実践プログラミング

3J23　椎名 泰之

# 目的

通常の線形探索、番兵版線形探索、ループによる二分探索、再起による二分探索の4つを実行し、処理時間について比較する。

# 実験方法

quickソートの処理前にtime.hによるclock関数で開始時間を取得する。

その処理が終了して、printfの処理前にclock関数で終了時間を取得し、差を求めることにより処理時間を計測する。

# 実行環境

CPU : Intel Core i5-3570 3.40GHz

RAM : 8.00 GB

OS : Windows 10 Enterprise 2015 LTSB

# 実行結果

|  |  |  |  |
| --- | --- | --- | --- |
| 回数 | 1 | 2 | 3 |
| 探索キー数 | 1000 | 1000 | 1000 |
| 線形探索（通常）:μs | 4808 | 4449 | 2511 |
| 線形探索（番兵）:μs | 3892 | 4056 | 1809 |
| 二分探索（ループ）:μs | 309 | 515 | 172 |
| 二分探索（再帰）:μs | 391 | 710 | 162 |

　以上より以下のことが挙げられる。

* 線形探索と二分探索では、二分探索のほうが速い。
* 線形探索の通常版と番兵版では、番兵版のほうが僅かに速い。
* 二分探索のloop版と再帰版では、大した違いがない。

# 考察

　二分探索は、ソート作業と探索作業の二つに分けることができるため、高速化ができたものと思われる。それだけでなく、比較の回数もそれに従って大きく減少するため、さらに処理時間が減ったものと思われる。大量のデータを扱うときは、このソートでも違いが出るため、二分探索を用いるほうが好ましいと考えた。更に、keyが末尾にあればあるほど、比較的高速になった。

　線形版は通常と番兵で差が大きく出た。途中でkeyが見つかった時にループから抜けた後の速さが違うからであるとわかった。If比較も少ないことがあげられる。

　再帰版はあまり差が出なかったが、関数呼出し後の1回の処理が増えるだけなのだと思う。しかし、再帰版では配列をメモリに置いておかなくてはならないため、レジスタの狭いマイコンなどでは、loop版を使用するとよいと考えた。