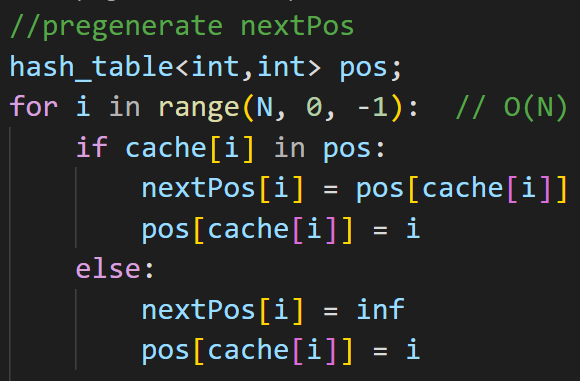
**演算法程式作業三**

**110502529陳柏燊**

**Pseudo code:**

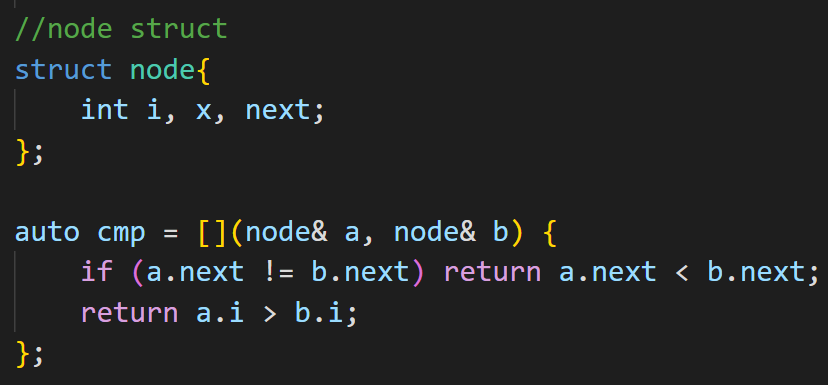
先從後往前跑，把每個cache下一次出現的位置找出來存在nextPos array,利用hash\_table 紀錄位置



Time complexity

Space complexity

等等有用到自訂結構,先說明結構



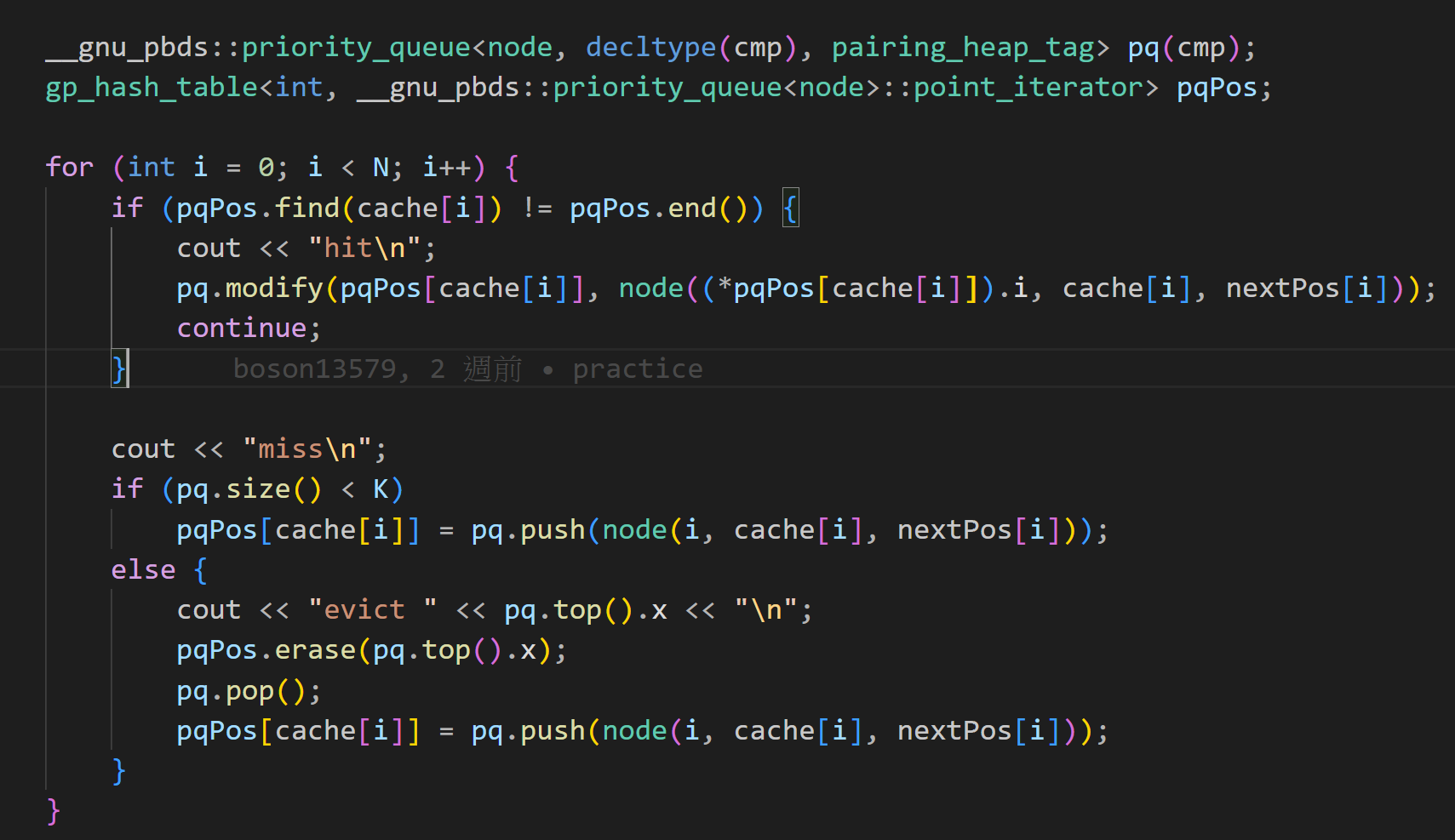
等等這個node會是priority\_queue裡的node, i代表進入cache的index, x代表cache, next 代表這個 cache 下一次出現的index  
 

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

Space complexity //Q是k 其他是N

程式實作方式



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

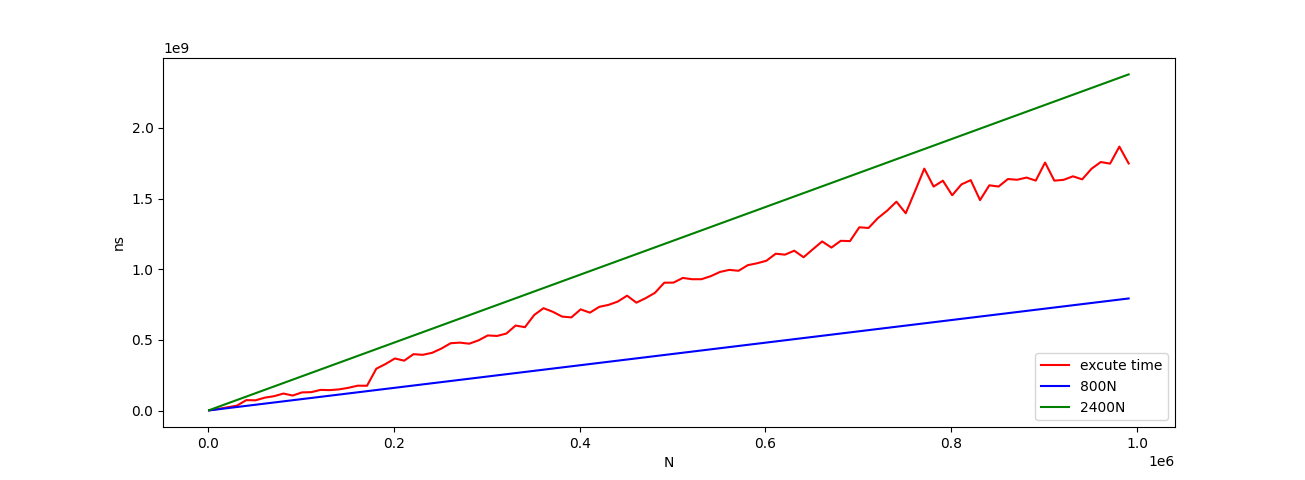
**複雜度分析**

Time complexity

Space complexity

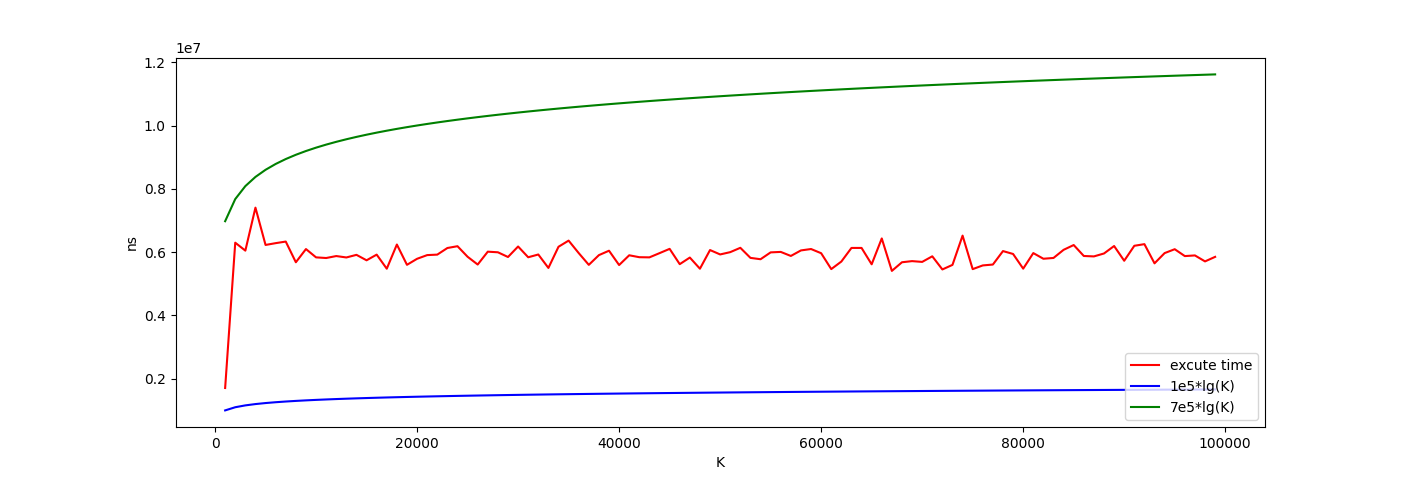
**時間實測**

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