3.1 Algorithms

Definition: An algorithm is a finite sequence of precise instructions for performing a computation or solving a problem.

For example, using the quadratic formula gives an algorithm to solve quadratic equations by hand. More complicated algorithms are usually carried out on a computer.

As a simple example to start with, let's look at an algorithm to find the biggest integer in a finite set.

So given the input 9, 13, 2, 6, 11 the algorithm should return the output 13.

Could our algorithm be just "look for the maximum integer". No, that's not precise enough. If the input was 1000 integers, how exactly could you or a machine find the maximum?

Here are precise steps:

(1) Set the temporary maximum equal to the first integer in the input list.
(2) Compare the temporary max with the next integer in the list. If the next integer is bigger, make it the new temporary max.

(3) Repeat the last step until you've got to the end of the input list.
(4) Return the temporary max as output. It is the maximum of the input list.

These steps were given in English. How could we code them for a machine? There are programming languages like C++, Python and Java that are all slightly different. Instead we use pseudocode which is easy to understand and convert to other programming languages.

Here is our algorithm in pseudocode:

```
procedure max (a_1, a_2, ..., a_n : integers)
max := a_1
for i := 2 to n
    if max < a_i then max := a_i

return max
```

On the first line "max" is the name of the procedure (algorithm) and the input is in parentheses. The notation ":=" means assignment so on line 2 the variable max is assigned the value of a_1. There is a for loop on lines 3, 4 which repeats as i goes from 2 to n. The "if p then something" means to do the something if the condition p

is true. Finally "return" gives the output.

Example (1) Show all the steps used by this procedure with input 9, 13, 2, 6, 11.

Solution: The input is stored as
$a_1 = 9 \quad a_2 = 13 \quad a_3 = 2 \quad a_4 = 6 \quad a_5 = 11$

and $n = 5$. Then $max := a_1$ means $max = 9$.

Starting the for loop, $i = 2$ and
if $\underbrace{max < a_2}_{9 < 13 \text{ true} \longrightarrow}$ then $\underbrace{max := a_2}_{\text{now } max = 13}$

Next $i = 3$ and if $\underbrace{max < a_3}_{13 < 2 \text{ false so don't do}}$ then $max := a_3$ ↑

For $i = 4$ and $i = 5$ condition $max < a_i$ also false so max is still 13. The for loop is finished.

Finally the procedure returns 13 as output.

It is clear that this algorithm is correct (does what it's supposed to) and works for any finite set of integers as input.

It would not work on an infinite set of integers, but for a finite set it always finishes and so is finite and effective.

- See p 193 for more on the properties algorithms should have.
- Also see Appendix 3 in the textbook for more information about pseudocode.

Searching algorithms

Suppose you look up a word in an online dictionary. How does the system find the word and then display its meaning? The words are stored in order from "a" to maybe "zyzzyva" with a file on each.

Instead of words we'll use a list of integers and search for a specific one.

First we assume the integers are not in order but are all different, for example

14, 6, 11, 2, 0, 9 .

If we are searching for 11 then we see it's the 3rd number in the list so the output should be 3.
If we search for 8 then it's not in the list and to signify this we can output 0.

Here is the Linear search algorithm in pseudocode:

```
procedure linear search (x: integer, a1, ..., an:
                         distinct integers)

i := 1
while (i ≤ n and x ≠ ai)
    i := i + 1
if i ≤ n then location := i
    else location := 0

return location
```

We used the while loop and "while p something" means to keep doing the something while p is true and stop when p becomes false. Also

"if p then A else B"

means do A if p is true and do B if p false.

Check that you understand why the linear search procedure works.

Now if the integers are distinct and in increasing order, like for example

3, 4, 9, 11, 17, 32

then it is much easier to search. That's why lists (like dictionaries or phone books) are ordered. At each step you could break the list into two equal parts and see if you need to search the

first part or the second. This is how the binary search algorithm works.

During the procedure it keeps narrowing the search to say between the i^{th} and j^{th} numbers in the list

$$a_i \quad a_{i+1} \quad \cdots \quad a_{j-1} \quad a_j .$$

↑ middle number a_m

then divide this list in two with $m = \lfloor \frac{i+j}{2} \rfloor$

and see if the number x you are searching for is bigger than the middle number a_m

Adjust i and j accordingly and keep going until $i = j$ and you've narrowed it to one possibility:

```
procedure binary search (x: integer, a_1, ..., a_n:
                         increasing integers)
i := 1
j := n
while i < j
    m := ⌊(i+j)/2⌋
    if x > a_m then i := m+1
       else j := m
if x = a_i then location := i
   else location := 0

return location
```

example ② Show the steps used by the binary search procedure to search for 6 in
2, 3, 6, 8.

Solution: In our set up $x=6$, $a_1=2$, $a_2=3$, $a_3=6$, $a_4=8$ and $n=4$. Also $i=1$ and $j=4$.

We have $i<j$ so the while loop begins

$$m = \left\lfloor \frac{i+j}{2} \right\rfloor = \lfloor 5/2 \rfloor = 2$$

if $x > a_m$ then $i := m+1$ else $j := m$

$6 > 3$
true so $\rightarrow i = 3$

we still have $i<j$ $(3<4)$ so the while loop continues

$$m = \left\lfloor \frac{i+j}{2} \right\rfloor = \lfloor 7/2 \rfloor = 3$$

if $x > a_m$ then $i := m+1$ else $j := m$

$6 > 6$
false so $\rightarrow j = 3$

now $i<j$ $(3<3)$ is false and while loop ends.

if $x = a_i$ then location $:=$ i else location $:= 0$

$6 = a_3$
true so $\rightarrow$ location $= 3$

procedure ends by returning the output 3. That means it found the number $x=6$ as the third number in the input list.