CSE 373: Homework 2

Algorithm Analysis

Out October 11th, 2017

Due October 18th, 2017, 11:30 pm

1 bigO notation

For each of the following, show that $f(n) \in O(g(n))$ by finding the values c, n_0 and demonstrating that the definition inequality holds.

1. $f(n) = 14n^3 + 3n^2$ $g(n) = n^3$

2. $f(n) = n * (4 + log(n))$ $g(n) = n^2$

2 Asymptotic Analysis

For the following methods, determine a tight bigO runtime bound in terms of n. An exact proof is not required, but some work must be shown.

1.
```
void problem1(int n) {
      int x = 0;
      for (int i = n/2 ; i < n; i++) {
           if(i % 5 == 0) break;
           for(int j = 1; j < n; j += 2) {
                x++;
                x*=2;
           }
      }
}
```

2.
```
void problem2(int n){
      for(int i = n; i>=0; i--){
           problemHelper(i);
      }
private void problemHelper(int n){
      if(n > 2) {
           problemHelper(n/2);
           problemHelper(n/2);
      }
}
```

3 Recurrences

Given the following recurrences, find the tight bigO bound for the function in terms of n. Note that the base cases are non-constant and the master theorem does not apply. Here, N refers to the initial call size of the recurrence. Show your work

1. $T(1) = N$
   $T(n) = 1 + 2 * T(n/2)$

2. $T(1) = log(N)$
   $T(n) = n + T(n - 1)$

4 Amortized Analysis

Consider the following function which adds to an array of Strings called data. You may assume the array is always non-null.

```
void insert(String t):
     if the array is not full:
          insert the element t into the next available position
     else:
          create an array of twice the size
          copy the old data into the new array
          insert the element to the end
          sort the array of elements (assume an O(n log n) sort)
          point "data" at the new array
```

1. Provide the best-case runtime for `insert(String t)`. No work needed

2. Provide the worst-case runtime for `insert(String t)`. No work needed.

3. Provide the amortized analysis for `insert(String t)`. Show your work
   Hint: Consider a sequence of n calls when the array is already full and has n elements

5 Algorithm Design

Design an algorithm which, given an array of integers, returns the two elements in the array that are numerically the closest together. If there is a tie, you may return any pair. Provide pseudocode for your algorithm and show the tight bound for runtime and memory usage where n is the size of the array.