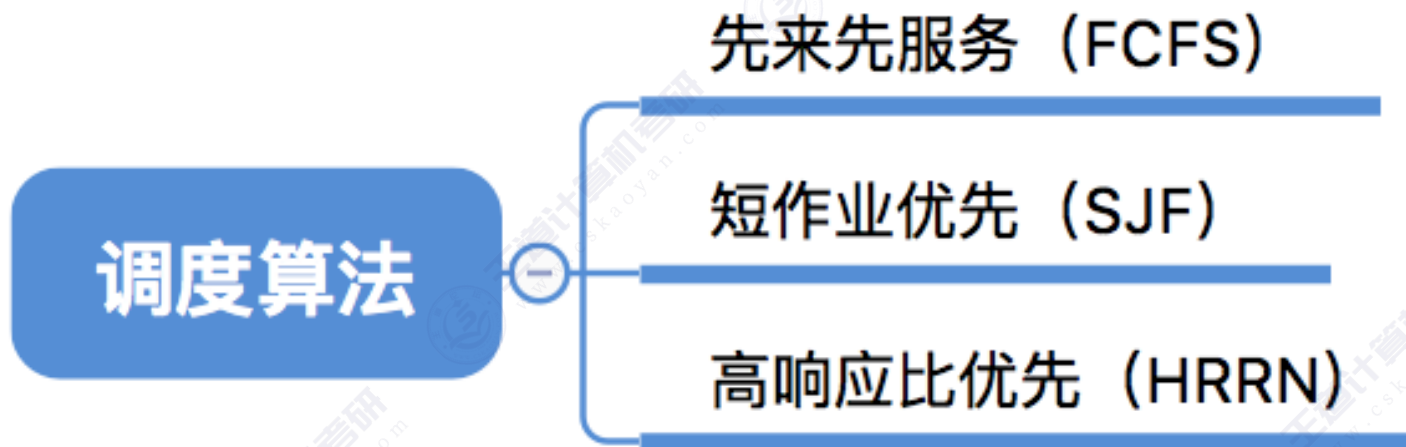


本节内容

# 调度算法

先来先服务  
最短作业优先  
最高响应比优先

# 知识总览

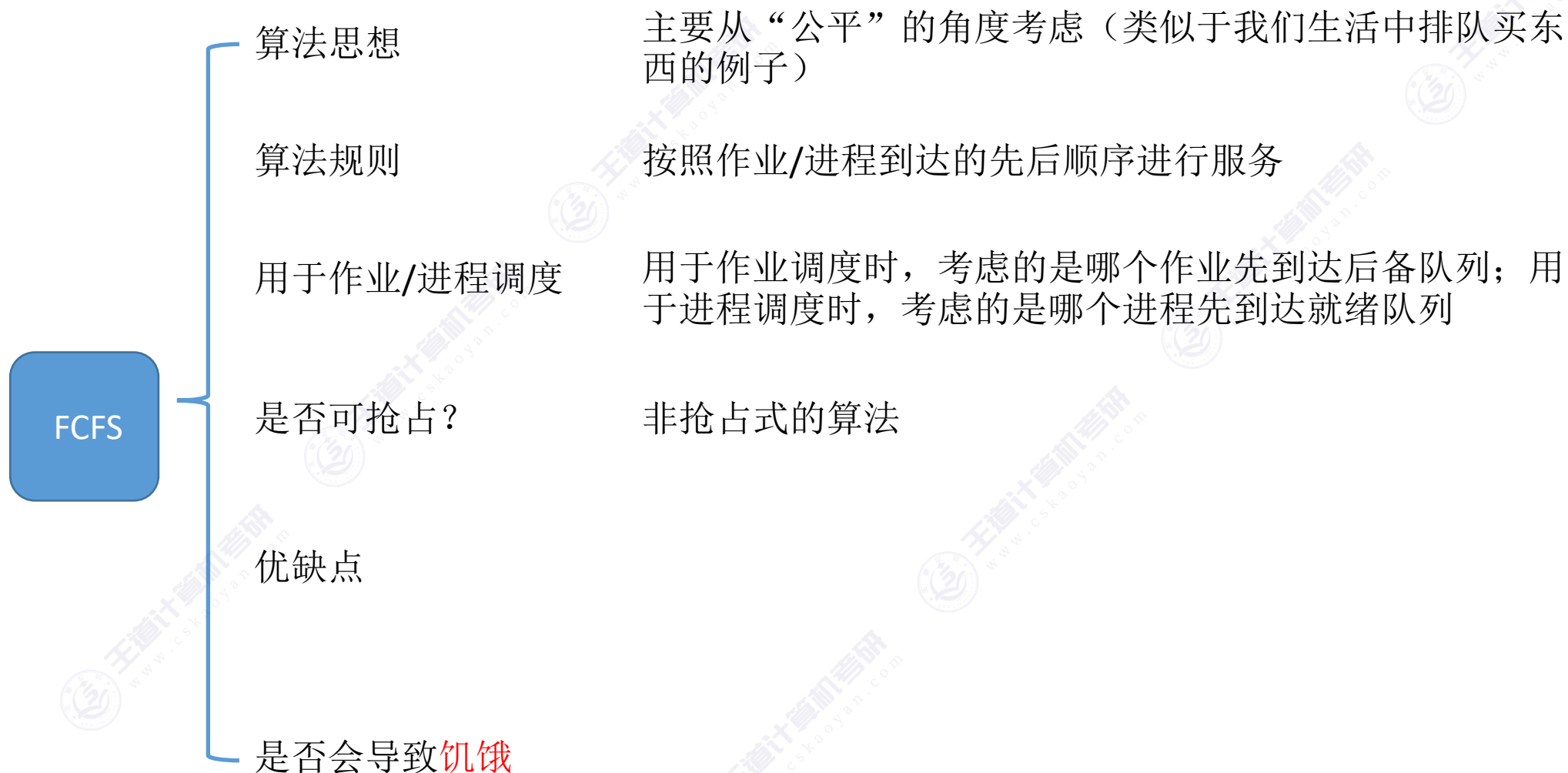


Tips: 各种调度算法的学习思路

1. 算法思想
2. 算法规则
3. 这种调度算法是用于 作业调度 还是 进程调度?
4. 抢占式? 非抢占式?
5. 优点和缺点
6. 是否会导致饥饿

某进程/作业长期  
得不到服务

# 先来先服务 (FCFS, First Come First Serve)



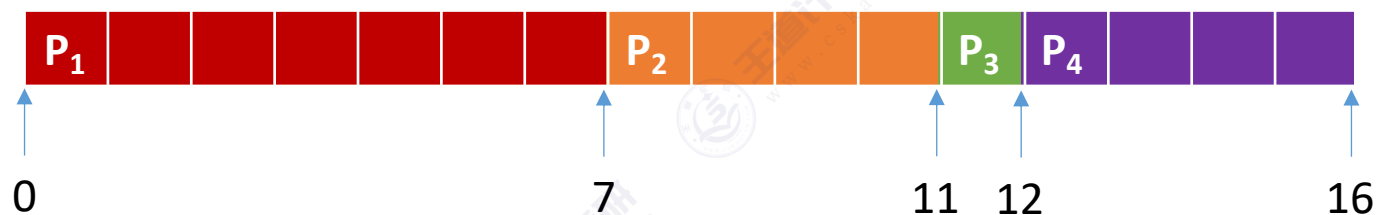
# 先来先服务 (FCFS, First Come First Serve)

例题：各进程到达就绪队列的时间、需要的运行时间如下表所示。使用**先来先服务**调度算法，计算各进程的等待时间、平均等待时间、周转时间、平均周转时间、带权周转时间、平均带权周转时间。

进程	到达时间	运行时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4

先来先服务调度算法：按照到达的先后顺序调度，事实上就是等待时间越久的越优先得到服务。

因此，**调度顺序**为：P1 → P2 → P3 → P4



周转时间 = 完成时间 - 到达时间

$$P1=7-0=7; P2=11-2=9; P3=12-4=8; P4=16-5=11$$

带权周转时间 = 周转时间/运行时间

$$P1=7/7=1; P2=9/4=2.25; P3=8/1=8; P4=11/4=2.75$$

等待时间 = 周转时间 - 运行时间

$$P1=7-7=0; P2=9-4=5; P3=8-1=7; P4=11-4=7$$

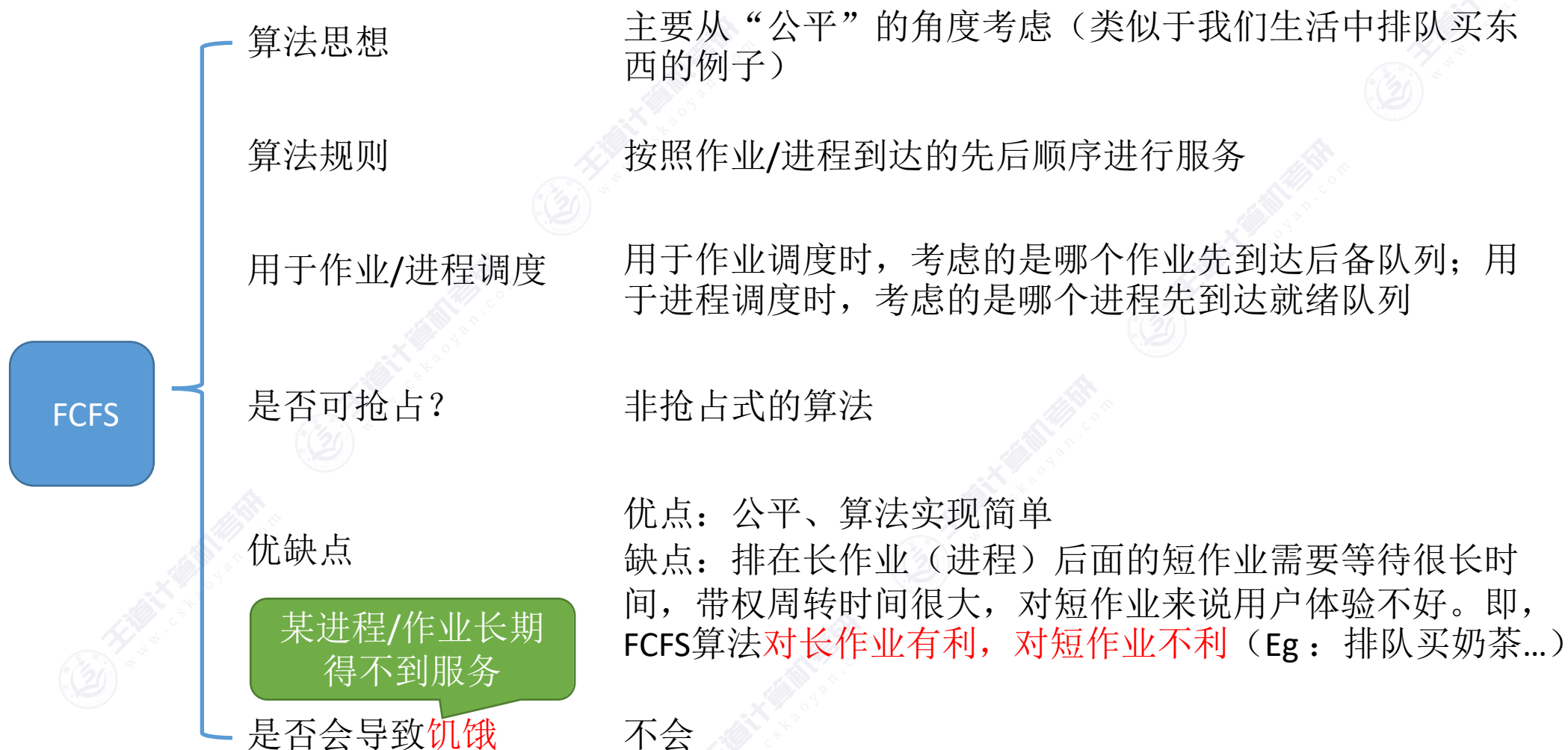
平均周转时间 =  $(7+9+8+11)/4 = 8.75$

平均带权周转时间 =  $(1+2.25+8+2.75)/4 = 3.5$

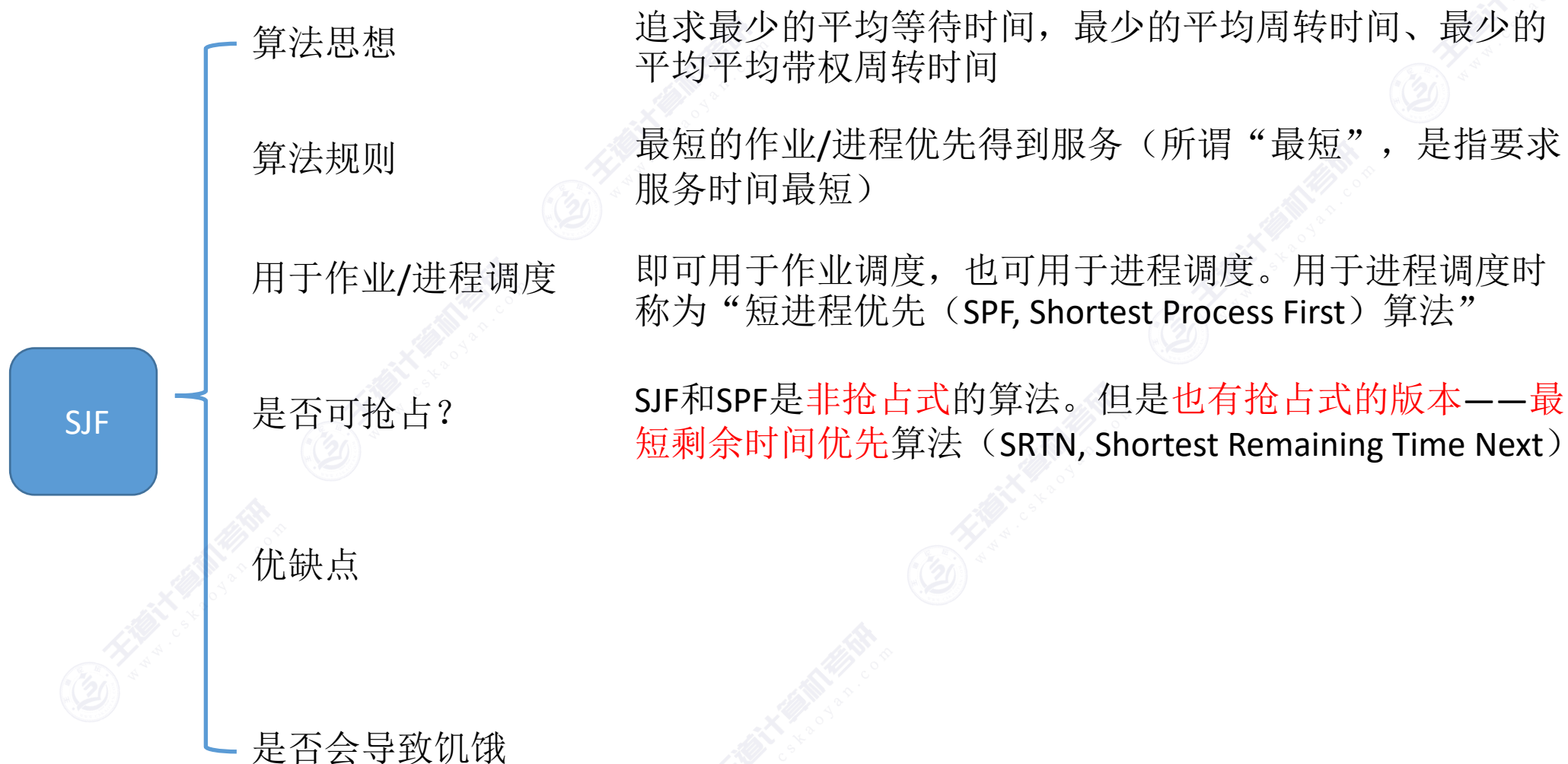
平均等待时间 =  $(0+5+7+7)/4 = 4.75$

注意：本例中的进程都是纯计算型的进程，一个进程到达后要么在等待，要么在运行。如果是又有计算、又有I/O操作的进程，其等待时间就是周转时间 - 运行时间 - I/O操作的时间

# 先来先服务 (FCFS, First Come First Serve)



# 短作业优先 (SJF, Shortest Job First)



# 短作业优先 (SJF, Shortest Job First)

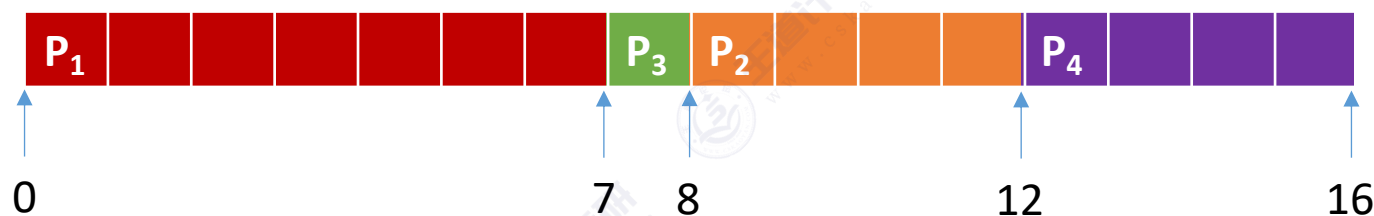
严格来说，用于进程调度应该称为短进程优先调度算法 (SPF)

例题：各进程到达就绪队列的时间、需要的运行时间如下表所示。使用**非抢占式**的**短作业优先**调度算法，计算各进程的等待时间、平均等待时间、周转时间、平均周转时间、带权周转时间、平均带权周转时间。

进程	到达时间	运行时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4

短作业/进程优先调度算法：每次调度时选择**当前已到达**且**运行时间最短**的作业/进程。

因此，**调度顺序**为：P1 → P3 → P2 → P4



周转时间 = 完成时间 - 到达时间

$$P1=7-0=7; \quad P3=8-4=4; \quad P2=12-2=10; \quad P4=16-5=11$$

带权周转时间 = 周转时间/运行时间

$$P1=7/7=1; \quad P3=4/1=4; \quad P2=10/4=2.5; \quad P4=11/4=2.75$$

等待时间 = 周转时间 - 运行时间

$$P1=7-7=0; \quad P3=4-1=3; \quad P2=10-4=6; \quad P4=11-4=7$$

平均周转时间 =  $(7+4+10+11)/4 = 8$

平均带权周转时间 =  $(1+4+2.5+2.75)/4 = 2.56$

平均等待时间 =  $(0+3+6+7)/4 = 4$

8.75  
3.5  
4.75

对比FCFS算法的结果，显然SPF算法的平均等待/周转/带权周转时间都要更低



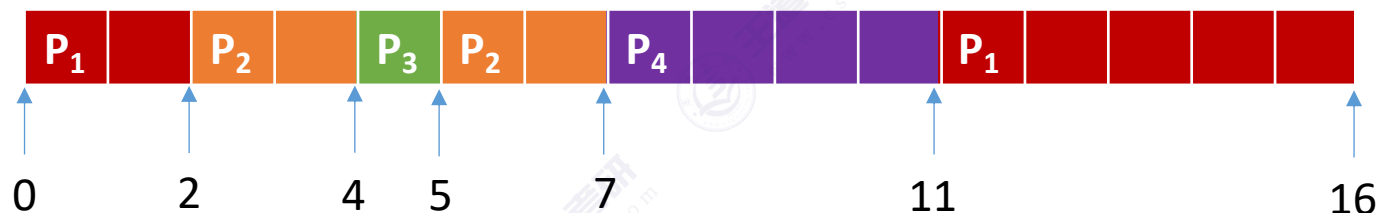
# 短作业优先 (SJF, Shortest Job First)

抢占式的短作业优先算法  
又称“最短剩余时间优先  
算法 (SRTN)”

例题：各进程到达就绪队列的时间、需要的运行时间如下表所示。使用**抢占式**的**短作业优先**调度算法，计算各进程的等待时间、平均等待时间、周转时间、平均周转时间、带权周转时间、平均带权周转时间。

进程	到达时间	运行时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4

**最短剩余时间优先算法**：每当有进程加入**就绪队列**改变时就需要调度，如果新到达的进程**剩余时间**比当前运行的进程剩余时间**更短**，则由新进程**抢占**处理机，当前运行进程重新回到就绪队列。另外，当一个**进程完成时**也需要调度



需要注意的是，当有新进程到达时就绪队列就会改变，就要按照上述规则进行检查。以下  $P_n(m)$  表示当前  $P_n$  进程剩余时间为  $m$ 。各个时刻的情况如下：

0时刻 (P1到达) :  $P_1(7)$

2时刻 (P2到达) :  $P_1(5)$ 、 $P_2(4)$

4时刻 (P3到达) :  $P_1(5)$ 、 $P_2(2)$ 、 $P_3(1)$

5时刻 (P3完成且P4刚好到达) :  $P_1(5)$ 、 $P_2(2)$ 、 $P_4(4)$

7时刻 (P2完成) :  $P_1(5)$ 、 $P_4(4)$

11时刻 (P4完成) :  $P_1(5)$



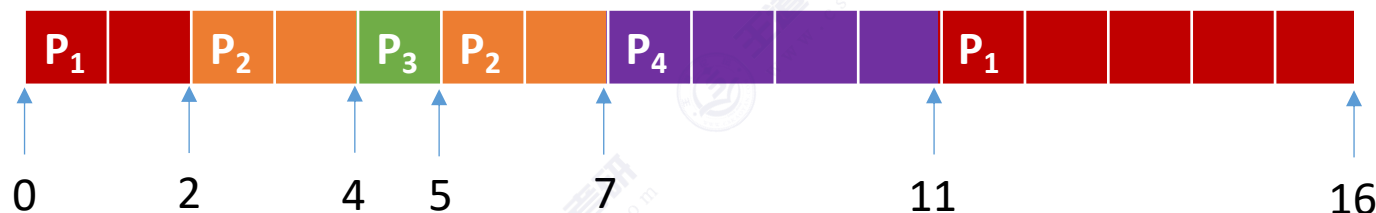
# 短作业优先 (SJF, Shortest Job First)

抢占式的短作业优先算法  
又称“最短剩余时间优先  
算法 (SRTN)”

例题：各进程到达就绪队列的时间、需要的运行时间如下表所示。使用**抢占式**的**短作业优先**调度算法，计算各进程的等待时间、平均等待时间、周转时间、平均周转时间、带权周转时间、平均带权周转时间。

进程	到达时间	运行时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4

**最短剩余时间优先**算法：每当有进程加入**就绪队列**改变时就需要**调度**，如果新到达的进程**剩余时间**比当前运行的进程**剩余时间****更短**，则由新进程**抢占**处理机，当前运行进程重新回到就绪队列。另外，当一个**进程完成时**也需要**调度**



周转时间 = 完成时间 - 到达时间

$$P1=16-0=16; P2=7-2=5; P3=5-4=1; P4=11-5=6$$

带权周转时间 = 周转时间/运行时间

$$P1=16/7=2.28; P2=5/4=1.25; P3=1/1=1; P4=6/4=1.5$$

等待时间 = 周转时间 - 运行时间

$$P1=16-7=9; P2=5-4=1; P3=1-1=0; P4=6-4=2$$

平均周转时间 =  $(16+5+1+6)/4 = 7$

平均带权周转时间 =  $(2.28+1.25+1+1.5)/4 = 1.50$

平均等待时间 =  $(9+1+0+2)/4 = 3$

8  
2.56  
4

对比非抢占式的短作业优先算法，显然抢占式的这几个指标又要更低

# 短作业优先 (SJF, Shortest Job First)

注意几个小细节:

1. 如果题目中未特别说明, 所提到的“短作业/进程优先算法”默认是非抢占式的

2. 很多书上都会说“SJF 调度算法的平均等待时间、平均周转时间最少”

严格来说, 这个表述是错误的, 不严谨的。之前的例子表明, 最短剩余时间优先算法得到的平均等待时间、平均周转时间还要更少

应该加上一个条件“在所有进程同时可运行时, 采用SJF调度算法的平均等待时间、平均周转时间最少”;

或者说“在所有进程都几乎同时到达时, 采用SJF调度算法的平均等待时间、平均周转时间最少”;

如果不加上上述前提条件, 则应该说“抢占式的短作业/进程优先调度算法 (最短剩余时间优先, SRNT算法) 的平均等待时间、平均周转时间最少”

3. 虽然严格来说, SJF的平均等待时间、平均周转时间并不一定最少, 但相比于其他算法 (如 FCFS), SJF依然可以获得较少的平均等待时间、平均周转时间

4. 如果选择题中遇到“SJF 算法的平均等待时间、平均周转时间最少”的选项, 那最好判断其他选项是不是有很明显的错误, 如果没有更合适的选项, 那也应该选择该选项

# 短作业优先 (SJF, Shortest Job First)

SJF

算法思想

追求最少的平均等待时间，最少的平均周转时间、最少的平均平均带权周转时间

算法规则

最短的作业/进程优先得到服务（所谓“最短”，是指要求服务时间最短）

用于作业/进程调度

即可用于作业调度，也可用于进程调度。用于进程调度时称为“短进程优先（SPF, Shortest Process First）算法”

是否可抢占？

SJF和SPF是**非抢占式**的算法。但是**也有抢占式的版本——最短剩余时间优先算法（SRTN, Shortest Remaining Time Next）**

优缺点

优点：“最短的”平均等待时间、平均周转时间  
缺点：不公平。**对短作业有利，对长作业不利**。可能产生**饥饿现象**。另外，作业/进程的运行时间是由用户提供的，并不一定真实，不一定能做到真正的短作业优先

是否会导致饥饿

会。如果源源不断地有短作业/进程到来，可能使长作业/进程长时间得不到服务，产生“**饥饿**”现象。如果一直得不到服务，则称为“**饿死**”

## 对FCFS和SJF两种算法的思考...



FCFS 算法是在每次调度的时候选择一个等待时间最长的作业（进程）为其服务。但是没有考虑到作业的运行时间，因此导致了对短作业不友好的问题

SJF 算法是选择一个执行时间最短的作业为其服务。但是又完全不考虑各个作业的等待时间，因此导致了对长作业不友好的问题，甚至还会造成饥饿问题

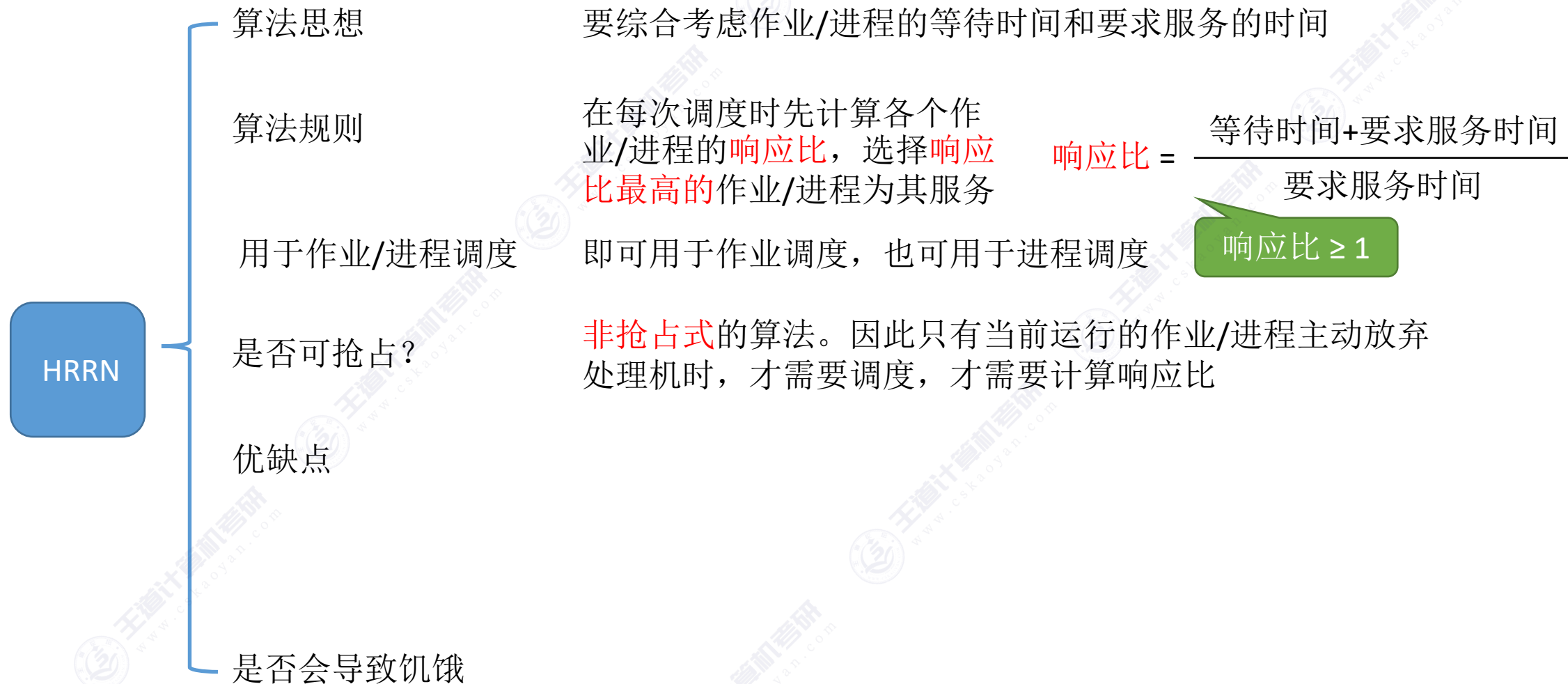
能不能设计一个算法，即考虑到各个作业的等待时间，也能兼顾运行时间呢？



厉害了，我的哥

高响应比优先算法

# 高响应比优先 (HRRN, Highest Response Ratio Next)

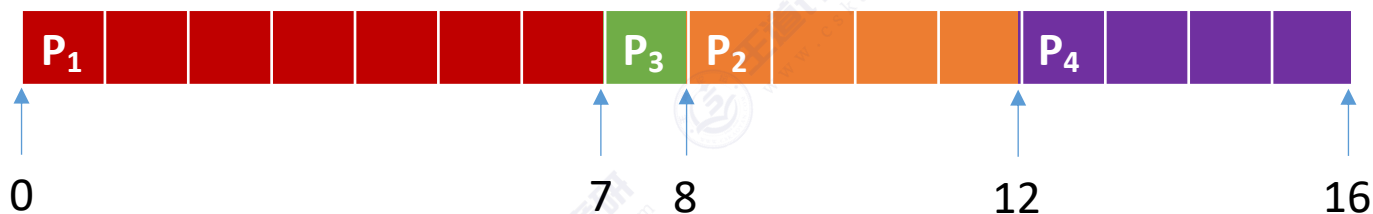


## 高响应比优先 (HRRN)

例题：各进程到达就绪队列的时间、需要的运行时间如下表所示。使用高响应比优先调度算法，计算各进程的等待时间、平均等待时间、周转时间、平均周转时间、带权周转时间、平均带权周转时间。

进程	到达时间	运行时间
P1	0	7
P2	2	4
P3	4	1
P4	5	4

高响应比优先算法：非抢占式的调度算法，只有当前运行的进程主动放弃CPU时（正常/异常完成，或主动阻塞），才需要进行调度，调度时计算所有就绪进程的响应比，选响应比最高的进程上处理机。



$$\text{响应比} = \frac{\text{等待时间} + \text{要求服务时间}}{\text{要求服务时间}}$$

P2和P4要求服务时间一样，  
但P2等待时间长，所以必然是P2响应比更大

0时刻：只有 P<sub>1</sub> 到达就绪队列，P<sub>1</sub> 上处理机

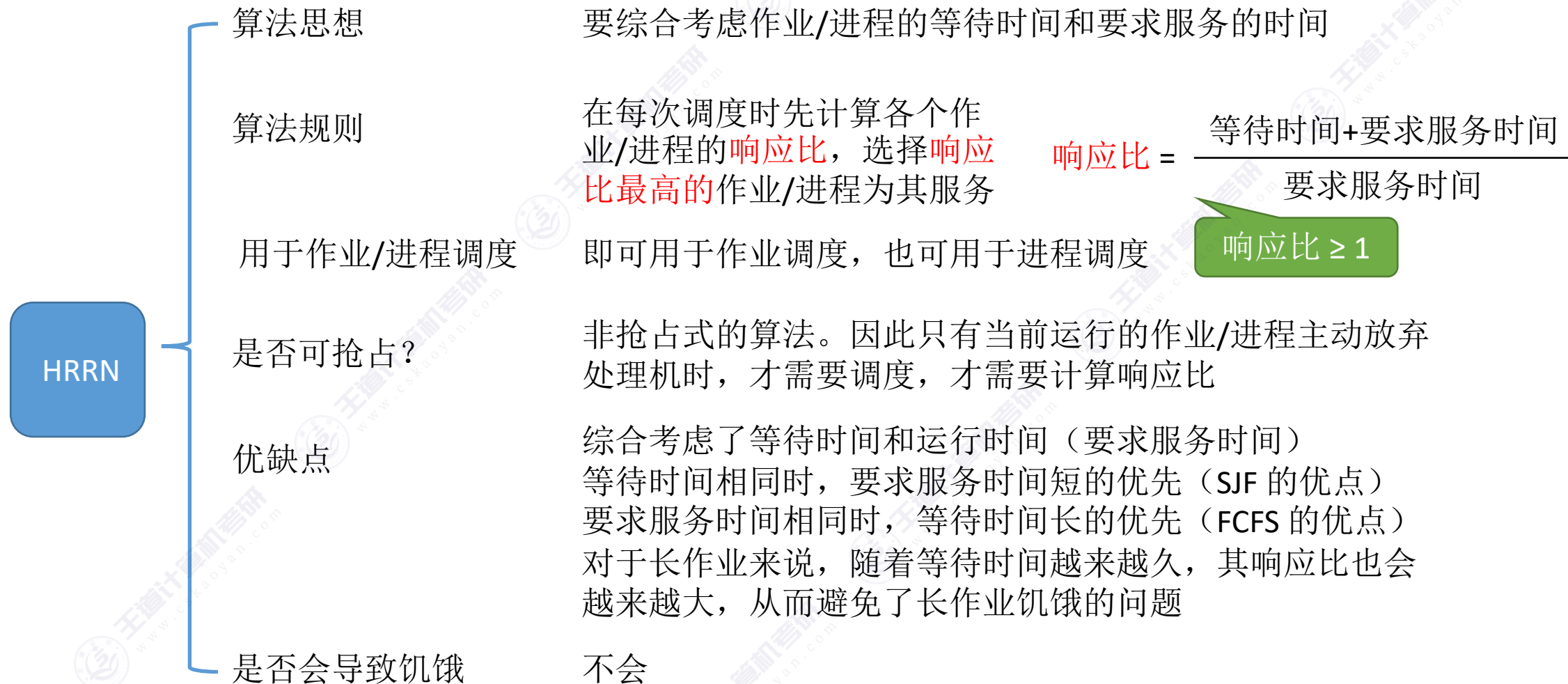
7时刻（P<sub>1</sub>主动放弃CPU）：就绪队列中有 P<sub>2</sub> (响应比=(5+4)/4=2.25)、P<sub>3</sub> ((3+1)/1=4)、P<sub>4</sub> ((2+4)/4=1.5)，

8时刻（P<sub>3</sub>完成）：P<sub>2</sub>(2.5)、P<sub>4</sub>(1.75)

12时刻（P<sub>2</sub>完成）：就绪队列中只剩下 P<sub>4</sub>



# 高响应比优先 (HRRN, Highest Response Ratio Next)





## 知识回顾与重要考点

算法	思想&规则	可抢占?	优点	缺点	考虑到等待时间&运行时间?	会导致饥饿?
FCFS	自己回忆	非抢占式	公平; 实现简单	对短作业不利	等待时间 $\sqrt{\quad}$ 运行时间 $\times$	不会
SJF/S PF	自己回忆	默认为非抢占式, 也有SJF的抢占式 版本最短剩余时间 优先算法 (SRTN)	“最短的” 平均等待 /周转时间;	对长作业不利, 可 能导致饥饿; 难以 做到真正的短作业 优先	等待时间 $\times$ 运行时间 $\sqrt{\quad}$	会
HRRN	自己回忆	非抢占式	上述两种算法的权衡 折中, 综合考虑的等 待时间和运行时间		等待时间 $\sqrt{\quad}$ 运行时间 $\sqrt{\quad}$	不会

注: 这几种算法主要关心对用户的公平性、平均周转时间、平均等待时间等评价系统整体性能的指标, 但是不关心“响应时间”, 也并不区分任务的紧急程度, 因此对于用户来说, 交互性很糟糕。因此这三种算法一般适合用于**早期的批处理系统**, 当然, FCFS算法也常结合其他的算法使用, 在现在也扮演着很重要的角色。而适合用于**交互式系统**的调度算法将在下一个小节介绍...

**提示: 一定要动手做课后习题!** 这些算法特性容易考小题, 算法的使用常结合调度算法的评价指标在大题中考察。



公众号：王道在线



b站：王道计算机教育



抖音：王道计算机考研