データ構造とアルゴリズム実験レポート
課題：最大公約数を求めるアルゴリズム

202110796 4 クラス 高橋大粋

締切日:2024 年 10 月 16 日
2024 年 10 月 16 日

1　基本課題

この課題では、教科書リスト 1.4(p.7) の「ユークリッドの互除法」に基づいた C プログラム gcd_euclid.c を作成し、作成したプログラムと教科書リスト 1.1(p.3) の「最大公約数を求める素朴なアルゴリズム」に基づいた C プログラム gcd_iter.c のリストおよび実行結果を示した。

1.1　gcd_euclid.c の作成

1.1.1　実装の方針

まず、ユークリッドの互除法を実行するための機能を gcd_euclid 関数に、実行結果を表示するための機能を main 関数に記述し、それぞれ実装した。main 関数は別ファイル main_euclid.c に実装した。また、main 関数は、int 型データ n, m をコマンドライン引数で渡すことによって動作する。もし、引数が 2 つ以外ならエラーを出力する。gcd_euclid 関数では入力された n, m の大小関係を固定するために大きい方の数を n にする処理を実装してある。これにより、ユークリッドの互除法を行いやすくなる。

1.1.2　実装コードおよびコードの説明

プログラム 1 に、gcd_euclid.c の主要部分を示す。1.1.1 で述べた、大きい方の数を n にする処理は 2〜6 行目の部分に相当する。

gcd_euclid 関数は整数型の引数を 2 つ取り、ユークリッドの互除法に基づき、n を m で割った余りが r のとき、n に m を代入、m に r を代入して n を m で割り、この操作を m が 0 になるまで繰り返す。m=0 のとき、n の値が最大公約数となるので int 型の n を返す。mian 関数では、コマンドラインで入力された n, m, そして gcd_euclid 関数で計算された最大公約数 gcd の 3 つの値が出力される。

```
1 int gcd_euclid(int n, int m) {
2   if (n < m) {
3       int tmp = m;
4       m = n;
5       n = tmp;
```

```
6     }
7     while (m != 0) {
8         int r = n % m;
9         n = m;
10        m = r;
11    }
12    return n;
13 }
```

プログラム 1　gcd_euclid.c の主要部

1.1.3　実行結果

まず、gcd_euclid.c を以下の make コマンドを実行してコンパイルし、引数として、自然数 24, 36 を与えて実行する。

```
1 PS C:\Users\daiki\Desktop\DSA\kadai1> make
2 gcc    gcd_iter.o main_iter.o    -o gcd_iter
3 gcc    gcd_euclid.o main_euclid.o    -o gcd_euclid
4 gcc    gcd_recursive.o main_recursive.o    -o gcd_recursive
5 PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_euclid 24 36
6 The GCD of 24 and 36 is 12.
```

5 行目の第一コマンドライン引数の 24, 第二コマンドライン引数の 36 はそれぞれ n, m に相当する。すなわち、このプログラムは自然数 n, m の最大公約数をユークリッドの互除法に基づき計算した結果 gcd と n, m を標準出力している。次に、コマンドライン引数が 2 つ以外のときのプログラムの挙動を確認する。

```
1 PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_euclid 10
2 Usage: C:\Users\daiki\Desktop\DSA\kadai1\gcd_euclid.exe <number1> <number2>
```

コマンドライン引数が 2 つ以外のときは、プログラム名と引数が 2 つ必要だという意味のエラーメッセージが標準出力されている。これにより、「ユークリッドの互除法」に基づいた C プログラム gcd_euclid が正しく実装されている事が確認できる。

次に授業で配布された c プログラム gcd_iter のプログラムの挙動も確認する。gcd_iter はユークリッドの互除法を使わず、自然数 n, m を 1 から順に割っていき、n, m を両方同時に割れたときの数を最大公約数 gcd とするプログラムである。

```
1 PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_iter 15 30
2 The GCD of 15 and 30 is 15.
3 PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_iter 15
4 Usage: C:\Users\daiki\Desktop\DSA\kadai1\gcd_iter.exe <number1> <number2>
```

gcd_euclid と同様にコマンドライン引数が 2 つのときには n, m, gcd を出力し、それ以外のときはエラーメッセージを出力する。gcd_euclid と gcd_iter はプログラムの中身は違うが動作そのものは同じである。

1.1.4 考察

gcd_iter は for ループの回数が n 回で固定なのに対し、gcd_euclid は n の値に依存し、$\frac{n}{2}$ 回以下であるので、計算量の面でみると gcd_euclid のほうが優れたアルゴリズムであると言える。また、今回は引数を 2 つしか取らなかったが、ユークリッドの互除法を繰り返し使うことにより、3 つ以上の自然数を引数に持つプログラムを実装することも容易であると考えられる。

2 発展課題

この課題では、基本課題で作成した gcd_euclid.c と gcd_iter.c について、それぞれの計算量（時間計算量、領域計算量）を議論した。また、gcd_euclid を、再帰的アルゴリズム（教科書 p.14）に基づいて書き換えた gcd_recursive.c を実装し、プログラムがどのように動作するかを基本課題と同様に説明した。

2.1 計算量について

一般に、時間計算量はループの回数に依存する。gcd_iter は for ループの回数が n 回で固定なので計算量は $O(n)$ となる。gcd_euclid は $n = mq + r$ と $q \geq 1$ より、$n - r = mq \geq m > r$ となり、$r < \frac{n}{2}$ が成り立つ。r の値は while ループの 2 回の繰り返しのあとに新たな n として取り扱われるため、ループを 2 周するごとに、n の値を半分未満の値にできることになる。したがって、最も回数が多い場合でも、$2\lfloor \log_2 n \rfloor - 1$ 回の繰り返しで、r=0 となり、計算が終了する。したがって、計算量は $O(\log n)$ となる。時間計算量を比較すると、ユークリッドの互除法の基づいたプログラムの方が時間計算量のオーダーが低く、優れていることがわかる。

領域計算量はアルゴリズムの実行に必要なコンピュータのメモリを表す計算の複雑さのことであり、たくさんの変数をメモリ内に保持するアルゴリズムは、領域計算量が大きく、あまりメモリを使わなければ領域計算量は小さいと言える。gcd_iter も gcd_euclid も領域計算量は $O(1)$ である。近年ではコンピュータのメモリが十分に増えたことにより領域計算量よりも時間計算量が重視されている。よって計算量はユークリッドの互除法に基づいたプログラムのほうが優れている。

2.2 gcd_recursive.c の作成

2.2.1 実装の方針

まず、gcd_euclid 関数を、再帰的アルゴリズム（教科書 p.14）に基づきいて書き換えた gcd_recursive 関数を実装した。その計算結果を表示するためのプログラムは授業で配布された main_recursive.c である。gcd_recursive も gcd_euclid と同様に引数として自然数を 2 つ取り、その最大公約数を出力する。違うのは関数内で自らを呼び出す再帰を使用しているところだ。再帰的アルゴリズムは構造そのものに再帰性が見られるデータ構造に対する操作や、計算過程が漸化式を用いて定義できるような処理などに非常にマッチしている。

2.2.2 実装コードおよびコードの説明

プログラム 2 に、gcd_recursive.c の主要部分を示す。2.2.1 で述べた、再帰を使用しているのは 9 行目の部分に相当する。

gcd_recursive 関数は 1.1.2 で述べた gcd_euclid 関数同様の処理を再帰を用いて実装した。9 行目で n に

m を、m に n%m を代入して再帰している。これにより、剰余の変数 r を用いずにユークリッドの互除法を実装できた。mian 関数では、コマンドラインで入力された n, m, そして gcd_recursive 関数で計算された最大公約数 gcd の 3 つの値が出力される。

```
if (n < m) {
    int tmp = m;
    m = n;
    n = tmp;
}
if (m == 0) {
    return n;
} else {
    return gcd_recursive(m, n%m);
}
return n;
```

プログラム 2　gcd_recursive.c の主要部

2.2.3　実行結果

1.1.3 と同様に make コマンドを実行してコンパイルし、コマンドライン引数として、自然数 18, 54 を与えて実行する。また、引数が 2 つ以外の場合の挙動も確認する。

```
PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_recursive 18 54
The GCD of 18 and 54 is 18.
PS C:\Users\daiki\Desktop\DSA\kadai1> ./gcd_recursive 2
Usage: C:\Users\daiki\Desktop\DSA\kadai1\gcd_recursive.exe <number1> <
    number2>
```

第一コマンドライン引数の 18, 第二コマンドライン引数の 54 がそれぞれ n, m に相当し gcd_recursive に渡され、計算された最大公約数 gcd が n, m とともに標準出力されている。また、コマンドライン引数が 2 つ以外の場合にはエラーを出力しているので gcd_recursive は正しく実装されていることが確認できる。

2.2.4　考察

再帰的アルゴリズムは、うまく利用すると非常に簡潔で理解しやすいプログラムを実装できるが、計算量という点では非再帰的アルゴリズムよりも悪いことが多い。そのため、n 重の for ループや計算過程が煩雑な場合に限定して再帰的アルゴリズムを利用するのが良いと考える。