CS-521-900, Assignment 3

Yiyun Zhang

August 17, 2020

1. During exam week.

2.
```
int i = 0
function constructTree(A, bound) {
  if(i >= n || A[i] >= bound) return null
  root = A[i++]
  root.left = constructTree(A, root.value)
  root.right = constructTree(A,bound)
  return root
}
constructTree(Array, +infinity)
```
   Loop invariant: $A[i]$ is less than bound.
   Initializaation: Before the recursion starts, $A[i]$ is $A[0]$ which is the first element of the array A, it is also the root of the BST. The bound is $+\infty$, therefore $A[i]$ is less than bound so it is true.
   Maintenance: For each recursion, $A[j]$'s left child is less than $A[j]$ because the bound is $A[j]$'s value, $A[j]$'s right child is larger than $A[j]$ because the bound is $+\infty$, it is also true for $j + 1$, so it is true.
   Termination: When the recursion stops, $A[n]$ is the last element of the array A, which is also the last element of the BST. The bound is $+\infty$, therefore $A[n]$ is less than bound so it is true.
   Running time: For function $contructTree()$, the recurrance relation is $T(n) = 2T(n) + c$, therefore the running time complexity is $O(n)$.

3.
```
function constructTree(A, i) {
  if(i == 0) return null
  root = A[i/2]
  root.left = constructTree(A, i/2)
  root.right = constructTree(A, 3i/2)
  return root
}
constructTree(Array, n)
```
   Loop invariant: $A[i]$ is larger than its left child and smaller than its right child.
   Initializaation: Before the recursion starts, $A[i]$ is $A[n]$ which is the last

element of the array A. Since there is no BST yet, so the statement is true.
Maintenance: For each recursion, $A[j]$'s left child is the $A[j/2]$ and the right child is $A[3j/2]$, since the array A is a sorted array therfore $A[j]$ is always larger than $A[j/2]$ and smaller than $A[3j/2]$. It is also true for $j+1$, so it is true.
Termination: When the recursion stops, $A[0]$ is the first element of the array A. Since it is the smallest number in the array and it has no children, therfore the statement is true.
Running time: For function $contructTree()$, the recurrance relation is $T(n) = 2T(n/2) + c$, therefore the running time complexity is $O(n)$.

4. For a directed graph $G = (V, E)$, computing out-degrees and in-degrees of every vertex both take $O(|E| + |V|)$ time, because we have to go through all adjacency lists to count the arcs incident from or directed away.

5. Based on $Theorem 22.10$: In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge. Therefore, if there is no back edge, there is only tree edge which means the undirected graph is acyclic. Thus, if there is a back edge exists, there is a circle contains in the given undirected graph G.

```
function hasCycle(G) {
  for each vertex u in G.V
    u.color = WHITE
    visited = NIL
  time = 0
  for each vertex u in G.V
    if u.color == WHITE
      if hasCycle-VISIT(G, u)
        return true
  return false
}
function hasCycle-VISIT(G, u) {
  time = time + 1
  u.d = time
  u.color = GRAY
  for each v in G.Adj[u]
    if (u, v) not in visited
      visited.add((u, v))
      visited.add((v, u))
      if v.color == GRAY
        return true
      if v.color == WHITE
        if hasCycle-VISIT(G, v)
          return true
```

```
  u.color = BLACK
  time = time + 1
  u.f = time
  return false
}
hasCycle(G)
```

According to the book: GRAY: indicates a back edge. (u, v) and (v, u) are really the same edge in an undirected graph.
Loop invariant: unvisited vertex's color is white Initializaation: Before the recursion starts, all vertex has no color so the statement is true. Maintenance: For each recursion, current vertex's color is set to black, it is also true for next recursion. So the statement is true. Termination: When the recursion stops, all vertex has been visited, if there is back edge exists, those vertex are grey, others are black. Running time: If there is no back edge, there will be maximum $|V| - 1$ distinct edges in the graph, therefore it is the upper bound before a back edge is found. Thus, the running time is $O(|V|)$