実践プログラミングⅠ　レポート（Pr1-2）

＊＊＊＊＊＊

1. 課題内容

Pr1-1で作成したプログラムにおいて

1．通常の線形探索

2．番兵利用版線形探索

3．ループによる二分探索

4．再帰処理による二分探索

それぞれを実行して処理時間について比較し、考察せよ。

1. 実験方法

Pr1-1で作成したプログラムを以下のように改良した。

1. 処理時間をプログラム内で計測する。

#include<time.h>、clock(); 等を使用する。

探索前に現在の時刻を保存し、探索後にもう一度時刻を保存する。この時刻の差を取ることによって、処理時間を計測した。この時、二分探索についてはクイックソートに係る時間も処理時間として含めている。

1. 計測する動作を10回繰り返し、平均処理時間を算出する。

コンピュータでの他のリソースの活用状況によっても変化するため、平均処理時間を算出する。

1. 実験結果

各探索方法と処理時間について、表1に示す。

（レコード数N=30000、探索するキーは1000個（N/2-500〜N/2+499）としている。）

表1：探索方法と処理時間

1. 考察
2. 線形探索と二分探索

表1に示した実験結果より、線形探索と二分探索を比較すると、二分探索は線形探索に比べ、圧倒的に処理時間が早いということが分かる。

線形探索のアルゴリズムは、キーが見つかるまで一つずつデータを比較するため、最大ですべてのデータを比較する。これに対して二分探索のアルゴリズムは、すべてのデータと比較する必要はない。また、二分探索の前に処理するソートについては、処理能力の高いクイックソートを用いているため、処理時間に大きく影響していない。

よって、比較回数の少ない二分探索は処理するデータ数が少ないため、探索の処理時間が早くなると考える。

1. 通常の線形探索と番兵利用版線形探索

　表1に示した実験結果より、番兵利用版線形探索のほうが、通常の線形探索より処理速度は0.0019秒早いことが分かる。まず、探索キーの数が少ないため、差はかなり小さくなっていると考える。

　番兵利用版線形探索の処理速度のほうが早いことについて、ループ1回の間に行う処理数の違いに着目する。

1. 通常版の場合の処理の数（forループ1回）

if ( target == data[i] ) //処理1：比較

i++ //処理2：変数の数字を1増やす

for ( i = 0; i < N; i++ ) //処理3：下線部、比較

1. 番兵利用版の場合（whileループ1回）

if(data[count]==target) //処理1：比較

count++; //処理2：比較

上記より、番兵利用版線形探索は通常版線形探索よりループ1回の間に行う処理回数が一つ少ないことが分かる。処理一つの問題ではあるが、レコード数・探索キーを増やした場合、処理時間の差は大きくなる。よって、番兵利用版線形探索のほうが処理速度は期待できるのではないか。

1. ループによる二分探索と再帰処理による二分探索

表1より、ループを用いた二分探索のほうが再帰処理を用いた二分探索と比べて、処理時間が短いことが分かる。

これについて、再帰処理のメモリの消費・関数呼び出しにかかる処理が関係しているのではないかと考える。Whileループは、メモリを消費しないが再帰処理ではメモリを消費する。わずかな差ではあるが処理能力に差が出ると考える。また、データ数がかなり大きい時スタックオーバーフローが発生してしまい正常に動作しない可能性が考えられるため、大量のデータを扱うときはループ版を利用したほうが良いのかもしれない。

1. 比較時に使用した計算機のスペック・実行環境

計算機のスペック

CPU : Intel® Core™ i5-4210U CPU @ 1.70Hz 2.40GHz

メモリ : 8.00GB

OS : Windows 10 64bit版

実行環境

Dev-C++ 5.11 Compile & Run

1. 課題に使用したプログラム

Pr1-1.c　// 今回の実験のベースのプログラム

Pr1-2.c　// Pr1-1.c を改良した、時間計測を含むプログラム