HW4　資工二　陳盈如　B05902118

[Report]

1. 在 main 裡面 pthread_create( ) 產生所需的thread，thread主要是負責建樹以及將 testing_data 在 traverse 整棵樹之後決定每個 testing_data 是好人還是壞人。我在 hw4.c 檔裡面的 thread number = 2, tree number = 2

```
while (tree_made < TREE_NUM) {
    for (int i = 0; i < THREAD; i++) {
        if (tree_made >= TREE_NUM) break;
        if (forest[tree_made] == NULL) {
            forest[tree_made] = (TREE *)malloc(sizeof(TREE));
            forest[tree_made]->left = forest[tree_made]->right = NULL;
        }
        pthread_create(&tid[i], NULL, (void *)make_tree, (void *)forest[tree_made]);
        tree_made++;
    }
    for (int i = 0; i < THREAD; i++) {
        pthread_join(tid[i], NULL);
    }
}
```

```
while (tree_check < TREE_NUM) {
    for (int i = 0; i < THREAD; i++) {
        if (tree_check >= TREE_NUM) break;
        pthread_create(&tid[i], NULL, (void *)testing, (void *)forest[tree_check]);
    }
    for (int i = 0; i < THREAD; i++) {
        pthread_join(tid[i], NULL);
        tree_check++;
    }
}
```

2.

thread number	time
10	1m52.419s
8	2m18.726s
6	2m35.330s
4	3m1.823s
2	3m30.549s

此表格可知道 thread 的數量越多，所需的時間越少，但是受限於系統本身核心能夠執行的量，thread 數量會有 upper bound，當達到某個特定的數量後即便開更多的 thread 也沒有辦法提升速度。

3.

thread number	instructions
10	12,098,448,420
8	9,633,944,595
6	12,161,169,162
4	10,482,765,401

2	11,697,384,393

由表格可以很明顯看出 thread 的數量和 instruction 沒有明顯的線性關係

4.

tree number	instructions
10	12,028,795,566
8	10,446,461876
6	7,749,525,658
4	6,002,907,976
2	4,395,250,557

由表格可以知道 tree 的數量越多，instruction 也會越多

5.

thread 的數量與正確性並無相關，thread 是用來減少所需的時間；而 tree number 當開到一定的數量之後，正確性提升的幅度會越來越小，然而建樹卻需要消耗較多的時間。權衡之下，便不會選擇建太大的樹，而是找一棵不大又能夠使正確性維持在一定的水平。