

**操作系统实验报告**

|  |  |
| --- | --- |
| **学 院：** | **计算机科学学院** |
| **专 业：** | **计算机科学与技术（创新实验班）** |
| **学 号：** | **42112255** |
| **姓 名：** | **何佳民** |
| **指导教师：** | **孙增国** |

**2023 年 12 月 19日**

**实验六 磁盘调度算法**

**一、实验目的**

通过这次实验，加深对磁盘调度算法的理解，进一步掌握先来先服务FCFS、最短寻道时间优先SSTF、SCAN和循环SCAN算法的实现方法。

**二、实验器材**

微机

**三、实验要求**

1）上机前认真复习磁盘调度算法，熟悉FCFS、SSTF、SCAN和循环SCAN算法的过程；

2）上机时独立编程、调试程序；

3）根据具体实验要求，完成好实验报告（包括实验的目的、内容、要求、源程序、实例运行结果截图、发现的问题以及解决方法）。

**四、实验内容**

**1.问题描述：**

设计程序模拟先来先服务FCFS、最短寻道时间优先SSTF、SCAN和循环SCAN算法的工作过程。假设有n个磁道号所组成的磁道访问序列，给定开始磁道号m和磁头移动的方向（正向或者反向），分别利用不同的磁盘调度算法访问磁道序列，给出每一次访问的磁头移动距离，计算每种算法的平均寻道长度。

**2.程序要求：**

1）利用先来先服务FCFS、最短寻道时间优先SSTF、SCAN和循环SCAN算法模拟磁道访问过程。

2）模拟四种算法的磁道访问过程，给出每个磁道访问的磁头移动距离。

3）输入：磁道个数n和磁道访问序列，开始磁道号m和磁头移动方向（对SCAN和循环SCAN算法有效），算法选择1-FCFS，2-SSTF，3-SCAN，4-循环SCAN。

4）输出：每种算法的平均寻道长度。

**3.问题分析：**

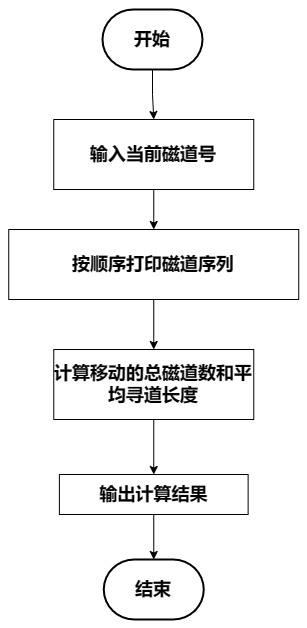


图 6-1 先来先服务FCFS流程图

先来先服务算法根据进程请求访问磁盘的先后次序进行调度。

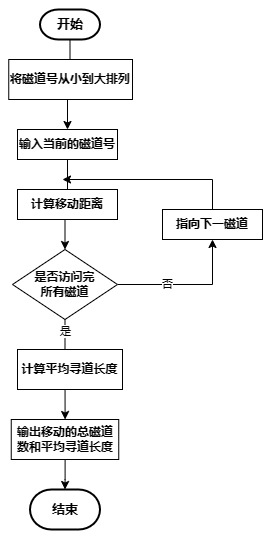


图 6-2 最短寻道时间优先SSTF流程图

该算法选择这样的进程，其要求访问的磁道与当前磁头所在的磁道距离最近，以使每次的寻道时间最短。首先将磁道号从小到大排列并输入当前的磁道号，然后计算移动距离直至访问完所有的磁道。最后计算平均寻道长度并输出移动的总磁道数和平均寻道长度。

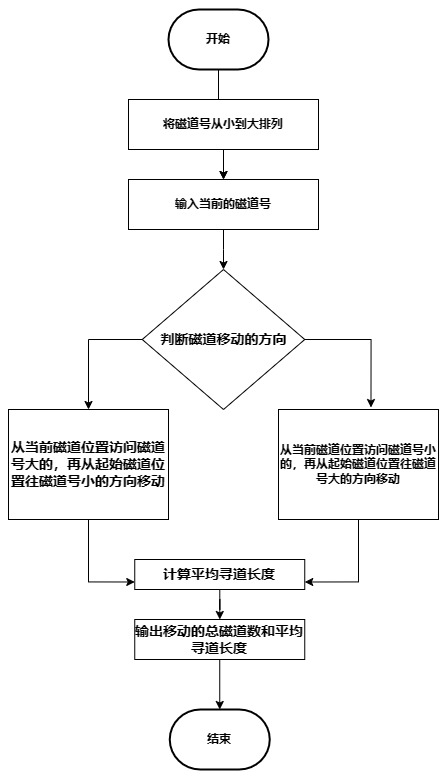


图 6-3 SCAN流程图

将磁道号从小到大排列并输入当前的磁道号，然后判断磁道移动的方向，从当前磁道位置访问磁道号大（小）的，再从起始磁道位置往磁道号小（大）的方向移动。最后计算平均寻道长度并输出移动的总磁道数和平均寻道长度。

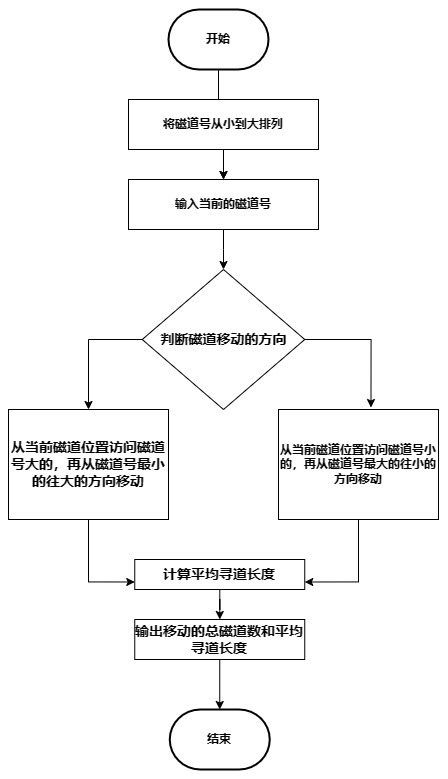


图 6-4 CSCAN流程图

首先将磁道号从小到大排列并输入当前的磁道号，然后判断磁道移动的方向，从当前磁道位置访问磁道号大（小）的，再从磁道号最小（大）方向的往大（小）的方向移动。最后计算平均寻道长度并输出移动的总磁道数和平均寻道长度。

**4.源程序：**

data.txt内容：  
9  
55 58 39 18 90 160 150 38 184  
100

Python代码（Python版本3.9）：

import numpy as np

# 定义最大数量

MaxNumber = 100

# 初始化变量

TrackNum = 0

StartTrack = 0

TrackOrder = np.zeros(MaxNumber, dtype=int)

VisitOrder = np.zeros(MaxNumber, dtype=int)

isVisited = np.zeros(MaxNumber, dtype=bool)

MoveDistance = np.zeros(MaxNumber, dtype=int)

totalDistance = 0

AverageDistance = 0.0

direction = False

def input\_data():

global TrackNum, StartTrack, TrackOrder

# 从文件中读取数据

with open("data.txt", "r") as readData:

TrackNum = int(readData.readline()) # 磁道个数

TrackOrder = list(map(int, readData.readline().split()))[:TrackNum] # 磁道访问序列

StartTrack = int(readData.readline()) # 开始磁道号

print("文件信息如下：")

print("磁道个数TrackNum =", TrackNum)

print("磁道访问序列：", ' '.join(map(str, TrackOrder)))

print("开始磁道号StartTrack =", StartTrack)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*开始\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

def getPositive(num1, num2):

return abs(num1 - num2)

def initial():

global totalDistance, AverageDistance, VisitOrder, MoveDistance, isVisited

MoveDistance.fill(0)

VisitOrder = np.copy(TrackOrder)

isVisited.fill(False)

totalDistance = 0

AverageDistance = 0.0

def FCFS():

print("\*\*\*\*\*\*\* 你选择了FCFS算法 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

initial()

MoveDistance[0] = getPositive(TrackOrder[0], StartTrack)

totalDistance = MoveDistance[0]

VisitOrder[0] = TrackOrder[0]

for i in range(1, TrackNum):

MoveDistance[i] = getPositive(TrackOrder[i], TrackOrder[i - 1])

totalDistance += MoveDistance[i]

VisitOrder[i] = TrackOrder[i]

global AverageDistance

AverageDistance = totalDistance / TrackNum

display()

print("\*\*\*\*\*\*\* FCFS算法结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

def SSTF():

global totalDistance # 添加这行来声明totalDistance为全局变量

print("\*\*\*\*\*\*\* 你选择了SSTF算法 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

initial()

CurrentTrack = StartTrack

disTemp = np.full(MaxNumber, 10000, dtype=int) # 初始化为一个大数

for i in range(TrackNum):

for j in range(TrackNum):

if not isVisited[j]:

disTemp[j] = getPositive(TrackOrder[j], CurrentTrack)

else:

disTemp[j] = 10000 # 访问过的磁道标记为大数

pointMin = np.argmin(disTemp)

if disTemp[pointMin] == 10000:

break # 如果最小值是10000，表示没有更多的磁道可以访问

VisitOrder[i] = TrackOrder[pointMin]

MoveDistance[i] = getPositive(TrackOrder[pointMin], CurrentTrack)

totalDistance += MoveDistance[i] # 更新总距离

CurrentTrack = TrackOrder[pointMin]

isVisited[pointMin] = True

disTemp[pointMin] = 10000 # 标记为已访问

global AverageDistance

AverageDistance = totalDistance / TrackNum

display()

print("\*\*\*\*\*\*\* SSTF算法结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

# SCAN算法

def SCAN():

global totalDistance # 确保使用全局变量

print("\*\*\*\*\*\*\* 你选择了SCAN算法 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

global direction

direction = int(input("请选择头移动方向（0表示向磁道增加的方向，1表示向磁道减小的方向）: "))

initial()

SortTrackOrder = sorted(TrackOrder)

point = next(i for i, v in enumerate(SortTrackOrder) if v >= StartTrack)

count = 0

currentTrack = StartTrack

# 向磁道增加的方向访问

if direction == 0:

for i in range(point, TrackNum):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

for i in range(point - 1, -1, -1):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

# 向磁道减小的方向访问

elif direction == 1:

for i in range(point - 1, -1, -1):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

for i in range(point, TrackNum):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

# 计算平均距离

global AverageDistance

if TrackNum > 0:

AverageDistance = totalDistance / TrackNum

else:

AverageDistance = 0

display()

print("\*\*\*\*\*\*\* SCAN算法结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

# CSCAN算法

def CSCAN():

global totalDistance # 使用全局变量

print("\*\*\*\*\*\*\* 你选择了CSCAN算法 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

global direction

direction = int(input("请选择头移动方向（0表示向磁道增加的方向，1表示向磁道减小的方向）: "))

initial()

SortTrackOrder = sorted(TrackOrder)

point = next(i for i, v in enumerate(SortTrackOrder) if v >= StartTrack)

count = 0

currentTrack = StartTrack

# 向磁道增加的方向访问

if direction == 0:

for i in range(point, TrackNum):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

for i in range(0, point):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

# 向磁道减小的方向访问

elif direction == 1:

for i in range(0, point):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

for i in range(point, TrackNum):

VisitOrder[count] = SortTrackOrder[i]

MoveDistance[count] = getPositive(VisitOrder[count], currentTrack)

totalDistance += MoveDistance[count]

currentTrack = VisitOrder[count]

count += 1

# 计算平均距离

global AverageDistance

if TrackNum > 0:

AverageDistance = totalDistance / TrackNum

else:

AverageDistance = 0

display()

print("\*\*\*\*\*\*\* CSCAN算法结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

# 显示结果的函数

def display():

print("从", StartTrack, "号磁道开始")

print("被访问的下一个磁道号", " 移到距离（磁道数）")

for i in range(TrackNum):

print(VisitOrder[i], " " \* 20, MoveDistance[i])

print("平均寻找长度 AverageDistance= ", AverageDistance)

# 主函数

def main():

input\_data()

IsContinue = 1

while IsContinue:

chooseAlgorithm = int(input("请选择算法（1-FCFS, 2-SSTF, 3-SCAN, 4-循环SCAN）: "))

if chooseAlgorithm == 1:

FCFS()

elif chooseAlgorithm == 2:

SSTF()

elif chooseAlgorithm == 3:

SCAN()

elif chooseAlgorithm == 4:

CSCAN()

else:

print("\*\*\*\*\*\*\* 警告：请输入正确的选择！ \*\*\*\*\*\*\*")

IsContinue = int(input("是否继续选择算法？（0代表退出，1代表继续）: "))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*结束\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**5.运行结果与分析：**

为了检查算法运行结果是否正确，用例题来说明，如下图。从磁道号100开始，访问先后序列为55 58 39 18 90 160 150 38 184。

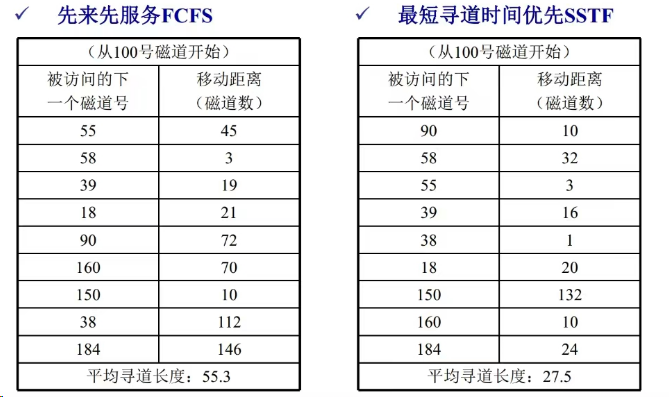


图 6-5 例题



图 6-6 例题

算法运行结果：  
输出界面如下：

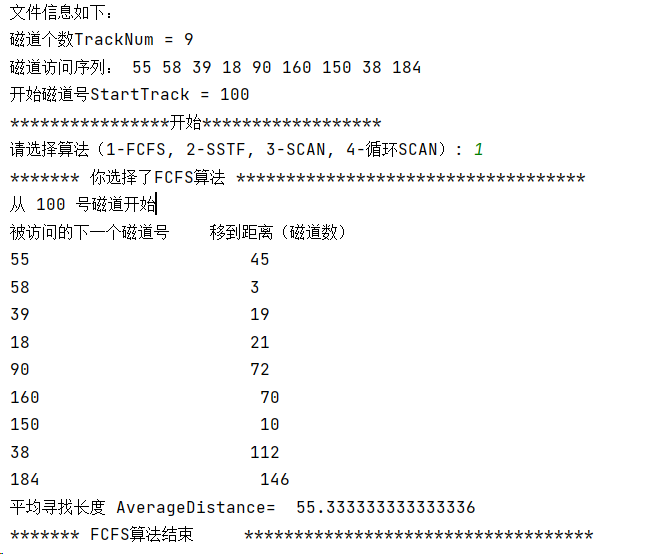


图 6-7 FCFS算法结果

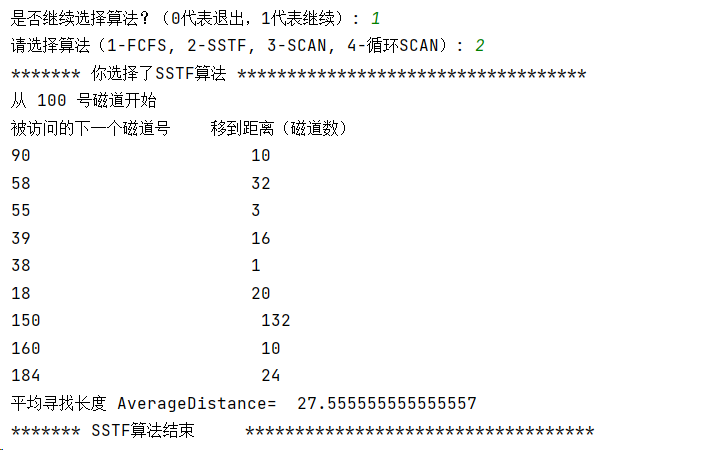


图 6-8 SSTF算法结果

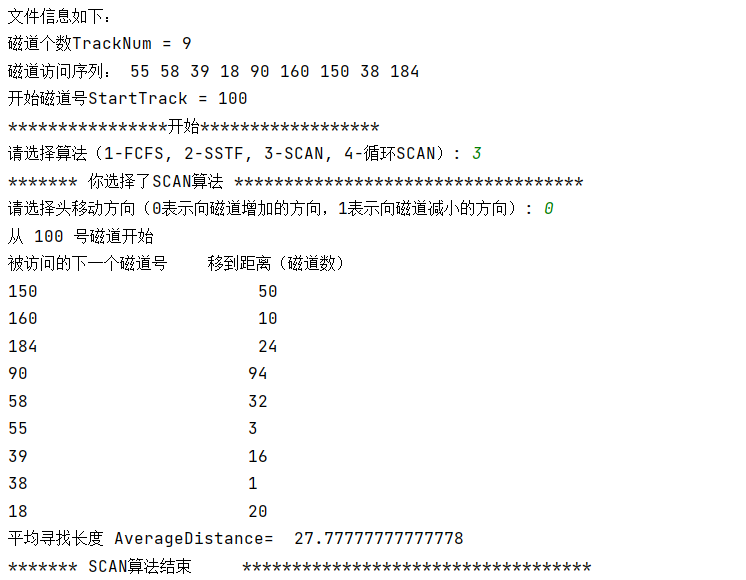


图 6-9 SCAN算法结果

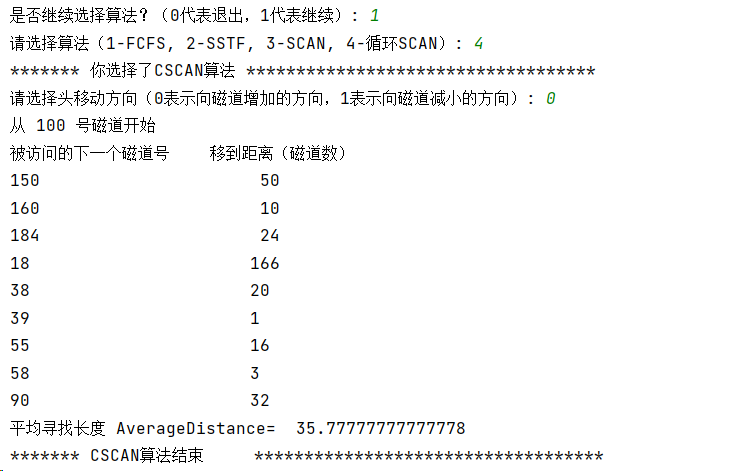


图 6-10 CSCAN算法结果

由四种算法的运行结果以及例题可知，除开数值的四舍五入，两者结果是一致的，然后是对每一种算法进行分析。

先来先服务（FCFS），此算法的优点是公平、简单，且每个进程的请求都能依次得到处理，不会出现某一进程的请求长期得不到满足的情况。

最短寻道时间优先算法（SSTF），这种调度算法不能保证平均寻道时间最短。

扫描算法（SCAN）考虑的是磁头的当前移动方向，基本上克服了最短寻道时间优先算法的服务集中于中间磁道和响应时间变化比较大的缺点。但是也有缺点，当磁头刚从里向外移动过某一磁道时，恰有一进程请求访问此磁道，这时该进程必须等待后才会处理其请求，致使该进程的请求被严重推迟。

循环扫描算法（CSCAN），CSCAN算法规定磁头单向移动，就减少了扫描算法（SCAN）的严重推迟。

**五、实验总结**

1.

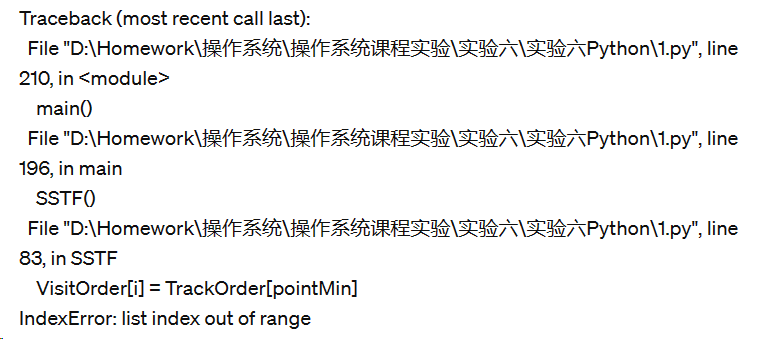
出现错误：  


图 6-11 错误截图

解决办法：  
 在SSTF函数中出现了“list index out of range”的错误。这通常发生在试图访问列表中不存在的索引时。需要确保pointMin的值始终在有效范围内。此外，我注意到在SSTF算法中，disTemp数组在每次迭代时需要重新初始化，以确保不会重复访问已经访问过的磁道。确保pointMin的值始终在有效范围内。此外，我注意到在SSTF算法中，disTemp数组在每次迭代时需要重新初始化，以确保不会重复访问已经访问过的磁道。

修改后，确保在每次迭代中，disTemp数组被正确初始化，并且一旦所有磁道都被访问过，循环就会终止。

2.

在1 的基础上，出现新的错误：

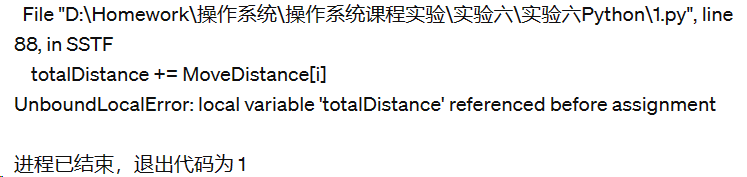


图 6-12 错误截图

解决办法：  
 错误是由于totalDistance被当作局部变量使用，需要在SSTF函数的开始处添加一行，声明totalDistance为全局变量。还有剩下的函数出现相同错误的地方都这么解决。

global totalDistance # 确保使用全局变量

3.  
选择算法SCAN和CSCAN时，出现错误：

平均寻找长度 AverageDistance= 0.0

解决办法：  
 检查SCAN函数中相关的逻辑：totalDistance在每次移动后都得到更新。在计算平均距离时，确保分母（TrackNum）不为零。

修改之后，totalDistance在每次移动后都得到更新，并且在计算平均距离时，如果TrackNum大于0，则进行计算，否则将平均距离设置为0。

4.

先来先服务算法是最简单的磁盘调度算法，它根据进程请求访问磁盘的顺序服务；

最短寻道时间优先算法根据磁头的当前位置首先将请求队列中距磁头最短的请求为之服务；

扫描算法也叫电梯调度算法，磁头固定从外向内然后从内向外逐柱面运动，如此往复；

循环扫描算法的磁头从盘面上的一端向另一端移动，遇到请求立即服务。

  在此次实验中让我对四种磁盘调度算法加深了理解，通过编程实现，把理论知识用于实践。