Algorithms

Introduction

Lecture 7 by *Marina Barsky*

So what is an algorithm?

Different definitions of an algorithm:

- *an unambiguous specification of how to solve a class of problems. Algorithms can perform calculation, data processing and automated reasoning* **Wikipedia**
- *a set of rules for solving a problem in a finite number of steps, as for finding the greatest common divisor* **Random House**
- *a procedure for solving a mathematical problem in a finite number of steps that frequently involves repetition of an operation* **Merriam-Webster**
- *a process or set of rules to be followed in calculations or other problem-solving operations, especially by a computer* **Oxford**

Algorithm:

- a sequence of steps for solving a computational problem

- If we want the machines to solve problems for us, then the sequence of steps must be *precise and unambiguous*

A Russian stamp showing Persian mathematician Muhammad ibn Musa **Al-Khwarizmi** whose last name was transliterated to ***Algorithmi***

Sample algorithm

Computational problem: compute max value in the array

Algorithm:
Declare the accumulator variable to hold ‘max so far’
Look at each element of the array in turn and compare it to ‘max so far’
If current element is greater than ‘max so far’ then
    Update max so far with current element
By the end max so far will hold the max

Translate into code

```
int max(int [] A) {
    Integer best = null;
    for (int n: A) {
        if( best ==null || n > best) {
            best = n;
        }
    }
    return best;
}
```

Problem vs. problem instance

- Problem instances:
  1. ***What is the position of element 11 in array A={2,10,4,1,3,11,33}?***
  2. ***What is the median of A?***

Median is the middle value in the sorted array

- General algorithmic problems:

We are interested in solving these

  1. **Given an array A of integers, and the target integer *t* find the position of the first occurrence of *t* in *A***
  2. **Given an array of integers *A* find its median**

Developing Algorithms: steps

1. Formalize the problem: input and output
2. Brainstorm solution
3. Express solution: pseudocode
4. Prove correctness (outside the scope of this course)
5. Estimate running time
6. Estimate space usage

1. Formalizing problem

- Sample problem instance: what is the Greatest Common Divisor (GCD) of 12 and 99?
- Formalized general problem: input and output

Problem: Compute GCD

Input: 2 integers *a, b*. $b > 1$, $a > 1$, $a > b$
Output: $\gcd(a, b)$.

We want it to work on large numbers:
gcd(3918848, 1653264)

Problem instance

2. Brainstorming the solution

GCD: Formal Definition

For integers, *a* and *b*, their ***greatest common divisor*** or ***gcd(a, b)*** is the largest integer *d* s.t. *d* divides both *a* and *b (without remainder)*.

Why would we want to compute it:

- Put fraction a/b into simplest form.

Need to check remainders of (a/d) (b/d)

- *d* should divide both *a* and *b*.
- Want *d* to be as large as possible.

a=45, b=15

both 45 and 15 are divisible by 3, 5,15

we want to find 15

Go over an example

Solution

Problem: Compute GCD

Input: 2 integers a, b. $b > 1$, $a > 1$, $a > b$

Output: $\gcd(a, b)$.

According to the problem and the definition of gcd:

- We need to go over integers 1, 2, …
- Check if each such integer d divides both a and b without remainder
- Keep the largest such number
- Stop when $d = min(a,b) = b$

a=45, b=15

both 45 and 15 are divisible by 3, 5,15

we want to find 15

This is algorithm in plain English

Three ways of expressing algorithmic solutions

- English
- Pseudocode
- Program

Increasing precision

Pseudocode: example

```
FOR i from 1 TO 100 DO
    IF i is divisible by 3 AND i is divisible by 5 THEN
        OUTPUT "Both"
    ELSE IF i is divisible by 3 THEN
        OUTPUT "By 3"
    ELSE IF i is divisible by 5 THEN
        OUTPUT "By 5"
    ELSE
        OUTPUT i
```

Python equivalent

```
def some_algorithm ():
  for i in range(1,101):
    if i%3 == 0 and i%5 == 0:
      print(i, "Both")
    elif i%3 == 0:
      print(i, "By 3")
    elif i%5 == 0:
      print(i, "By 5")
    else:
      print(i)
```

Python – the most pseudocode-like language

Pseudocode does not have specific syntax requirements: it just has to be **clear and unambiguous**

Some specifics

- Assignment operator:
  X **:=** 5
  X ← 5 (you can use x=5, but then use == for equality)
- Comparing for equality:
  if x **=** y (you can use x==y)
- *FOR* loops:
  for each element x in sequence:
  for i from 1 to n:
  for i from 1 to n step 2:
  for i from n down to 1:
- *WHILE* loop:
  same as if

Pseudocode does not have specific syntax

But keep in mind the goal:
pseudocode **must be easily translatable into a working program** (in any language).

Avoid language-specific instructions

Pseudocode for GCD

English:

Try every integer from 1 to *b (b* < *a* without lost of generality).
If the integer divides both *a* and *b*, remember the best *gcd* so far.
Since the integers we test are increasing,
the algorithm will remember the last – the greatest common divisor for *a* and *b.*

Pseudocode:

```
Algorithm GCD(a, b)

best = 1
for d from 2 to b:
    if (d divides a) and (d divides b):
        best = d
return best
```

Exercise: Develop algorithm for searching in the array

- Formalize the problem: input, output
- Brainstorming?
- Now write the pseudocode

Pseudocode for search in Array and Linked List

Algorithm find (array A, target)

```
n = length of A
for i from 0 to n-1:
        if A[i] == target:
                return i
return -1
```

Algorithm find (Linked List head, target)

```
current = head
i = 0
while current is not null:
        if current.data == target:
                return i
        current = current.next
        i = i + 1
return -1
```