CS 325
Due Friday, 4 Nov

Homework # 5

Divide and Conquer Recurrences

For the following algorithms given in schematic form, **write down** a divide-and-conquer recurrence relation for the run time, and **solve** for the run time in Big Theta form.

(a)

```
PROCEDURE   MULT( a, b, n)
               Split a into  a0, a1, and b into b0, b1
               MULT( a0, b0, n/2):   MULT( a0, b1, n/2)
               MULT( a1, b1, n/2):   MULT( a1, b0, n/2)
               MULT( a1 + a0, b1 + b0, n/2)
               COMBINE the results of the subproblems
```

If Split and COMBINE have $\Theta(n)$ run time then MULT has runtime

---

(b)

```
FUNCTION   FOOL( X, n)
               Split X into  X1 and X2
               Y1 = FOOL(X1, n/2)
               Y2 = FOOL(X2, n/2)
               COMBINE(Y1, Y2)
```

If SPLIT and COMBINE are both $\Theta(n)$ then FOOL has runtime

---

(c)

```
FUNCTION   MESS( X, n)
           Split X into  Y1,  Y2,  Y3
           TEST( Y1, Y2)
           depending on the outcome of TEST
             DO      MESS( Y1, n/3)
                     or MESS( Y2, n/3)
                     or MESS( Y3, n/3)
```

If SPLIT takes $\Theta(n^2)$ and TEST takes $\Theta(1)$ then MESS has runtime

____

(d) If SPLIT takes $\Theta(\log n)$ and TEST takes $\Theta(1)$ then MESS from (c) has runtime

____

(e)

```
PROCEDURE   MULT( a, b, n)
               Split a into  a0, a1, and b into b0, b1
               MULT( a0, b0, n/2)
               MULT( a1, b1, n/2)
               MULT( a1 + a0, b1 + b0, n/2)
               COMBINE the results of the subproblems
```

If SPLIT has $O(1)$ and COMBINE has $\Theta(n)$ run time then MULT has runtime

____

Construction of a Tree from a Degree List

A *graph* is a set of *vertices* (singular: *vertex*) and a set of *edges*, each edge connecting two vertices together. The *degree* of a vertex is the number of edges which connect to the vertex. Said another way, the degree of a vertex is the number of vertices *adjacent* to the given vertex.

In a graph with n vertices $v_1, v_2, \cdots, v_n$, let $d_1, d_2, \cdots, d_n$ be the respective degrees of those vertices. The degree of a vertex v_i in a *connected* graph is bounded, $1 \leq d_i < n$.

A *tree* is a connected graph with n vertices and $n-1$ edges. Since each edge connects to two vertices, it easy to see that in a tree,

$$\sum_{i=1}^{n} d_i = 2(n-1)$$

because each of the $n-1$ edges contribute 2 to the sum of degrees. We want to turn this around and show how to find a tree, given a set of degrees which satisfy the above summation constraint. This leads to the following *Tree Construction Problem.*

Given a set of positive integers, $d_1, d_2, \cdots, d_n$, such that $\sum_{i=1}^{n} d_i = 2(n-1)$, construct a tree with n vertices in which $degree(v_i) = d_i$.

Problems

1. Write an inductive proof of
   If $\sum_{i=1}^{n} d_i = 2(n-1)$, where $d_1, d_2, \cdots, d_n$ are positive integers, then there is a tree with n vertices in which $degree(v_i) = d_i$.
2. What are you inducting on ? What is the inductive variable?
3. State your inductive hypothesis as a function of your inductive variable.
4. What is the BASE case?
5. Prove the BASE case.
6. Carry out the INDUCTIVE STEP of the proof.
   This may involve some algebraic manipulation.
7. Use you inductive proof to design a divide and conquer algorithm for the tree construction problem.
8. Program your algorithm and demonstrate by examples that your program produces correct output.
9. Write and solve a difference equation for the running time of your program.
10. Run your program for different problem sizes and plot the running times to show that your program behaves according to your analysis in #9. Remember to choose the appropriate graph type.

Hints

The simplest way to represent a tree is by a parent array, in which PARENT[i]=j if and only if v_j is the parent of v_i in the tree hierarchy. Of course the root, v_r, will have no parent, which can be represented by PARENT[r] = 0.

A vertex at the end of the hierarchy in a directed tree, which has no children vertices, is called a *leaf*. What can you say about the degrees of leaf vertices?