演算法程式作業三

Offline Caching Problem

繳交期限：05/28

Description

參考作業 7-1 使用 furthest-in-future 的概念設計有效率的演算法，實作 cache 管理機制。

在電腦系統中，會利用 cache 機制減少存取資料所花的時間。當程式執行時，會有一連串的資料請求，若請求的資料在 cache 中，稱為 cache hit，若不在則稱為 cache miss。當 cache miss 時，若 cache 內還有空間會直接將此資料放入 cache 中，如果 cache 已滿則會將 cache 內的其中一個元素移出，放入當前的資料。

Furthest-in-future 是一種處理 offline caching 的 greedy 策略。在已知請求資料序列(request sequence) 的情況下，每當 cache miss 且 cache 已滿，會選擇替換掉 cache 中在 request sequence 中距離最遠的元素。

目前系統有大小為 k 個 block 的 cache C，資料請求序列為 $b_1, b_2, \dots, b_N$。使用 Furthest-in-future 的機制管理，對於每一筆 request，若 cache hit 輸出 “hit”，cache miss 輸出 “miss”，其中若有將 block 移出則在 “miss”的下一行輸出 “evict x“ x 為移出的元素。如 cache 內有多個未來都不會用到的 block，則優先移出先進入到 cache 的 block。

(詳細說明參考作業 7-1、Introduction to Algorithms - Fourth Edition 15.4)

需繳交報告及程式碼，要求如下

Report

- Pseudo Code 與演算法說明
- Cache 管理資料結構說明
- 自行設計測資評估執行時間，並分析時間與空間複雜度

Code

- 以 C/C++ 實作
- 繳交的程式碼請參考以下格式
- Input 說明

  第一行輸入兩正整數 $K, N((1 \le K \le 1000, 1 \le N \le 10000000)$代表 cache 大小及 request sequence 的長度，接下來輸入 N 個正整數 b_i $(0 < b_i < 2{,}147{,}483{,}647)$ 代表每一筆 request

  - Output 說明

    對於每一筆 request，若 cache hit 輸出 “hit”、cache miss 輸出 “miss”，其中若有將 block 移出則在 “miss” 的下一行輸出 “evict X“ X 為移出的 block

- Sample Input
  3 6
  1 2 3 2 4 3
- Sample Output
  miss
  miss
  miss
  hit
  miss
  evict 1
  hit

繳交作業檔名格式 ：

- 程式碼：學號_姓名_hw3.cpp
- 報告：學號_姓名_report3.pdf