

演算法程式作業三

110403518 林晉宇

一、furthest-in-future

本次作業題目為 furthest-in-future 管理 cache 的演算法，並透過 C++ 實作來模擬 cache 的運作。

報告包含：

1. 實作程式的 pseudo code 以及資料結構的說明
2. 複雜度分析
3. 如何透過 python 生成測資
4. 程式執行時間圖

二、實作方法及資料結構說明

我參考楊弘笠同學在作業 7-1 簡報中的作法：

1. 建一個 **record** 陣列，用來記錄下一個相同 request 的 index。
 2. 每次讀入新的 request 時，判斷是否在 cache [1]:
 - (1) 在 cache，則 cache hit。
 - (2) 不在 cache，且 cache 還有空間，則 cache miss，並把 request 值加入 cache。
 - (3) 不再 cache 且 cache 已滿，則 cache miss，並選出在 request sequence 中最遠的元素並移除 [2]。
- 資料結構說明：
 - [1] 使用 C++ 的 `unordered_map` (hash map)，增刪查找為近乎常數時間。
 - [2] 使用 C++ 的 `priority_queue<pair<int, int>>`，pair 中第一個值代表下一個相同 request 的 index(即 record 陣列所存的值)，

第二個代表該 request 的值。priority_queue 目的在於確保距離最遠的 request 會在 priority_queue 的 top()，其在新增及刪除元素為 $O(\log n)$ 。

- Pseudo code:

```
unordered_map<int,bool> cache;
int cur_cache_size = 0; // 當前 cache 有多少元素
priority_queue<pii> pq;
for(i = 1 ~ n) // 1-based
{
    if(cache[request[i]]) // cache hit
        pq.push({record[i], request[i]});
    else if(cur_cache_size < k)
    {
        cur_cache_size++;
        cache[request[i]] = true;
        pq.push({record[i], request[i]});
    }
    else
    {
        int ev = pq.top().second; // 要 evict 的 request 值
        cache[request[i]] = true;
        pq.push({record[i], request[i]});
    }
}
```

三、分析複雜度

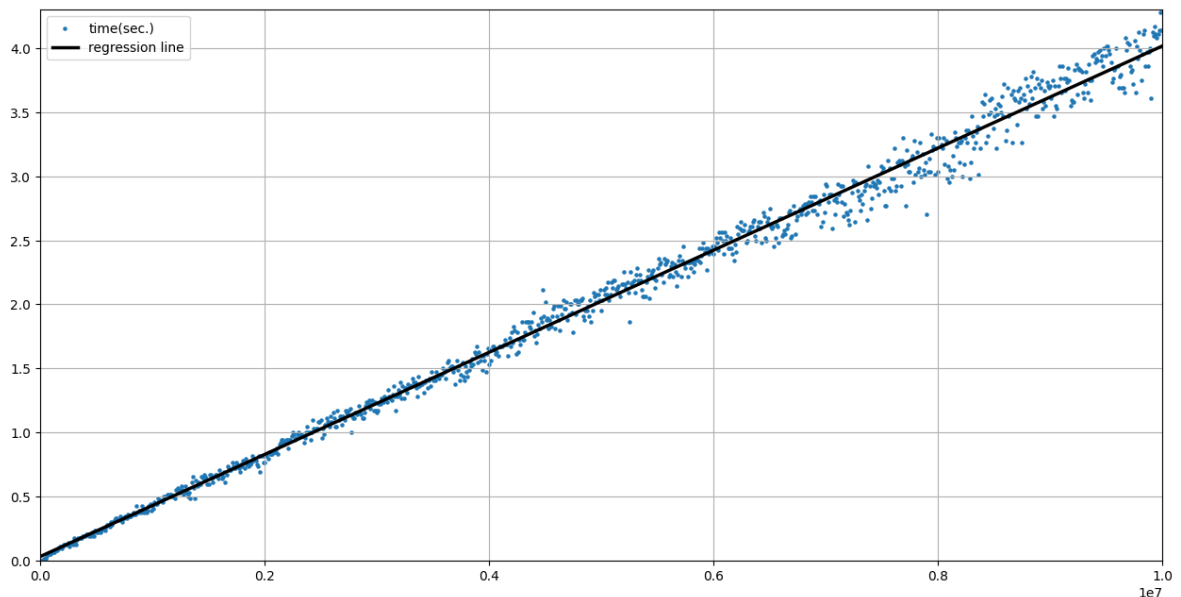
- 時間複雜度：一共有 n 個 request，每次 request 需要將資訊 push 進 priority_queue，時間為 $O(n \log n)$ 。
- 空間複雜度： $O(n)$ 。

四、如何生成測資

```
test.py > ...
1  import os
2  import random
3  import numpy as np
4
5  os.system("g++ 110403518_林晉宇_hw3.cpp -o t") # compile
6
7  open('output.txt', 'w').close()
8
9  for i in range(1, 10000000, 10000):
10     print("case: ", i)
11     K = random.randint(1, 1000)
12     N = i
13     f = open("testcase.in", "w")
14     f.write(f"{K} {N}\n")
15     B = np.random.randint(100000, size = N)
16     for j in B:
17         f.write(f"{j} ")
18     os.system("./t")
```

以上為生成測資的 python 程式碼， n 從 1 開始慢慢地增至 10^7 (每次增加 10000)， k 及 b 皆為隨機產生，總共產出 1000 筆資料。

五、結果



將 1000 筆測資讀入程式後，將每一次的執行時間畫成圖表，橫軸為 N 的大小，縱軸為時間(秒數)，藍點為實際時間，黑線為回歸線，可以看到當 N 大至 10^7 時，執行時間大約需要四秒。