アルゴリズム論 2&3

概要

- アルゴリズムの定義
- C言語の復習
- データ構造について
- 計算量について
- 再帰呼び出し

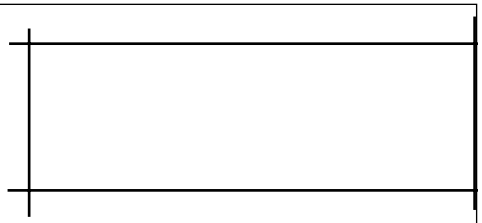
1

algorithm(広辞苑より)

- アラビアの数学者:アル・フワリズミー(Al-Khwarizmi)に因む
- 問題を解決する定型的な手法・技法
- コンピュータなどで、演算手続きを指示する規則。算法。

2

アルゴリズムとは？



- 与えられた問題を解くための、機械的操作からなる有限の「手続き」
- コンピュータが情報を処理する
  - 処理手順をコンピュータに指示する
  - プログラム：決められた言語（計算機言語）で記述
- 処理手順の良し悪し（一般的な基準）

  良い処理手順とは
  - 処理時間が短い
  - メモリ使用領域が小さい
  - わかりやすい記述(保守しやすい)

3

良いプログラムを書くために

- プログラムを書く段階において
- 処理手順を検討
  - 基本的な考え方
  - 問題解決方法
  - 全体の処理
- 良いプログラムは

  良い処理手順(アルゴリズム)

  を使用する
  - 先人が開発した定型的な処理手順を組み合わせるのが近道

4

本授業の構成及び特徴

- 各種の問題解決に際して...
- 適用できる各種の定型的手法を紹介
- 各種手法の特徴、計算量の比較を行う
- サンプルプログラムを使用して理解を深める

5

アルゴリズムの記述

- フローチャートまたは疑似プログラミング言語として記述するのが一般的
- フローチャートの構造(構造化言語)
  - 順次構造：決められた順番通りにいくつかの手順を実行する構造
  - 選択構造：ある条件を調べて、その結果に応じて手順が変化する構造
  - 反復構造：ある条件が成立している間、決められた手順を繰り返す構造

6

C言語の復習

- 変数定義
- ブロック構造
- 代入文
- 選択構造
  - if文
  - switch文
- 反復構造
  - for文
  - while文
  - do while文
- 関数

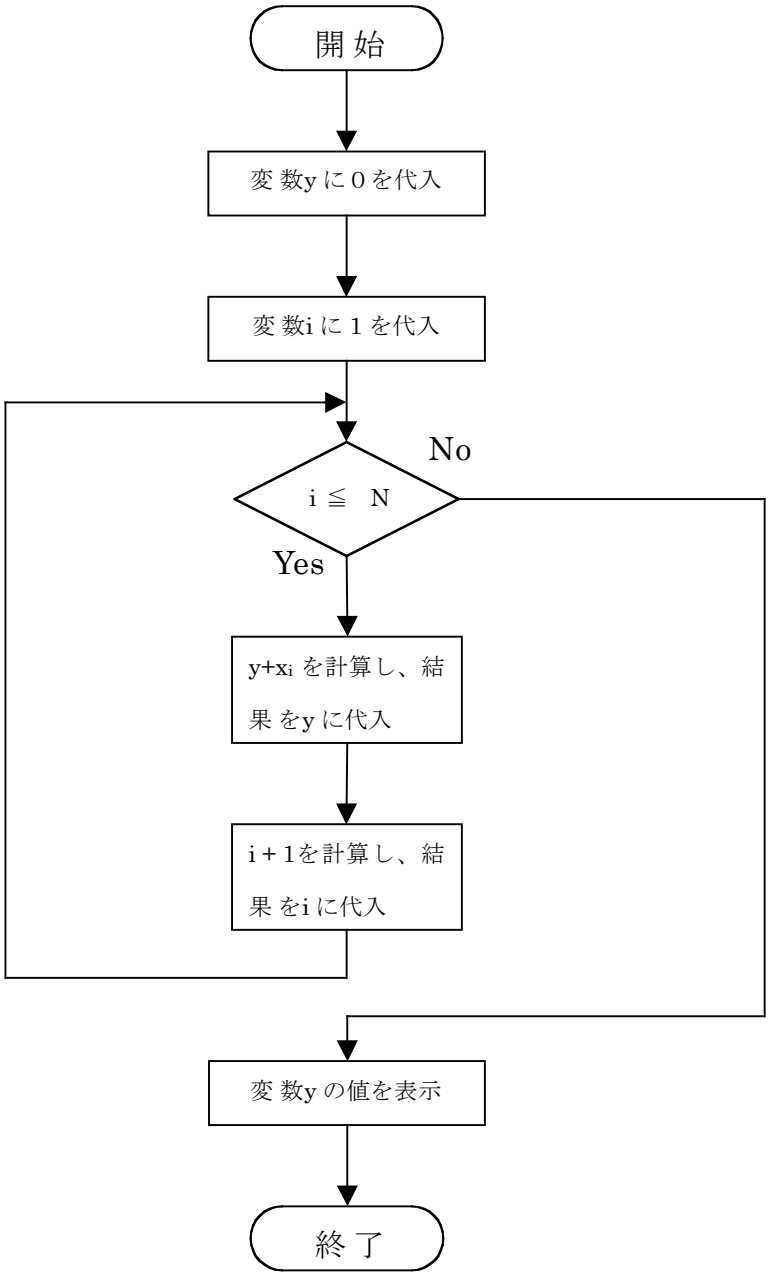
7

N個の整数の合計を求める

```
#include <stdio.h>

int main(void)
{
    int x[11]={0,1,2,3,4,5,6,7,8,9,10};
    int  N=10;
    int  y=0;
    int i;

    for (i=1;i<=N;i++) {
        y=y+x[i];
    }
    printf("¥n Sum is %d ", y);
    return(0);
}
```



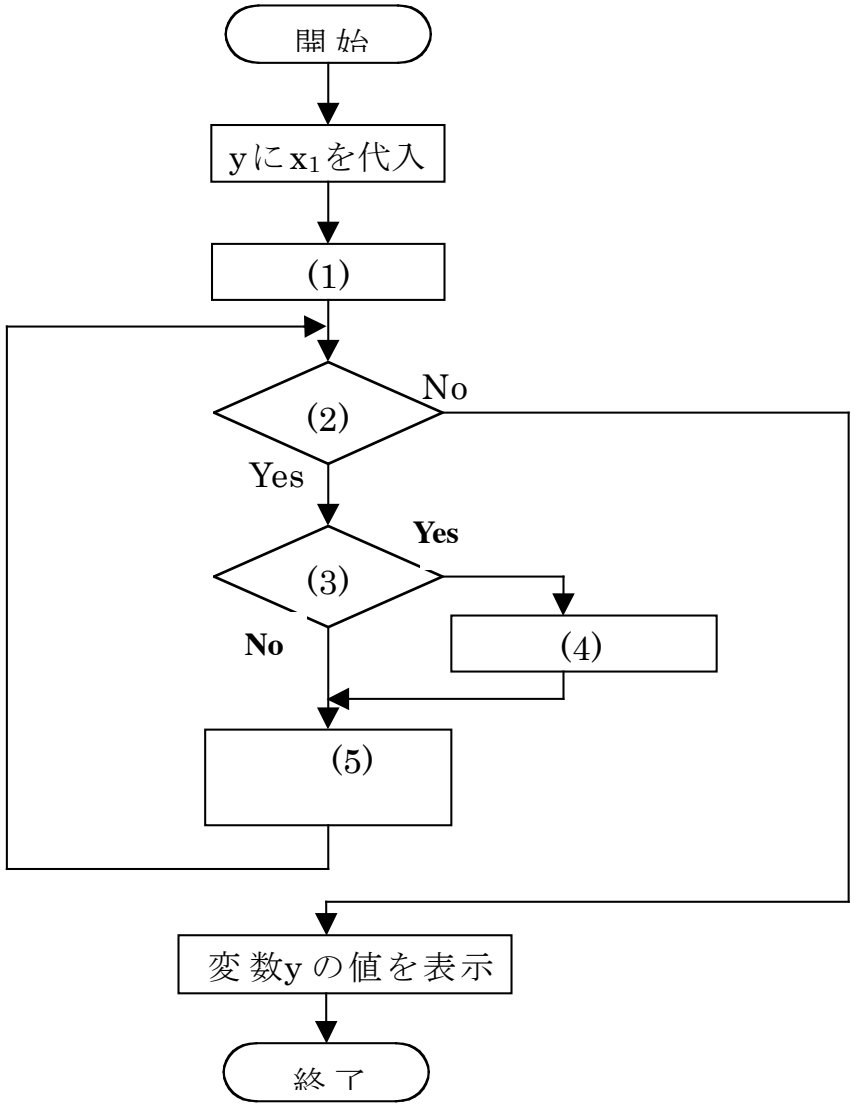
8

N個の整数の最大値を求める

```
#include <stdio.h>

int main(void)
{
    int x[11]={0,1,3,5,7,10,9,8,6,4,2};
    int  N=10;
    int  y=x[1];
    int i;

    for (i=2;i<=N ;i++) {
        if (y<x[i]) y=x[i];
    }
    printf("¥n Max is %d ", y);
    return(0);
}
```



9

C言語のデータ構造

型		バイト数	範囲
整数型	char	1	-127～127
	unsigned char	1	0～255
	short int	2	-32767～32767
	unsigned short int	2	0～65535
	int	4	-2147483647～ 2147483647
	unsigned int	4	0～4294967295
	long int	4	-2147483647～ 2147483647
	unsigned long int	4	0～4294967295
実数型	float	4	
	double	8	
	long double	8	

注:計算機の種類によって多少異なる

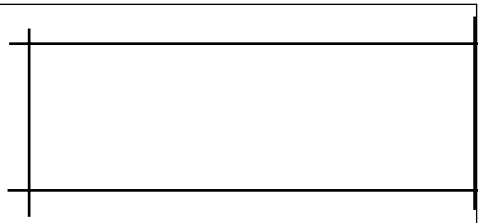
10

計算量について(前提)

- ランダムアクセスメモリを持つ逐次実行型のコンピュータ(ノイマン型コンピュータ)が前提
  - メモリアクセス時間が場所(アドレス)によらず一定
  - １回に１つの命令のみを実行する
  - 個々のコンピュータ固有のものは考慮しない
  - 各命令は単位時間で実行できる
  - 計算時間：アルゴリズムを実行した際の命令数
  - 記憶領域：使用した変数の数
- その他のコンピュータ？→非ノイマン型
  - データフロー型コンピュータ
  - 並列型コンピュータ
  - ベクトル型コンピュータ

11

計算量について



- 時間計算量(time complexity)
  - 所要演算量
    - 最大時間計算量（worst case time complexity）

      入力データ数nに対する最悪(大)の時間計算量：T(n)
    - 平均時間計算量（average case time complexity）
- 領域計算量(space complexity)
  - 所要記憶容量

12

最大時間計算量 T(n) の例

n個の整数の最大値を求める

```
#include <stdio.h>

int main(void)
{
      int x[11]={0,1,3,5,7,10,9,8,6,4,2};
      int  n=10;
      int  y;
      int i;


      y=x[1];
      for (i=2;i<=n ;i++) {
            if (y<x[i])
                  y=x[i];
      }
      printf("¥n Max is %d ", y);
      return(0);
}
```

	計算量
	1
	n
	n-1
最大	n-1　平均 (n-1)/2

合計:最大　3n-1　平均　(5n-1)/2

T(n)=3n-1

13

計算量の漸近的評価(オーダ記法)

- 計算量評価の関心事:
  nの値が大きくなった時にどうなるのか?
- nの値が十分大きい場合に計算量を漸近的に評価する必要がある
- オーダ記法
  - n_0以上の全てのnに対して$T(n) \leqq cf(n)$が成立する
    - c:定数
  - 「オーダ f(n)の計算量」 O(f(n)) と呼ぶ
    - 定数は無視する
  - f(n)の種類
    $n, n^2, n^3, \log_2 n, n \log_2 n, 2^n, n!, n^n$ 等
  - オーダ記法による計算量評価:計算量の第1次評価

14

計算量増加量の例

$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	n!	n^n
3.3	10	33	100	1000	1024	3.6×10^6	1.0×10^{10}
6.6	100	660	10000	1.0×10^6	1.27×10^{30}		
9.9	1000	9900	1.0×10^6	1.0×10^9			
13.2	10000	1.33×10^5	1.0×10^8	1.0×10^{12}			
16.6	100000	1.66×10^6	1.0×10^{10}	1.0×10^{15}			
19.9	1000000	1.99×10^7	1.0×10^{12}	1.0×10^{18}			

15

計算量の漸近的評価(続き)

- 多項式時間アルゴリズム
  - $O(n)$, $O(n \log_2 n)$, $O(n^2)$, $O(n^3)$

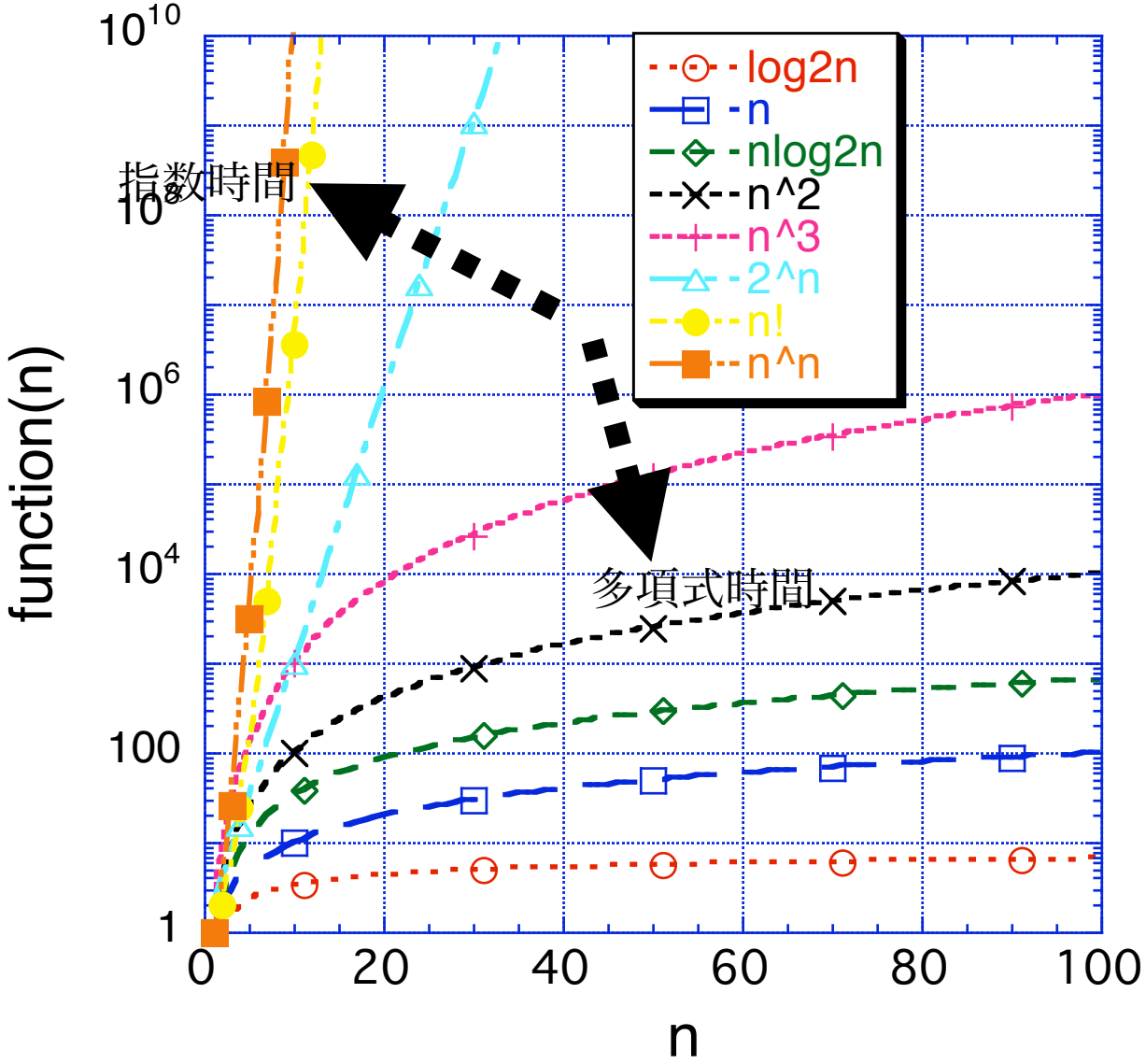
- 指数時間アルゴリズム
  - $O(2^n)$, $O(n!)$, $O(n^n)$

教科書p.13のグラフ

1秒で10^9の計算を実行するコンピュータが前提

指数時間アルゴリズムは非現実的

16



関数の形

増加量の順番

理解すること

素数の抽出

- 3種類のアルゴリズムによる計算量の違いに注目
- 2～100までの素数を抽出する

(1)単純版

自分より小さい数で順番に除算を行う

自分自身でのみ割り切れた場合は素数

自分以外の数で割り切れた場合は素数ではない

(2)改良版1

奇数のみを対象にする

自分より小さい素数で除算を行う

自分自身でのみ割り切れた場合は素数

自分以外の数で割り切れた場合は素数ではない

(3)改良版2

奇数のみを対象にする

自分の平方根より小さい素数で除算を行う

自分自身でのみ割り切れた場合は素数

自分以外の数で割り切れた場合は素数ではない

18

単純版(prime1.c)

ソースファイル

```
#include <stdio.h>

int main(void)
{
      int   i,n;
      int         N=100;        /* 最大の数 */
      int         counter=0;    /* 除算の回数 */

      for (n=2;n<=N;n++)     {
            for (i=2;i<n;i++)    {
                  counter++;
                  if (n%i==0)     /* 割り切れたら素数でない   */
                  break;
            }
            if (n==i)   printf("%d¥n",n);   /* 最後まで割り切れない */
      }
      printf("Total number of calculation : %d¥n",counter);

      return(0);
}
```

実行結果

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
Total number of calculation : 1133
```


19

改良版1(prime2.c)

ソースファイル

```
#include <stdio.h>
int main(void)
{
	int	i,n;
	int	N=100; /* 最大の数 */
	int	prime[500]; /* 素数を格納する配列 */
	int	ptr=0;	/* 抽出した素数の数 */
	int	counter=0; /* 除算の回数 */

	prime[0]=2; /* 最初の素数　*/
	ptr++;

	for (n=3;n<=N;n+=2)　{　/* 奇数のみを対象　*/
		for (i=1;i<ptr;i++) {　/* すでに抽出した素数で除算　*/
			counter++;
			if (n%prime[i]==0)　/* 割り切れたら素数でない　*/
				break;
		}
		if (ptr==i)
			prime[ptr++]=n;	/* 最後まで割り切れない */
	}
	for (i=0;i<ptr;i++) printf("%d¥n",prime[i]);
	printf("Total number of calculation : %d¥n",counter);
	return(0);
}
```

実行結果

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
Total number of calculation : 313
```


20

改良版2(prime3.c)

ソースファイル

```
#include <stdio.h>
int main(void)
{
        int    i,n;
        int    N=100; /* 最大の数 */
        int    prime[500]; /* 素数を格納する配列 */
        int    ptr=0; /* 抽出した素数の数 */
        int    counter=0; /* 乗算+除算の回数 */
        int    flag; /* 除算できたか否かのフラグ   */

        prime[ptr++]=2;   /* 最初の素数   */
        prime[ptr++]=3;   /* ２番目の素数   */

        for (n=5;n<=N;n+=2)     { /* 奇数のみを対象   */
                flag=0;
                for (i=1; counter++, prime[i]*prime[i]<=n;i++)     {
                        counter++;
                        if (n%prime[i]==0) { /* 割り切れたら素数でない   */
                                flag=1;
                                break;
                        }
                }
                if (flag==0)
                        prime[ptr++]=n;  /* 最後まで割り切れない */
        }
        for (i=0;i<ptr;i++) printf("%d¥n",prime[i]);
        printf("Total number of calculation : %d¥n",counter);
        return(0);
}
```

実行結果

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
Total number of calculation : 191
```


21

課題問題1

課題の目的:素数を抽出する各種アルゴリズムを使用して計算量の概念を理解する

1. 各種アルゴリズムのプログラム作成
2. 各種アルゴリズムでN=10,50,100,500,1000として計算量を算出
3. 計算量をグラフ化する
4. 考察を記述する

★レポートを作成し次の講義(2009/4/30)に提出

★レポートのフォーマットは自由とする