University of Nevada, Las Vegas Computer Science 477/677 Spring 2022

Answers to Assignment 5: Due Wednesday April 6, 2022

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet. Please follow Mr. Nicholas Heerdt's instructions on how to submit your completed assignment.

1. Each answer is either *divide and conquer*, *dynamic programming* or *greedy.*

   (a) Binary search is **divide and conquer**.

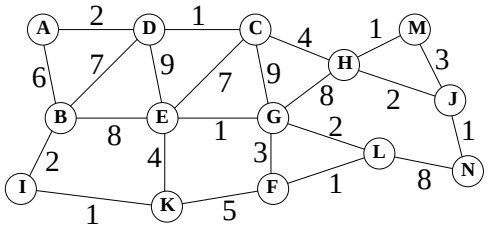
   (b) Huffman's algorithm is **greedy.**

   (c) Kruskal's algorithm is **greedy.**

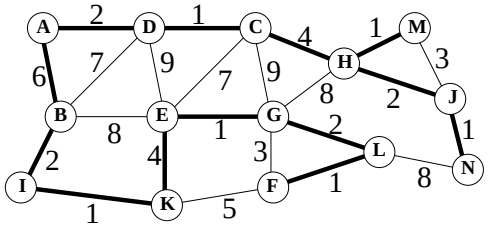
   (d) The Bellman-Ford algorithm is **dynamic programming.**

   (e) Mergesort is **divide and conquer.**

2. Walk through Kruskal's algorithm to find the minimum spanning tree of the weighted graph shown below.



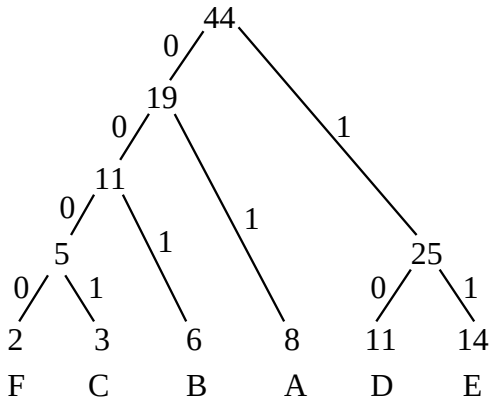
Kruskal's algorithm builds a spanning tree for a weighted connected graph one edge at a time. Start with all vertices and no edges. At each step, add the minimum weight remaining edge which does not create a cycle. When the graph is connected, you're done.



This is the solution. But how would you write the program? It is easy to eyball the correct choice at each step for this example. But what if there were thousands of vertices and tens of thousands of edges? For that, we would have to use something more sophisticated, such as the union/find structure.

3. Find an optimal Huffman code on the alphabet A,B,C,D,E,F where frequencies are given in the following table.

A	8	01
B	6	001
C	3	0001
D	11	10
E	14	11
F	2	0000



4. I did almost the same problem as this one on the board, so (almost) all you have to do is remember what I did and write it down.

   Consider the function f(n) computed by the code below.

```
int f(int n)
{
 // input condition: n >= 0
 if(n<7) return 1;
 else return f(n/2)+f(n/2+1)+f(n/2+2)+f(n/2+3)+n*n;
}
```

   The function can be computed by recursion, as given in the C++ code above. However, we could also compute f(n) using dynamic programming or memoization.

   (a) What is the asymptotic value of f(n)? (The value itself, not the time to compute it.) Write the recurrence.

   $f(n) = 4f(n/2) + n^2$, $f(n) = \Theta(n^2 \log n)$ by the Master Theorem.

   (b) What is the asymptotic time complexity of the recursive computation of f(n)? Write the recurrence. (You should be able to solve this problem using one of the theorems we've covered, but if you can't, try programming it for various values of n.)

   $T(n) = 4T(n/2) + \Theta(1)$, $T(n) = \Theta(n^2)$ by the Master Theorem.

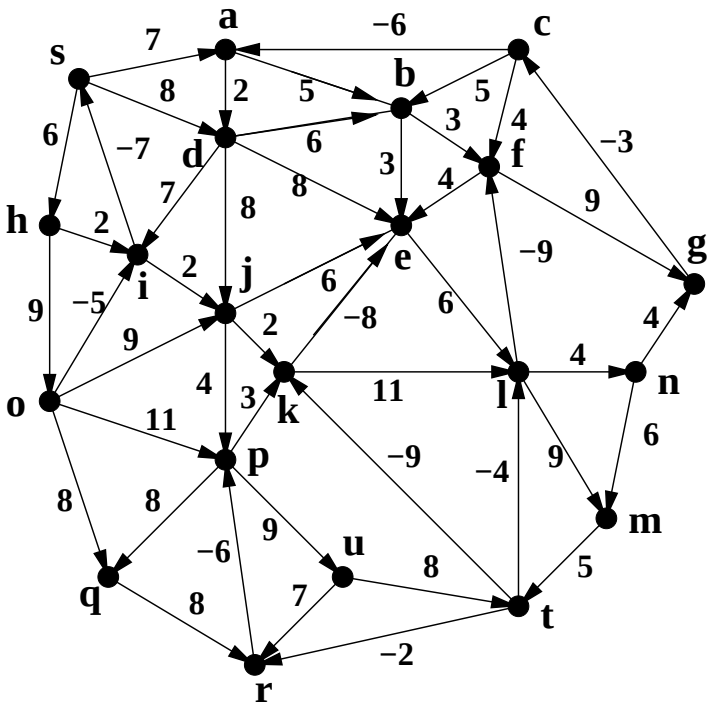
   (c) What is the asymptotic time complexity of the dynamic programming computation of f(n)? (There is no excuse for not being able to figure this out without writing a program.)

   $\Theta(n)$

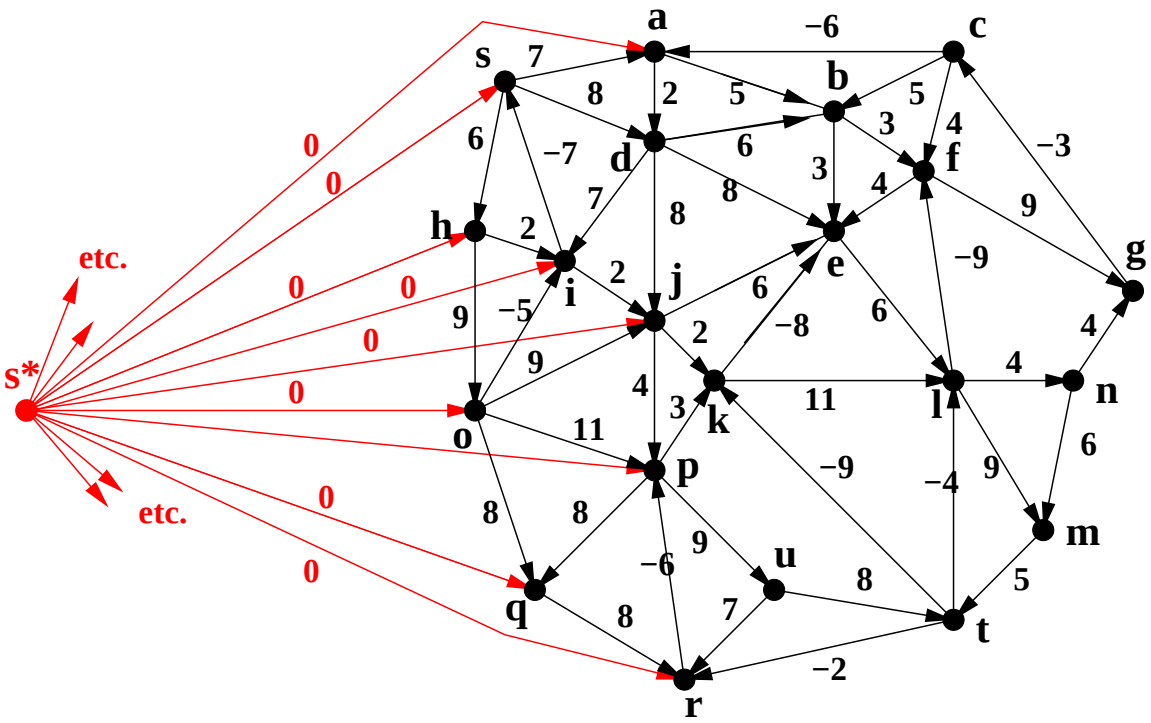
   (d) What is the asymptotic time complexity of the computation of f(n) using memoization? (This is harder than the others. If you write a program and try various values of n, you need a range of large values, like 1024 and up, to get the picture. Remember: memoization uses a sparse array structure.)

   Just as in class, the answer is $\Theta(\log n)$.

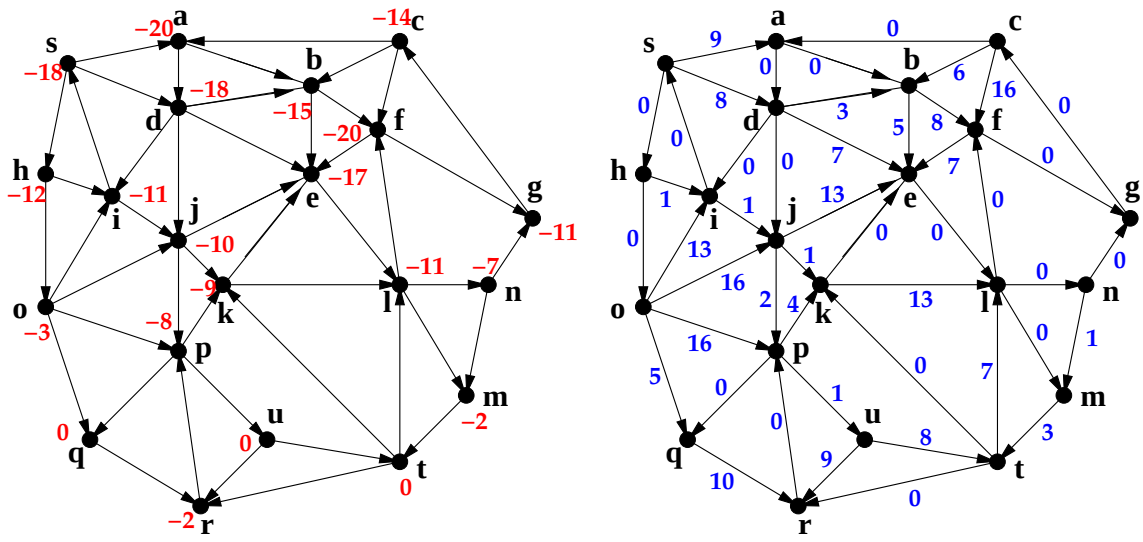
5. Consider Johnson's algorithm on the weighted directed graph G shown below.



The first step is the augment the graph by adding an arc of weight 0 to every vertex from a new source vertex, s^*, as shown below.



Using the Bellman-Ford algorithm to compute h, label every vertex v with $h(v)$, and then label every arc of G with the non-negative adjusted weight.



What additional steps would you take to finish Johnson's algorithm for G? (Do not do those steps.)

Run Dijkstra's algorithm n times, once for each vertex being the source.