**資料結構 作業一**

**資科二 簡傑**

1. **新增資料**

* 註 (一) : 數據的時間單位為毫秒(ms)
* 註 (三) : Sorted Array在k = 21時超時，以前數項求得公比後估計k = 22 ~ 30。
* 註 (三) : 本次試驗之BTree在 k = 28以後無法生成，以前數項求得公比後估計k = 29、30；Binary Search Tree在k = 27以後無法生成，同樣以前數項求得公比後估計k = 28、29、30 ; HashTable 在k = 27以後無法生成，以前向乘以2的倍率估計k = 28、29、30。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Insert (毫秒) | | | | | Search (毫秒) | | | | |
|  | Array | BST | Btree | Hash |  | Array | BST | Btree | Hash |
| 10 | 1.162 | 0.494 | 0.332 | 0.413 | 10 | 34.126 | 35.067 | 63.57 | 18.883 |
| 11 | 3.923 | 1.018 | 0.685 | 0.749 | 11 | 35.817 | 36.636 | 66.619 | 18.379 |
| 12 | 14.123 | 2.083 | 1.436 | 1.521 | 12 | 36.992 | 37.888 | 71.291 | 18.719 |
| 13 | 53.738 | 4.299 | 3.022 | 2.99 | 13 | 39.083 | 39.611 | 76.723 | 18.395 |
| 14 | 213.04 | 8.995 | 6.403 | 6.015 | 14 | 40.643 | 41.985 | 83.617 | 18.677 |
| 15 | 818.964 | 18.91 | 13.399 | 12.113 | 15 | 42.025 | 45.266 | 96.149 | 18.537 |
| 16 | 3240.1 | 39.305 | 28.653 | 24.116 | 16 | 43.89 | 48.651 | 117.063 | 18.629 |
| 17 | 13028.6 | 83.618 | 60.981 | 48.634 | 17 | 46.313 | 49.615 | 151.763 | 18.801 |
| 18 | 52450.5 | 174.213 | 128.269 | 97.249 | 18 | 48.574 | 55.571 | 226.1 | 18.399 |
| 19 | 250819 | 408.477 | 271.986 | 217.336 | 19 | 50.319 | 72.254 | 388.726 | 18.704 |
| 20 | 1.04E+06 | 1078.38 | 681.066 | 531.118 | 20 | 54.486 | 92.079 | 831.718 | 18.483 |
| 21 | 4.15E+06 | 2391.7 | 1700.38 | 1248.94 | 21 | 55.476 | 112.276 | 1879.23 | 18.903 |
| 22 |  | 5502.58 | 4214.48 | 2692.06 | 22 | 54.857 | 129.192 | 4417.2 | 18.41 |
| 23 |  | 12101.2 | 8875.88 | 5580.49 | 23 | 54.28 | 139.17 | 9096.46 | 18.773 |
| 24 |  | 30764.8 | 20977 | 11433.8 | 24 | 54.532 | 170.042 | 21226.1 | 18.59 |
| 25 |  | 67989.7 | 44795.8 | 23298 | 25 | 54.292 | 190.104 | 45068.4 | 18.549 |
| 26 |  | 151095 | 101479 | 48376.9 | 26 | 54.669 | 214.523 | 101783 | 18.919 |
| 27 |  | 350836 | 287554 | 103383 | 27 | 54.211 | 251.678 | 287658 | 18.629 |
| 28 |  | 771839.2 | 584636 | 206766 | 28 | 54.211 | 260.9994 | 899410 | 18.5 |
| 29 |  | 1698046 | 1286199 | 413532 | 29 | 54.211 | 270.3208 | 2698230 | 18.5 |
| 30 |  | 3735702 | 2829638 | 827064 | 30 | 54.211 | 279.6422 | 8094690 | 18.5 |

1. **搜尋資料**

* 註 (一) : 數據的時間單位為毫秒(ms)
* 註 (二) : Sorted Array搜索時使用二分搜尋法。
* 註 (三) : Binary Search Tree在k = 27以後，以 之公比推估k = 28、29、30 ; BTree在k = 28以後，以前數項求得公比後估計k = 28、29、30 ; HashTable 在k = 27以後無法生成，觀察前數項後估計k = 28、29、30。
* 註 (四) : 由於Sorted Array、Binary Search Tree和Hash Table重合度高，故附上 20和之圖表

1. **實驗環境**
   1. 系統:

Distributor ID: Ubuntu

Description: Ubuntu 20.04 LTS

Release: 20.04

Codename: focal

* 1. 主機規格:

Cpu model name : AMD Ryzen 3 3300X 4-Core Processor

cpu MHz : 3999.844

cache size : 512 KB

MemTotal : 32862476 kB

MemFree : 21720228 kB

MemAvailable : 30287388 kB

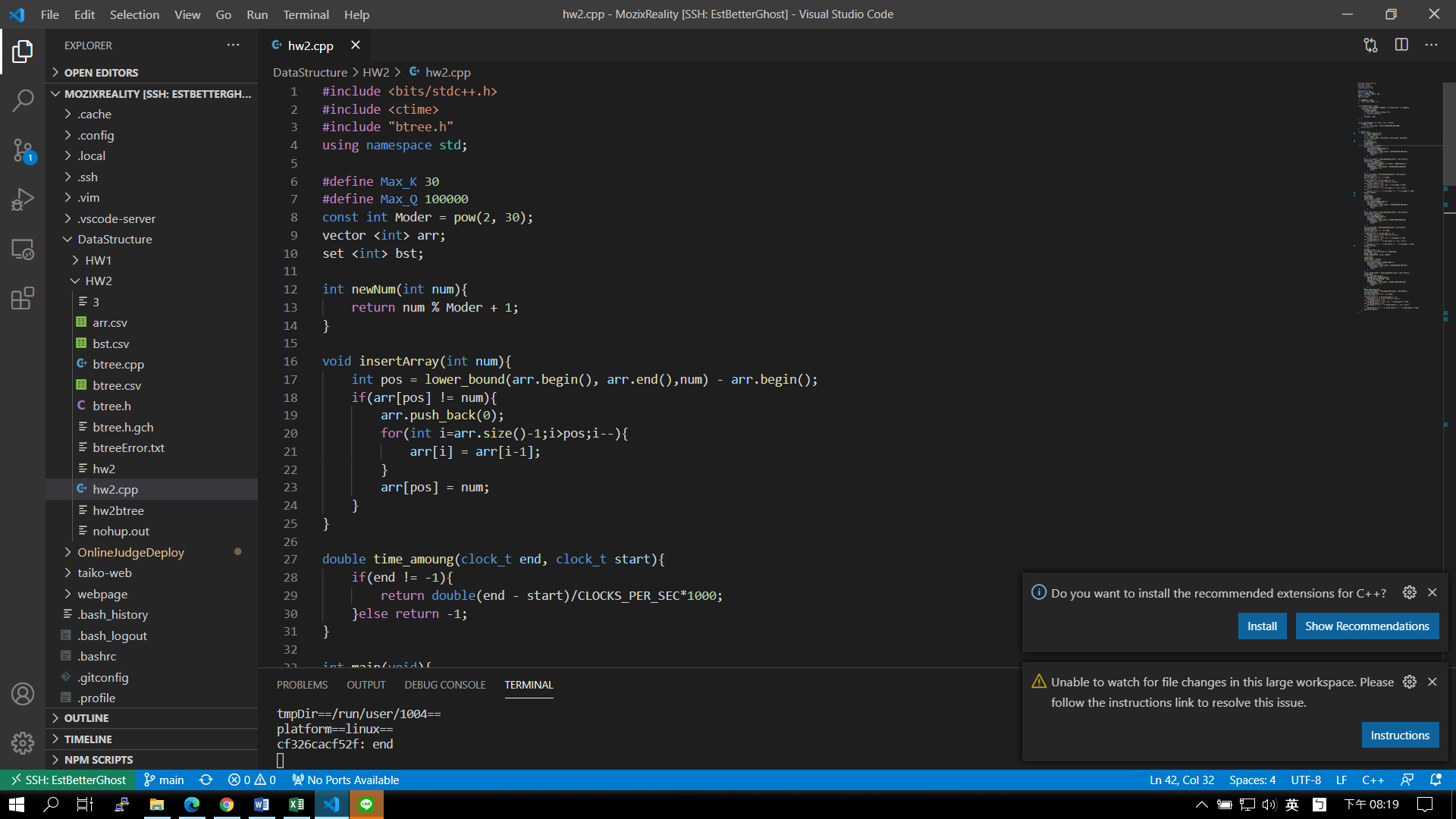
Buffers : 3809784 kB

Cached : 4799232 kB

1. **實驗程式碼**
   1. Btree程式碼來源

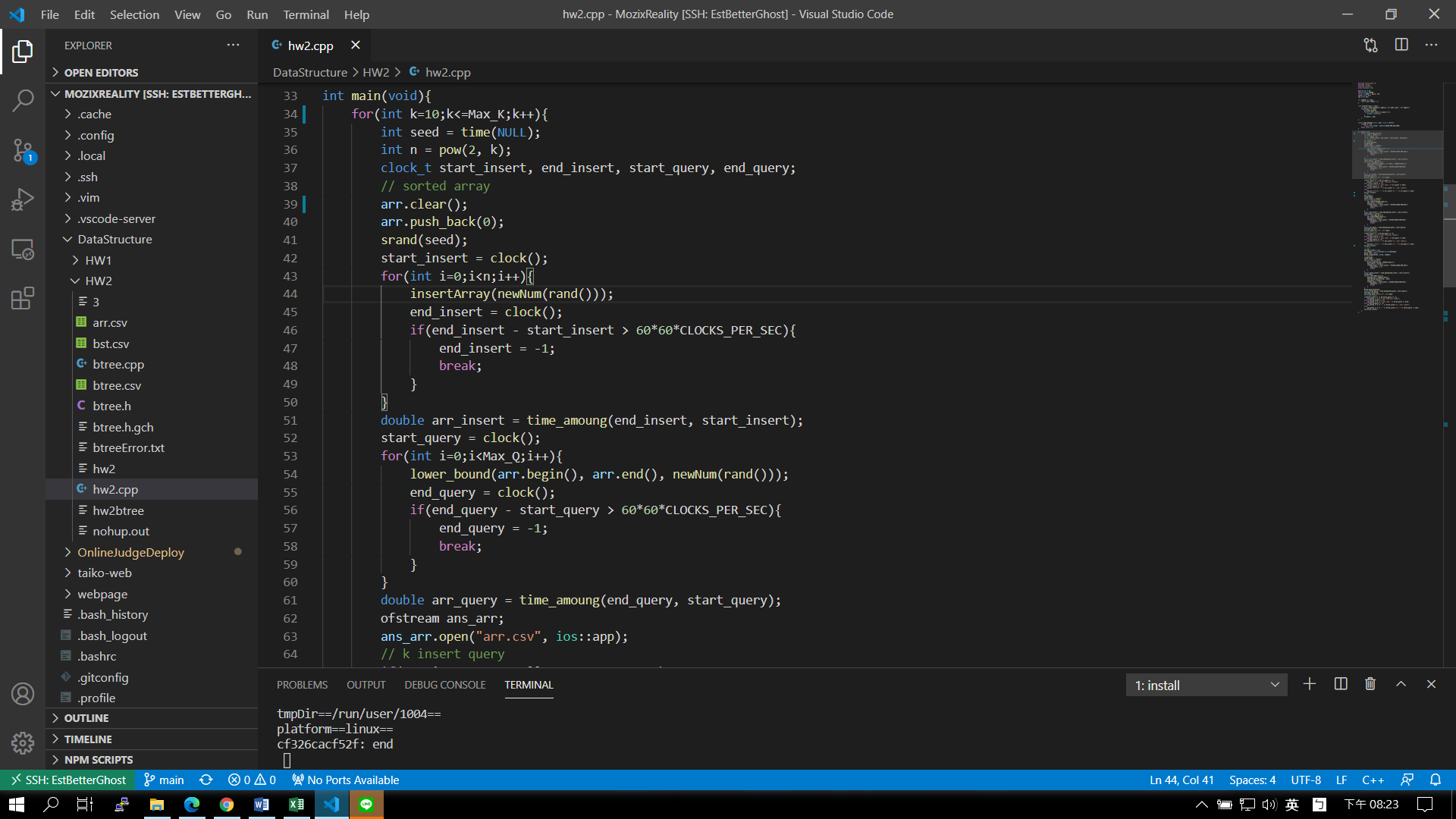
https://www.cnblogs.com/cthon/p/9281664.html

* 1. Function



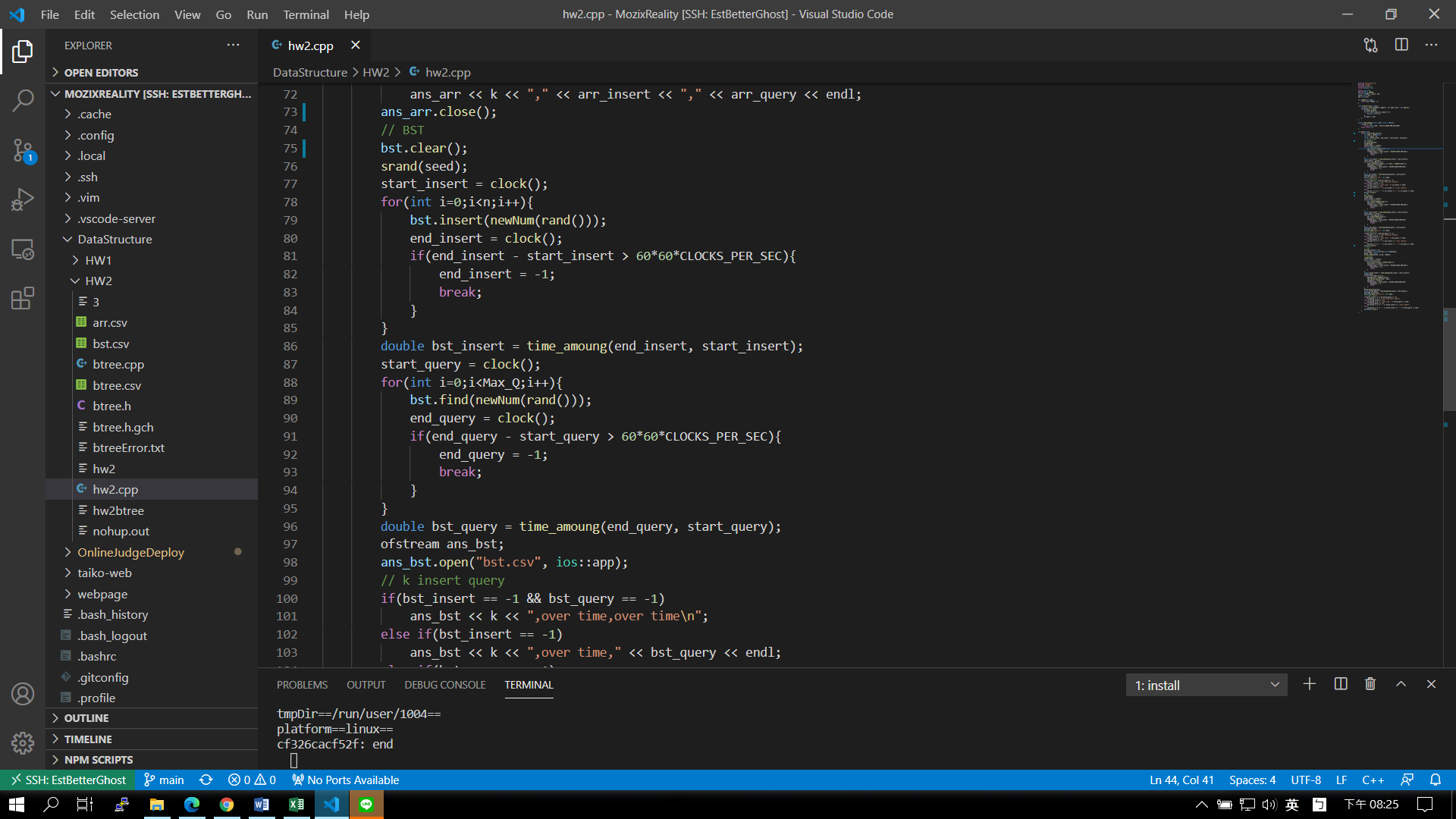
說明:

* + - newNum() : 隨機產生1 ~ 2^31 的整數
    - insertArray() : 插入一整數至array，由二分搜找到其正確位置後插入，時間複雜度為O(N)
    - time\_amoung() : 判定程式經過時間以及是否超時。
  1. Main
     + Sorted Array



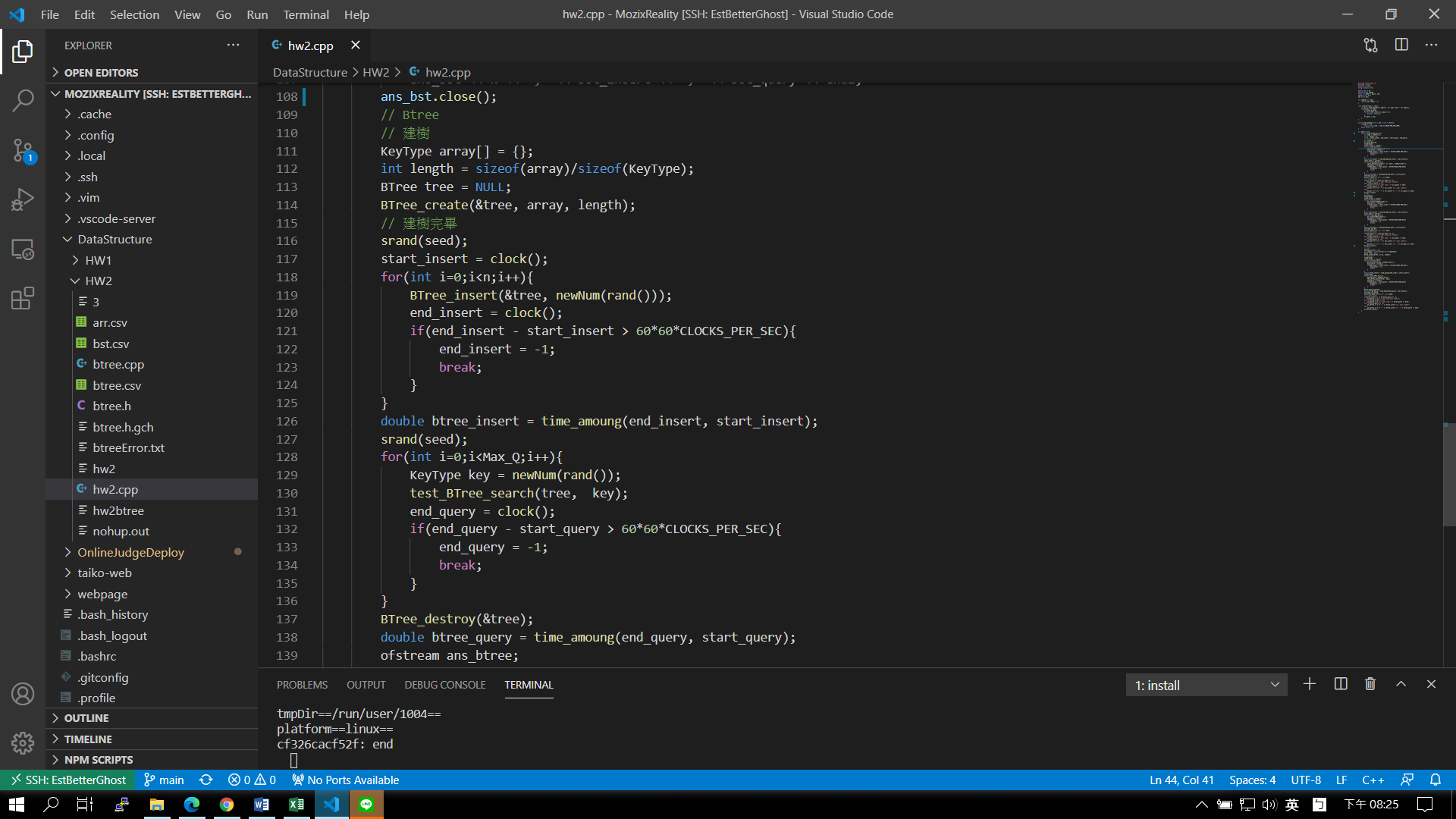
說明 : 對於每個k清空array後，由insertArray()作新增(O(n))，搜尋時使用二分搜尋(O(log n))

* + - Binary Search Tree



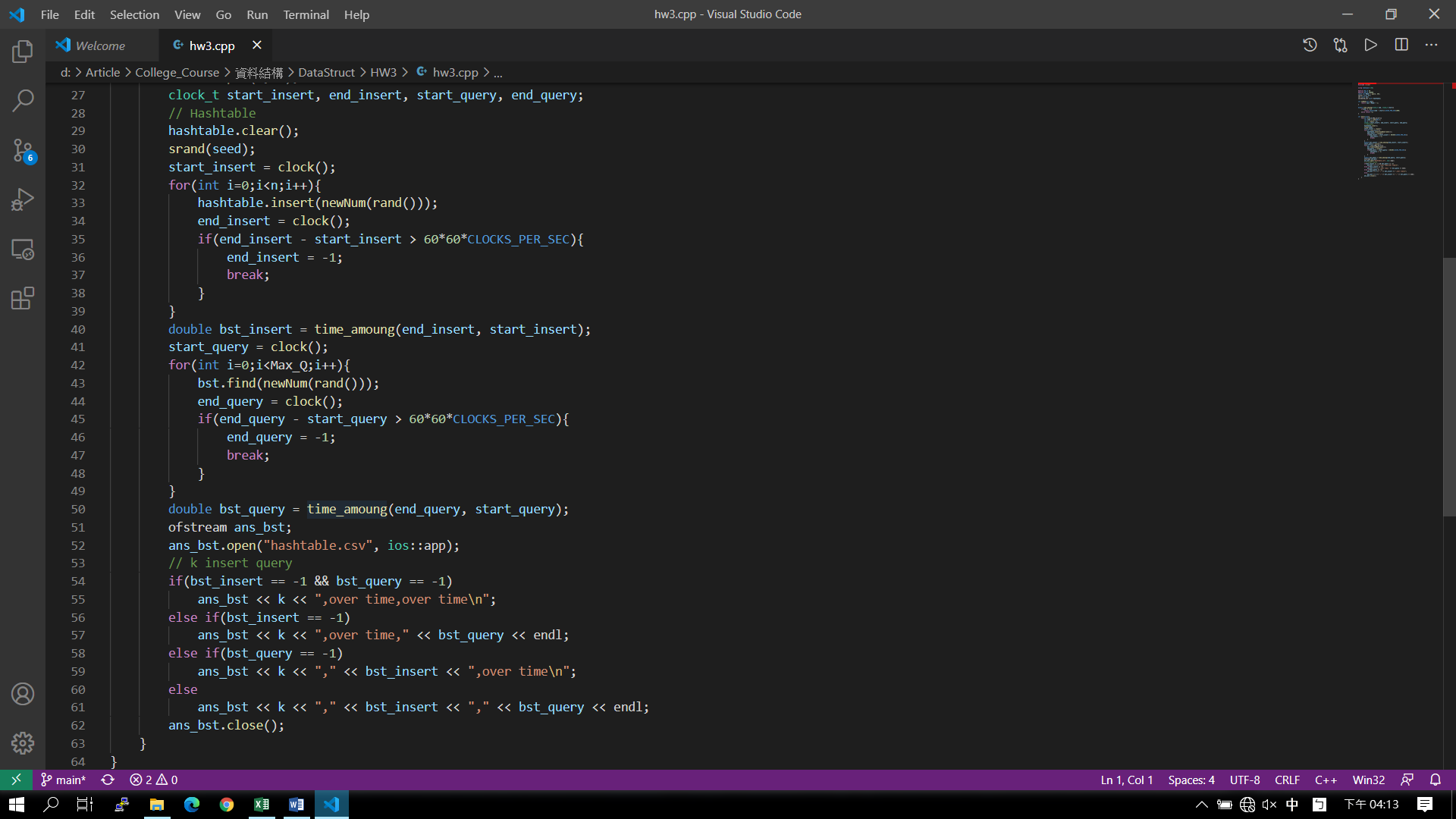
說明 : 使用c++內建BST set實作，由insert新增元素(O(log n))，搜尋時使用find(O(log n))

* + - Btree



說明 : 以BTree\_create興建BTree，由BTree\_insert新增元素(O(log n))，搜尋時使用test\_BTree\_search(O(log n))

* + - Hash Table



說明 : 以std::unordered\_map興建Hash Table，由Hash Table新增元素(O(1))，搜尋時使用test\_BTree\_search(O(log n))

1. 心得、疑問、與遇到個困難

本次試驗令我感到奇怪的是Btree，雖然在測試時順利的跑道K=28，然而其搜尋時的時間並不規則，因此在作估計時，是以前數項求得公比後才估計。

其次是本以為能夠順利完成的Binary Search Tree，竟然在k=27時便陣亡了，由於伺服器限制記憶體大小的關係，超過一定上限的程序會被砍掉，因此沒有辦法順利跑到k=30實在是萬分可惜。

不過本次實驗相較於上次作業，我使用相較熟悉的C++語言，在操作以及速度上，都有了相當大的提升，或許也是因為第一次作業的關係，更熟習作業流程，伺服器的操作以及遇到錯誤狀況時的處理，以下附上Github連結，方便檢視整個實驗程式檔及錯誤資料。