1. This question concerns Dijkstra’s shortest (i.e. least cost) paths algorithm. (See section 4.4).

   (a) Construct an example of an edge weighted directed graph having no negative cycles (but having some negative weight edges) for which Dijkstra’s algorithm will not compute the optimal set of paths.

   (b) Consider the following alternative definition of the “cost” of a path $\pi$. Define $\text{cost}(\pi) = \max_{e \in \pi} \ell_e$. Show how to modify Dijkstra’s algorithm so as to compute the least cost paths from $s$ to all other vertices. Briefly justify why your algorithm correctly computes the optimal paths. (It is sufficient, for example, to set up an appropriate induction.)

   (c) Consider the (usual) least cost paths and MST problems. Construct an example of an undirected graph with positive edge weights where the optimal solutions for these problems differ.

   SOLUTION: This problem wasn’t phrased as carefully as it should have been. Any reasonable answer will be accepted. A more precise question would be: “Construct an example of an undirected graph with positive edge weights where the optimal solution for the MST is unique and where no matter what vertex is chosen as a start vertex, the least cost paths tree (when viewed as an undirected tree) is different from the MST tree.” A good example is the following graph: Let $G = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{\text{all edges } (u, v) \text{ except edge } (1, 4)\}$. Think of the graph being layed out as a rectangle with one diagonal edge.

   \[ 
   \begin{array}{c}
   1------2 \\
   |     | \\
   |     | \\
   |     | \\
   |     | \\
   3 ---- 4
   \end{array} 
   \]

   Let edges (1,2), (2,3), and (3,4) have cost 2; edges (1,3) and (2,4) have cost 3. The unique MST is the zig-zag path $Z$ formed by cost 2 edges. But since every node $s$ is adjacent to a cost 3 edge, the min cost tree starting at $s$ will contain a cost 3 edge.

2. Consider the following one machine flexible intervals scheduling problem. An interval $I_j$ is now a triple $s_j, f_j, k_j$ where $s_j$ and $f_j$ are (as before) the starting and finishing times and $k_j$ is the “client” to which the interval belongs. A feasible schedule now is one where (as before) intervals do not intersect and for every client, at most one interval is scheduled.
NOTE: This problem seemed to have caused the most concern and hence I am providing a complete solution.

(a) Describe a greedy algorithm $ALG$ for the one machine flexible interval scheduling problem which is a $\frac{1}{2}$ approximation algorithm. That is, for every input set $\mathcal{I}$, $|ALG(\mathcal{I})| \geq \frac{1}{2}|OPT(\mathcal{I})|$. 

SOLUTION: An adaption of the interval selection EFT algorithm can be used. Namely, we have:
Sort the input intervals so that $f_1 \leq f_2 \ldots \leq f_n$; Initialize $S := \emptyset$
For $j = 1, \ldots, n$
    If $I_j$ does not “conflict” with any interval in $S$ then $S := S \cup I_j$
End If
End For

Now the meaning of “conflicts” is that $I_j$ does not intersect any previously selected interval and additionally does not have the same client number of any previously selected interval.

(b) Sketch a charging argument to prove that your algorithm achieves a $\frac{1}{2}$ approximation.

Hint: The charging argument should consider clients in OPT-ALG.

SOLUTION: As indicated in the hint, we can define the profit of an algorithm to be the number of different clients selected (which is the same as the number of different intervals selected by the definition of the problem). Now let OPT-EFT denote the clients selected by OPT but not by EFT and let EFT be the clients selected by EFT. Now we want to define a 1-1 function $h$ from OPT-EFT into EFT. This will show that $|OPT-EFT| \leq |EFT|$ and hence that $|OPT| \leq 2|EFT|$ as desired. Let say that client $K$ is in OPT-EFT and it is scheduled by some interval $I = (s, f, K)$. Let $I'$ be the leftmost interval scheduled by EFT that intersects $I$. Then $h(K) = \text{the client of } I'$. We need to prove two things: 1) $h$ is a function and 2) $h$ is 1-1.

1) $h$ is a function since some interval $I'$ in the EFT schedule must intersect $I$ or else EFT would have selected interval $I$ contradicting the fact the $K$ is not in EFT (since $K$ was assumed to be not in EFT and hence $I$ would not be conflicting a previously selected interval). Hence $I'$ is well defined and then so is its client.

2) $h$ is 1-1. I claim that the finishing time $f'$ of $I'$ must satisfy $f' \leq f$. If not, then once again, EFT would have scheduled $I$. Now just as we observed in the interval selection problem we cannot have two intervals scheduled by OPT that can intersect an interval in EFT that ends earlier.

3. This question concerns Huffman coding.

(a) Let $p_a = p_b = p_c = p_d = 1/8; p_e = p_f = 1/4$. Construct an optimal prefix code.

(b) Let $p_a = p_b = 1/6; p_c = p_d = 1/3$. Construct an optimal prefix code.
(c) Suppose we have a set of symbols $a_1, \ldots, a_n$ where $p(a_i) < 1/3$ for all $i$ and $\sum_{1 \leq i \leq n} p(a_i) = 1$. Sketch an argument to show that the Huffman tree cannot have a code word of length 1.

4. Modify the argument in section 11.1 to show that the makespan greedy algorithm achieves an approximation ratio of $2 - \frac{1}{m}$ where $m$ is the number of machines. That is show for every input set $\mathcal{I}, GREEDY(\mathcal{I}) \leq (2 - \frac{1}{m}) \cdot OPT(\mathcal{I})$.

SOLUTION: Here GREEDY and OPT refer respectively to the makespan cost for these solutions. Let $p_j$ be the processing time of the job that causes the makespan bound and say that this job is scheduled on machine $M_k$. Then we can say that the final makespan $= p_j + \text{load } L$ on machine $k$ just before scheduling $p_j$. That is, the makespan cost for the online greedy algorithm is $p_j + L$.

$L \leq (\sum_{i \neq j} p_i)/m$ since the right hand side is the average load/machine for all jobs other than the $j^{th}$ job.

As stated in class OPT $\geq p_j + \text{average load} = (\sum_i p_i)/m$. And we also know that OPT $\geq \max_i p_i \geq p_j$.

Putting this all together, GRREDY $= L + p_j \leq \sum_{i \neq j} /m + p_j = (\sum_{1 \leq i \leq n} p_i)/m - (1/m)p_j + p_j \leq OPT + (1 - 1/m)OPT = (2 - 1/m)OPT.$