HW4 report

B05902050
黃子源

1. 我把 thread 開在 maketree 的部分，每種一棵樹就 create 一個 thread，然後一等到某一個 threadid 有空就再分配一棵樹給他（pthread_tryjoin_np），直到所有樹都種完。所以假如樹種很多顆，就可以用 thread 同時 make 很多樹不用一顆一顆建，降低執行時間。

2. thread v.s time :

為了突顯 thread 數跟時間的關係，我建樹每次取出的 data 量為 1000，樹的數量設 300 :

Thread 數	1	2	4	16	32
Real time	21m57.186s	11m54.044s	7m13.868s	4m42.951s	4m15.782s

我們發現 thread 的數量跟時間呈現明顯的正相關，但隨著 thread 數量增大而越來越不明顯，可能是因為理想狀況下假設 make 一顆 tree 要花時間 t，要花的總種樹時間就是：tree/thread*t，所以總時間遞減的幅度的確會隨這 thread 增加而減少，直得注意的是因為 make 一顆 tree 的時間不會變，因此當 tree 的數量很少時，create thread 並不能有效降低時間。

3. thread v.s. instructions :

我設樹每次取出的 data 量為 100，樹的量設為 500 :

Thread	1	4	16	32
Instructions	87,699,094,734	55,555,701,361	47,701,713,968	44,245,106,477

我們發現 instruction 數量隨著 thread 增加而減少，但減少幅度越來越少。我其實不太確定確切的原因，但我猜是因為 thread 可以同時多線程去跑這個 process，因此可以有效降低工作量，而工作量又跟 instruction 成正比，因此才會跟 thread 的量呈反比。

4. tree num v.s. instructions :

我設每次取出的 data 量為 100，thread 數為 8 :

Tree num	100	200	500	1000
instructions	11,121,243,705	21,235,961,799	49,259,286,810	95,658,501,097

我們發現 instruction 數量跟 tree 的數量幾乎成正比，tree num 100 跟 tree num 200 的 instruction 數量差差不多剛好兩倍，tree num 500 跟 1000 的也是如此。原因我想是因為 instruction 數量跟工作量成正比，而在隨機取的 data 數量和 thread 不變的情況下，instructions 幾乎只和 tree num 相關，因此兩者才會成高相關。

5. 其他發現！

在這部分我將探討 tree 數量跟正確率的比較，在這部分我們設每次取出的 data 量為 100，thread 數為 8，並寫了一個能測 ans.csv 跟 submission.csv 一樣比率的 code，因為正確率每次執行都不太相同，因此我會執行五次並取平均：

Tree num	1	1	1	1	1	average
正確率	79.04%	86.58%	81.06%	71.27%	72.68%	78.126%

Tree num	5	5	5	5	5	average
正確率	88.89%	88.56%	86.03%	86.68%	89.48%	87.93%

Tree num	100	100	100	100	100	average
正確率	90.06%	89.74%	89.06%	88.82%	89.66%	89.27%

Tree num	500	500	500	500	500	average
正確率	89.45%	89.41%	89.52%	89.18%	89.50%	89.41%

我們發現正確率的確跟 tree 成正比，但並不是說 tree 越大準確率越高，可以看出 treenum 在 100 跟 500 時正確率並沒有太大的差異，但卻可以明顯看出，相較於 tree num 只有個位數的狀況，tree num 很大時的 output 相對穩定很多。對此狀況我們可以解釋說因為建樹時我們是隨機取 data，因此難免會有狀況比較極端的樹拉低準確率（如上表 thread 為 1 時正確率為 71.27%的 case），但當建的樹越來越多時，最後每一棵樹在投票時所佔的權重比就會比較低，可以降低極端意見的狀況，最終的正確率也會趨近於此演算法的正確率，也就是 89%左右。