EC 504 – Fall 2019 – Final

Put your name and BU ID on the first page. This exam is closed book, no notes – except for 3 pages of personal notes to be passed in at the end. No use of laptops or internet.

1. (20 pts) Answer True or False to each of the questions below. Each question is worth 2 points. Answer true only if it is true as stated, with no additional assumptions. No explanation is needed, but any explanation is likely to earn partial credit, and no explanation will not earn any credit if the answer is wrong

   (a) Consider a minimum spanning tree T in a connected, weighted undirected graph (V, E) where the weights have different values. Then, for every node i, the minimum spanning tree must contain the arc $\{i, j\}$ with the smallest weight connected to node i.

   (b) Any sorting algorithm on a set of 32 bit positive integers must have a run time complexity $f(N) \in \Omega(N \log(N))$.

   (c) There are instances of the integer knapsack problem which can be solved in polynomial time by a greedy algorithm.
   1

   (d) Consider a binary search tree with n keys. Finding whether a key value is already in the tree can be done in $O(\log(n))$.

   (e) Consider a binary search tree that was constructed by branching on the median value at each level, storing the keys at the leaves of the tree. Assume there are n keys in the tree, where each key is an ordered pair of values. The worst case complexity to find a key is $O(\log(n))$.

   (f) For the master equation $T(n) = aT(n/b) + n^k$ with $\gamma = \log(a)/\log(b)$ the solution is always a sum of terms powers n^γ and n^k.

   (g) The adjacency matrix for a graphs $G(N, A)$ is always a symmetric matrix, i.e the same above and below the diagonal.

   (h) Given a sorted set of numbers the dictionary search will always perform have complexity $T(n) \in o(log(n))$.

   (i) If you have a minimum distance between (i,j) in an undirected graph $d(i, j) = d(i, k) + d(k, j)$ then either $d(i, k)$ or $d(k, j)$ is a minimum distance but not both.

   (j) The best union-find algorithm can perform a sequence random sequence of n uniona and m finds in $Tn, m) \in O(n + m)$.

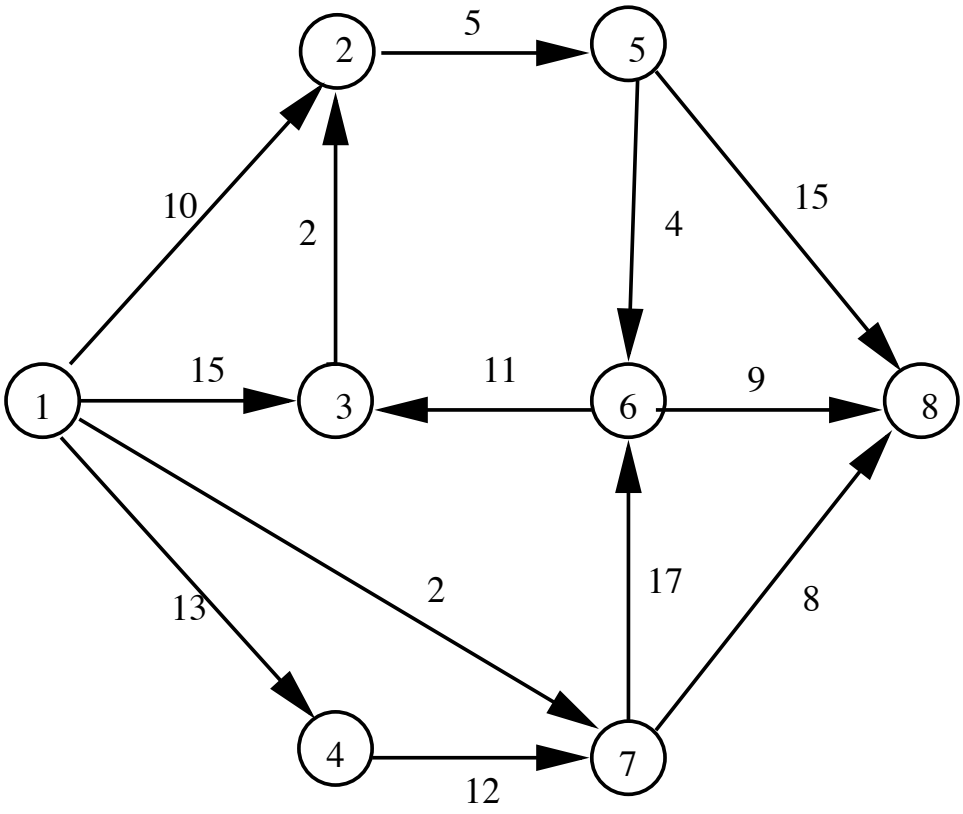


Figure 1:

2. (15 pts) Consider Fig. 1 as a directed, capacitated graph, where the numbers on an arc now indicate an arc's capacity to carry flow from node 1 to node 8. In the max-flow algorithm of Ford and Fulkerson, the key step is, once a path has been found, to augment the flow and construct the residual graph for the next iteration.

   (a) Suppose your program picks the first augmentation path to be $1 \to 7 \to 6 \to 3 \to 2 \to 5 \to 8$. What is the capacity of this path?

   (b) Now be smarter and beginning with a min hop $1 \to 2 \to 5 \to 8$ for the first path enumerate the additional paths that bring you to maximun flow. Draw **all** the residual and **all** flow graphs after **each** augmentation. What is the maximum flow? Draw the minimum cut to show this right.

3. (15 pts) You are interested in compression. Given a file with characters, you want to find the binary code which satisfies the prefix property (no conflicts) and which minimizes the number of bits required. As an example, consider an alphabet with 8 symbols, with relative frequency of appearance in a text file give below [1].

alphabet:	V	A	M	P	I	R	E	S
frequency:	4	40	16	7	30	18	75	8

(a) Determine the Huffman code by constructing a tree with **minimum external path length**. (Please use the "smaller weight to the left" convention.)

(b) With 0 bit to the left and 1 bit to right, identity the code for each letter and list the number of bits for each letter. (Note: as you descend the tree the bits are code for a letter goes right to left!)

(c) Compute the average number of bits per symbol in this code? Is it less than 3? (You can leave the answer as a fraction since getting the decimal value is difficult without a calculator.) Give an example of frequencies for these 8 symbols that would saturate 3 bits per letter?

[1] vampire: Origin mid 18th cent: form French, from Hungarian **vampir**, perhaps form Turkish **uber "witch"**. Meaning: a corpse supposed, in European folklore, to leave its grave at night to drink the blood of the living by biting their necks with long pointed canine teeth!

4. (15 pts) Consider searching in the "text" of length $N = 25$

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
a	b	c	a	a	b	c	a	b	a	b	c	a	b	a	b	c	a	b	c	a	b	a	b	c

for the following string of length $M = 8$,

a	b	c	a	b	a	b	c

using the KMP algorithm.

(a) Give the prefix function, $\pi(i)$, for above string: That is the value of $\pi(i)$ for $i = 1, \cdots 8$.

(b) Find all the instances of this pattern in the text. That is give the start index in the text for the aligned pattern.

(c) Specify all the non-trivial shift (i.e grater than 1 unit) that occurred in the scan using the KMP algorithm.

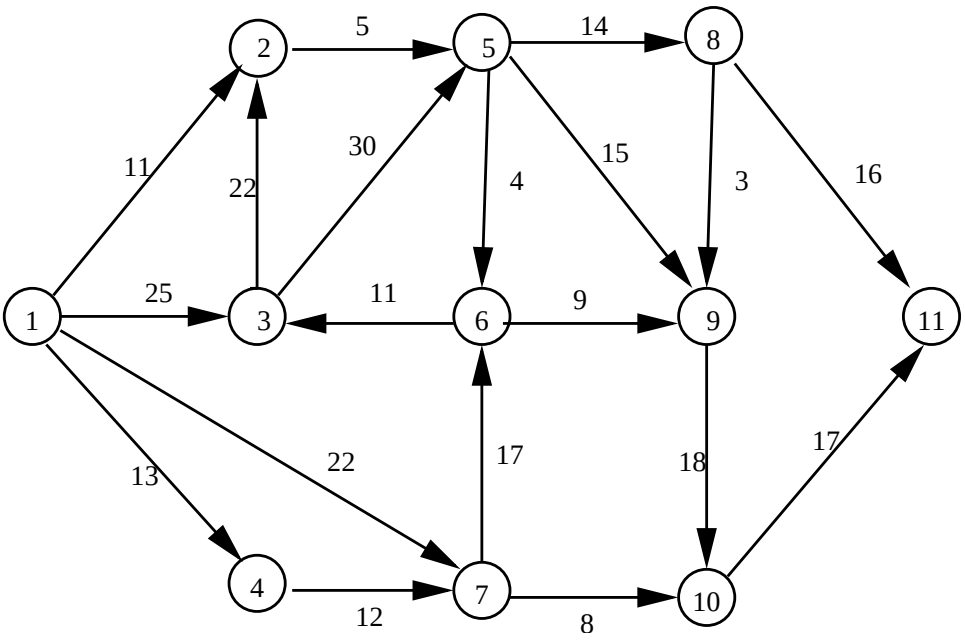


Figure 2:

5. (20 pts) Consider the Fig. 2 above.

   (a) Computed the minimum spanning tree considering all arcs to be bi-directional (in spite of the figure!) using Prims algorithm starting form node 1, listing in order the total weight and predecessors as they modified.

   (b) Now with taking into account the directed arcs compute the minimum distance from 1 to all other nodes using Dijkstras algorithm specifying the predecessor as each arc is added to the tree. What is the weigh to this tree relative to the minimum spanning tree above.

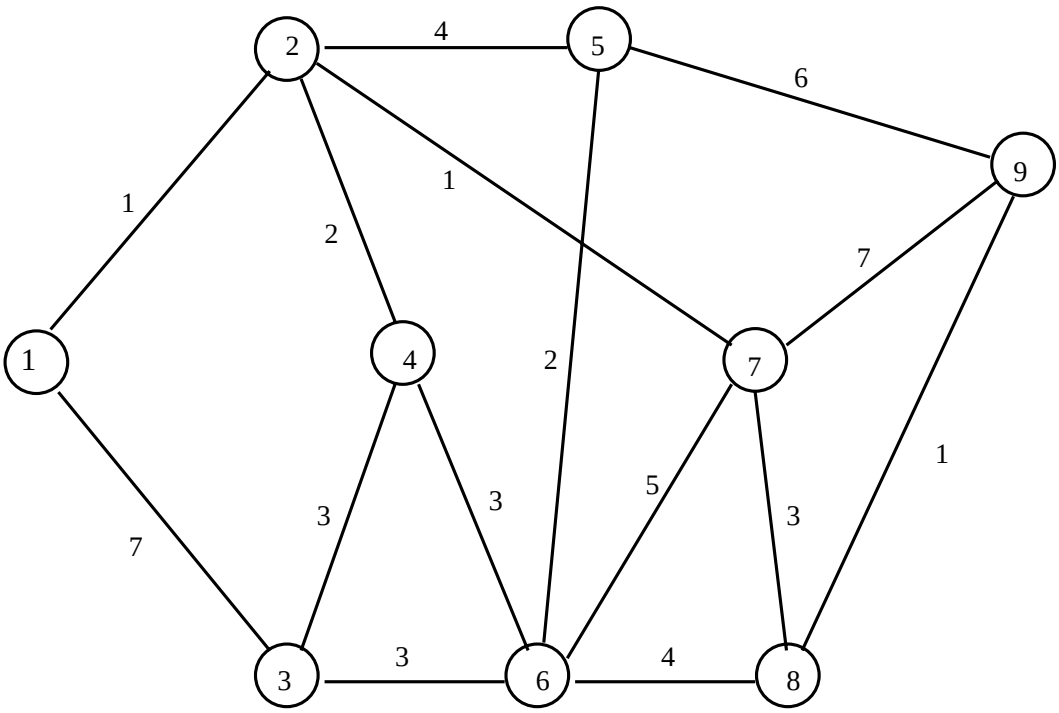


Figure 3:

6. (15 pts) Consider the undirected weighted graph in Figure 3. (There are copies of this graph at the end of the exam.)

   (a) Use Dijkstra's algorithm to find the shortest paths from node 1 to all of the nodes. (Dijkstra's, like Prim's minimum spaning tree algorithm, adds one node at a time.)

   (a) Make a series of tables showing the values of the distance array $d(i)$ and the predecessor array $\pi(i)$ after each update.

   (b) Draw the final tree on the figure with the final values of $d(i)$ and $p(i)$ at each node.

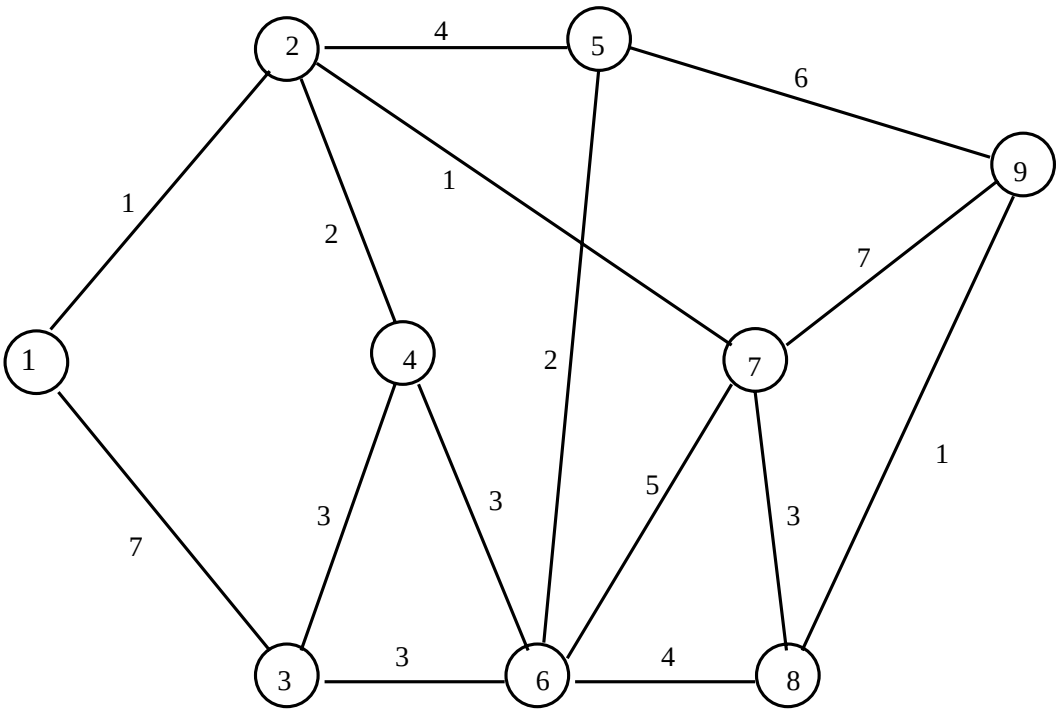
   (b) Repeat the exercise above except now use using Belllman-Ford this time (Bellman-Ford, like Kruskal's minimum spaning tree algorithm, adds one arc at a time.)

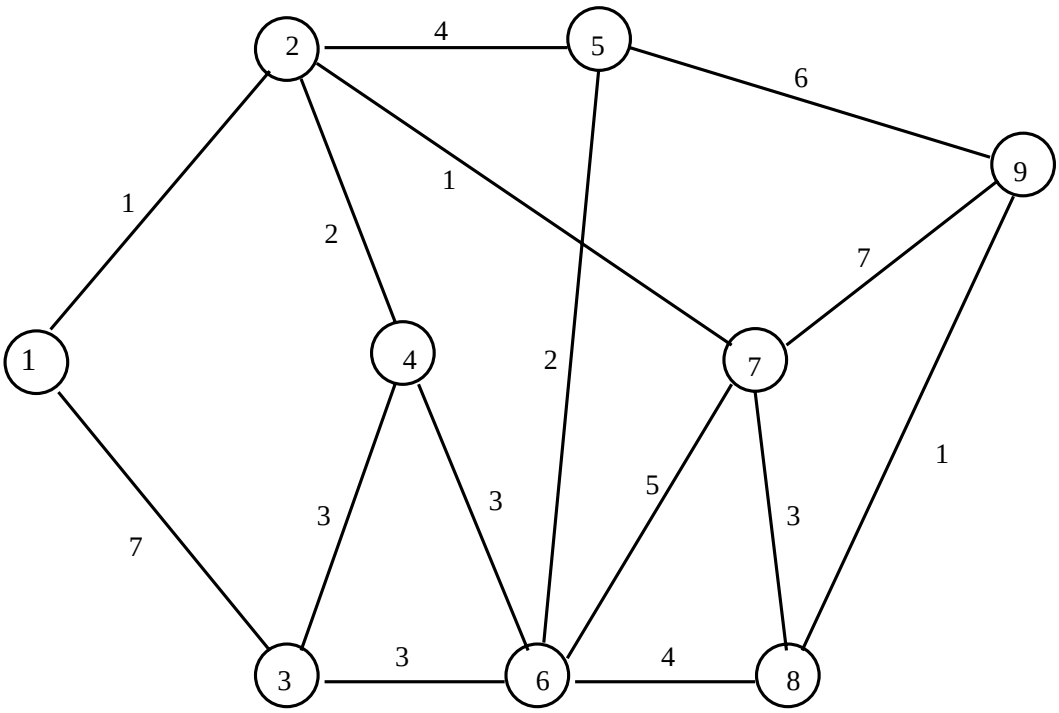
   (a) Make a series of tables showing the values of the distance array $d(i)$ and the predecessor array $\pi(i)$ after each update.

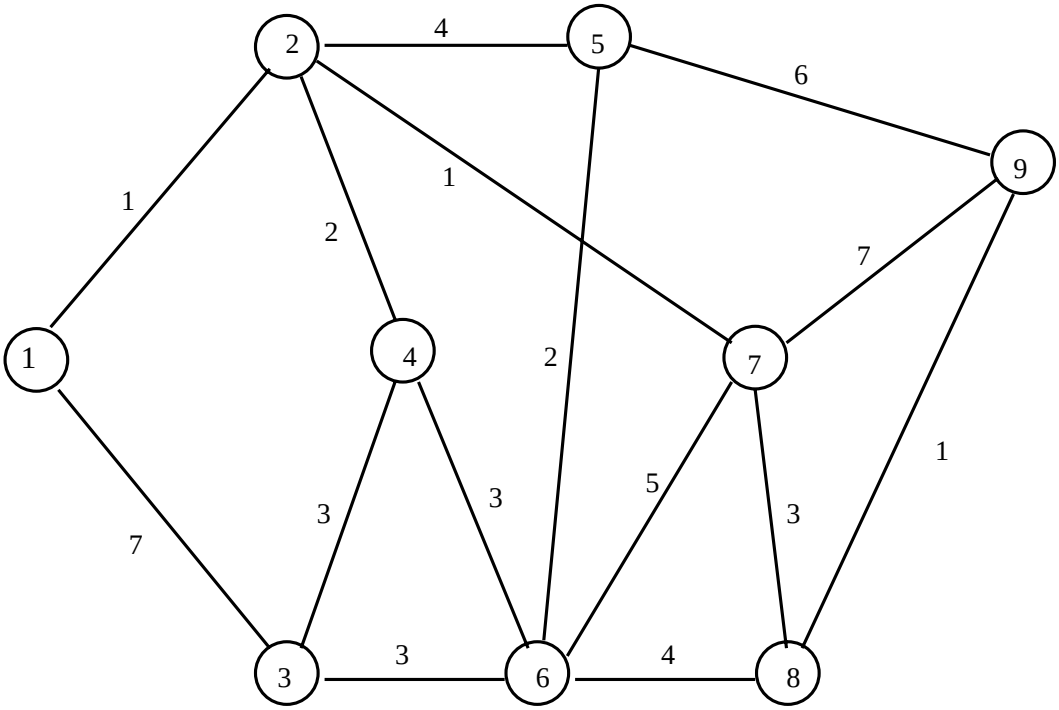
   (b) Draw the final tree on the figure with the final values of $d(i)$ and $p(i)$ at each node.

   (c) Finally find the minimum spaning tree.

   Draw the final tree on the figure with the final values of of $p(i)$ at each node. Are the final trees in all these case the same. Explain.


2
4
5
6
9
1
1
2
7
1
4
2
7
1
3
3
5
3
7
3
3
4
8


2
4
5
6
9
1
1
2
7
1
4
2
7
1
3
3
5
3
7
3
3
4
6
8


2
4
5
6
9
1
1
2
7
1
4
2
7
7
3
3
3
5
3
1
3
3
6
4
8