Cairo University
Faculty of Engineering
Computer Engineering
Fall 2019

Design and Analysis of Algorithms
Sheet 2

1. Sort the following functions in increasing order of growth:
   a. $\lg\sqrt{n}$
   b. $\lg^2 n$
   c. $n \lg n$
   d. n
   e. $\lg n$
   f. $\sqrt{n}$
   g. n^2
   h. $\lg\lg n$
   i. $\sqrt{n}\lg n$

2. Solve the recurrences:
   a. $T(n) = 3T\left(\frac{n}{2}\right) + n$
   b. $T(n) = 4T\left(\frac{n}{2}\right) + n\lg n$
   c. $T(n) = 8T\left(\frac{n}{3}\right) + n^2$
   d. $T(n) = 8T\left(\frac{n}{2}\right) + n^3$
   e. $T(n) = T\left(\frac{3n}{4}\right) + T\left(\frac{n}{4}\right) + n$
   f. $T(n) = T\left(\frac{n}{2}\right) + 2T\left(\frac{n}{4}\right) + 1$
   g. $T(n) = T(n-1) + T\left(\sqrt{n}\right) + n$
   h. $T(n) = 2T(n-1) + 1$ (Tower of Hanoi)

3. For each of the functions below:

   a. Trace the execution of the function when called with $n = 4$. In text, describe briefly what it performs.

   b. Give a recurrence relation that describes the running time of the function.

   c. Solve the recurrence.

   i.
```
int recursive1(int n)
{
   if (n == 0)
      return 0;
   else
      return 1 + recursive1(n-1);
}
```

   ii.
```
int recursive2(int n)
{
   if (n == 0)
      return 0;
   return recursive1(n) + recursive2(n-1);
}
```

   Note that this function calls the function from part 3.i.

   iii.
```
int recursive3(int n)
{
   if (n == 0)
      return 1;
   else
      return recursive3(n-1) + recursive3(n-1);
}
```

   iv.
```
int recursive4(int n)
{
   if (n == 0)
      return 1;
   else
      return 2 * recursive4(n-1);
}
```

4. Implement an algorithm to find the kth to last element in a singly linked list.

5. Implement an algorithm to delete a node in the middle of a singly linked list given only a pointer to this node.

6. Implement an algorithm to partition a linked list around a given value x, such that all nodes less than x come before all nodes greater than or equal to x.

7. Adapt one of the studied sorting algorithms to sort a linked-list in the best possible time complexity. What is the time complexity?

8. Given an UNSORTED array of (distinct) integers and given also a variable n, find ALL pairs of values in the array that sum to n.

   Example:

   Input: $[7\ 2\ 5\ 8\ 1\ 3\ 0]$ and $n = 5$

   Output: $\{(2,3), (5,0)\}$

   a. Implement an algorithm to solve the problem WITHOUT using hash tables. What are the time and space complexities?
   b. Implement an algorithm to solve the problem using hash tables. What are the time and space complexities?