第三十七章-磁盘驱动器

预备知识

该作业使用 disk.py 使您熟悉现代磁盘驱动器的工作方式。
它具有许多不同的选项，并且与大多数其他模拟器不同，
它具有图形动画，以向您确切显示磁盘运行时会发生什么。

首先让我们看一个简单的例子。要运行模拟器并计算一些基本的寻道，旋转和传输时间，您首先必须提供对模拟器的请求列表。
可以通过指定确切的请求或通过模拟器随机生成一些请求来完成。

我们首先指定一些请求列表。 让我们先开始一个单一请求：

```
prompt> python2 disk.py -a 10
```

你会看到下面的输出:

```
REQUESTS ['10']
For the requests above, compute the seek, rotate, and transfer times.
Use -c or the graphical mode (-G) to see the answers.
```

为了能够计算该请求的查找，旋转和传输时间，您必须了解扇区布局，磁盘头的起始位置等的更多信息。
要查看许多此类信息，请以图形方式（-G）运行模拟器：

```
prompt> python2 disk.py -a 10 -G
```

此时将出现一个窗口，上面有简单磁盘。 磁盘头位于外部磁道的扇区 6 上。
如您所见，扇区 10（我们的示例扇区）位于同一磁道上，距离大约是磁道的三分之一。
旋转方向为逆时针方向。在键盘上按 s 键运行模拟器

模拟器运行完成后，您应该能够看到，为了访问扇区 10，磁盘花费了 105 个单位时间用于旋转，
30 个单位时间用于传输，寻道时间为 0。 按 “q“ 关闭模拟器窗口。

要计算这些值（而不是仅运行模拟），您需要了解有关磁盘的一些详细信息。
首先，转速默认设置为每单位时间旋转 1 度。 因此，完整旋转一圈需要 360 个单位时间。
其次，传输在扇区之间的中点开始和结束。 因此，要读取扇区 10，从 9 到 10 之间中点开始读取，
到 10 与 11 之间中点结束。最后，在默认磁盘中，每个磁道有 12 个扇区，这意味着每个扇区占用 30 度的旋转空间 。
因此，要读取一个扇区，它需要 30 个时间单位（根据我们默认的旋转速度）。

有了这些信息，您现在应该能够计算出访问扇区 10 的寻道时间，旋转和传输时间。
因为磁头从与 10 相同的磁道开始，所以没有寻道时间。 由于磁盘以 1 度/时间单位旋转，
因此需要 105 个时间单位才能到达扇区 10 的开始，介于 9 与 10 之间（请注意，它与第 9 区的中间正好 90 度，与中点又有 15 度）。
最后得出，读取一个扇区的数据需要 30 个时间单位。

译者注：读取数据的时间是包括在目标扇区间旋转的时间的

现在让我们看一个稍微复杂的例子：

```
prompt> python2 disk.py -a 10,11 -G
```

在这种情况下，我们要读取两个扇区数据，即 10 和 11。需要多长时间？在运行模拟之前尝试猜测！

您可能猜到了，此模拟仅多花费 30 个单位时间来传输下一个扇区 11。
因此，搜索和旋转时间保持不变，但是请求的传输时间却增加了一倍。
实际上，您可以在模拟器窗口的顶部看到这些总和。 它们还可以打印到控制台，如下所示：

```
...
Sector:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Sector:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
TOTALS      Seek:  0  Rotate:105  Transfer: 60  Total: 165
```

现在，让我们用一个寻道示例。 尝试以下请求：

```
prompt> python2 disk.py -a 10,18 -G
```

要计算这将花费多长时间，您需要知道寻道将花费多长时间。
默认情况下，每条轨道之间的距离为 40 个距离单位，默认搜索速度为每单位时间移动 1 个距离单位。
因此，从外磁道到中间磁道的搜寻需要 40 个时间单位。

您还必须了解调度策略。默认调度策略为 FIFO.
因此，现在您可以假设请求处理顺序可以通过 "-a" 标志指定，以此来计算请求时间。

要计算磁盘处理这些请求所需的时间，我们首先计算访问扇区 10 所需的时间，
我们可以知道这是 135 个时间单位（105 个旋转单位，30 个传输单位）。
完成此请求后，磁盘开始寻找扇区 18 所在的中间磁道，耗时 40 个 时间单位。
然后，磁盘旋转到扇区 18，并以 30 个时间单位传输它，从而完成了模拟。 但是最后一次旋转需要多长时间？

要计算 18 的旋转延迟，首先要计算出磁盘从结束访问第 10 扇区到开始访问第 18 扇区所需要的时间，
假设进行零成本寻道。从模拟器上可以看到，外侧轨道上的第 10 扇区与中间轨道上的第 22 扇区对齐，
有 7 个扇区将 22 和 18 分开(23、12、13、14、15、16 和 17，当圆盘逆时针旋转时)。
旋转 7 个扇区需要 210 个时间单位(每个扇区 30 个时间单位)。
然而，这个旋转的第一部分实际上是花了 40 个时间单位去寻找中间的轨道。
因此，进入第 18 扇区的实际旋转延迟是 210 减 40，即 170 个时间单位。
(磁盘头寻道和磁盘旋转是同时进行的，也就是说当寻道完成时，磁盘已经转过了一定角度，需要在旋转时间中减去这部分开销)

注意，您可以在没有图形的情况下运行，并使用“-c”标志来只查看结果而不查看图形。

```
prompt> ./disk.py -a 10,18 -c
...
Sector:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Sector:  18  Seek: 40  Rotate:170  Transfer: 30  Total: 240
TOTALS      Seek: 40  Rotate:275  Transfer: 60  Total: 375
```

您现在应该对模拟器的工作原理有一个基本的了解。 课后作业的问题将探讨一些不同的选项，以更好地帮助您建立磁盘实际工作方式的模型。

Problem1

问题描述

计算以下几组请求的寻道、旋转和传输时间: `-a 0, -a 6, -a 30, -a 7, 30, 8`,
最后计算 `-a 10,11,12,13`

问题分析

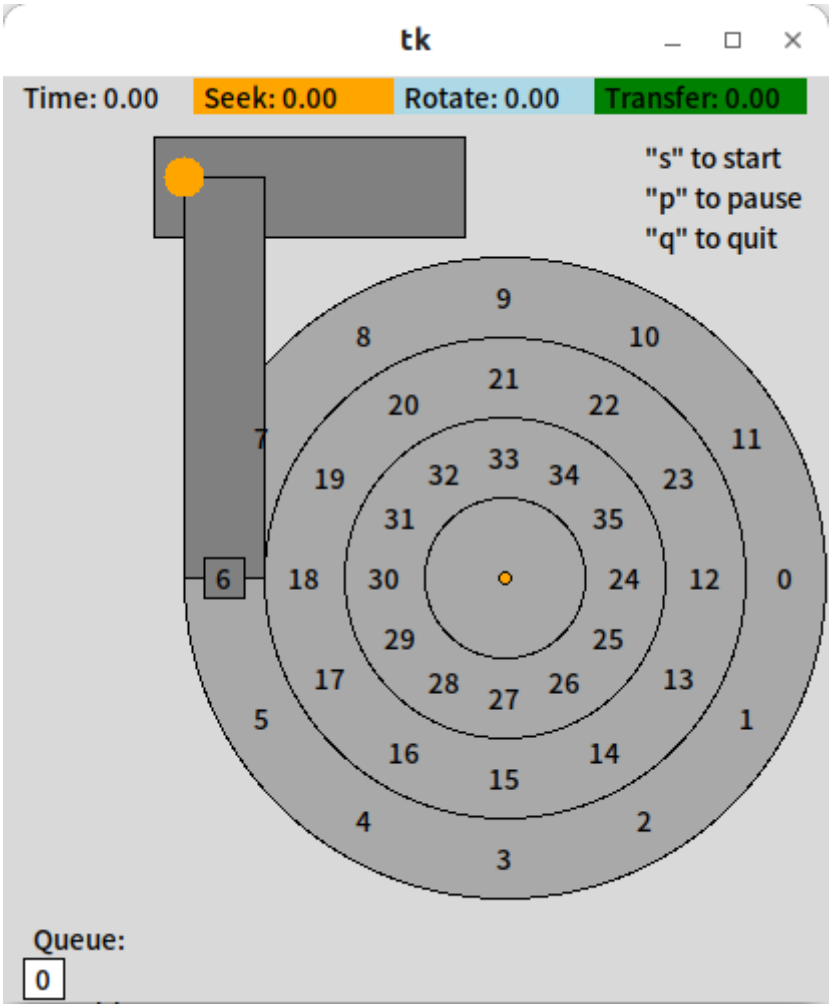
每条轨道之间的距离为 40 个距离单位，默认搜索速度为每单位时间移动 1 个距离单位。因此，从外磁道到中间磁道的搜寻需要 40 个时间单位。

每个磁道有 12 个扇区，默认转速为每单位时间旋转 1 度。因此，转过一个扇区需要 30 个时间单位 。

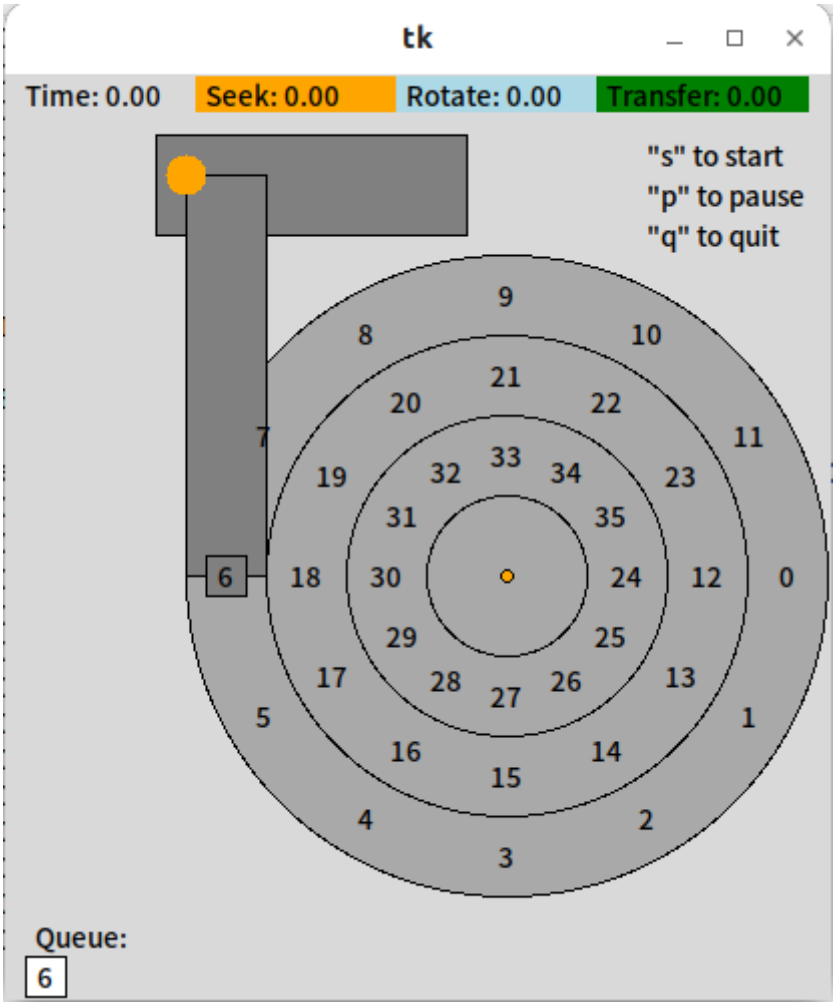
磁盘头寻道和磁盘旋转是同时进行的，也就是说当寻道完成时，磁盘已经转过了一定角度，需要在旋转时间中减去这部分角度造成的开销。

读取一个扇区的数据需要 30 个时间单位。

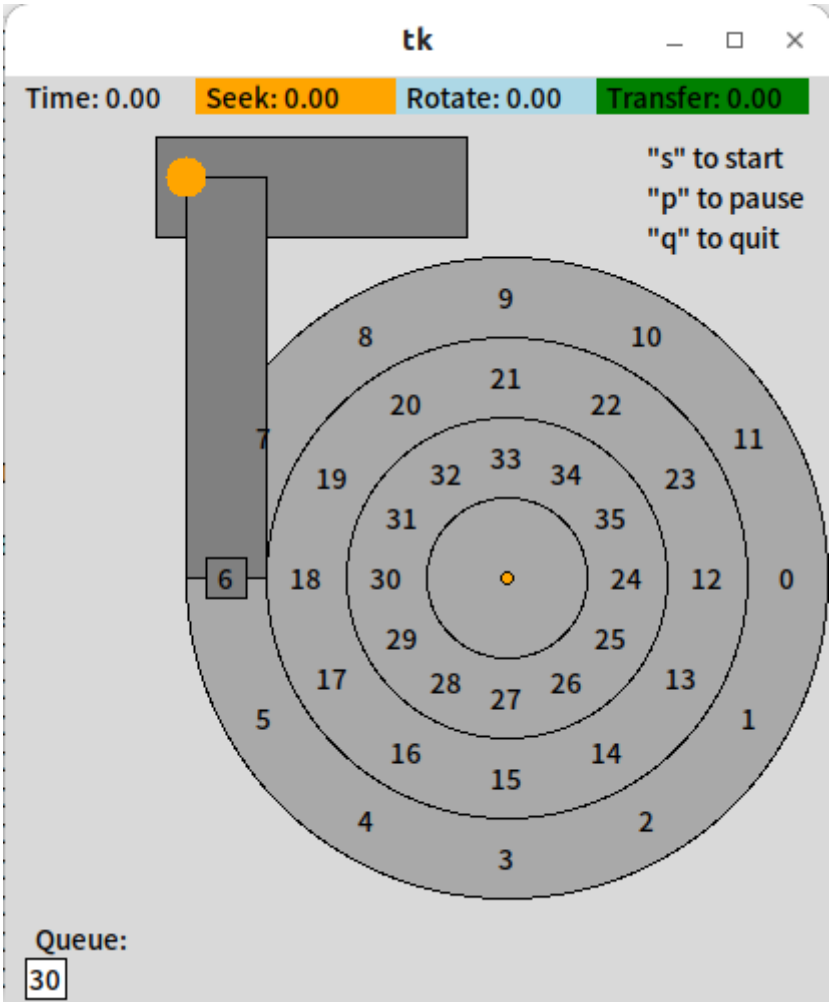
问题解答

1. 

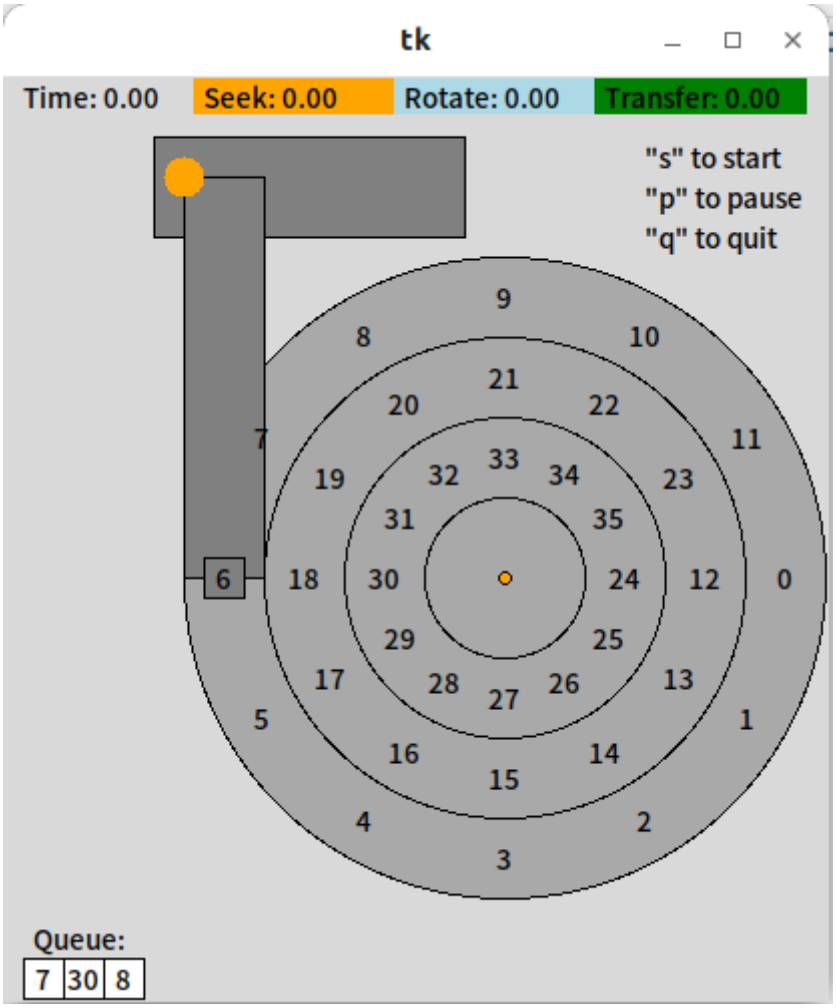
   不需要寻道，旋转5.5个扇区，5.5 * 30 + 30 = 195.0

2. 

不需要寻道，旋转11.5个扇区，11.5 * 30 + 30 = 375.0

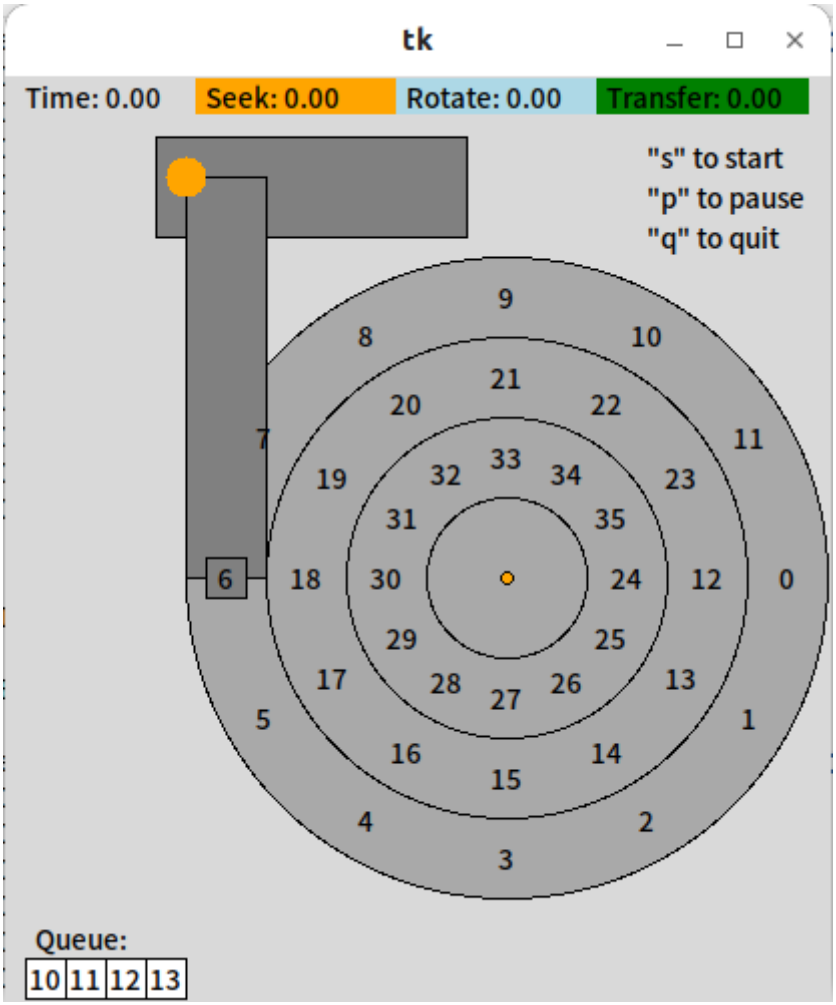
3. 

先寻道，跨越两个磁道，无寻道成本时旋转345度，减去寻道时转过的角度80，2*40+345-80+30=375

4. 

先旋转15度，然后传输7；再寻道到内圈磁道，无成本寻道旋转300度，减去寻道时转过的角度80，传输30；寻道到外圈磁道，当寻道完成时已经转过了扇区8的起始位置，还需要旋转310度，传输8

0.5 * 30 + 30 + 40 * 2 + 300 - 80 + 30 + 40 * 2 + 310 + 30 = 795.0

5. 

   先旋转105度，然后传输10；接着直接传输11；再寻道到中间磁道，当寻道完成时已经转过了扇区12的起始位置，还需要旋转320度，传输12；接着直接传输13

   3.5 * 30 + 30 + 30 + 40 + 320 + 30 + 30 = 585.0

答案验证

```
python2 disk.py  -a 0 -G

REQUESTS ['0']

Block:   0  Seek:  0  Rotate:165  Transfer: 30  Total: 195

TOTALS      Seek:  0  Rotate:165  Transfer: 30  Total: 195
```

```
python2 disk.py  -a 6 -G

REQUESTS ['6']

Block:   6  Seek:  0  Rotate:345  Transfer: 30  Total: 375

TOTALS      Seek:  0  Rotate:345  Transfer: 30  Total: 375
```

```
python2 disk.py  -a 30 -G

REQUESTS ['30']

Block:  30  Seek: 80  Rotate:265  Transfer: 30  Total: 375

TOTALS      Seek: 80  Rotate:265  Transfer: 30  Total: 375
```

```
python2 disk.py  -a 7,30,8 -G

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795
```

```
python2 disk.py  -a 10,11,12,13 -G

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate:320  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:425  Transfer:120  Total: 585
```

经验证，答案正确。

Problem2

问题描述

执行上述相同请求,但将寻道速率更改为不同值：-S 2,-S 4,-S 8, -S 10,-S 40, -S 0.1。时间如何变化?

问题分析

每条轨道之间的距离为 40 个距离单位，搜索速度可变为每单位时间移动 n 个距离单位。

每个磁道有 12 个扇区，默认转速为每单位时间旋转 1 度。因此，转过一个扇区需要 30 个时间单位 。

磁盘头寻道和磁盘旋转是同时进行的，也就是说当寻道完成时，磁盘已经转过了一定角度，需要在旋转时间中减去这部分角度造成的开销。

读取一个扇区的数据需要 30 个时间单位。

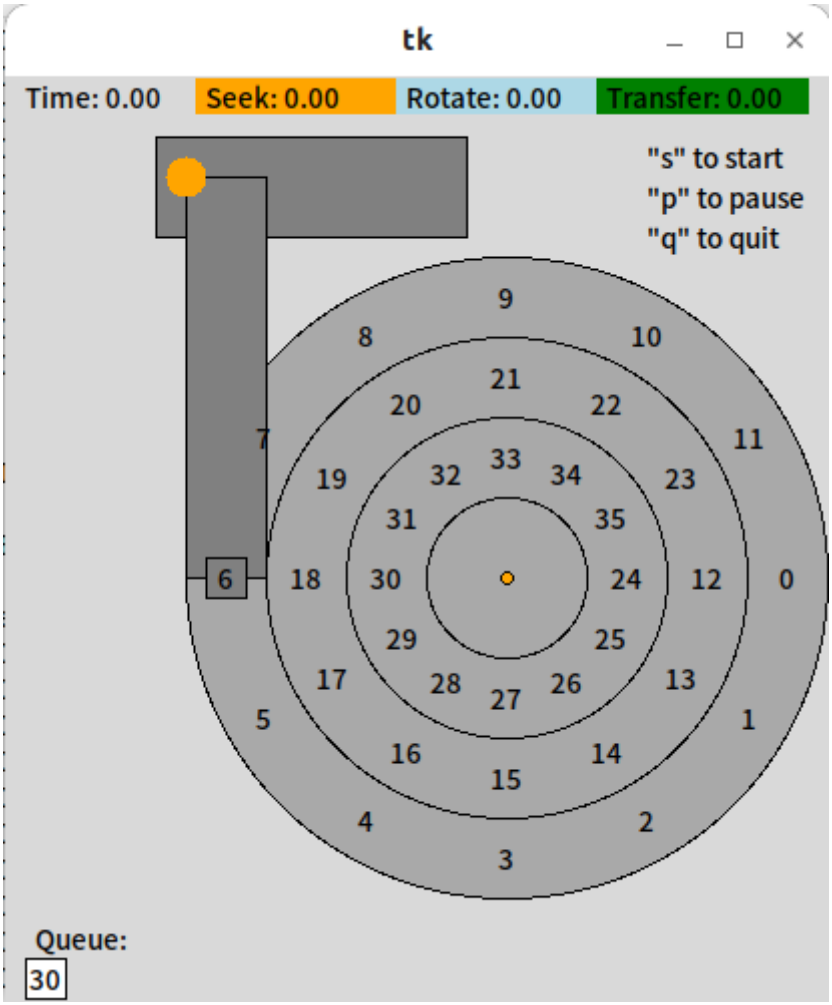
问题解答

答案矩阵如下：

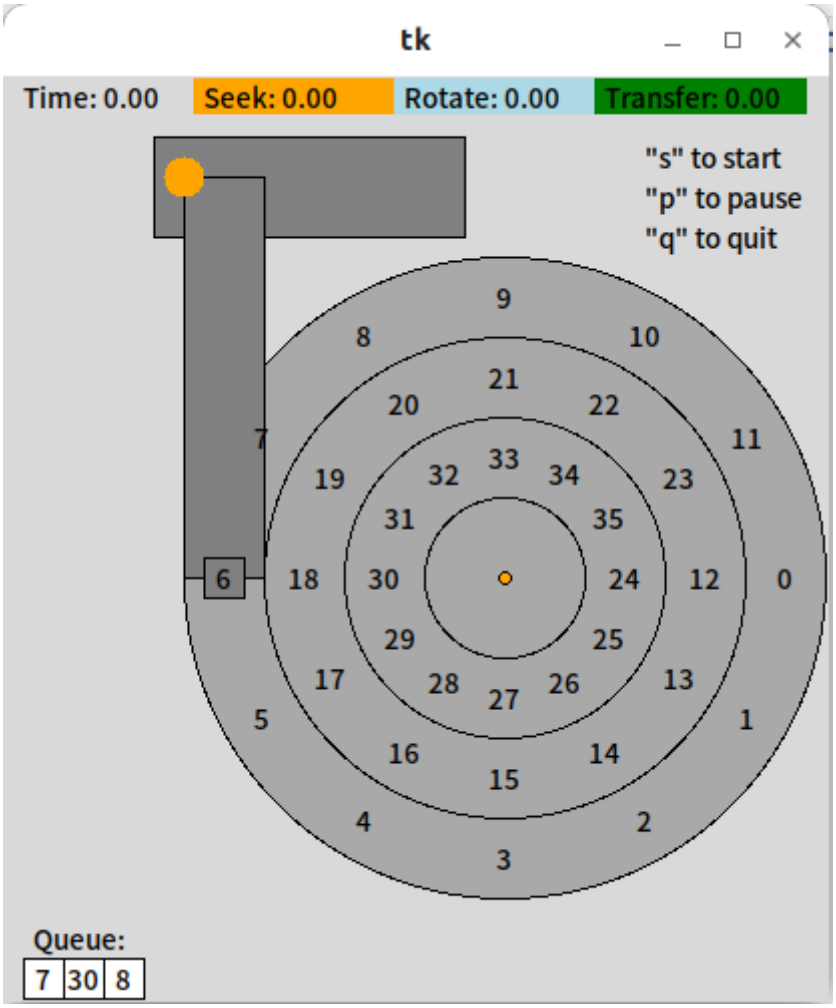
['195', '195', '195', '195', '195', '195']
['375', '375', '375', '375', '375', '375']
['375', '375', '375', '375', '375', '1095']
['795', '435', '435', '435', '435', '2235']
['585', '585', '585', '585', '585', '945']

从上述结果可以发现，不是寻道速率越快，总时间就越短，也可能不变，还是要具体问题具体分析。

将寻道速率更改为不同值，所以在此我们只考虑需要寻道的情况3、4、5。情况1、2时间不变。在此只给出部分分析。

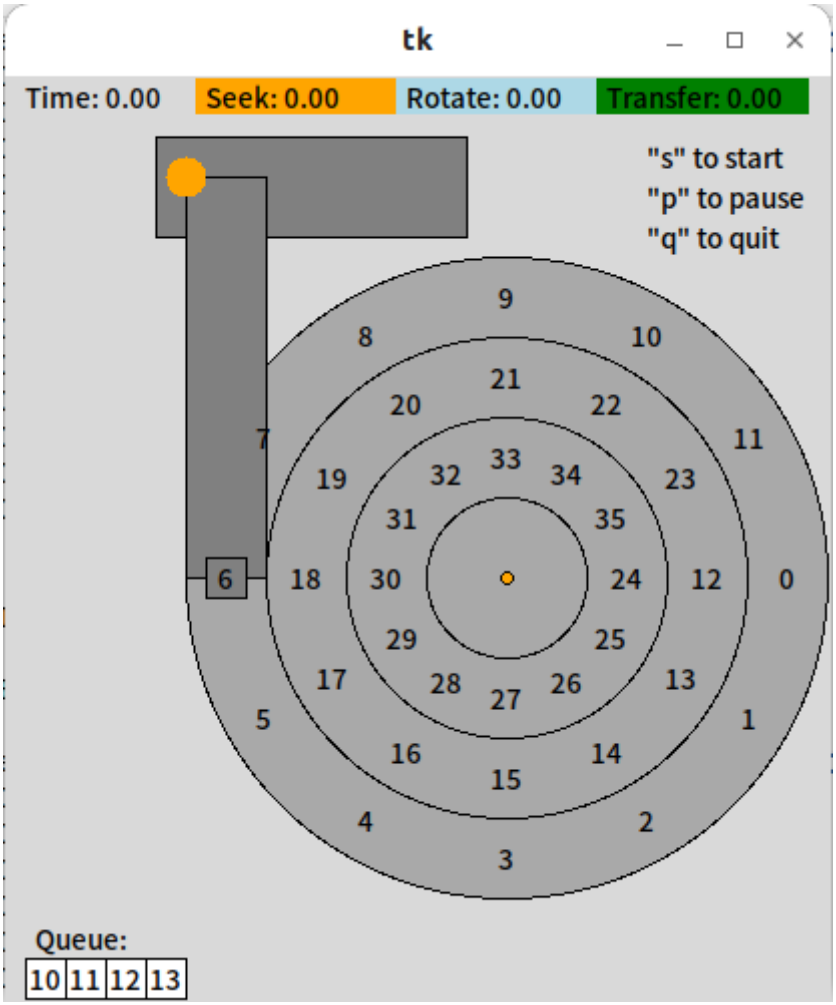
1. 

   先寻道，跨越两个磁道，耗时40个时间单位，无寻道成本时旋转345度，减去寻道时转过的角度40，40+345-40+30=375

2. 

先旋转15度，然后传输7；再寻道到内圈磁道，耗时2个时间单位，无成本寻道旋转300度，减去寻道时转过的角度2，传输30；寻道到外圈磁道，无成本寻道旋转30度，减去寻道时转过的角度2，传输8

0.5 * 30 + 30 + 2 + 300 - 2 + 30 + 2 + 30 - 2 + 30 = 435.0

3. 

   先旋转105度，然后传输10；接着直接传输11；再寻道到中间磁道，耗时400个时间单位，当寻道完成时已经转过了400度，已经转过了扇区12，还需要旋转320度，传输12；接着直接传输13

   3.5 * 30 + 30 + 30 + 400 + 320 + 30 + 30 = 945.0

答案验证

```
python2 disk.py  -a 30 -G -S 2

REQUESTS ['30']

Block:  30  Seek: 40  Rotate:305  Transfer: 30  Total: 375

TOTALS      Seek: 40  Rotate:305  Transfer: 30  Total: 375
```

```
python2 disk.py  -a 7,30,8 -G -S 40

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek:  2  Rotate:298  Transfer: 30  Total: 330
Block:   8  Seek:  2  Rotate: 28  Transfer: 30  Total:  60

TOTALS      Seek:  4  Rotate:341  Transfer: 90  Total: 435
```

```
python2 disk.py  -a 10,11,12,13 -G -S 0.1

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek:401  Rotate:319  Transfer: 30  Total: 750
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek:401  Rotate:424  Transfer:120  Total: 945
```

经验证，答案正确。 *(PS:允许少量误差)*

Problem3

问题描述

同样的请求,但改变旋转速率：-R 0.1,-R 0.5,-R 0.01。时间如何变化?

问题分析

每条轨道之间的距离为 40 个距离单位，默认搜索速度为每单位时间移动 1 个距离单位。因此，从外磁道到中间磁道的搜寻需要 40 个时间单位。

每个磁道有 12 个扇区，转速为每单位时间旋转 n 度。

磁盘头寻道和磁盘旋转是同时进行的，也就是说当寻道完成时，磁盘已经转过了一定角度，需要在旋转时间中减去这部分角度造成的开销。

读取一个扇区的数据需要 30/n 个时间单位。

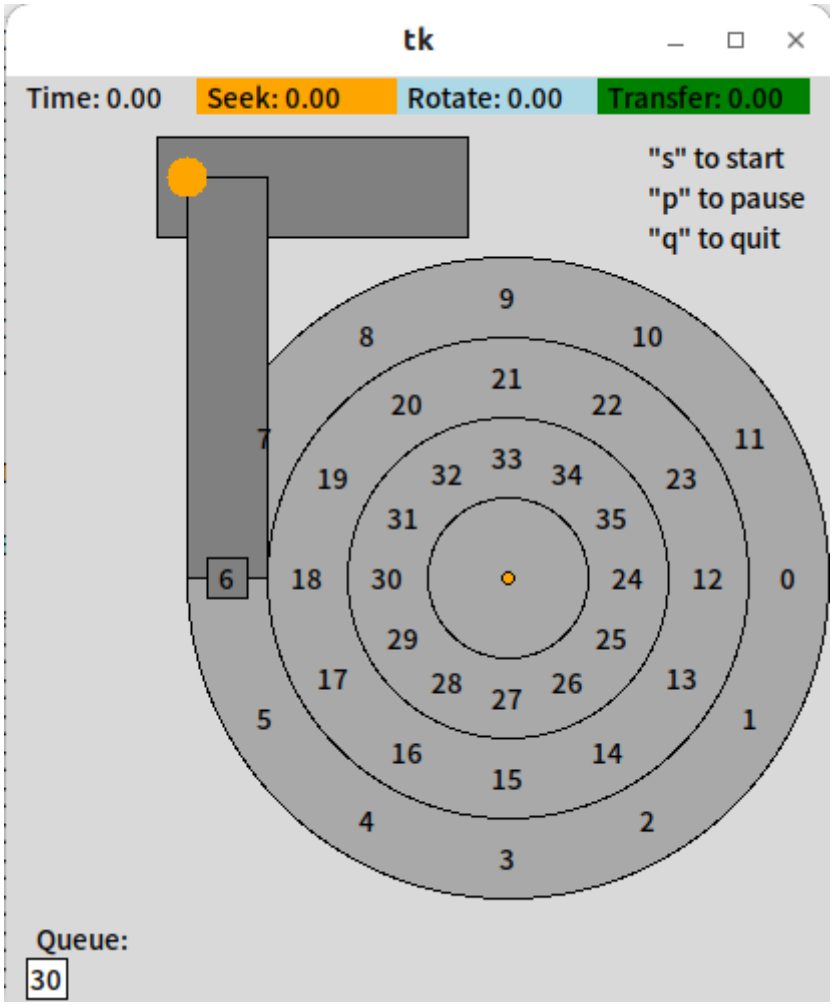
问题解答

答案矩阵如下:

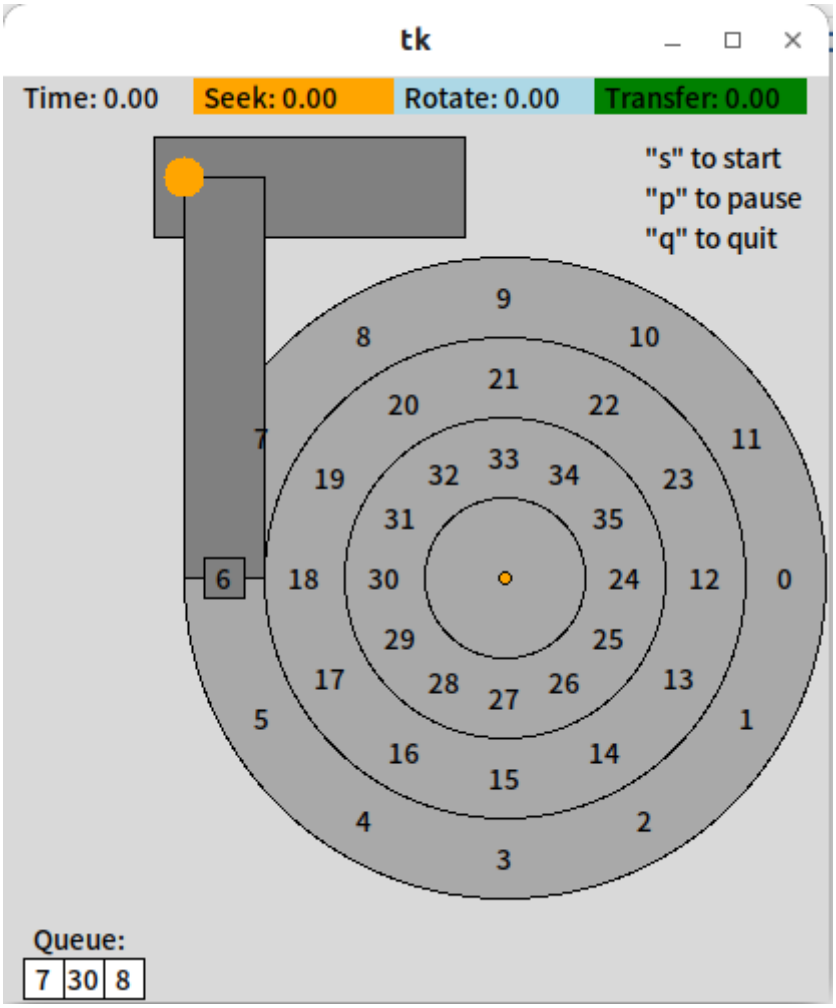
['1950', '390', '19500']
['3750', '750', '37501']
['3750', '750', '37501']
['4349', '1590', '43500']
['5850', '1170', '58501']

从上述结果可以发现，旋转速率越快，总时间就越短。

在此只分析情况3、4、5。

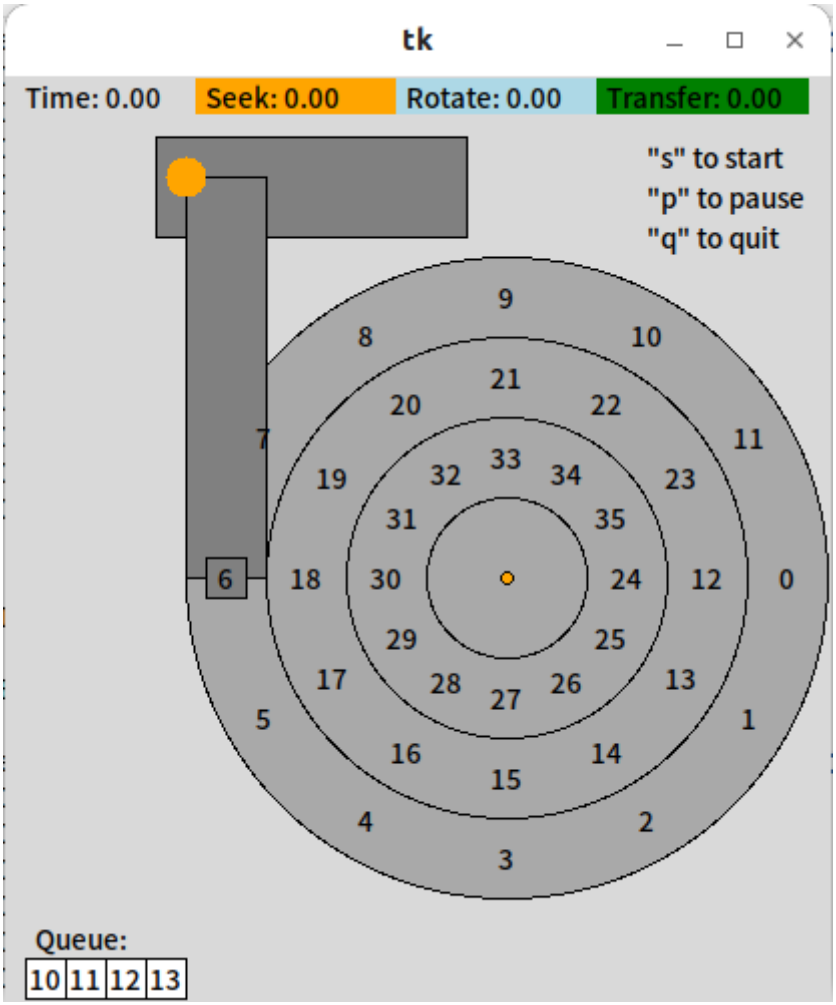
1. 

先寻道，跨越两个磁道，耗时80个时间单位，无寻道成本时旋转345度，减去寻道时转过的角度8，2*40+(345-8)/0.1+300=3750

2. 

先旋转15度，然后传输7；再寻道到内圈磁道，耗时80个时间单位，无成本寻道旋转300度，减去寻道时转过的角度40，传输30；寻道到外圈磁道，当寻道完成时已经转过了扇区8的起始位置，还需要旋转350度，传输8

0.5 * 60 + 60 + 40 * 2 + (300 - 40)/0.5 + 60 + 40 * 2 + 350/0.5 + 60 = 1590.0

3. 

   先旋转105度，然后传输10；接着直接传输11；再寻道到中间磁道，耗时40个时间单位，当寻道完成时已经转过了扇区12的起始位置，还需要旋转359.6度，传输12；接着直接传输13

   3.5 * 3000 + 3000 + 3000 + 40 + 359.6/0.01 + 3000 + 3000 = 58500.0

答案验证

```
python2 disk.py  -a 30 -G -R 0.1

REQUESTS ['30']

Block:  30  Seek: 80  Rotate:3369  Transfer:301  Total:3750

TOTALS      Seek: 80  Rotate:3369  Transfer:301  Total:3750
```

```
python2 disk.py  -a 7,30,8 -G -R 0.5

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 30  Transfer: 60  Total:  90
Block:  30  Seek: 80  Rotate:520  Transfer: 60  Total: 660
Block:   8  Seek: 80  Rotate:700  Transfer: 60  Total: 840

TOTALS      Seek:160  Rotate:1250  Transfer:180  Total:1590
```

```
python2 disk.py  -a 10,11,12,13 -G -R 0.01

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:10499  Transfer:3000  Total:13499
Block:  11  Seek:  0  Rotate:  0  Transfer:3001  Total:3001
Block:  12  Seek: 40  Rotate:35961  Transfer:3000  Total:39001
Block:  13  Seek:  0  Rotate:  0  Transfer:3000  Total:3000

TOTALS      Seek: 40  Rotate:46460  Transfer:12001  Total:58501
```

经验证，答案正确。 *（PS:允许少量误差）*

Problem4

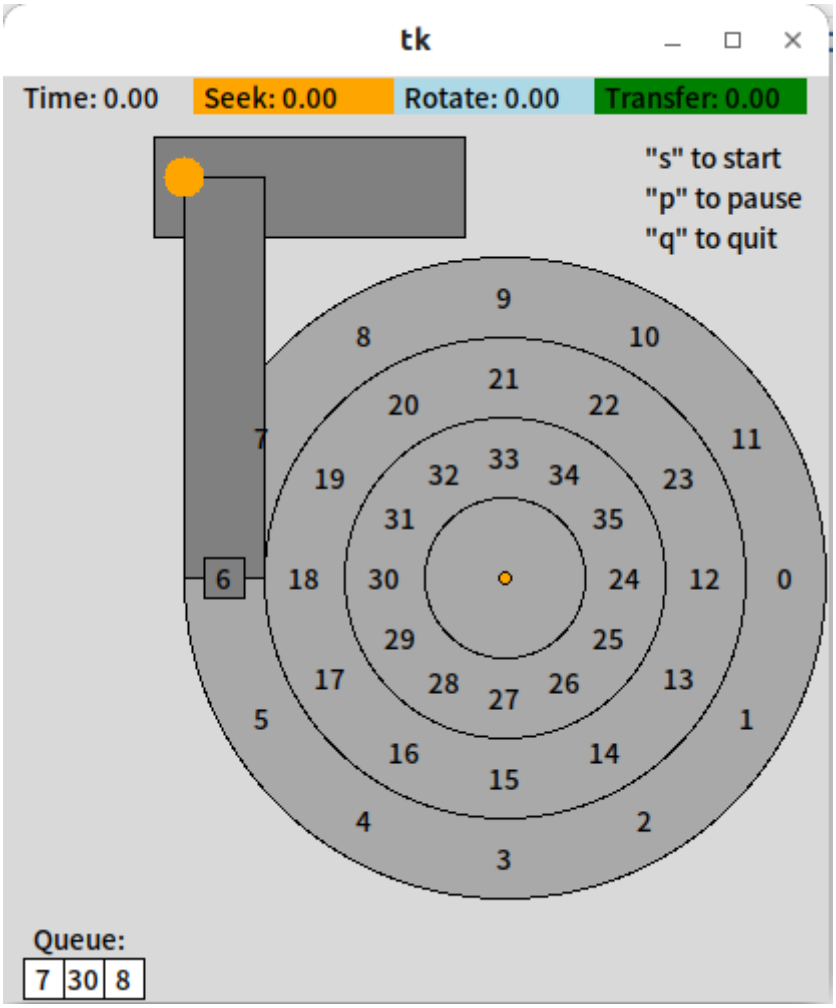
问题描述

你可能已经注意到,对于一些请求流,一些策略比 FIFO 更好。例如,对于请求流-a 7,30,8 ，处理请求的顺序是什么?现在在相同的工作负载上运行最短寻道时间优先
(SSTF)调度程序(-p SSTF)。每个请求服务需要多长时间(寻道、旋转、传输)?

问题分析

SSTF按磁道对I/O请求序列排序，选择在最近磁道上的请求先完成。这种调度算法会造成饥饿问题，即如果对当前磁道有稳定请求，那么对其他磁道的请求将永远不会被满足。

问题解答



- FIFO处理请求的顺序是: 7,30,8
  先旋转15度，然后传输7；再寻道到内圈磁道，无成本寻道旋转300度，减去寻道时转过的角度80，传输30；寻道到外圈磁道，当寻道完成时已经转过了扇区8的起始位置，还需要旋转310度，传输8
  0.5 * 30 + 30 + 40 * 2 + 300 - 80 + 30 + 40 * 2 + 310 + 30 = 795.0
- SSTF处理请求的顺序是: 7,8,30（最短寻道时间优先，扇区7和扇区8位于当前磁道，所以优先处理请求）
  先旋转15度，然后传输7；再直接传输8；再寻道到内圈磁道，耗时80个时间单位，无成本寻道旋转270度，减去寻道时转过的角度80，传输30
  0.5 * 30 + 30 + 30 + 40 * 2 + 270 - 80 + 30 = 375.0

答案验证

```
python2 disk.py -a 7,30,8 -c

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795
```

```
python2 disk.py -a 7,30,8 -c -p SSTF

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

经验证，答案正确。

Problem5

问题描述

现在做同样的事情,但使用最短的访问时间优先(SATF)调度程序(-p SATF)。它是否对 -a 7,30,8 指定的一组请求有所不同？找到 SATF 明显优于 SSTF 的一组请求。出现显著差异的条件是什么?

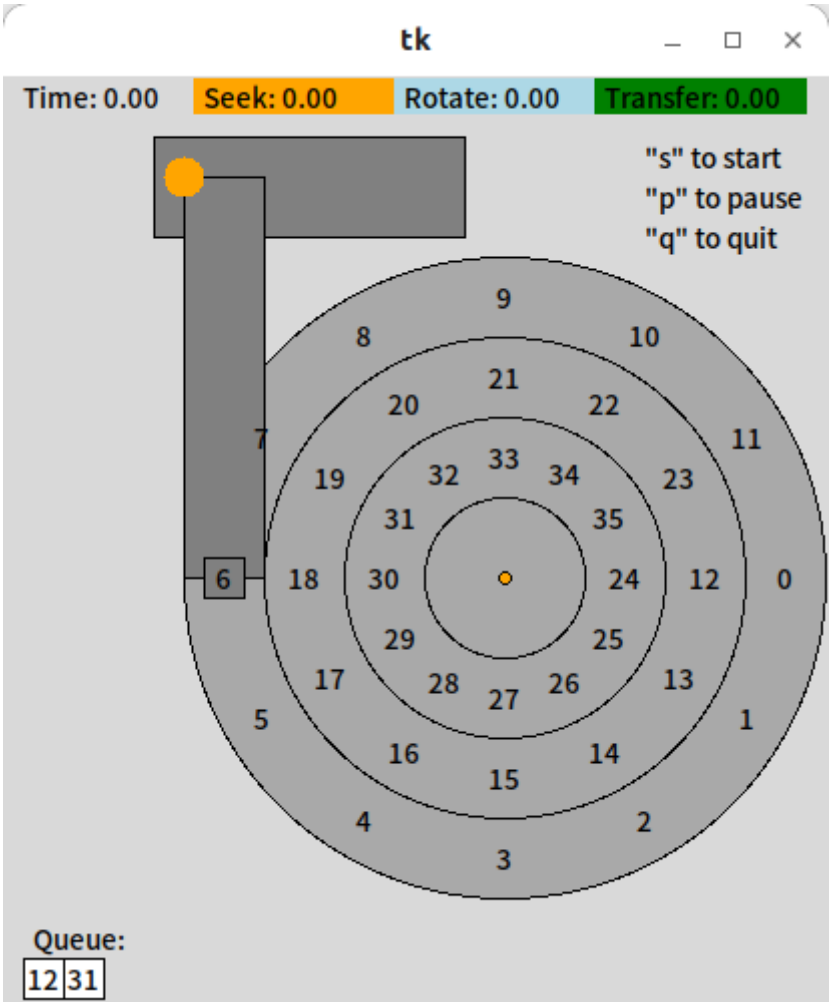
问题分析

SATF不仅考虑寻道时间，还考虑旋转时间，考虑的因素更加全面。

问题解答

- 对 -a 7,30,8，SATF 与 SSTF 没有区别（寻道速率等于旋转速率）
- SATF 明显优于 SSTF 的一组请求：

```
python2 disk.py -a 12,31 -c -S 40 -R 3 -p SSTF
python2 disk.py -a 12,31 -c -S 40 -R 3 -p SATF
```



当使用 SSTF 时，会先满足最近磁道上的请求，所以先调度12，再调度31

当使用 SATF 时，会综合考虑寻道时间和旋转时间，因为31更“近”，所以先调度31，再调度12

- 当寻道时间远小于旋转时间时，SATF 与 SSTF 出现显著差异，SATF 明显优于 SSTF。因为此时I/O的主要成本在旋转时间，而SSTF没有考虑旋转时间。

答案验证

```
python2 disk.py -a 12,31 -c -S 40 -R 3 -p SSTF

REQUESTS ['12', '31']

Block:  12  Seek:  1  Rotate: 54  Transfer: 10  Total:  65
Block:  31  Seek:  1  Rotate: 59  Transfer: 10  Total:  70

TOTALS      Seek:  2  Rotate:113  Transfer: 20  Total: 135
```

```
python2 disk.py -a 12,31 -c -S 40 -R 3 -p SATF

REQUESTS ['12', '31']

Block:  31  Seek:  2  Rotate:  3  Transfer: 10  Total:  15
Block:  12  Seek:  1  Rotate: 39  Transfer: 10  Total:  50

TOTALS      Seek:  3  Rotate: 42  Transfer: 20  Total:  65
```

经验证，答案正确。

Problem6

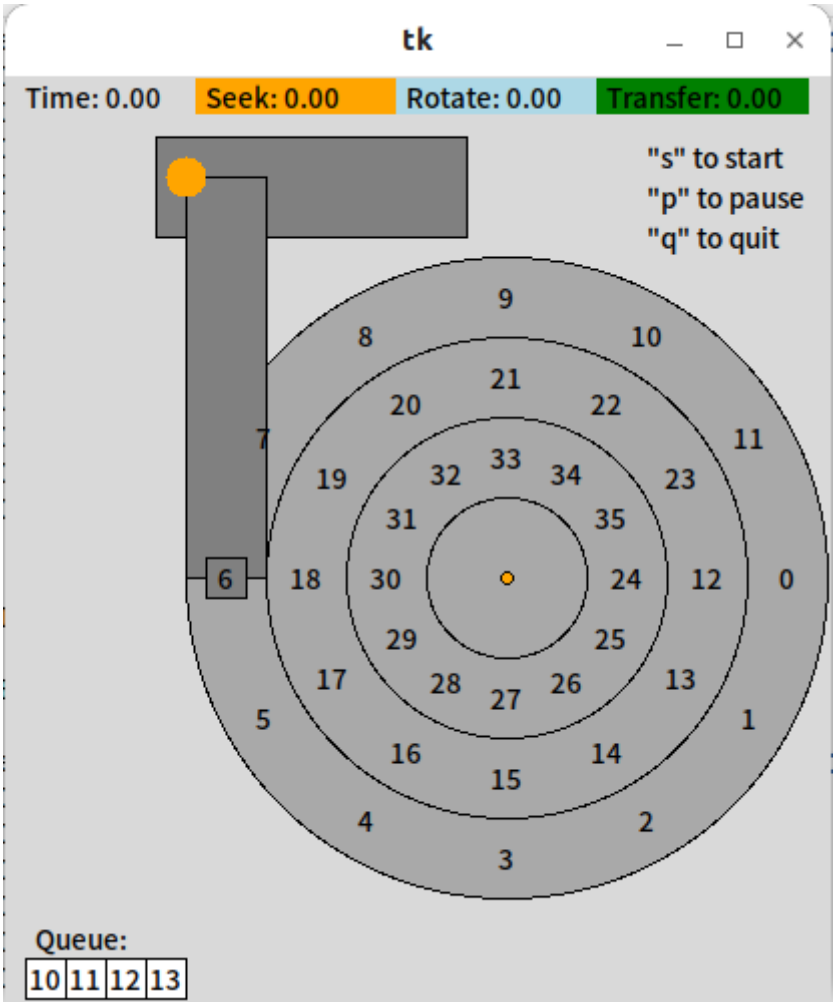
问题描述

你可能已经注意到,该磁盘没有特别好地处理请求流 -a 10,11,12,13。这是为什么?你可以引入一个磁道偏斜来解决这个问题(-o skew,其中 skew 是一个非负整数)?考
虑到默认寻道速率,偏斜应该是多少,才能尽量减少这一组请求的总时间?对于不同的寻道速率(例如,-S 2,-S 4)呢?一般来说,考虑到寻道速率和扇区布局信息,你能否写出一个公式来计算偏斜?

问题分析

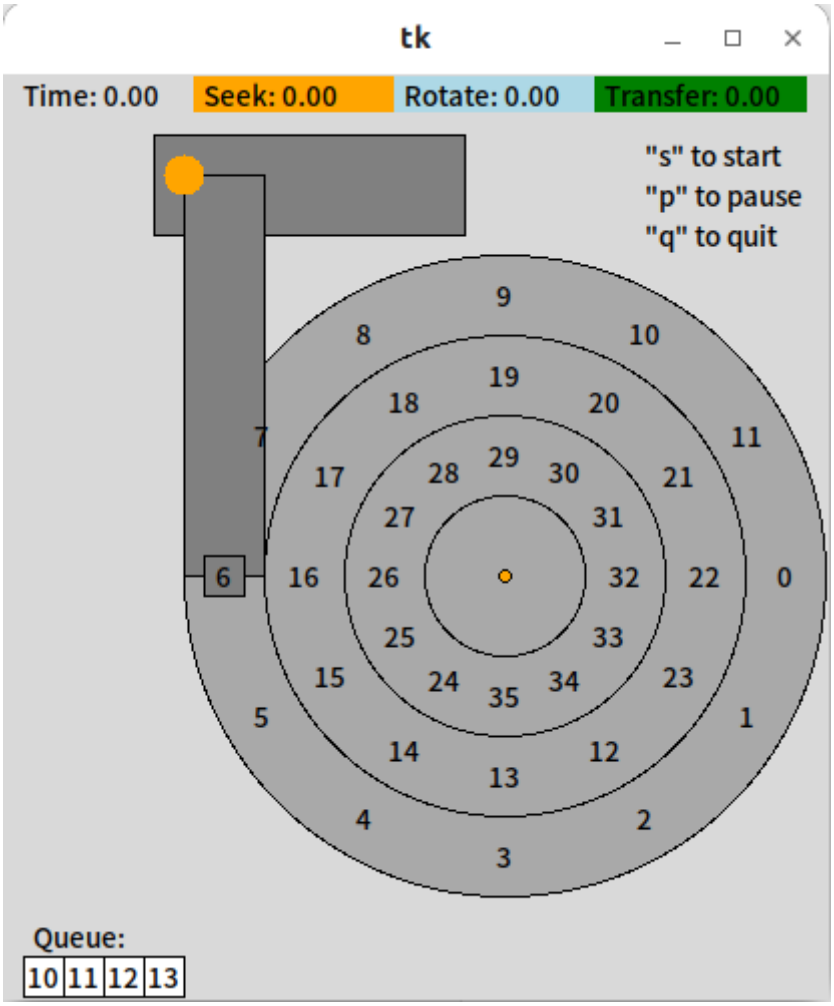
磁道偏斜用于解决在顺序读写中磁头移到下一磁道后不需要等待整个旋转延迟。

问题解答



因为寻道时间太长，导致更换磁道时，刚好旋转超过了扇区12的起始位置，导致需要重新旋转一个周期。

```
python2 disk.py  -a 10,11,12,13 -G -o 2
# 可以看到内圈磁道与外圈磁道偏移了两个扇区
```



先旋转105度，然后传输10；接着直接传输11；再寻道到中间磁道，消耗40个时间单位，还需要旋转20度，传输12；接着直接传输13

3.5 * 30 + 30 + 30 + 40 + 20 + 30 + 30 = 285.0

对于不同的寻道速率，寻道速率越快，寻道时间越短，寻道过程中磁盘转过的角度越小，磁道偏斜可以相应的减小。

答案验证

```
python2 disk.py  -a 10,11,12,13 -G -o 2

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate: 20  Transfer: 30  Total:  90
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:125  Transfer:120  Total: 285
```

经验证，答案正确。