Data Structures - 67109
Exercise 2

Due: 27/03/2019

Question 1

Let $T : \mathbb{N} \to \mathbb{R}^+$and write an expression for the tightest upper asymptotic bound of $T(n)$ you can find (i.e. a function $g : \mathbb{N} \to \mathbb{R}^+$ such that $T(n) = O(g(n))$) and prove its correctness by induction for the following cases. Assume for all of these $T(1) = 1$:

1. $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$
2. $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$
3. Explain the logic for the correctness of the bound, without a formal proof: $T(n) = 2T\left(\left\lfloor \frac{n}{5} \right\rfloor\right) + 3T\left(\left\lfloor \frac{n}{10} \right\rfloor\right) + n$ (Hint: use the fact that $\sum_{i=0}^{\infty} x^i$ converges for $|x| < 1$)

Question 2

Find Θ bounds for the following recurrence relations (in any way you'd like), prove your answers:

1. $T(n) = 2T\left(\lfloor n/4 \rfloor\right) + \sqrt{n}$
2. $T(n) = 2T\left(\lfloor n/4 \rfloor\right) + n^{0.51}$
3. $T(n) = \sqrt{2}T(\lfloor n/2 \rfloor) + \log n$
4. $T(n) = 16T\left(\lfloor n/4 \rfloor\right) + n!$
5. $T(n) = 14T\left(\lfloor n/2 \rfloor\right) + 50n^3 + 4n^2 - 1$

Question 3

Let $f, g : \mathbb{N} \to \mathbb{R}_{>0}$. Prove or give a counterexample:

1. $10log(n) + 10 \in \Theta(log(n))$
2. Assume $f \in O(g)$, then there exists $c > 0$ s.t. $\forall n \in \mathbb{N}\ f(n) \leq c \cdot g(n)$
3. Assume $f \in o(g)$, then $f \notin \Theta(g)$

Question 4

Write down the recurrence relations to each one of the following algorithms. Find and prove (in any way you'd like) Θ bounds for the Min algorithm and O bound for the Fib algorithm:

Algorithm 1 Min(A[1:n])

1. If n==1: return A[1]
2. return min{A[1], Min(A[2:n])}

Algorithm 2 Max(A[1:n])

1. If n==1: return A[1]
2. return max{Max(A[1:$\frac{n}{2}$]), Max(A[$\frac{n}{2}$:n])}

Algorithm 3 Fib(n)

1. If n==1 or n==2: return 1
2. return Fib(n-1)+Fib(n-2)

Algorithm 4 Power1(x,n)

1. If n==0: return 1
2. return x·Power1(x,n-1)

Algorithm 5 Power2(x,n)

1. If n==0: return 1
2. If n%2==0:
   - return Power2(x^2, $\frac{n}{2}$)
3. else:
   - return x·Power2(x^2, $\frac{n-1}{2}$)

Question 5

Recall the MergeSort you have seen in class:

Algorithm 6 MergeSort(arr)

1. if arr.length == 1 return arr
2. m ← $\lfloor (arr.length - 1)/2 \rfloor$
3. first half ← MergeSort(arr[0 : m])
4. second half ← MergeSort(arr[m+1 : arr.length])
5. return Merge(first half, second half)

Algorithm 7 Merge(arr1, arr2)

1. result ← new array of size (arr1.length + arr2.length) initialized with zeros
2. i ← 0, j ← 0
3. while i < arr1.length or j < arr2.length
   (a) if i==arr1.length
      - result[i+j] ← arr2[j]
      - j ← j+1

   (b) else if j==arr2.length or arr1[i] < arr2[j]
      - result[i+j] ← arr1[i]
      - i ← i+1

   (c) else
      - result[i+j] ← arr2[j]
      - j ← j+1
4. return result

Prove its correctness. When proving the correctness of the Merge(arr1, arr2) routine - define and use a loop invariant.