理当前工件后所用总时间，则将时间更新*
47. bf[current-1]=j;*//记录此次进行完工序后的时间线标记*
48. break;
49. }
50. }
51. *// system("pause");*
52. *//观察当前运行的数据结果*
53. }
54. printf("%d\a\n",res);
55. *//    system("pause");*
56. }

以下是另一版本代码：

1. #include<cstdio>
2. #include<iostream>
3. using namespace std;
4. int i,j,k,m,n,ans;
5. int a[405]; *//安排顺序*
6. int mord[21][21],t[21][21];
7. int cnt[21],last\_[21]; *//last\_ 每个工件最晚时间*
8. bool rec[21][8001]; *//每台机器的每个时间是否被占用*
9. int main()
10. {
11. scanf("%d%d",&m,&n);
12. for (i=1; i<=m\*n; i++)
13. scanf("%d",&a[i]);
14. for (i=1; i<=n; i++)
15. for (j=1; j<=m; j++)
16. scanf("%d",&mord[i][j]);
17. for (i=1; i<=n; i++)
18. for (j=1; j<=m; j++)
19. scanf("%d",&t[i][j]);
20. for (i=1; i<=m\*n; i++) {
21. cnt[a[i]]++;
22. int tmp1=a[i],tmp2=cnt[tmp1]; *//tmp1:工件 tmp2:工序*
23. int tmpm=mord[tmp1][tmp2]; *//机器号 //t[tmp1][tmp2]*
24. for (j=last\_[tmp1];;j++) {
25. bool flag = 1;
26. for (k=j+1; k<=j+t[tmp1][tmp2]; k++)
27. if (rec[tmpm][k]) {
28. flag = 0;
29. break;
30. }
31. if (flag) {
32. for (k=j+1; k<=j+t[tmp1][tmp2]; k++)
33. rec[tmpm][k] = 1; *//扔入空档*
34. last\_[tmp1] = max(last\_[tmp1],j+t[tmp1][tmp2]);
35. ans = max(ans,last\_[tmp1]); *//将新覆盖的结束点更新*
36. *//printf("%d : %d %d\n",i,j+1,j+t[tmp1][tmp2]);*
37. break;
38. }
39. }
40. }
41. printf("%d",ans);
42. return 0;
43. }

🌓