Programming Design Homework 02 資工2B 408410120 鍾博丞

環境配置

Operating System: Ubuntu 20.04 LTS using KDE plasma

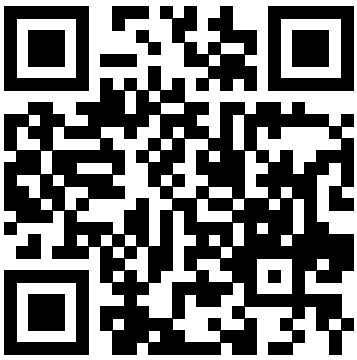
CPU: AMD R9 3900X 12C 24T @ 3.8GHz

RAM: 32GB DDR4 3600MHz (Double channel)

SSD: WD Black 256G WDS256G1X0C TLC (Seq. R: 2050MB/s, Seq. W: 700MB/s, Random R: 170K IOPS, Random W: 130K IOPS)

Github Link

https://reurl.cc/AgVqNE



A. 操作方法

DS_perf 每次都會生成給定大小的 int 資料，故執行此程式時，建議 RAM 有 8GB 或以上，否則很容易使 swap space 被填滿，造成 CPU 使用大量的時間做 I/O，甚至會直接被 OS kill 掉

DS_perf 最少需接收 6 個參數

```
./DS_perf -d <N> -q <M> [-arr] [-ll] [-bst] [-bs] [-hash]
```

產生 N 筆唯一的資料，注意大小不要太過於接近 INT_MAX = 2,147,483,647，否則產生效率會非常低，下面將詳細說明為何會如此

產生 M 筆資料進行查詢

N 與 M 支援科學記號，即 10^6 可寫作 1e6, 1E6, 1e+6, 1E+6 四種寫法，其他數字以此類推

至少選擇一種方法

B. 亂數資料生成方法

已知 rand() 產生的值介於 0 ~ INT_MAX，故產生一個大小為 INT_MAX 的 bool array，初始值設定為 false，之後每次產生亂數時檢查該數有沒有出現過，若沒有，則在該亂數的 index 位置設為 true，若有，則重新產生，直到沒有碰撞為止

故而，產生越多筆資料，碰撞機率會越大，建議 N 設定為 1e6 即可

其實，還有一種方法，就是先產生連續的整數，然後像洗牌一樣，隨機挑選兩個不同的 index 做交換，只是這樣做的話，很難保證 Binary Search Tree 的平衡，所以在這裡不採用此方法

C. 演算法實作及複雜度

1. Linear Search (-arr)

直接依序將產生的亂數填入陣列，每次搜尋都需走訪陣列中的所有元素

Time complexity: $O(n)$

2. Linked list (-ll)

直接依序將產生的亂數填入每一個 node，然後插入在頭端，所以填入完成後會是反序的，每次搜尋都需走訪 linked list 中的所有元素

與陣列相比，因為資料不連續，每次走訪都需查詢 next 在哪裡，故 linked list search 會比 array linear search 來的慢

Time complexity: $O(n)$

3. Binary Search (-bs)

將產生的亂數填入陣列後，做 qsort

Time complexity: $O(\lg n)$

4. Binary Search Tree (-bst)

直接依序將產生的亂數填入每一個 node，然後依照 BST 的規則，插入在左邊或右邊

雖然 in-order traversal 就等同於已經排序完成，且假設其平衡狀態接近 AVL，但是與 array 中的 binary search 比較，多出了查詢下一個節點的時間，故 BST search 會比 array binary search 來的慢

Time complexity: $O(\lg n)$

5. Hash (-hash)

將產生的亂數以下列非線性函數做轉換，並開一個比資料量兩倍大的陣列，假設為 2N

使用 FNV-1a Algorithm 實作

```
hash = FNV_offset_basis
for each octetOfData to be hashed
    hash = hash xor octetOfData
    hash = hash * FNV_prime
return hash
```

Where the constants `FNV_offset_basis` and `FNV_prime` depend on the return hash size you want:

```
Hash Size
==========
32-bit
    prime: 2^24 + 2^8 + 0x93 = 16777619
    offset: 2166136261
64-bit
    prime: 2^40 + 2^8 + 0xb3 = 1099511628211
    offset: 14695981039346656037
128-bit
    prime: 2^88 + 2^8 + 0x3b = 309485009821345068724781371
    offset: 144066263297769815596495629667062367629
256-bit
    prime: 2^168 + 2^8 + 0x63 =
374144419156711147060143317175368453031918731002211
    offset:
1000292579580525809070709686206257048370927960142411939452252845017414719255557
```

產生出該 key 的 address，然後將 key 填入 Node 裡，並將該 address 的內容指向該 Node，若未來發生 collison，則以 linked list 接在其後面

Time complexity: $O(1)$

D. 測量結果

./DS_perf -d 1e6 -q 1e5 -arr -ll -bs -bst -hash

(1) Linear Search

```
❯ time strace -c ./DS_perf -d 1e6 -q 1e5 -arr
Time for random integer generator: 225663713ns = 0.23s

arr:
Time for insert key to array: 95069764ns = 0.10s
We have found 45 keys.
Time for linear search to array: 182001055156ns = 182.00s

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 93.24    0.039091       13030         3           munmap
  5.46    0.002288           0      2565           write
  1.16    0.000486           0      2561           read
  0.12    0.000050          12         4           openat
  0.01    0.000005           1         5           fstat
  0.01    0.000004           0         9           mmap
  0.00    0.000000           0         4           close
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         6           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.041924                  5168         2 total
strace -c ./DS_perf -d 1e6 -q 1e5 -arr  182.28s user 0.40s system 99% cpu 3:02
.71 total
```

(2) Linked list

```
❯ time strace -c ./DS_perf -d 1e6 -q 1e5 -ll
Time for random integer generator: 256212372ns = 0.26s

ll:
Time for insert key to linked list: 175001998ns = 0.18s
We have found 39 keys.
Time for search in linked list: 286279595788ns = 286.28s

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 94.35    0.054077       27038         2           munmap
  2.30    0.001320           0      2565           write
  2.01    0.001152           0      2561           read
  1.14    0.000656         164         4           openat
  0.18    0.000101           0       239           brk
  0.01    0.000007           1         5           fstat
  0.00    0.000000           0         4           close
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         6           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.057313                  5402         2 total
strace -c ./DS_perf -d 1e6 -q 1e5 -ll  286.56s user 0.42s system 99% cpu 4:47.
12 total
```

(3) Binary Search

```
❯ time strace -c ./DS_perf -d 1e6 -q 1e5 -bs
Time for random integer generator: 217953653ns = 0.22s

bs:
Time for insert key to array: 86674555ns = 0.09s
Time for qsort: 91635735ns = 0.09s
We have found 58 keys.
Time for binary search to array: 20885939ns = 0.02s

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 94.57    0.044696       11174         4           munmap
  2.03    0.000960           0      2566           write
  1.96    0.000928           0      2561           read
  1.41    0.000668         167         4           openat
  0.01    0.000006           0        10           mmap
  0.01    0.000005           1         5           fstat
  0.00    0.000000           0         4           close
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         3           brk
  0.00    0.000000           0         6           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         1           sysinfo
  0.00    0.000000           0         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.047263                  5172         2 total
strace -c ./DS_perf -d 1e6 -q 1e5 -bs  0.43s user 0.38s system 100% cpu 0.813
total
```

(4) Binary Search Tree

```
❯ time strace -c ./DS_perf -d 1e6 -q 1e5 -bst
Time for random integer generator: 211354027ns = 0.21s

bst:
Time for insert key to BST: 591797900ns = 0.59s
We have found 55 keys.
Time for search in BST: 50598831ns = 0.05s

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 93.45    0.046242       23121         2           munmap
  2.76    0.001364           0      2565           write
  1.75    0.000866         216         4           openat
  1.60    0.000791         197         4           close
  0.40    0.000198           0      2561           read
  0.02    0.000012           0       239           brk
  0.02    0.000009           1         5           fstat
  0.00    0.000000           0         8           mmap
  0.00    0.000000           0         4           mprotect
  0.00    0.000000           0         6           pread64
  0.00    0.000000           0         1         1 access
  0.00    0.000000           0         1           execve
  0.00    0.000000           0         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.049482                  5402         2 total
strace -c ./DS_perf -d 1e6 -q 1e5 -bst  0.93s user 0.40s system 100% cpu 1.321
 total
```

(5) Hash

```
❯ time strace -c ./DS_perf -d 1e6 -q 1e5 -hash
Time for random integer generator: 202325264ns = 0.20s

hash:
Hash collision: 212771
Time for insert key to hashtable: 221688104ns = 0.22s
We have found 48 keys.
Time for searching keys to hashtable: 7292147ns = 0.01s

% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 94.62    0.042286       14095         3           munmap
  2.42    0.001080           0      2566           write
  1.36    0.000608         152         4           openat
  0.76    0.000339           0      2561           read
  0.47    0.000212         212         1           execve
  0.11    0.000048           5         9           mmap
  0.09    0.000042           0       239           brk
  0.06    0.000028           7         4           mprotect
  0.04    0.000017           2         6           pread64
  0.03    0.000012           2         5           fstat
  0.01    0.000006           1         4           close
  0.01    0.000006           6         1         1 access
  0.01    0.000005           2         2         1 arch_prctl
------ ----------- ----------- --------- --------- ----------------
100.00    0.044689                  5405         2 total
strace -c ./DS_perf -d 1e6 -q 1e5 -hash  0.54s user 0.36s system 100% cpu 0.90
5 total
```

最後使用 valgrind 檢查一下 memory leak

```
valgrind --leak-check=full --show-leak-kinds=all --verbose ./DS_perf -d 1e6 -q 1
-arr -ll -bs -bst -hash
```

```
==66578==
==66578== HEAP SUMMARY:
==66578==     in use at exit: 0 bytes in 0 blocks
==66578==   total heap usage: 3,000,018 allocs, 3,000,018 frees, 2,231,512,079
 bytes allocated
==66578==
==66578== All heap blocks were freed -- no leaks are possible
==66578==
==66578== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

E. 結論

使用 integer 做 hash 其實是有點違反 hash 的實做目的的，既然已經是 integer 了，何不直接將 integer 當作 address 直接插入資料呢？

只是，我們還是要實作 hash function，這對於學習比較有幫助

就結果而論，因為資料是使用 integer，所以使得所有比較方法都不需要做 strcmp，大幅發揮 qsort 的能力，所以 binary search 最快，接著是 hash，相信如果是字串處理的話，hash 會是最快的

接著是 binary search tree，因為他每次要插入新的值時，都要先產生 new node，從 strace 中也可看出，brk 被呼叫的次數與 linked list 差不多，遠高於其他的比較方法，也因此，雖然 linked list traversal 和 array linear search 時間複雜度是一樣的，但是 linear search 還是比較快

F. Reference

FNV-1a Algorithm: https://reurl.cc/NX5Nvp

4102150_02 System Programming, instructor: Prof. Shi-Wu Lo