EC 504 – Fall 2023 – Practrice Exam

This is sample of question. But ti is equally valuable to reveiw the Homework Exercise, slides on GitHUB and of course the disconsions in class.

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. Give an explanation to earn partial credit even if the answer is wrong.

   (a) $2^{\log_{10} n} \in O(n)$

   (b) $10^{\log_2(n)} \in O(n^3)$

   (c) If $f_i(n) \in O(n^2)$, then $\sum_{i=1}^{n} log_2(f_i(n)) \in O(n^2)$

   (d) In a binary search tree with no repeated keys, deleting the node with key x, followed by deleting the node with key y, will result in the same search tree as deleting the node with key y, then deleting the node with key x, under the assumption that you replace the deleted key with its predecessor if it exists, or with its successor otherwise.

   (e) The second smallest element in a binary min-heap with all elements with distinct values will always be a child of the root.

   (f) Inserting numbers $1, \ldots, n$ into a binary min-heap in that order will take $\Theta(n)$ time.

   (g) 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.

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

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

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

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

(l) There are exactly 5 distinct binary search trees that include the numbers 1,2 anad 3.

(m) The Johnson All to All distance algorithm for $G(N, A)$ introduces an augmented graph with an extra node and potential function to remove all the negative cycles so that Dijkstras one to all can be used.

(n) The scheduling algorithm with deadlines and unit tasks when you us the *Maximun Procrastination* method is an example of amortized analysis.

2. (10 pts) Consider a list of n elements with distinct values in the range $\{1, \ldots, n^2\}$. Identify which of the following sorting algorithms will produce a sorted list in worst case time $O(n \log n)$ (note I said $O()$, not $\Theta()$).

   (a) Counting (or Bin) sort

   (b) Radix sort with radix n

   (c) Merge sort

   (d) Insertion sort

   (e) Quicksort

3. (15 pts) For each of these recursions, please give the tightest upper bound for the recursion. You can write your answer as $T(n) \in O(g(n))$ for your best choice of function $g(n)$.

   (a) $T(n) = 8T(n/2) + n^2$

   (b) $T(n) = 5T(n/2) + (n \log n)^2$

   (c) $T(n) = 2T(n/2) + n \log n$

   (d) $T(n) = T(n/2) + n$

   (e) $T(n) = nT(n/2)^2$, with $T(1) = 1$. (HINT: Find the exact solution to the recursion relation for $S(n) = \log_2(T(n)$ and set $T(n) = 2^{S(n)}$.)

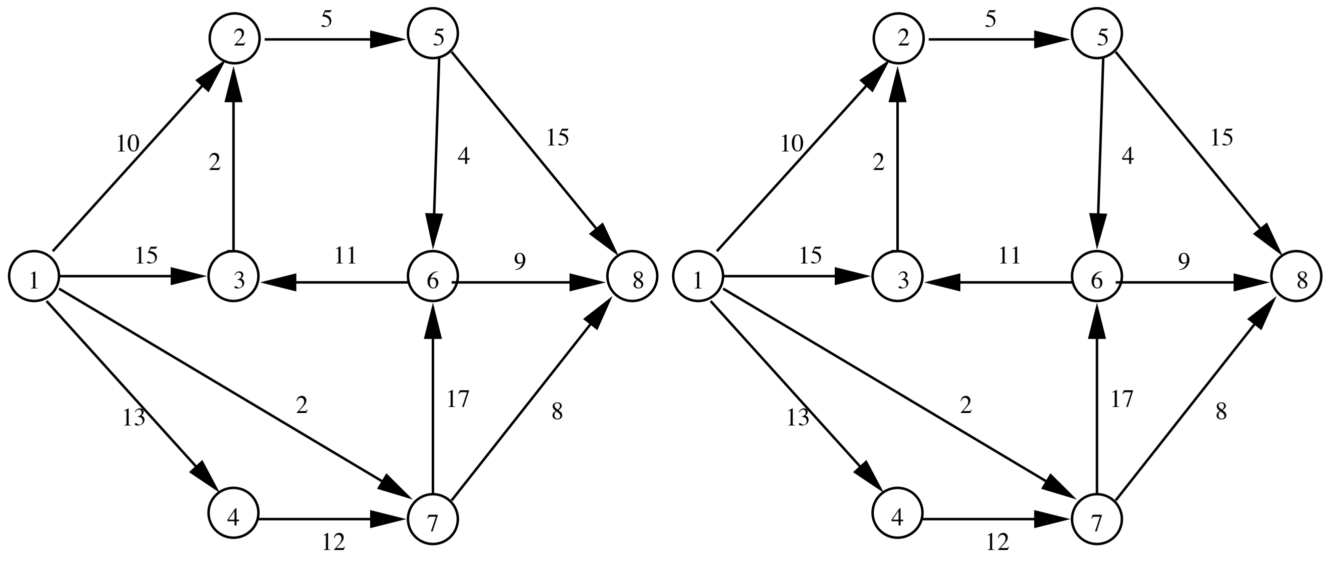
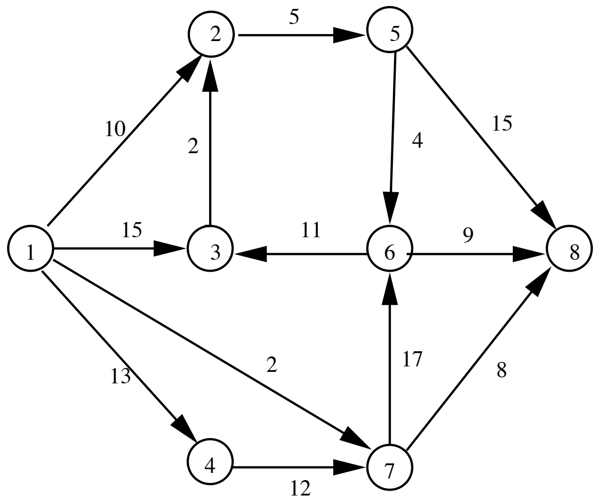
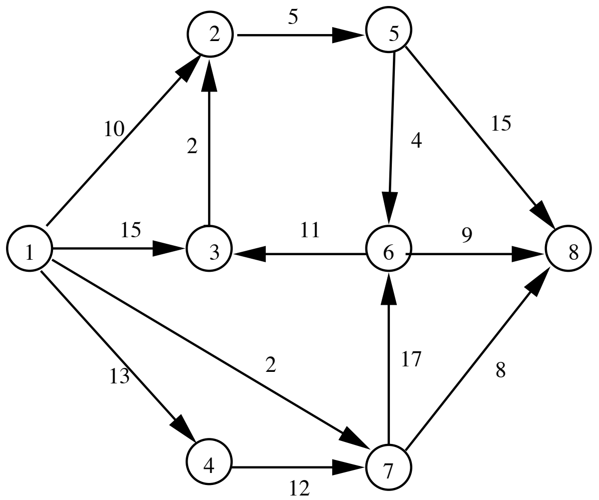
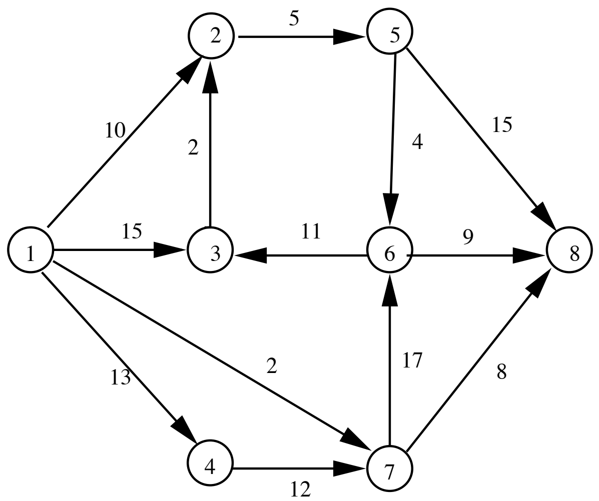
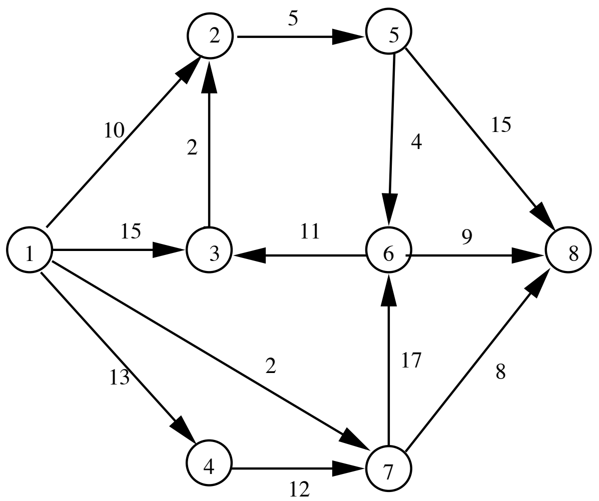
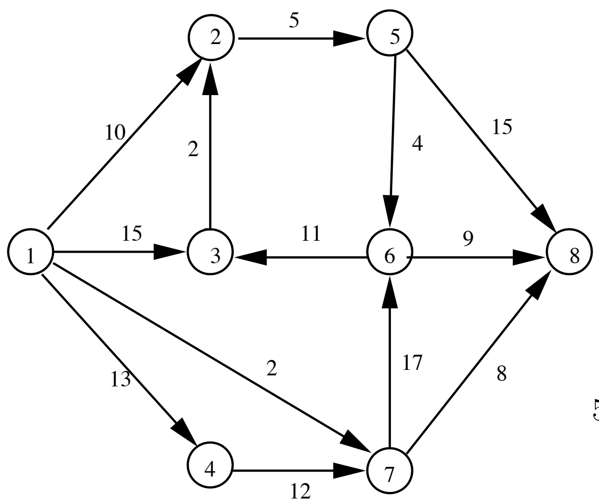
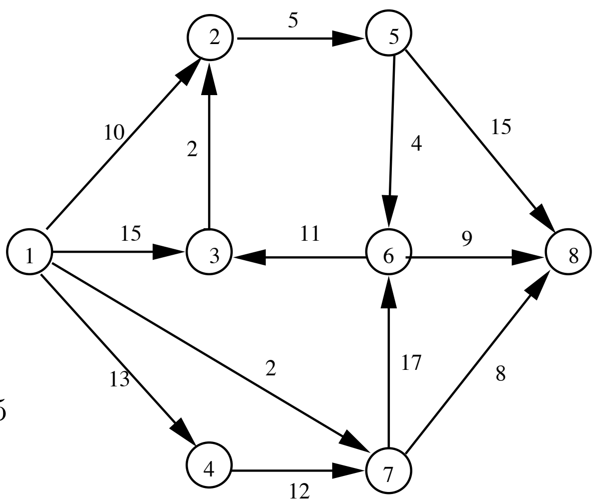


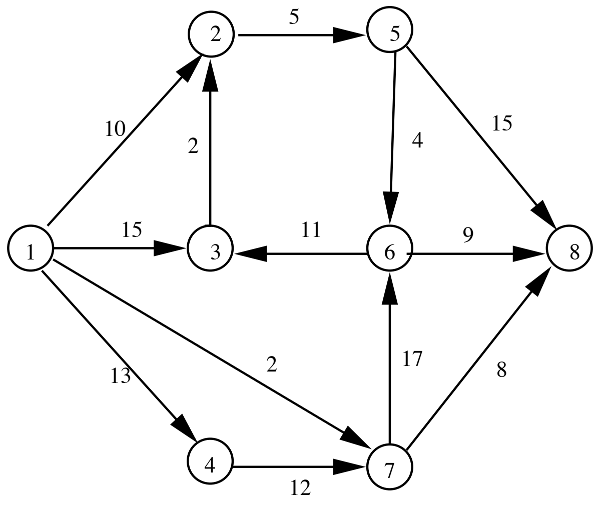
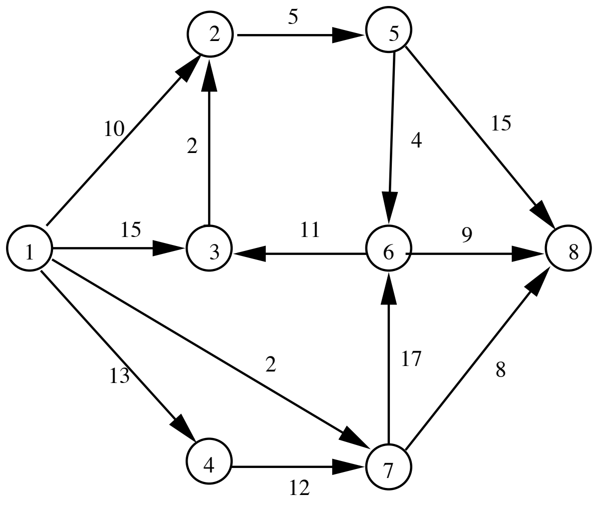
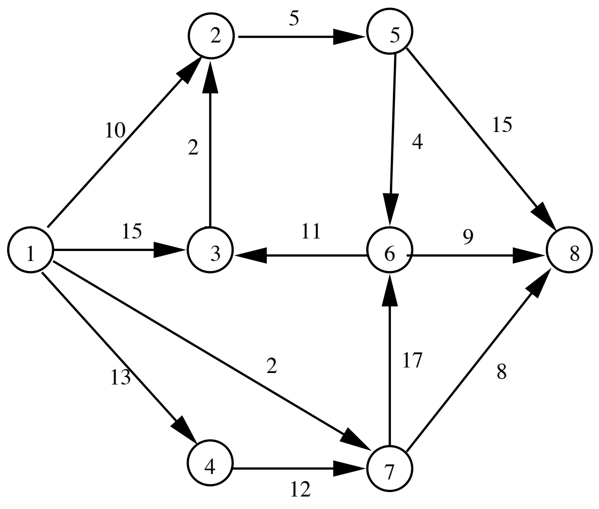
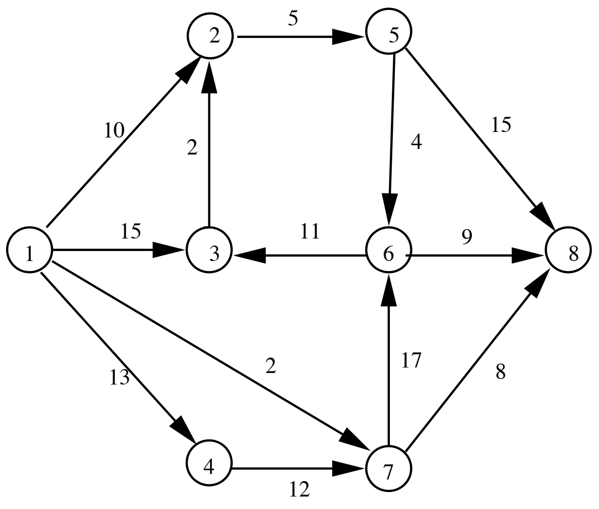
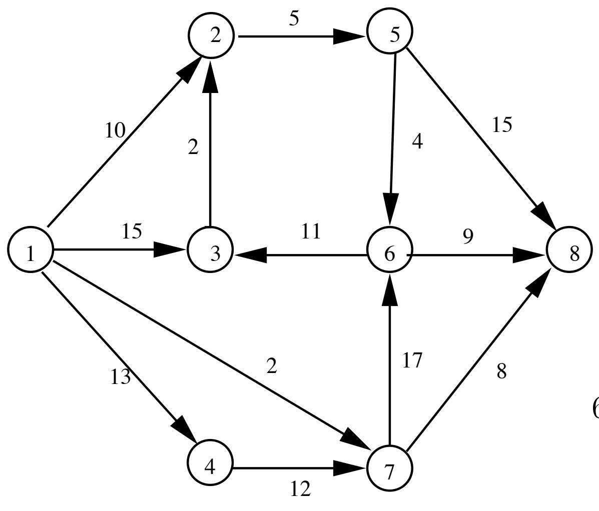
Figure 1:

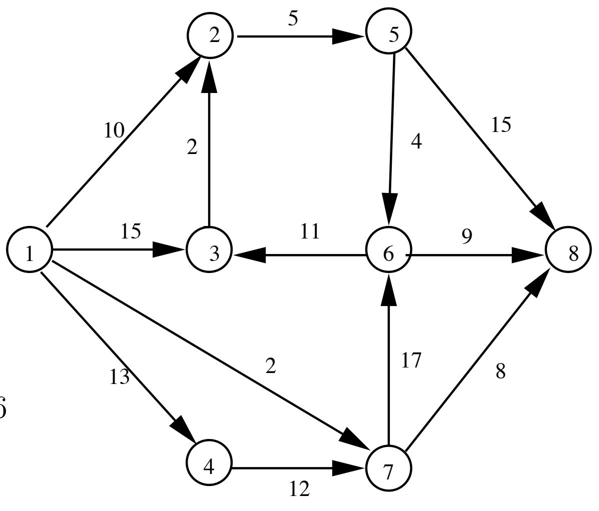
4. (15 pts) Consider Fig. 1 as a directed, capacitated graph, where the numbers on an arc now indicate an arc's capacity to carry flow from node 1 to node 8. In the max-flow algorithm of Ford and Fulkerson, the key step is, once a path has been found, to augment the flow and construct the residual graph for the next iteration.

   (a) Suppose your program picks the first augmentation path to be $1 \to 7 \to 6 \to 3 \to 2 \to 5 \to 8$. Draw the augmented flow path on the left graph above and the residual graph above in the right side. 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** flow graphs and **all** the residual and after **each** augmentation. What is the maximum flow? Draw the minimum cut to show this right.


5
10
2
4
15
15
11
9
13
2
17
8
12

5
10
2
4
15
15
11
9
13
2
17
8
12

5
10
2
4
15
15
11
9
13
2
17
8
12

5
10
2
4
15
15
11
9
13
2
17
8
12

5
10
2
4
15
15
11
9
13
2
17
8
12

5
10
2
4
15
15
11
9
13
2
17
8
12


2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12

2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12

2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12

2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12

2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12


2
5
5
10
2
4
15
1
15
3
11
6
9
8
2
17
8
13
4
7
12

5. (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:	F	I	B	O	N	A	C	E
frequency:	4	20	6	14	8	25	15	37

(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!) **You may want draw first on extra paper and then copy it below to get it this standare the form.**

(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] Fibonacci https://en.wikipedia.org/wiki/Fibonacci_sequence Ok I mispelled the name to make it 8 different letters!

6. (15 pts) Consider searching in the "text" $T(1:N)$ 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 $M = 8$ as an array $P(1:M)$ (i.e. $P(i), i = 1, 2, \cdots M$)

b	a	b	c	a	b	c	a

using the KMP algorithm.

(a) Give the prefix function for above string: That is the value of $\pi(i)$ for $i = 1, \cdots 8.$ [2] It is useful to copy the pattern in $P(1:M)$ to slide it against $P(1:M)$ After a failure at $q+1$ you can safely shift by $\pi(q)$ before starting to search again. Note overlapping finds are ok!)

(b) Find all the instances of this pattern in the text. That is give the start index in the text for the aligned pattern. (You can do this even if you have not got the right prefex function. Slide $P(1:M)$ against $T(1:N)$.)

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

[2] **Recall that the prefex function looks at the first q values of $P(1:q)$ and ask how far you can shift to match to have largest prefi match the suffix in these q terms. $\pi(q) = MAX\{k < q$ such that $P(1:k) = P(1+q-k:q)\}$.**

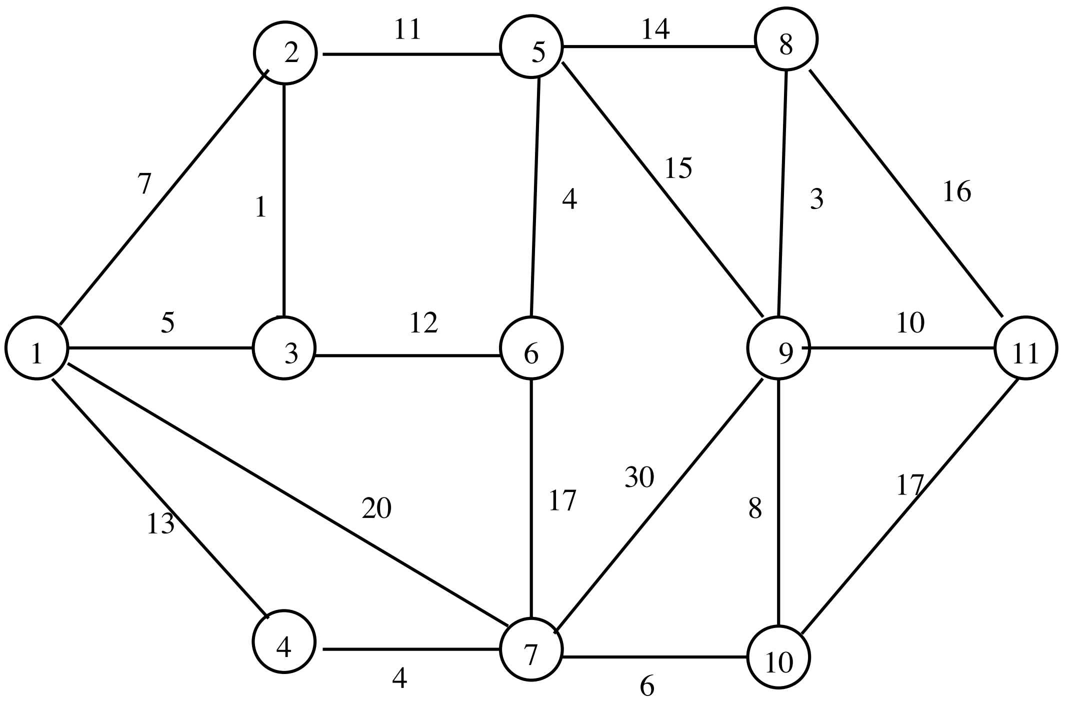


Figure 2:

7. (20 pts) Consider the undirected weighted graph in Figure 2. (There are copies of this graph at the end of the exam.) [3]

   (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.)

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

   (ii.) 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.)

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

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

[3]**Note in part a and b below you start with $d(1) = 0$, $d(i > 1) = \infty$ and $\pi(i) = -1$. The table at each step only needs to show the values of $d(i)$ and $\pi(i)$ that change!**

(c) Computed the minimum spanning tree considering all arcs to be bi-directional (in spite of the figure!) using Prim's algorithm starting form node 1, listing in order the total weight and predecessors as they are modified at each step.

(d) Are the final trees in all these cases the same? Explain

8. (15 pts) Quick questions:

   (a) How many distinct binary search trees are there that includes the numbers 1,2,3,and 4?

   (b) What are the sizes of the binomial trees in a binomial heap that has 13 elements?

   (c) If you insert the numbers 1,2,3,4,5,6,7 into a red-black tree, what is the tree that results?

   (d) if you insert the numbers 1,2,3,4,5,6,7 into an AVL tree, what is the tree that results?

9. (15 pts) Assume that there is a set of tasks with deadlines and values listed below, each of which takes unit time to complete.

Task #:	1	2	3	4	5	6	7	8
Values:	2	4	3	8	3	6	5	3
Deadline:	1	8	2	6	4	1	3	5

(a) Explain clearly in four **short** statements the optimization algorithm that maximizes the value for task that can be completed by the deadlines. **(Unnecessarily long explanations will get LESS credit!)**

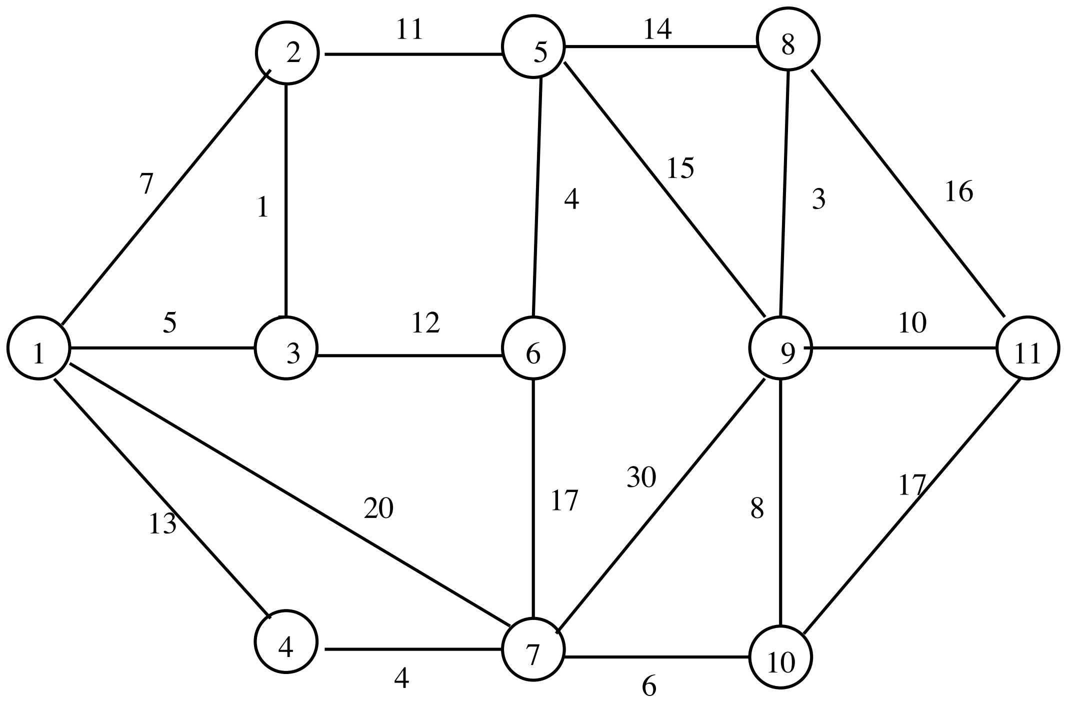
(b) Use this algorithm to find the schedule for this set of tasks that maximizes the value for all tasks completed by the deadline.

10. (15 pts) Consider the following scheduling problem: There are 10 tasks, each of which require a certain amount of processing time and have a value, as illustrated in the table below:

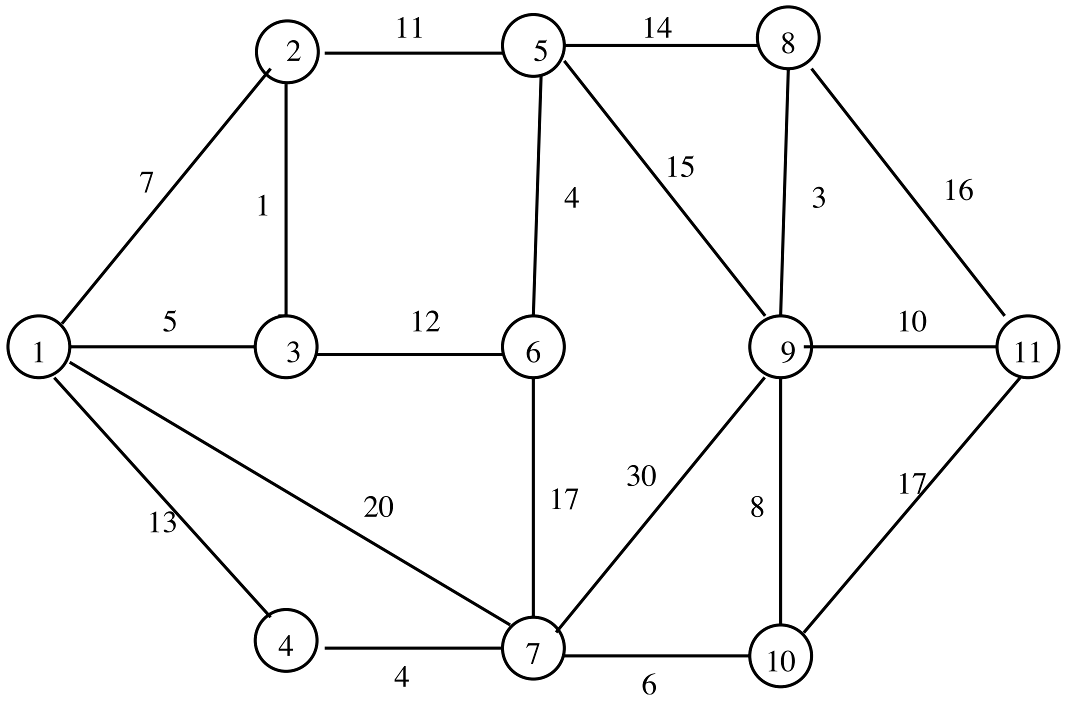
Task #:	1	2	3	4	5	6	7	8	9	10
Value:	7	3	4	6	3	4	8	5	7	2
Time:	3	9	3	6	2	6	8	4	5	7

(a) Suppose that there is a single machine with total capacity of 25 units of time, and that one gets partial credit for partially processing a task, so that processing a task of value V_i for time $x_i t_i$ when the requirements are t_i units provides value $x_i V_i$. **What is the optimal set of tasks to schedule, and the value achieved by this optimal schedule?**

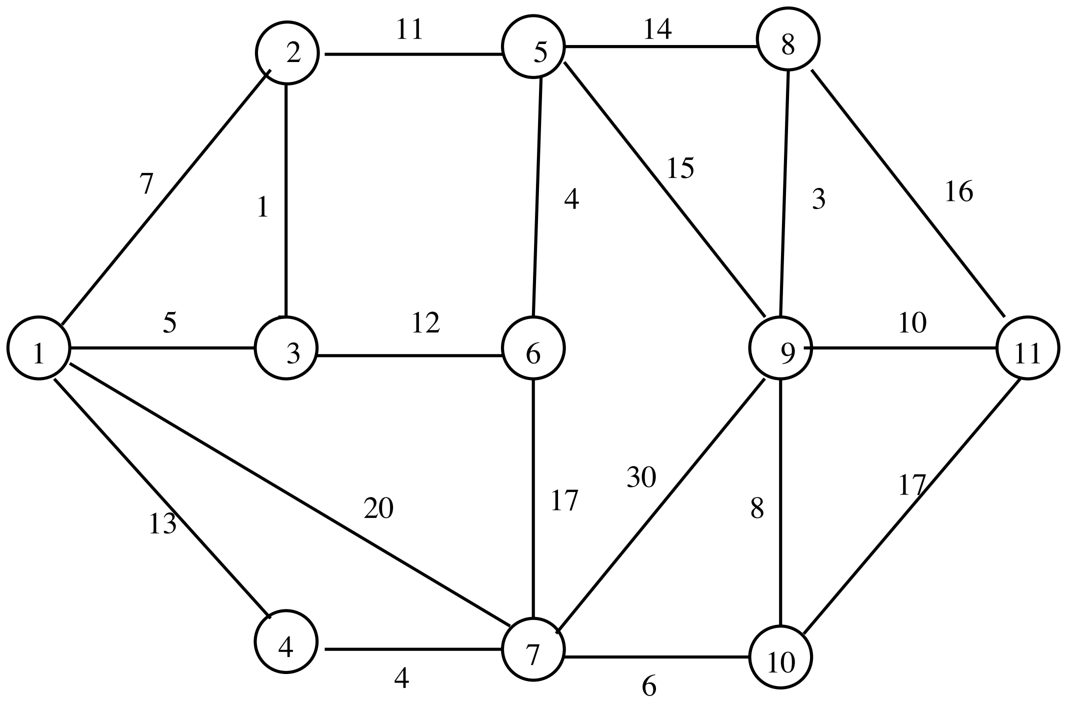
(b) Suppose that there is a one machine with total capacity of 15 units of time on which you get no partial credit. Suppose there is another machine with 10 units on which you get partial credit. Modify the above result to do your best to schedule on the two machine. (You will get credit for any reasonable efficiency.)

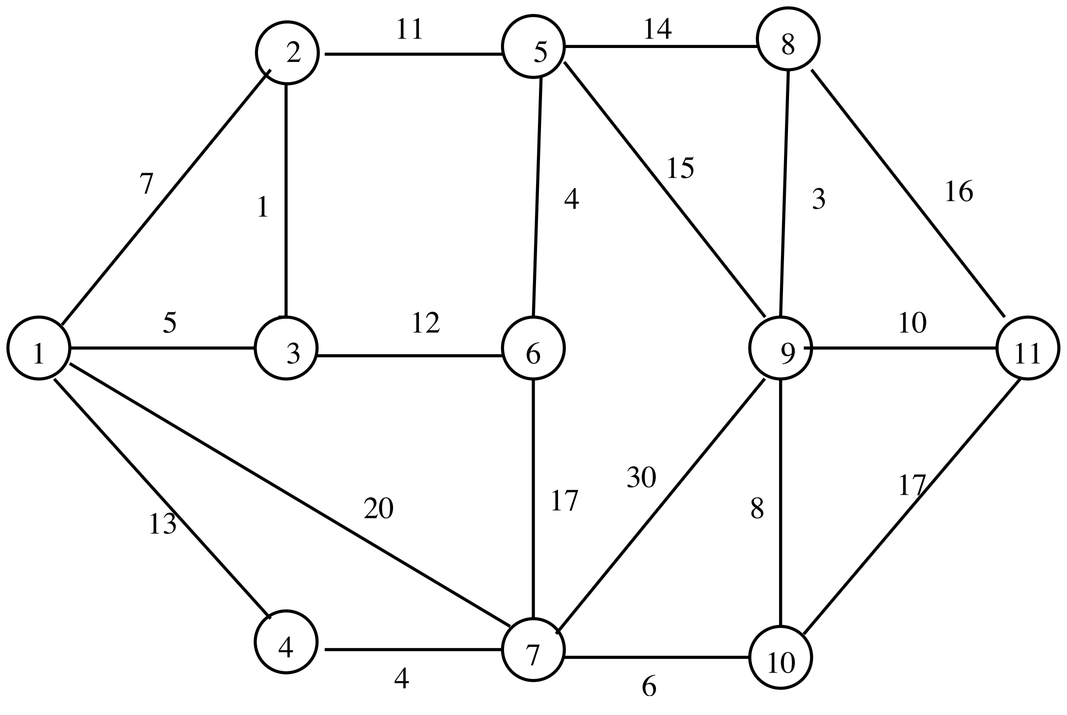


11
14
7
1
4
15
3
16
5
12
10
13
20
17
30
8
17
4
6
1
2
3
4
5
6
7
8
9
10
11



11
14
7
1
4
15
3
16
5
12
10
13
20
17
30
8
17
4
6
1
2
3
4
5
6
7
8
9
10
11


1
2
3
4
5
6
7
8
9
10
11
7
5
13
20
1
11
12
4
4
17
6
14
15
3
30
8
10
16
17


1
2
3
4
5
6
7
8
9
10
11
7
5
13
20
1
11
12
4
4
17
6
14
15
3
30
8
10
16
17