

Algorithms and Complexity

Key properties of algorithms:

- Precise instructions for a computation.
- Correctness.
- Complexity. Running time. Space.

To determine time complexity:

- count operations, and
- convert to big-Oh notation.

We will use “pseudocode” in the style of the book to describe algorithms. Read Appendix 2.

max returns the maximum element in a sequence.

```

procedure max( $a_1, a_2, \dots, a_n$ : R)
    max :=  $a_1$ 
    for  $i := 2$  to  $n$ 
        if  $max < a_i$  then  $max := a_i$ 
    end for
    return max
end procedure

```

The comparison $max < a_i$ is performed $n - 1$ times. $n - 1$ is $O(n)$.

linear_search returns the location of *key* in a sequence, or 0 if *key* is not in the sequence.

```
procedure linear_search (key,  $a_1, \dots, a_n$ : R)
  loc := 0
  i := 1
  while  $i \leq n \wedge loc = 0$ 
    if  $key = a_i$  then loc := i
    i := i + 1
  return loc
```

The comparison $key = a_i$ is performed from 1 to n times. Both 1 and n are $O(n)$.

binary_search returns the location of *key* in a sorted sequence, or 0 if *key* is missing.

```
procedure binary_search (key: R;
                           $a_1, a_2, \dots, a_n$ : increasing R)
  left := 1; right := n
  while  $left < right$ 
    mid :=  $\lfloor (left + right) / 2 \rfloor$ 
    if  $key > a_{mid}$ 
      then left := mid + 1
      else right := mid
  if  $key = a_{left}$ 
    then return left
  else return 0
```

$key > a_{mid}$ is performed $\leq \lceil \log_2 n \rceil$ times. $O(\log_2 n)$.

bubble_sort sorts a sequence of numbers.

```

procedure bubble_sort ( $a_1, \dots, a_n: \mathbf{R}$ )
  for  $i := 1$  to  $n - 1$ 
    for  $j := 1$  to  $n - i$ 
      if  $a_j > a_{j+1}$ 
        then  $temp := a_j$ 
            $a_j := a_{j+1}$ 
            $a_{j+1} := temp$ 

```

$a_j > a_{j+1}$ performed $\sum_{i=1}^{n-1} (n - i)$ times. $O(n^2)$.

insertion_sort sorts a sequence of numbers.

```

procedure insertion_sort ( $a_1, \dots, a_n: \mathbf{R}$ )
  for  $j := 2$  to  $n$ 
     $i := j - 1$ 
     $temp := a_j$ 
    while  $i \geq 1$  and  $a_i > a_j$ 
       $a_{i+1} := a_i$ 
       $i := i - 1$ 
     $a_{i+1} := temp$ 

```

$a_i > a_j$ at most $\sum_{j=2}^n (j - 1)$ times. $O(n^2)$.

change makes change. c_i is a denomination.

procedure *change* ($n: \mathbf{Z}^+$;
 c_1, \dots, c_r : decreasing \mathbf{Z}^+ with $c_r = 1$)
for $i := 1$ **to** r
 while $n \geq c_i$
 add a coin with value c_i to the change
 $n := n - c_i$

$n \geq c_i$ performed less than $r + c_1 + \lfloor n/c_1 \rfloor$ times.