演算法程式作業三

110502529 陳柏燊

Pseudo code:

先從後往前跑，把每個 cache 下一次出現的位置找出來存在 nextPos array,利用 hash_table 紀錄位置

```
//pregenerate nextPos
hash_table<int,int> pos;
for i in range(N, 0, -1):  // O(N)
    if cache[i] in pos:
        nextPos[i] = pos[cache[i]]
        pos[cache[i]] = i
    else:
        nextPos[i] = inf
        pos[cache[i]] = i
```

Time complexity $\theta(N)$

Space complexity $\theta(N)$

等等有用到自訂結構,先說明結構

```
//node struct
struct node{
    int i, x, next;
};

auto cmp = [](node& a, node& b) {
    if (a.next != b.next) return a.next < b.next;
    return a.i > b.i;
};
```

等等這個 node 會是 priority_queue 裡的 node, i 代表進入 cache 的 index, x 代表 cache, next 代表這個 cache 下一次出現的 index

```
//main algorithm
priority_queue<node> Q(cmp);
for i in range(N):  O(N)
    if cache[i] in Q    //這裡用hash table實作 O(1)
        cout<<"hit\n"
        Q.modify(cache[i], node(Q[cache[i]].i, cache[i], nextPos[i])) //O(log(k))
    else
        cout<<"miss\n"
        if Q.size() < k
            Q.push(node(i, cache[i], nextPos[i])) //O(log(k))
        else
            cout << "evict " << Q.top().x << "\n";
            Q.pop();
            Q.push(node(i, cache[i], nextPos[i])) //O(log(k))
```

主要程式,從 0 開始遍歷到 N-1, 如果 cache[i] 在 cache 裡, 則 modify Q 裡 cache[i].next 不然就先 check cache 是否滿了, 沒滿就直接放進去, 滿了就 evict Q.top() , 然後再放進去

Time complexity $\theta(N\log(K))$

Space complexity $\theta(N+K)$ //Q 是 k 其他是 N

程式實作方式

```
__gnu_pbds::priority_queue<node, decltype(cmp), pairing_heap_tag> pq(cmp);
gp_hash_table<int, __gnu_pbds::priority_queue<node>::point_iterator> pqPos;

for (int i = 0; i < N; i++) {
    if (pqPos.find(cache[i]) != pqPos.end()) {
        cout << "hit\n";
        pq.modify(pqPos[cache[i]], node((*pqPos[cache[i]]).i, cache[i], nextPos[i]));
        continue;
    }       boson13579, 2 週前 • practice

    cout << "miss\n";
    if (pq.size() < K)
        pqPos[cache[i]] = pq.push(node(i, cache[i], nextPos[i]));
    else {
        cout << "evict " << pq.top().x << "\n";
        pqPos.erase(pq.top().x);
        pq.pop();
        pqPos[cache[i]] = pq.push(node(i, cache[i], nextPos[i]));
    }
}
```

利用__gnu__pbds namespace 下的 priority_queue (std 的 priority_queue 不能 modify) , pbds 的 pq, push element 進去的時候會 return 一個 pointer, 利用 hash map 把 cache 對應到的 pointer 存下來, 下次 modify 的時候, 直接更改這個 pointer 的值就好

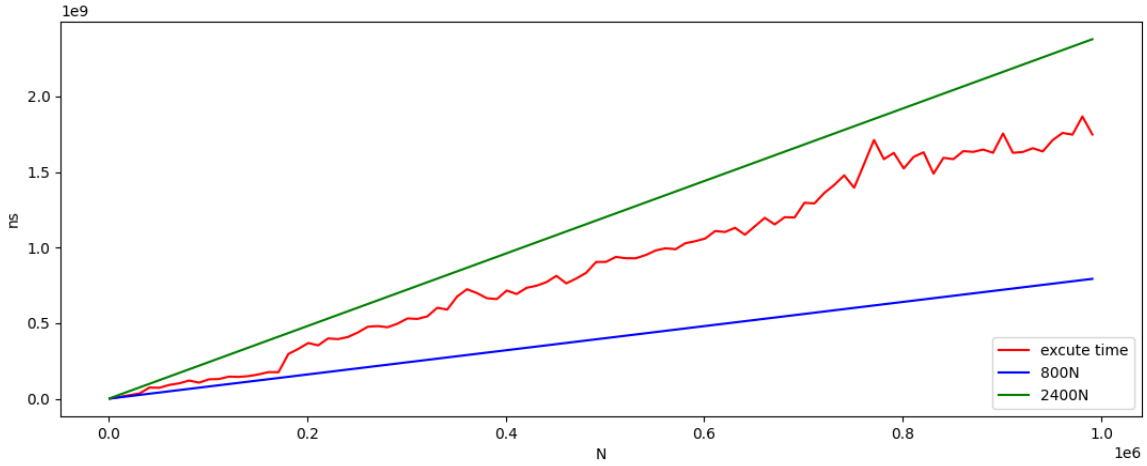
複雜度分析

Time complexity $\theta(N\log(K))$

Space complexity $\theta(N+K)$

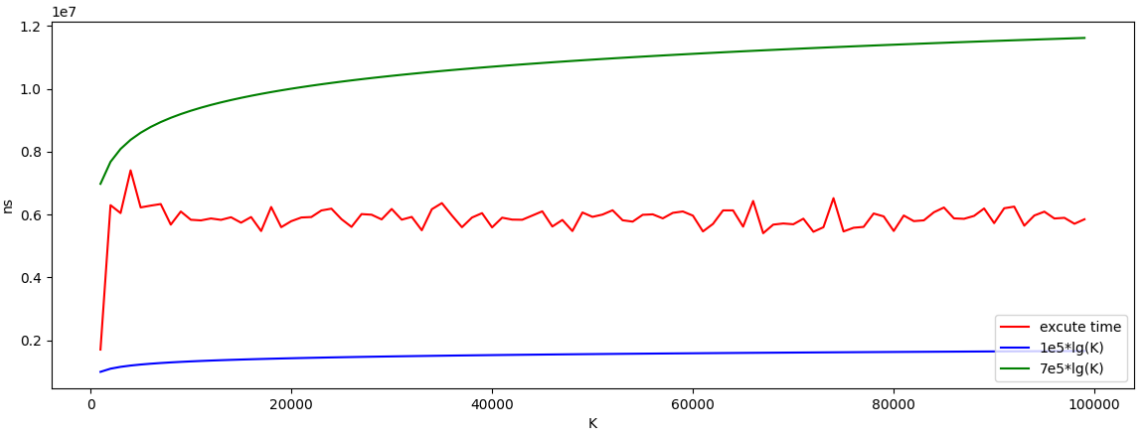
時間實測

K 固定 100, N in Range (1000, 1001000)



我們可以發現固定 K 時，我們可以用 800N 及 2400N 夾住執行時間，故 time complexity 裡的 N 應該是 N^1 沒錯

N 固定 100000, K in Range(1000, 100000)



我們可以發現固定 N 時，我們可以用 1e5*lg(K) 及 7e5*lg(K) 夾住執行時間，故 time complexity 裡的 K 應該是 lg(K) 沒錯