Programming Design In-class Practices

Algorithms and Recursion

Ling-Chieh Kung

Department of Information Management
National Taiwan University

Complexity

- Let’s watch a video!

Problem 1: linear search

- Given a size-n array A of integers and a target integer t, check whether t is in A.
- Please design a function that:
  - Takes an integer array A, its length n, and an integer t as parameters.
  - Returns true if t is in A and false otherwise.
- Please write **pseudocode** by writing a **loop**.

```
Search(A, n, t)
    for i from 1 to n
        if A_i = t
            return true
    return false
```

Problem 2: linear search (recursion)

- Given a size-n array A of integers and a target integer t, check whether t is in A.
- Please design a function that:
  - Takes an integer array A, its length n, and an integer t as parameters.
  - Returns true if t is in A and false otherwise.
- Please write **pseudocode** with **recursion**.

```
Search(A, n, t)
    t_0 = Search(A, n − 1, t)
    if t_0 = true or A_n = t
        return true
    else
        return false
```

Problem 3: Hanoi Tower

- Given an integer n, the fastest way to solve the Hanoi Tower problem is unique.
- Let's watch a **video**!
- Though you have read the code solving the Hanoi Tower problem, please still write **pseudocode** for it.

Problem 3: Hanoi Tower

Hanoi(f, v, t, d)
  if $d = 1$
    Move the disc from pillar f to pillar t
  else
    Move the top $d - 1$ discs from pillar f to pillar v by utilizing pillar t
    Move the last disc from pillar f to pillar t
    Move the $d - 1$ discs from pillar v to pillar t by utilizing pillar f

Hanoi(f, v, t, d)
  if $d = 1$
    Move the disc from pillar f to pillar t
  else
    Hanoi(f, t, v, $d - 1$)
    Move the last disc from pillar f to pillar t
    Hanoi(v, f, t, $d - 1$)

Problem 4: insertion sort

- Let's watch a **video**!
- Idea: Given a size-n integer array A whose first k elements are sorted (from small to large) and the remaining is not, insert the $(k+1)$th element into the first half to make the first $k+1$ element sorted.

```
insertionSort(a non-repetitive array A, the array length n)
    for i from 1 to n
        insert A_j to the proper place within A_{1..(i−1)} // how?
        // now A_{1..i} is sorted
```

- Please refine the pseudocode to make it **precise**.
  - Make you (and your friend) know how to execute each step.

Problem 4: insertion sort

insertionSort(a non-repetitive array A, the array length n)
    for i from 1 to n
        // $A_{1..(i-1)}$ is sorted
        insert A_j to the proper place within $A_{1..(i-1)}$
        // now $A_{1..i}$ is sorted

insertionSort(a non-repetitive array A, the array length n)
    for i from 1 to n
        // $A_{1..(i-1)}$ is sorted
        for j from i to 1
            if $A_j < A_{j-1}$
                swap A_j and A_{j-1} // precise?
            else
                break
        // now $A_{1..i}$ is sorted