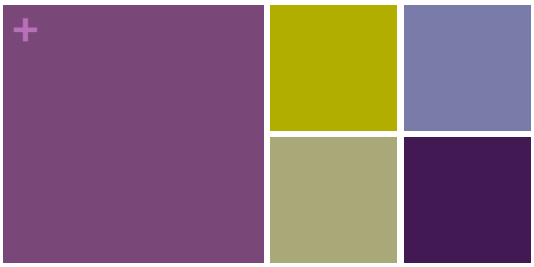

Describing and Evaluating Algorithms

Computer programs

- **procedures**
  - **call**ed
    - with **parameters** (inputs)
  - **return** a value (output)
- **array**
  - aggregate of data of the same type in one entity
  - each **entry**
    - has an **index** (indices start at 1)
    - has an **element** (value)
    - square brackets [] indicate the index of an array
      - for instance the *i*-th element of array *A* is written *A*[*i*]

Example Procedure Linear Search

Procedure LINEAR-SEARCH(*A,n,x*)
Inputs:
- *A*: an array
- *n*: the number of elements in *A* to search through
- *x:* the value being searched for

Output: Either an index *I* for which *A*[*i*] = *x*, or the special value NOT-FOUND, which could be any invalid index into the array, such as 0 or any negative integer.

1. Set *answer* to NOT-FOUND.
2. For each index *i,* going from 1 to *n*, in order:
   A. If *A*[*i*] = *x,* then set *answer* to the value of *i.*
3. Return the value of *answer* as the output.

Example Procedure Better Linear Search

Procedure BETTER-LINEAR-SEARCH(*A,n,x*)
Inputs:
- *A*: an array
- *n*: the number of elements in *A* to search through
- *x:* the value being searched for

Output: Either an index *I* for which *A*[*i*] = *x*, or the special value NOT-FOUND, which could be any invalid index into the array, such as 0 or any negative integer.

1. For each index *i,* going from 1 to *n*, in order:
   A. If *A*[*i*] = *x,* then return the value of *i* as output.
2. Return NOT-FOUND as the output.

Example Procedure Better Linear Search

Procedure BETTER-LINEAR-SEARCH(*A,n,x*)
Inputs:
- *A*: an array
- *n*: the number of elements in *A*
- *x:* the value being searched for

Can we do better? How?

Output: Either an index *I* for which *A*[*i*] = *x*, or the special value NOT-FOUND, which could be any invalid index into the array, such as 0 or any negative integer.

1. For each index *i,* going from 1 to *n*, in order:
   A. If *A*[*i*] = *x,* then return the value of *i* as output.
2. Return NOT-FOUND as the output.

How can we do better?

- Notice
  - we do 2 comparisons every time
- Question
  - can we do just one comparison each time?

How can we do better?

- Notice
  - we do 2 comparisons every time
- Question
  - can we do just one comparison each time?
- Insight
  - we know it stops if it's the item is in the array

How can we do better?

- Notice
  - we do 2 comparisons every time
- Question
  - can we do just one comparison each time?
- Insight
  - we know it stops if it's the item is in the array
- Answer
  - replace the last value with a sentinel

Sentinel? Huh?

- A sentinel is the value that will stop a while loop, but is not the 'true' item searched for.
- Let's say you have X = "ABCDE" and you are searching for 'G'
  - set last to 'E' set X[n-1] to 'G'
  - resulting in X = "ABCDG" and last = 'E'
- Now you know that you will find 'G'
- but you still have to do at least one additional test.

Example Procedure Sentinel Linear Search

Procedure SENTINEL-LINEAR-SEARCH(*A,n,x)*
Inputs and Output: Same as LINEAR-SEARCH
1. Save *A*[*n*] into *last* and then put *x* into *A*[*i*].
2. Set *i* to 1.
3. While *A*[*i*] = *x*, do the following
   A. Incement *i* .
4. Restore *A*[*n*] from *last* .
5. If *i* < *n* or *A*[*i*] = *x*, then return the value of *i as the output.*
6. Otherwise, return NOT-FOUND as the output.

Analyzing time complexity

- Theta notation
- Big O notation
- little o notation

Showing correctness

- Using Loop invariants
  - Initialization – invariant is true before first iteration
  - Maintenance – as long as it is true before the first iteration, the invariant is also true before the next iteratoin
  - Termination – the loop ends, and when it does, the loop invariant along with the reason that the loop terminated, gives us a useful property
- Using Recursion
  - Induction
  - prove true for base case(s)
  - assuming true for n(or n-1), prove true for n+1(or n)