Design Document 4

作者 or 組員：陳育霖 B113040052

目標：Disk Scheduling Algorithms

演算法：

FCFS：先進先出，依任務序列進行移動

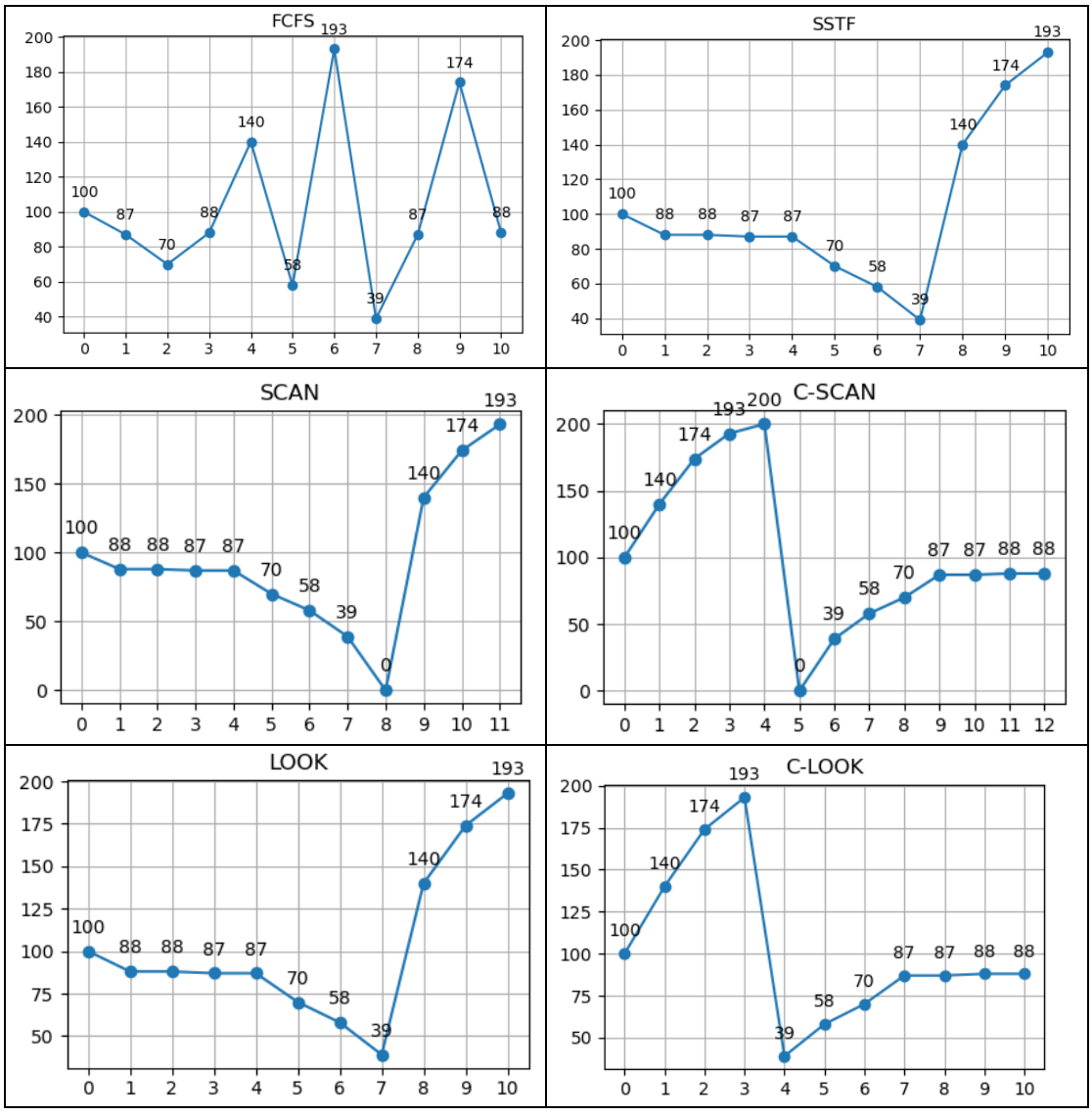
SSTF：尋找當下最近的位置，再進行移動

SCAN：移動至外圈距離最近的位置；到最外圈時，移動至內圈距離最近的位置

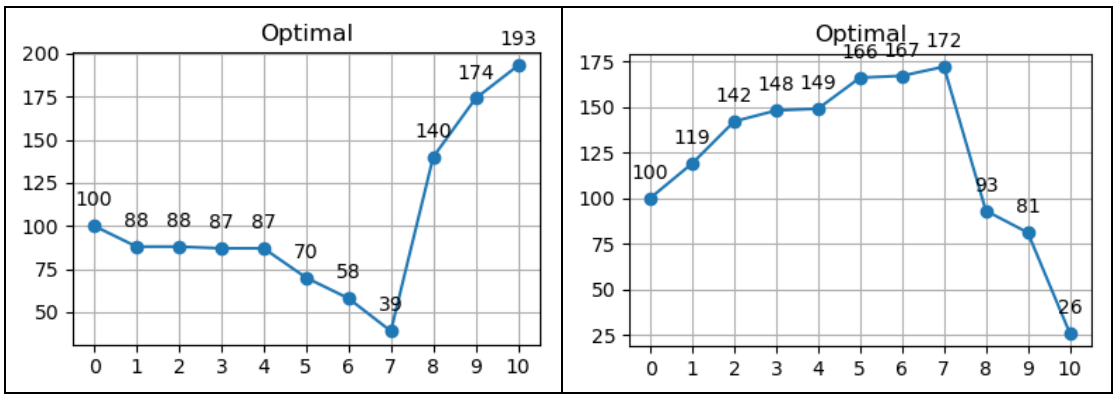
C-SCAN：移動至外圈距離最近的位置；到最外圈時，直接跳到最內圈，再往外移動到距離最近的位置

LOOK：類似於 SCAN，但只跑到最外圈的 request 就停止，不須跑到最外層

C-LOOK：類似於 C-SCAN 及 LOOK，一樣只跑到最外圈的 request，接著從最內圈的 request 開始往外跑



Optimal：先判斷最左邊和最右邊的 request 哪個離起始點最近，先走距離較近的那邊，再回到起始點，走沒完成的另一邊。以下圖片為兩種狀況的示意圖：



註: 以上圖片皆使用 python 模擬演算法並繪製

演算法的設計目的及優缺點：

FCFS：易於設計及實作

優點：簡單、公平

缺點：位置落差大，導致 request 等待時間長

SSTF：優先處理離當前最近的 request，減少尋道時間

優點：較 FCFS 移動量少

缺點：可能導致 request 較外圍的會飢餓(starvation)

SCAN：提供更平均的等待時間

優點：時間較可預測，飢餓發生的可能較 SSTF 低

缺點：如果在最外圈或最內圈沒有 request 時，效率會較低

C-SCAN：相較於 SCAN 來說，每個位置的處理時間與頻率會更平均

優點：所有 request 的等待時間非常平均

缺點：僅提供一個掃描方向，整體時間會較 SCAN 高一些

LOOK：與 SCAN 的目的相似，減少 SCAN 中不必要的移動

優點：移動距離及時間較 SCAN 少

缺點：設計較複雜

C-LOOK：與 C-SCAN 相似，提供更均勻的等待時間與減少不必要的移動

優點：花費時間及移動距離較 C-SCAN 少

缺點：設計較複雜，僅提供一個掃描方向，時間較 LOOK 高

Optimal：依起始位置與 request 分布，來選擇最佳路徑

優點：平均所需時間及移動量最少

缺點：設計最複雜，需先判斷怎麼走會最佳

檔案內容：

index.c　模擬 Disk 調度器演算法，以下為可自由調整之變數：

```
#define request_num 1000    // 設定 request 的生成數量
#define cylinders 5000    // 設定 cylinders 的範圍
#define initial 0    // 設定初始位置
#define run_times 10    // 設定測試次數
```

graph.py　生成 Disk 調度器演算法示意圖，以下為主要設定：

```
num_requests = 10    # 設定 request 數量
requests = np.random.randint(0, 200, size=num_requests)    # cylinders 的範圍
initial_head_position = 100    # 初始位置
```

程式流程圖：

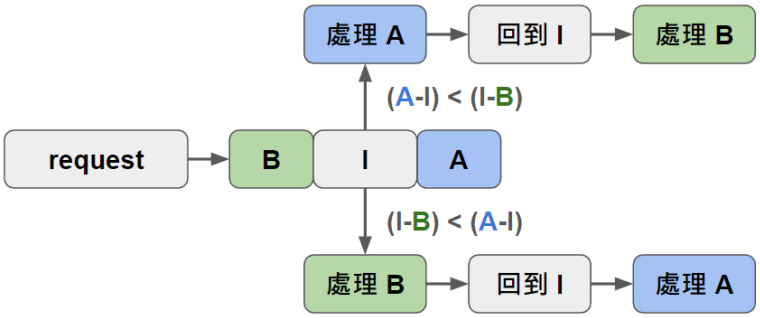


圖 1　Optimal 演算法流程圖

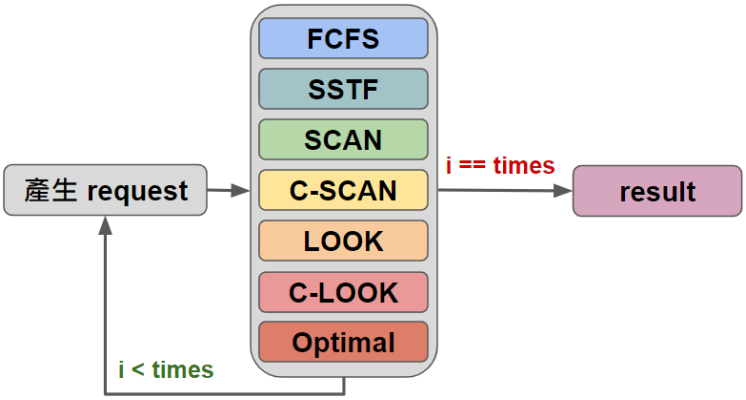


圖 2　程式測試流程圖 (times 為執行次數)

程式測試步驟：

clang index.c -o index

./index

測試結果：

時間 (單位: 秒)

start at	FCFS	SSTF	SCAN	CSCAN	LOOK	CLOOK	Optimal
0	16.698s	0.050s	0.050s	0.050s	0.050s	0.050s	0.050s
1000	16.692s	0.060s	0.090s	0.100s	0.090s	0.099s	0.060s
2500	16.522s	0.075s	0.075s	0.100s	0.075s	0.099s	0.075s
5000	16.427s	0.050s	0.050s	0.100s	0.050s	0.099s	0.050s

移動量

start at	FCFS	SSTF	SCAN	CSCAN	LOOK	CLOOK	Optimal
0	1669836	4998	4998	4998	4998	4998	4998
1000	1669220	5977	8987	9995	8987	9973	5977
2500	1652246	7523	7498	9997	7495	9991	7493
5000	1642741	4998	4998	9995	4998	9991	4998

演算法應用場景：

FCFS：適合應用在 I/O 請求數量不多、且請求間隔時間較長的場景。

例如：**個人電腦、小型伺服器**

SSTF：適合應用在 I/O 請求位置相近的場景。

例如：**伺服器的磁碟陣列、資料庫系統**

SCAN：適合應用在 I/O 請求位置分散的場景。

例如：**工作負載繁重的伺服器、大型資料中心**

C-SCAN：適合應用在需要兼顧尋道時間和公平性的場景。

例如：**多使用者共用的磁碟機、效能要求較高的伺服器**

LOOK：是 SCAN 的改良版本，效能優於 SCAN，應用場景與 SCAN 相似

C-LOOK：C-SCAN 的改良版本，效能優於 C-SCAN，應用場景與 C-SCAN 相似

Optimal：是效率最高的，但僅限於理論研究，現實無法預測所有 I/O 的位置

例如：**離線磁碟排程、模擬和仿真**