CS211: Algorithms & Data structures

Dr. Sameer M. Alrehaili

September 12, 2021

1 Introduction

In order to understand problem-solving skills, particularly algorithmic-solving skills, the following are a number of problems for you to tackle. Note that we use the basic approach to develop algorithms in this section. Next section presents same algorithms, but with analysis and performance improvment.

You may need to know the following list, which explains some mathematical notations found in some of the following algorithms.
$\boldsymbol{N} = \{0, 1, 2, 3, \ldots\}$ the set of natural numberrs
$\boldsymbol{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ the set of integers
$\boldsymbol{Z}^+ = \{1, 2, 3, \ldots\}$ the set of positive integers
$\boldsymbol{R}$ the set of real numbers
$\boldsymbol{R}^+$ the set of positive real numbers

1.1 The maximum of three numbers

Finding the maximum number among a set requires comparing them with each other.

Algorithm 1: Finding the maximum of three numbers

Input:a, b, and c, are three numbers
Output: max, the maximum number

1: $max \leftarrow a$
2: **if** $b > max$ **then**
3: $\quad max \leftarrow b$
4: **end if**
5: **if** $c > max$ **then**
6: $\quad max \leftarrow c$
7: **end if**
8: **return** max

1.2 The maximum number in an array

The following algorithm is finding the maximum element in a set.

Algorithm 2: Finding the maximum number

Input:$(a_1, a_2, \ldots, a_n)$, an array of n elements
Output: max, the maximum number in a

```
1: max ← a_1
2: for i ← 2 to n do
3:    if a_i > max then
4:       max ← a_i
5:    end if
6: end for
7: return max
```

This algorithm begins with assigning the first element of the array, a, into a variable named max. Then loop is used to compare the rest of elements in the array with max's value. When the ith element's value is greater than the value of max, then max is assigned with the value of the ith element. Think of how to find the second largest number?

1.3 Even or Odd numbers

1.3.1 Even or Odd for a given positive number

Find whether a given number is even or odd?

Algorithm 3: Finding whether a given number is even or odd

Input:a, is a positive integer number ($\boldsymbol{Z}^+$)
Output: "*even*" or "*odd*"

```
1: if a mod 2 = 0 then
2:    return "even"
3: else
4:    return "odd"
5: end if
```

1.3.2 Even or Odd numbers for first n

Wrtite a pseudocode to find odd and even numbers of first n ?

Algorithm 4: Finding odd and even numbers of first n

Input:n, is a positive integer number ($\boldsymbol{Z}^+$)
Output: "$even$" or "odd"

```
1: for i ← 1 to n do
2:     if i mod 2 = 0 then
3:         Print i is "even"
4:     else
5:         Print i is "odd"
6:     end if
7: end for
```

1.3.3 Odd numbers for first n

Wrtite a pseudocode to find odd numbers of first n ?

Algorithm 5: Finding odd numbers of first n

Input:n, is a positive integer number ($\boldsymbol{Z}^+$)
Output: the first n odd numbers

```
1: for i ← 1 to 2 × n do
2:     if i mod 2 ≠ 0 then
3:         Print i
4:     end if
5: end for
```

1.4 Prime numbers

1.4.1 Check a given number

Finding that a given number is prime or not? a positive integer

Algorithm 6: Finding whether a number is prime or not

Input:a, is a positive integer number ($\boldsymbol{Z}^+$)
Output: "$true$", if a is prime, "$false$" , if a is not prime

```
1: for i ← 2 to a − 1 do
2:     if a mod i = 0 then
3:         return "true"
4:     end if
5: end for
6: return "false"
```

1.5 Factorial $n!$

Find factorial of a given number?

Algorithm 7: Finding factorial of a number

Input:n, is a positive integer number ($\boldsymbol{Z}^+$)
Output: f, is the factorial of n, $n!$

1: $f \leftarrow 1$
2: **for** $i \leftarrow 2$ to n **do**
3: $\quad f \leftarrow f \times i$
4: **end for**
5: **return** f

1.6 The power of a number

Suppose you want to compute x^n, where x is any real number ($\boldsymbol{R}$), and n is any integer number ($\boldsymbol{Z}$).

Calculate the power of a given number?

Algorithm 8: Computing the power of a number

Input:x, is a real number $x \in \boldsymbol{R}$, n is an integer number, $n \in \boldsymbol{Z}$
Output: x^n

1: $p \leftarrow 1$
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad p \leftarrow p \times x$
4: **end for**
5: **return** p

1.7 The sum function

Calculate the sum of a given set of numbers?

Algorithm 9: Computing the sum of numbers

Input:$(a_1, a_2, \ldots, a_n)$, is a an array of numbers
Output: sum

1: $sum \leftarrow 0$
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad sum \leftarrow sum + a_i$
4: **end for**
5: **return** sum

1.8 The average function

Find the average of a given set of numbers?

Algorithm 10: Computing the average of numbers

Input:$(a_1, a_2, \ldots, a_n)$, is a an array of numbers
Output: avg

```
1: sum ← 0
2: for i ← 1 to n do
3:     sum ← sum + a_i
4: end for
5: avg ← sum/n
6: return avg
```

1.9 Selection problem

Let's say we have a set of n numbers, and we would like to determine the kth largest. For example, the 3rd largest element in $\{2, 4, 8, 1, 9, 4, 5\}$ is 5 and the 2nd smallest is 2.

Write an algorithm to find the k-th largest element in A

Read the N numbers into an array, sort the array in decreasing order by some simple algorithm such as bubblesort, and then return the element in position k

Read the first k elements into an array and sort them (in decreasing order). Next, each remaining element is read one by one. As a new element arrives, it is ignored if it is smaller than the kth element in the array. Otherwise, it is placed in its correct spot in the array, bumping one element out of the array. When the algorithm ends, the element in the kth position is returned as the answer.

Algorithm 11: Selecting problem

Input: an array A and k
Output

```
1: A ←n numbers such that A ← {x_1, ..., x_n},
2: S ← [9, 7, 5, 4, 3, 2, 1, 0]
3: if i ≥ 5 then
4:     i ← i − 1
5: else
6:     if i ≤ 3 then
7:         i ← i + 2
8:     end if
9: end if
```

1.10 Greatest Common Divisor (GCD)

The Greatest Common Divisor (GCD) of two integers, a and b, is the largest positive integer number that divides both of them without leaving any reminder.

For example, 6 is the GCD of 12 and 18

The problem of GCD has been discovered

1.10.1 Brute Force Method

Find the greatest common divisor (GCD) of two integers, a and b.

Algorithm 12: GCD

Input: Two integer numbers a and b
Output: gcd

1: m $\leftarrow$ The minimum number of a and b.
2: gcd $\leftarrow 0$
3: i $\leftarrow 2$
4: **while** $(i \leq m)$ **do**
5: **if** $a \ mod \ i = 0$ and $b \ mod \ i = 0$ **then**
6: $gcd \leftarrow i$
7: **end if**
8: i $\leftarrow i + 1$
9: **end while**
10: **return** gcd

?? Brute Force method loops for 48 times to find gcd(80, 48), while recursive method cost 4 iteration and subtraction method only cost 3 iterations. This algorithm is inefficient and tedious if a and b are big and the gcd of the two numbers is equal to one.

Listing 1: GCD

```
// Hello.java
public static int GCD1_1(int a, int b){
    int m = Math.min(a,b);
    int gcd=0;
    int i=2;
    while(i<=m)
    {
        if(a%i ==0 && b%i==0)
            gcd=i;
    i++;
    }
    return gcd;
}
```

1.10.2 Subtraction Method

Algorithm 13: GCD

Input: Two integer numbers a and b
Output: a

```
1: while (a ≠ b) do
2:     if a > b then
3:         a ← a − b
4:     else
5:         b ← b − a
6:     end if
7: end while
8: return a
```

1.10.3 Euclid's algorithm using iterative method

$gcd(a, b) = gcd(b_i, a_i \bmod b_i)d$
$ax_0 + by_0 = d$
$a_{i+1} = b_i$
$b_{i+1} = a_i \bmod b_i$

Algorithm 14: GCD

Input: Two integer numbers a and b
Output: gcd

```
1: while (b ≠ 0) do
2:     r ← a mod b
3:     a ← b
4:     b ← r
5: end while
6: return gcd
```

1.10.4 Euclid's algorithm using recursion method

1.11 Bubble sort

Solution

Algorithm 15: Bubble Sort

Input: An array A of n numbers $A \leftarrow \{x_1, \ldots, x_n\}$
Output: sorted A

```
1: for i ← 1 to n − 1 do
2:     for j ← 2 to n − i do
3:         if A_j − 1 > A_j then
4:             swap A_j − 1 and A_j
5:         end if
6:     end for
7: end for
```

java /*run*/*media*/*salrehaili*/*c*6*b*5*e*3*a*6−85*e*4−4*bae*−*a*742−8*afb*7118*f*09*c*/*backup*/*uni*/*courses*/*fall*2021/*C*

2 Algorithm Analysis