CS 261 HW7

Prof. Jared Saia, University of New Mexico

Due April 16th

1. Imagine that Mergesort is changed so that it partitions the input list into 3 sub-lists instead of just 2, that it recursively calls Mergesort on each of these three lists and that it then calls a Merge subroutine on the 3 sorted lists to merge them into a single sorted list. Assume that this Merge subroutine works in $O(n)$ time where n is the total number of elements in all 3 lists. Write and solve (using recursion trees) a recurrence relation for the run time of this new version of Mergesort.

2. Consider the recurrence $T(n) = 5T(n/2) + n$. Use the recursion tree method to solve this recurrence to within tight big-O notation.

3. Consider the following function:

```
int f (int n){
  if (n==0) return 3;
  else if (n==1) return 5;
  else{
    int val = 2*f (n-1);
    val = val - f (n-2);
    return val;
  }
}
```

   (a) Write a recurrence relation for the *value* returned by f. Solve the recurrence exactly. (Don't forget to check it)

   (b) Write a recurrence relation for the *running time* of f. Get a tight upperbound (i.e. big-O) on the solution to this recurrence.

4. A person deposits 1 dollar in an account that yields 8% interest annually. Set up a recurrence relation for the amount in the account at

the end of n years. Now find an explicit formula for the amount in the account after n years.

5. Exercise 7.1.20

6. Write a recurrence relation for the number of ways to climb n stairs if the person climbing the stairs can take either 1 stair or 3 stairs at a time. What are the initial conditions? How many ways can the person climb 8 stairs?

7. Challenge: Exercise 7.2.46