Data Structures and Algorithms – (COMP SCI 2C03)
Winter 2021
Tutorial-I

January 25, 2021

Notes for the tutorial: When writing code involving linked lists, we must always be careful to properly handle the exceptional cases (when the linked list is empty, when the list has only one or two nodes) and the boundary cases (dealing with the first or last items). This is usually much trickier than handling the normal cases.

Question 1 Write an algorithm that traverses a linked list to print the data value in each node, in linear time in the length of the linked list.

procedure $List\_Traverse(L)$
  $x = L.head$
  while $x \neq NIL$ **do**
    Print $x.value$ ▷ Here we assume that value is the identifier for the data item in the linked list
    $x = x.next$

Question 2 Write an algorithm that inserts a node with data value = 10 in a double ordered linked list L, containing the data values $1, 3, 6, 9, 11, 15, 21, 32, 45$ in linear time in the length of the linked list.

procedure $List\_Insert(L)$
  $x = L.head$
  while $x \leq 10$ **do**
    $x = x.next$
  $y = newnode$
  $y.next = x$
  $y.prev = x.prev$

$x.prev = y$
$y.prev.next = y$

Question 3 9.[15] Using ONLY the definition of $O(f(n))$ prove that the following statements are TRUE:

(a) $(7000n^3 + 3n + 2)/n = O(n^3)$ - choose $c = 7005$ and $n_0 = 1$

(b) $O(n^5 + n^4 \log n) = O(n^5)$ - choose $c = 2$ and $n_0 = 1$

Question 4 Using ONLY the definition of $O(f(n))$ prove that the following statements are FALSE:

(a) $n2^n + 3n + 2 = O(2^n)$ - The proof is by contradiction. Let c, n_0 be the least constants, such that the below inequality holds for all $n \geq n_0$

$$n2^n + 3n + 2 \leq c \cdot 2^n.$$

However, for all $n \geq c+1$, $n2^n + 3n + 2 > c \cdot 2^n$ – a contradiction.

(b) $n^{1.5} = O(n)$ - The proof is by contradiction. Let c, n_0 be the least constants, such that the below inequality holds for all $n \geq n_0$

$$n^{1.5} \leq c \cdot n.$$

However, for all $n \geq c^4$, $n^{1.5} > c \cdot n$ – a contradiction.

Question 5 Write an implementation of the operations *Push* and *Pop* for a Stack data structure using a singly linked list. The solution follows the style of CLRS.

procedure $Push(S, x)$
    if !$S.IsEmpty$ **then**
        $x.next = S.head$
    $S.head = x$
procedure $Pop(S)$
    if !$S.IsEmpty$ **then**
        $x = S.head$
        $S.head = S.head.next$
        return x
    else
        return NIL

Question 6 Write an implementation of the operations *Enqueue* and *Dequeue* for a Queue data structure using a singly linked list. The solution follows the style of CLRS.

procedure $Enqueue(Q, x)$
    if $!Q.IsEmpty$ **then**
        $Q.tail.next = x$
    $Q.tail = x$
procedure $Dequeue(Q)$
    if $!Q.IsEmpty$ **then**
        $x = Q.head$
        $Q.head = Q.head.next$
        return x
    else
        return NIL

Question 7 Compute the running time function $T(n)$ for the below algorithm. Your solution should compute the frequencies as discussed in class to compute $T(n)$. Also, provide the big-Oh estimation for $T(n)$, and explain your answer.

```
int sum = 0;
for (int k = n; k > 0; k /= 2)
  for (int i = 0; i < k; i++)
     sum++;
```

Frequency of running sum++ in the inner loop:
$1st$ iteration $= n$
$2nd$ iteration $= n/2$
$3rd$ iteration $= n/4$
...
$n'th$ iteration $= 1$
Sum: $n + \frac{n}{2} + \frac{n}{4} + ... + 1 = 2n - 1$