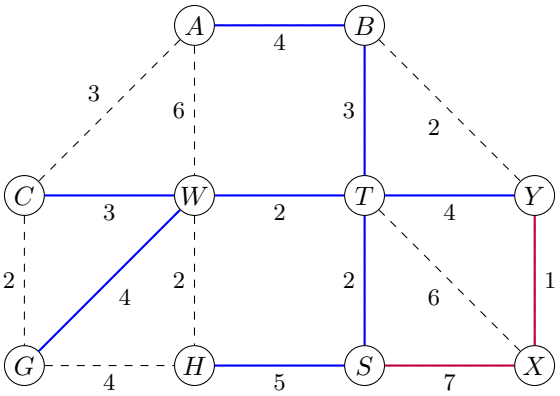
1. (a) (1 points) In the following graph, clearly fill in the edges that are in the single-source shortest path tree starting at s, such as one that would be output by Dijkstra's Algorithm. You do not need to show all of your work, but it is recommended that you keep your work reasonably well organized in the chart provided.



All blue edges plus exactly one purple edge

(b) (0.5 points) For the above graph, is the single-source shortest path tree rooted at s unique? Why or why not?

No. There are two equally short paths from S to X: (S, X) or $S \to T \to Y \to X$.

2. (1.5 points) Suppose we have a hash table of size $m = 13$, implemented with quadratic probing (defined here to mean that the ith choice is $(h(k) + i^2)\%m$, with the initial choice being $i = 0$) and with a hash function of $h(k) = k\%m$. The table stores unsigned values. Insert the following values into the hash table in the order they are given. Do not resize the hash table, regardless of load factor. The values to insert are: $8, 21, 34, 18, 5, 9, 25$ (in the order listed).

(a) Mark the entries of the following table with the values. Leave blank any spot that will be blank.

0	1	2	3	4	5	6	7	8	9	10	11	12
25					18	5		8	21	9		34

(b) If I do not resize the table, there is a value not currently in the table that I cannot insert into the table. Give the hash value of it. Your answer should be a value in the range $[0, 12]$, inclusive.

9. Note that $9, 9+1, 9+4, 9+16, 9+25, 9+36, 9+49, \ldots$ are all taken. A hazard of quadratic probing!

3. (1 point) Let G be a simple, undirected, connected graph with positive distinct integer edge weights. Suppose we want to find a minimum spanning tree of G. However, suppose we also know that $m = n+2$ for this graph. That is, there are two more edges than there are vertices. Give a *linear time* algorithm to find a minimum spanning tree of G. Briefly describe why it is linear time.

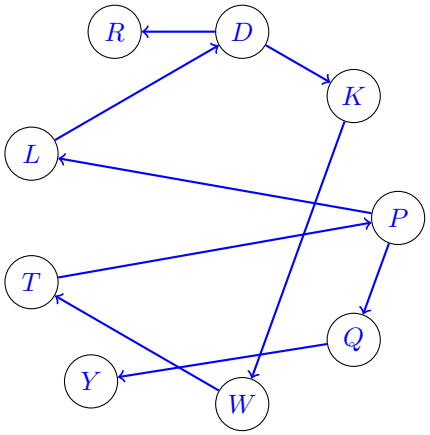
Note: the algorithms in lecture and the reading all take time $\Omega(n)$ but none are $\mathcal{O}(n)$.

We can actually use reverse-delete here. In $\mathcal{O}(n)$ time, we can find a cycle, find the max weight edge in the cycle, and delete it. In a general graph, we might need to do this $\mathcal{O}(n^2)$ times, which is why this is not an efficient algorithm in general. However, because we know $m = n + 2$, we know we need only do this three times to get a tree, so this is linear time.

We can't, however, run the algorithms from class directly without explicitly stating how we are modifying them, as they take superlinear time for general graphs.

4. (1 point) Consider the following graph. As drawn, this is an undirected graph. Re-draw the graph on your answer sheet as a *directed* graph in such a way that each vertex has *at most* one incoming edge. Make sure the direction of your edges is clear!

   Direct the cycle's edges, then direct every remaining edge away from the cycle. The key observation is that there is exactly one cycle. I mentioned this aspect in two lectures. The following is one answer; it would also be correct to reverse all the edges in the cycle (while retaining the direction of edges not in the cycle).



5. Suppose we have a Cuckoo Hash Table with each table having room for $m = 11$ entries each. Our hash functions are $h_0(x) = x\%11$ and $h_1(x) = (x/11)\%11$, where the / is integer division (floor of division; discard remainder). For example, $h_0(1289) = 2$ and $h_1(1289) = 7$. We insert the keys $46, 51, 84, 200, 134, 138, 52, 184, 181, 179, 25, 28, 36, 72, 85$, in that order, into the Hash Table

   (a) (1 point) After the first five keys are inserted, are they in the upper array (indexed by h_0) or the lower array (indexed by h_1) after all of first five keys have been inserted? Circle your choice on each clearly.

   Tables after part A:

0	1	2	3	4	5	6	7	8	9	10
		46					84			

0	1	2	3	4	5	6	7	8	9	10
	134			51			200			

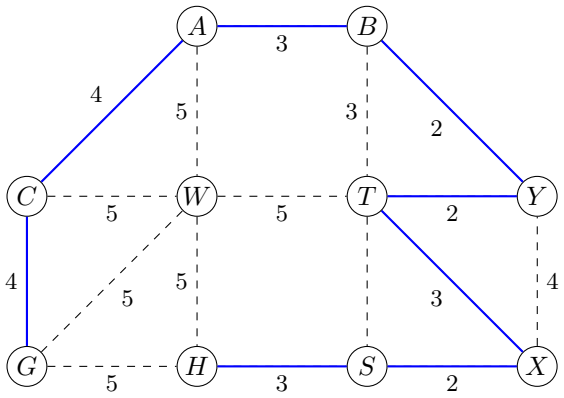
   (b) (1 point) Which value will be the last one **inserted successfully** into the table prior to the rehash/resize? Recall that we are rehashing / resizing only after an element cannot be inserted; there is no *a priori* maximum eviction length as there was in project three.

   28 which is t6 / b2 was the last one.

6. (1 point) Give a valid topological ordering of the following graph. Write only the eight letters on the response page (no arrows, commas, etc).

   There are many correct answers; among them are YTWPKLQR.

7. (1 point) Find the minimum spanning tree of the following graph:



MST is the blue edges, plus exactly one edge incident to W

8. (2.5 points) Amy and Barbara are playing a cooperative game. Each player has a vector of n positive integers, arranged as a row of squares from left to right. Each player has a token, which starts on the leftmost square of their row (represented by the first element of their vector). The goal is to move *both* tokens to the rightmost squares of their respective rows/vectors.

   On each turn, the two players must decide whether to move their tokens right or left. When a token moves, it moves exactly a number of squares equal to the number written on the square on which it is placed, either to the right or left. If either token moves past either end of the row, both players immediately lose. If ever both are at the right-most square at the same time, they both win.

   Describe an algorithm to determine whether Amy and Barbara can win the game they are playing, given their input vectors $A[1 \ldots n]$ and $B[1 \ldots n]$. Your algorithm *description* should be as simple as possible, but no simpler.

   Give the running time of your algorithm and briefly explain why it has that running time.

Create a graph with n^2 vertices, one for each possible configuration of the tokens. Place directed edges (u, v) if it is possible to move from configuration u to configuration v in one move. Creating this graph from the input takes $\mathcal{O}(n^2)$ time. Note that deciding the set of edges takes $\mathcal{O}(1)$ for each vertex, as there are only two possible outgoing edges: players move left and players move right. This also means that $|E| \leq 2n^2$.

We can use (BFS or DFS, doesn't matter) to see if there is a path from $(1, 1)$ to (n, n). If there is, Amy and Barbara can win. If not, they cannot. WFS takes $\mathcal{O}(V + E)$ time, which is $\mathcal{O}(n^2)$ in this case, as there are n^2 vertices.