資料結構 HW2-Heap

B12508026 戴偉璿

May 29, 2025

1 程式內容解析

1.1 前處理

1.1.1 輸出檔案

首先使用 `init()` 函數宣告一個 `ofstream` 物件以清空 `output` 檔案，接下來宣告一個 `append` 模式的 `ofstream` 物件 `fout` 以便於後續寫入排程結果。

1.1.2 輸入檔案

接著使用 `read_data()` 函數，調用 `ifstream` 物件讀取 `data` 檔案，這邊我把.csv 檔案換成以空格分隔的純文字檔案方便處理

1.1.3 型態別

由於一個病人有三個數值（id，抵達時間，優先度）要紀錄，因此使用 `struct` 來定義一個 `patient` 結構體，方便處理

1.1.4 檢查

設計了一個 `check()` 函數來檢查是否有正確輸入，避免讀取到一堆亂碼或是錯誤的數值，這個函數會輸出所有輸入的資料在終端機，由於我們處理過後的資料是直接使用檔案流輸出到檔案，因此不影響結果。

1.2 Heap

1.2.1 Heap 結構體

Heap 使用一個先前宣告過的 `patient` 結構體陣列來儲存病人資料，並且使用一個整數變數 `pt` 來紀錄目前 Heap 的大小，`pt` 的數值本身也代表這個 Heap 最後一個元素的 index。

1.2.2 插入

插入到 Heap 的最後面，接下來再一一與他的父節點比較，如果比父節點大就交換位置，直到不需要交換為止。

使用了位元運算計算父節點的 index，這樣可以節省一些運算時間。

1.2.3 頂端元素

取得 Heap 的頂端元素就是取得 `hp[1]`

1.2.4 刪除頂端元素

通常取得頂端元素之後會將其刪除，刪除頂端元素就是將最後一個元素放到頂端，將 `pt` 減一，接下來再一一與他的子節點比較，如果比子節點小就交換位置，直到不需要交換為止。

1.2.5 是否為空

判斷 Heap 是否為空就是判斷 `pt` 是否為 0。

1.2.6 除錯

這個結構體提供了一個 `show()` 函數，可以列印出 Heap 的內容，方便除錯。

1.3 排程

排程的過程比較麻煩，我們維護一個 `current_time` 變數來紀錄目前的時間，每次從 Heap 中取出頂端元素（也就是目前優先度最高的病人）之後，就將時間增加他需要的治療時間，接著將這個病人寫入 `output` 檔案中。接下來再將所有抵達時間小於等於目前時間的病人加入 Heap 中，然後再從 Heap 中取出頂端元素，重複這個過程直到 Heap 為空。如果 Heap 為空但是還有病人沒有處理完，就將時間增加到下一個病人抵達的時間，迴圈會自動判斷並姐將這個病人加入 Heap 中。

這一部份要非常小心，因為如果一口氣將所有小於當前時間的病人丟到 Heap 中，然後一口氣處理完之後才加入下一組病人，可能會導致處理過程中有優先度更高的病人到來。

2 排程結果（如 output 檔案）

Patient ID	Time	Priority
1	0∼3	3
2	3∼6	3
3	6∼9	3
5	9∼14	2
4	14∼17	3
8	17∼22	2
9	22∼29	1
13	29∼34	2
15	34∼39	2
17	39∼46	1
20	46∼51	2
23	51∼56	2
6	56∼59	3
26	59∼64	2
7	64∼67	3
29	67∼74	1
10	74∼77	3
11	77∼80	3
12	80∼83	3
14	83∼86	3
16	86∼89	3
18	89∼92	3
19	92∼95	3
21	95∼98	3
22	98∼101	3
24	101∼104	3
25	104∼107	3
27	107∼110	3
28	110∼113	3
30	113∼116	3