There are 115 points total (So first exam is about 20% and this is about 25%.)

1. (25 pts.) Recall that Prim’s algorithm finds a minimum spanning tree by greedily growing a tree starting with $v_1$, whereas Kruskal’s algorithm greedily adds edges in a way that avoids cycles. For the graph shown below, list the edges in the order they are chosen by each algorithm. Edges are labeled with upper case letters. (Two copies of the graph are provided so you can use them as “worksheets” if you wish to.)

(a) Prim’s algorithm: A K E R M L F B J

(b) Kruskal’s algorithm: M F A R E L K B J

2. (25 pts.) The worst-case running time for an algorithm is an increasing function of $n$ and satisfies $T(n) = 3T(n/2) + 2n$ when $n$ is a power of two. Furthermore, $T(1) = 1$. Determine the complexity class of $T(n)$.

Ans. Apply Theorem B.5 on page 492 (or Theorem B.6): $a = 3$, $b = 2$, $c = 2$, and $k = 1$. Hence $a > b^k = 2$ and so $T(n) \in \Theta(n^{\log_2 3}) = \Theta(n^{\log 3})$. 

MORE
3. (25 pts.) Problem 3.33 says "...write an algorithm to find the maximum sum in any contiguous sublist of a given list of \( n \) real numbers. Analyze your algorithm, and show the results using order notation." We present an algorithm below. **Analyze it.** You should give both **average-case** and **worst-case** complexity information.

```plaintext
MaxSum(list, \( n \))
    best = 0  // Best sum so far
    right = 0  // Best sum ending on the end right of 1\ldots i
    For \( i = 1 \) to \( n \)  // \( i \) is the right end
        right = right + list[\( i \)]  // Extend sum to the right
        If (right > best) best = right
        If (right < 0) right = 0  // Empty sum is better
    End for
End
```

**Ans.** The basic operation can be anything inside the loop, including the incrementing of \( i \) required for the loop. Since the loop is executed \( n \) times, the every-case time complexity is \( \Theta(n) \). Since this is every-case, it is also average-case and worst-case.

4. (40 pts.) Indicate whether true or false. Beware of guessing:

- correct answer +5pts.  incorrect answer --3pts.  no answer 0pts

(a) **F** Greedy algorithms are called "greedy" because they often require a lot of storage.

(b) **F** Dynamic programming algorithms usually split the problem into a few smaller problems, which are solved by recursive calls.

(c) **F** Usually it is easier to prove that a greedy algorithm is correct than it is to prove that a dynamic programming algorithm is correct.

(d) **F** If we find a good dynamic programming algorithm for a problem, there will probably not be a good greedy algorithm.

(e) **T** The "principle of optimality" is a good method for proving that a dynamic programming algorithm is correct.

(f) **T** A dynamic programming approach is better than a divide and conquer approach for solving a recursion such as \( S(n, k) = S(n-1, k) + (k-1)S(n-1, k-1) \). (If \( k = 1 \) or \( n = k \), then \( S(n, k) = 1 \).)

(g) **T** Kruskal’s algorithm is better than Prim’s when the graph has relatively few edges.

(h) **F** A greedy algorithm for the 0-1 Knapsack Problem is at least as good as a dynamic programming algorithm.

END