Data Structure
Homework: Priority Queue

B12508025 盧雅筠

一、演算法說明

使用 Binary min-heap 實作Priority Queue

1. 資料讀取與預處理

   從 Scheduling for ER Patients.csv 讀入 30 位患者的
   - patient ID
   - arrival time
   - priority: 1 = 最高、2 = 次中、3 = 最低
2. Binary min-heap
   - vector<Patient> 儲存二元樹，以一個節點 i 來討論
     - 其父節點為 $(i-1)/2$ 小數點無條件去除取整數
     - 其左子節點為 $2*i+1$
     - 其右子節點為 $2*i+2$
   - push()
     - 將新患者置於陣列末端
     - 呼叫 heapifyUp
   - pop()
     - 用末端元素覆蓋 root
     - pop_back() 刪除末端
     - 呼叫 heapifyDown
   - top()
     - 取出堆頂（最優先患者）
   - lessThan(a,b)
     - 若 a.priority != b.priority，則 a.priority < b.priority 較優先
     - 否則 a.arrival < b.arrival 較優先
   - heapifyUp(int i)
     - 用途：當 push() 後，用來把這個節點「往上」移動，直到min-heap
     - 以 lessThan(child, parent)
       - 若子節點更高優先級，就與父節點交換
       - 否則代表正確放置，直接跳出迴圈
     - 重複：每次交換後，把 i 更新為剛剛的父節點位置，繼續往上看新的父節點，直到min-heap
   - heapifyDown(int i)
     - 用途：當你 pop() 後，root 可能不再最小，用此函式把它「往下」移動，直到min-heap
     - 以 lessThan() 找smallest
       - 在父節點和它的左右子節點三者中，找出優先最高的那一個，記作 smallest。
     - 判斷並交換
       - 若 smallest 不是 i（代表某子節點比父節點更優先），就把父節點與最輕的子節點互換，並把 i 更新為 smallest
       - 否則代表父節點已經是三者之中最輕
     - 重複：每次交換後，新的 i 位置可能仍需和它的子節點比較，直到min-heap

3. int main()

- 設 currentTime = 0，指標 idx = 0（指向下一位尚未入堆的患者）
- 重複以下步驟直到所有患者皆被處理
    - 將所有 arrival ≤ currentTime 的患者即，代表該患者已經在候診區，按序 push 進 heap
    - 如果沒有患者在候診區，則跳到下一位患者的到達時間
    - pop the top of heap
        - 計算該患者的服務時間（priority=1→7、2→5、3→3
        - 記錄 start = currentTime、finish = currentTime + serviceTime
        - 再更新 currentTime = finish
        - 如此保證每次取出的，都是尚在候診且「最緊急、最先到」的患者

二、結果

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