Lecture Outline for Recurrences

- Recursive definition of sequences
- Recursive definition of sets
- Recursive definition of operations
- Recursive definition of algorithms
- Next Part: Solving recurrences
  - Expand, guess, verify
  - Solution formula
- Section 2.4 of text

Recursive definition

- Define something in terms of itself!!!
- Consists of two steps:
  1. Basis step: Define something simple *not* in terms of itself
  2. Inductive or recursive step: New cases are defined in terms of previous cases

Recursive definition of sequence

- Sequence: Ordered set of objects
- If you define the first value of the sequence (or first few values) [Basis step]

AND

- Define later values in terms of earlier values [Inductive step]
- That is a Recursively defined sequence

Recursive definition of sequence

- Example:
  - $T(1) = 1$
  - $T(n) = T(n-1) + 3, n \geq 2$

- Example:
  - $F(1) = 1, F(2) = 1$
  - $F(n) = F(n-1) + F(n-2), n>2$

Fibonacci Sequence

Prove: $F(n) = 5\,F(n-4) + 3\,F(n-5),\ n \geq 6$

Recursive definition of set

- A set is an unordered sequence
- Recursive definition of set
- Example:
  1. Set of ancestors of James
  2. Set of strings made out of the alphabet *A*, called *A**

Recursive definition of set

- Example:
  - Define all binary strings that are palindromes

  - All identifiers that must begin with a letter and can have number or letter after that

Recursive definition of operation

- Define an operation performed on an object in terms of a basis step and in terms of smaller sized objects
- Example:
  - Exponentiation by a positive integer

  - Concatenation of a string with itself *n* times, i.e., x^n

Recursive definition of algorithm

- Given the sequence
  - S(1) = 2
  - S(n) = 2 S(n-1), n≥2
- Example 1: Iterative algorithm
- Example 2: Recursive algorithm

Iterative definition of algorithm

```
ALGORITHM
S(integer n)
//function that iteratively computes the value S(n)
//for the sequence S of Example 29
Local variables:
integer i              //loop index
CurrentValue           //current value of function S
   if n = 1 then
      return 2
   else
      i = 2
      CurrentValue = 2
      while i <= n do
         CurrentValue = 2*CurrentValue
         i = i + 1
      end while
      //CurrentValue now has the value S(n)
      return CurrentValue
   end if
end function S
```

Recursive definition of algorithm

ALGORITHM

S(integer *n*)
//function that recursively computes the value *S*(*n*)
//for the sequence *S* of Example 29

```
    if n = 1 then
        return 2
    else
        return 2*S(n − 1)
    end if
end function S
```


Recursive definition of algorithm

- Selection Sort

ALGORITHM SELECTIONSORT

SelectionSort(list *L*; integer *j*)
//recursively sorts the items from 1 to *j* in list *L* into increasing order

```
    if j = 1 then
        sort is complete, write out the sorted list
    else
        find the index i of the maximum item in L between 1 and j
        exchange L[i] and L[j]
        SelectionSort(L, j − 1)
    end if
end function SelectionSort
```

Complexity of Selection Sort

- Give the recursive definition for the number of steps in Selection Sort when it has to work on a list of size *n*.
- Hint: The work is in terms of the work needed on a list of size *n-1*, plus some term.

T(n) =

- Does this amount of work depend on the data in the specific list that is being sorted?

Recursive definition of algorithm

- Binary search

```
ALGORITHM BinarySearch
BinarySearch(list L; integer i; integer j; itemtype x)
//searches sorted list L from L[i] to L[j] for item x
   if i > j then
      write("not found")
   else
      find the index k of the middle item in the list L[i]–L[j]
      if x = middle item then
         write("found")
      else
         if x < middle item then
            BinarySearch(L, i, k - 1, x)
         else
            BinarySearch(L, k + 1, j, x)
         end if
      end if
   end if
end function BinarySearch
```

Binary Search: Example

- Given list L
  - {3, 6, 11, 17, 19, 24, 26}
  - Search for 15
  - Search for 24

Complexity of Binary Search

- Give the recursive definition for the number of steps in Binary Search when it has to work on a list of size *n*.

- *Hint*: The work is in terms of the number of steps needed on a list of size *n/2*, plus some term.

T(n) =

- Does this amount of work depend on the data in the specific list that is being searched?