There are 210 points possible.

1. (30 pts.) Recall that Dijkstra’s algorithm finds shortest paths from $v_1$ to all other vertices by adding edges linking in the closest points. In the graph shown below, each edge is bidirectional; that is, you can travel in either direction on it. **Edges are labeled with upper case letters.** (Two copies of the graph are provided so you can use one as a “worksheet” if you wish.)

   (a) List edges in order chosen by algorithm: ____________________________

   (b) At each vertex, give the length of the shortest path from $v_1$ to the vertex. *Indicate which graph has your answer.*

   ![Graph](image)

2. (25 pts.) Consider the following eight complexity categories (remember $\lg = \log_2$):

   $\Theta(n) \quad \Theta(n^2) \quad \Theta(2^n) \quad \Theta(3^{\lg n}) \quad \Theta(n^{\lg n}) \quad \Theta(n \lg n) \quad \Theta((\sqrt{n} + \ln n)^2) \quad \Theta(2^{n + \lg n})$.

   (a) Which are equal?

   (b) Arrange the distinct classes in order from slowest growing to fastest growing. In other words, if $\Theta(f(n))$ is to the left of $\Theta(g(n))$, then $f(n) \in o(g(n))$. 
3. (30 pts.) The average running time for an algorithm is a nondecreasing function of $n$ and satisfies $T(4n) = T(2n) + 2T(n)$ for all $n > 0$. Furthermore, $T(1) = 1$ and $T(2) = 3$.

(a) Determine $T(2^k)$ as a function of the integer $k$.

Hint: Set $t_k = T(2^k)$.

(b) Determine the complexity class of $T(n)$.

4. (30 pts.) Suppose we have two sorted lists $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, both of length $n$, that we want to merge to obtain a sorted list of length $2n$, say $c_1, \ldots, c_{2n}$. To do this, we must decide where the $a_i$’s fit among the $b_j$’s to produce the $c$ list. The number of choices for this is $\binom{2n}{n} \geq 4^n/(2n^{1/2})$.

Suppose the merge is done comparisons of keys. Using the above information, derive a lower bound for the worst case number of key comparisons that are needed. Explain your reasoning; don’t just give an answer.
5. (30 pts.) Here is an informal description of a routine \texttt{Proc} that is looking for \( x \) in a sorted list \( S \). The parameters are the ends of the list. While it is looking it does some processing in \texttt{ProcLow} and \texttt{ProcHigh}.

\begin{verbatim}
Proc(lo, hi)
    If lo > hi we are done.
    k = [(lo + hi)/2].
    If S[k] = x, we are done.
    If S[k] < x
        Call ProcHigh(k, hi) andProc(k + 1, hi)
    Else
        Call ProcLow(lo, k) andProc(lo, k - 1)
    Endif.
End
\end{verbatim}

We begin by calling \texttt{Proc(1,n)}. Most of the time is spent in \texttt{ProcLow} and \texttt{ProcHigh}. In fact, \texttt{ProcLow(a,b)} requires \( \log(b - a + 1) \) basic operations and \texttt{ProcHigh(a,b)} requires \( (b - a + 1) \) basic operations. (You do not need to know what any of this code is supposed to do.)

(a) Let \( W(n) \) be the worst case running time for \texttt{Proc(1,n)}. Give a recursion and initial condition for \( W(2^n) \). (In the worst case, \( x \) is not in the list.)

(b) Let \( A(n) \) be the average running time for \texttt{Proc(1, n)}. Assuming \( x \) is not in the list and the probability that \( S[k] < x \) is \( 1/2 \), give a recursion for \( A(n) \). You need not give an initial condition.
6. (65 pts.) Indicate whether true or false. Beware of guessing:

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

___ $\Theta(2^{n+2}) = \Theta(2^n)$.

___ $\Theta((n + 2)^2) = \Theta(n^2)$.

___ $\Theta(2^{n+\log n}) = \Theta(2^n)$.

___ $\Theta((n + \log n)^2) = \Theta(n^2)$.

___ Greedy algorithms are called “greedy” because they make the best choice at the present time, without concern for the future.

___ Dynamic programming algorithms use a bottom up approach.

___ Divide and conquer algorithms use a bottom up approach.

___ If a divide and conquer algorithm requires recomputing the same quantity many times, it is a good idea to look for a dynamic programming algorithm.

___ No greedy algorithm is known for the 0-1 Knapsack Problem.

___ It is usually fairly easy to determine average and worst-case time complexities for backtracking algorithms.

___ There is a search algorithm that uses comparison of keys and is significantly faster on average and in the worst case than binary search.

___ There is a sorting algorithm that uses comparison of keys and is significantly faster on average than and in the worst case than mergesort.

___ Quicksort has a good average run time and a poor worst-case run time.