EC 504 – Fall 2023 – Homework 6

Due Thursday Nov 30, 2023 by 11:59AM submitted to gradescope.

Reading Assignment: CLRS Chapters 26 and 32 as well as other section in Selected Topics VII may give idea for Project in addition to googling of course.

1. (20 pts) Answer True or False to each of the questions below, and explain briefly why you think the answer is correct..

   (a) There are fractional knapsack problem gives an upper bound on maximum value, ($V = \sum_{i=1}^{N} x_i t_i$ for $\sum_{i=1}^{N} t_i \leq T$) but never the same maximum value as the integer knapsack.

   (b) Given a very large (random) sequence of N coin flips $\{HTHTTH...\}$ the problem of finding as sequence of $M = \Theta(N)$ head (H) is $O(N)$.

   (c) Suppose you have an unsorted array of $2n + 1$ elements. One can determine the n smallest elements of the array in $O(n)$ time.

   (d) The largest element in a binary search tree can be found by recursive search to the right subtree.

   (e) Consider a minimum spanning tree T in a connected, weighted undirected graph $G(N, A)$ 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.

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

   (g) 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.

   (h) The best union-find algorithm can perform a sequence random sequence of N unions and M finds in $T(N, M) \in O(N + M)$.

   (i) 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.

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

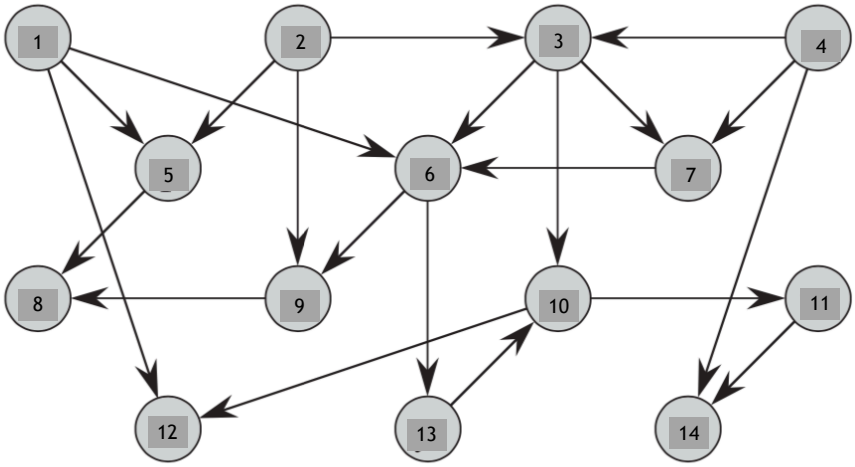


Figure 1:

2. (15 pts) Do a topological sort on this directed graph. (YES this is exactly the same graph done in class. Still a good exercise.)

   (a) Starting from node 1 do a topological sort using a stack to enumerate the nodes. Alway prefer the smaller node number when pushing into the stack. When the stack is empty start with the lowest node number not yet visited. You should show enough of the individual stack operatinos to trace the enumeration (crossing out element is ok for pop operations).

   (b) Show on the graph that order of execution that has no deadlocks. What is the graphs property that guarantees this lack of deadlocks?

   (c) Give the number and size of the directed connected components in this directed graph. How many connected components in the graph if all arcs are allowed to be undirected?

   (d) If we did the topological sort after permuting the labels and still using the low label first rule for the new lables, would the result be the same order of execution and the same number of directed connected components?

3. (10 pts) Consider Floyd-Warshall algorithm for a directed graph with 4 nodes. The first value for $D^{(0)}(i,j) = W(i,j)$ is the following:

D(0)(i,j)	j=1	j=2	j=3	j=4	j=5
i=1	0	5	9	inf	3
i=2	inf	0	8	inf	inf
i=3	inf	1	0	7	4
i=4	inf	inf	inf	0	inf
i=5	11	inf	inf	6	0

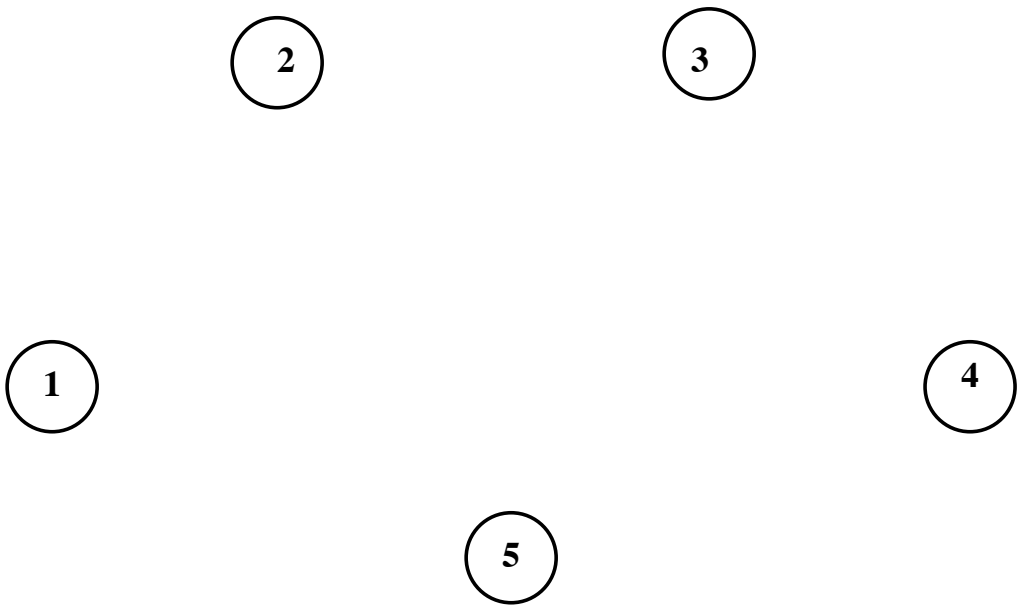


Figure 2:

(a) Draw the directed graphs on Fig. 3 with arcs labeled by their weights. Give the iterations k = 1, 2, ... for the Floyd-Warshall algorithm. (Recall that the k = 1 iteration considers paths from i to j thru node 1 and k = 2 paths thru node 2, etc.) Show your work as a sequence of two 5 by 5 tables.

(b) Solve the all pairs shortest path by two iterations of "repeated squaring", i.e. for $i, j, k = 1, 2, 3, 4, 5$

$$D^{(2)}(i,j) = MIN[\ D^{(0)}(i,k) + D^{(0)}(k,j)\ ]\ ,$$

$$D^{(4)}(i,j) = MIN[\ D^{(2)}(i,k) + D^{(2)}(k,j)\ ]\ .$$

(c) Show your work as a sequence of two 5 by 5 tables.

4. (15 pts) This is a problem in the arithmetic of **Binomial queues** – also called a **Binomial Heap** when enforcing min-Heap like order. This is nice collectoi of trees *e.g. forest* that are have either zero nodes (just a null pointer) or 2^k nodes for each tree.

   (a) Draw two data structures for a binomial queues with a number of nodes give by $N = n_0 + 2n_1 + 4n_2 + 8n_3 + 16n_4$ and binary numbers $\{n_4 n_3 n_2 n_1 n_0\} = \{10101\}$ and $\{n_4 n_3 n_2 n_1 n_0\} = \{11110\}$.

   (b) If we demand that the value at the root each tree or subtree is a minimum for that tree each of the separate queues, describe rule for addition that maintains this property in the sum. For binomial queue with this property and $O(N)$ nodes what is the complexity of bound on finding the minimum value?

   (c) In general adding two binomial queues with $O(N_1)$ and $O(N_2)$ nodes bound on the complexity of the complexity of this add operation?

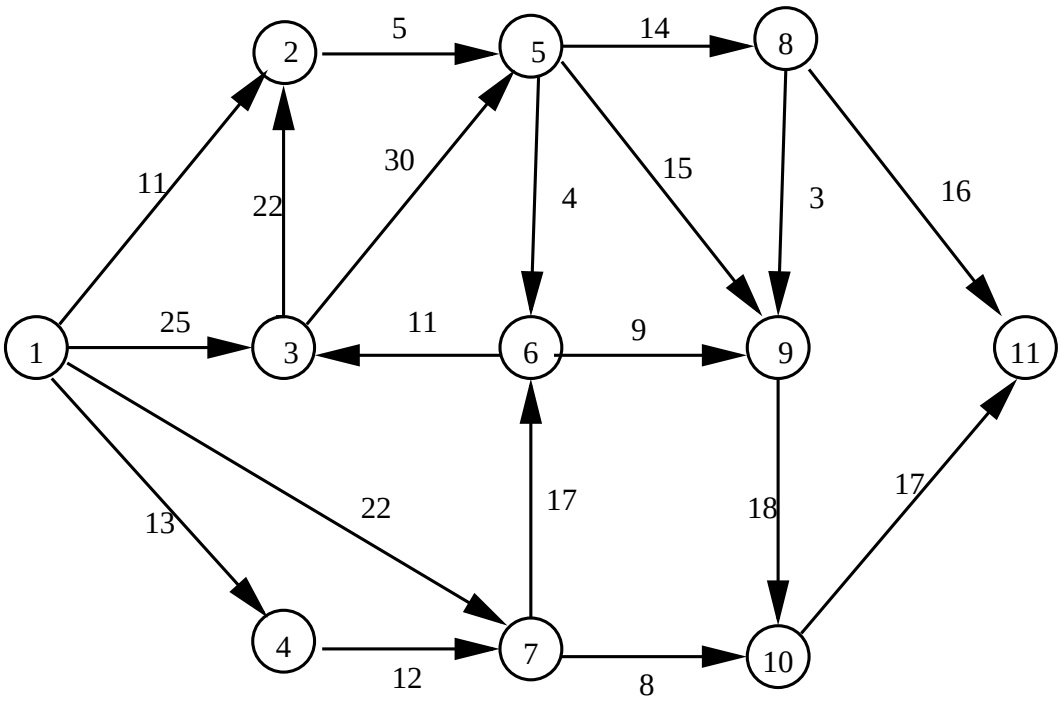


Figure 3:

5. (20 pts) Consider the directed weighted graph in Figure 3. (There are copies of this graph on the next page [1].

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

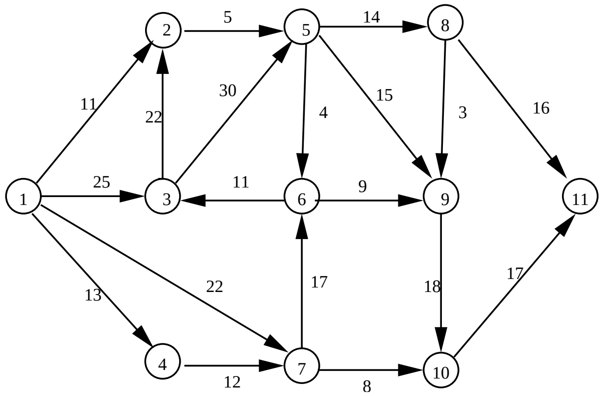
   (i.) Make a series of tables showing the values of the distance array $d(i)$ and the predecessor array $\pi(i)$ after each update.

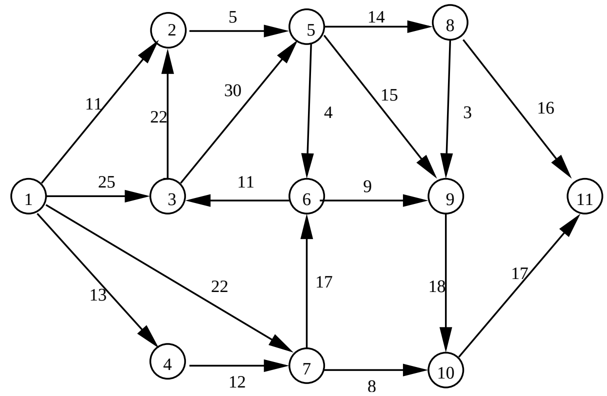
   (ii.) Draw the final tree on the figure with the final values of $d(i)$ and $p(i)$ at each node.

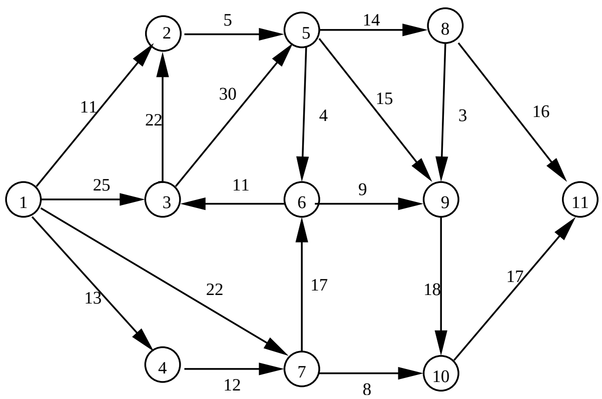
   (c) Computed the minimum spanning tree considering all arcs to be bi-directional (in spite of the figure!) using Prim's algorithm starting form node 1, listing in order the total weight and predecessors as they are modified at each step.

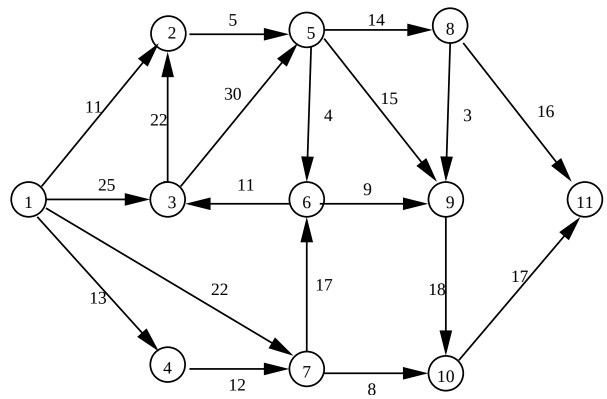
   (d) Are the final trees in all these cases the same? Explain

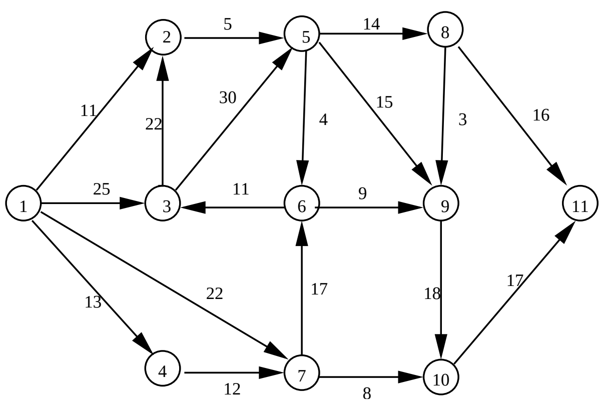
[1] **Note in part a and b below you start with $d(1) = 0$, $d(i > 1) = \infty$ and $\pi(i) = -1$. The table at each step only needs to show the values of $d(i)$ and $\pi(i)$ that change!**

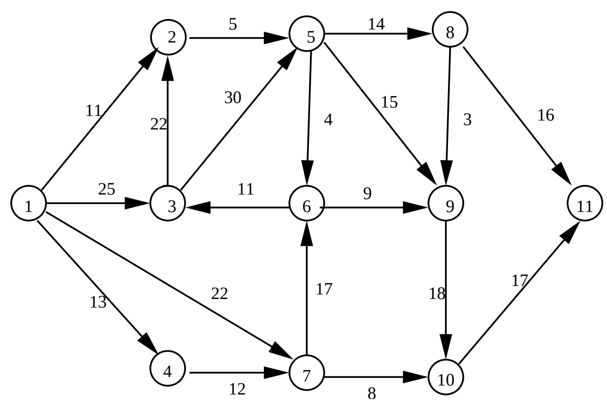

2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10


2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10


2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10


2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10


2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10


2
5
5
14
8
11
22
30
4
15
3
16
1
25
3
11
6
9
9
11
13
22
17
18
17
4
12
7
8
10

Object number	1	2	3	4	5	6
Value	10	11	12	13	14	15
Size	1	1	2	3	4	3

6. (10 pts) Consider the following knapsack problem. We have six objects with different values and Vi and sizes wi. The object values and sizes are listed in the table below:

   (a) Suppose we are given a knapsack with total size 11, and you are allowed to use fractional assignments. What is the maximum value that fits in the knapsack for the above tasks?

   (b) What is the maximum value of the tasks that fit entirely in the knapsack with size 11 — i.e. the integer knapsack.

7. (15 pts) Consider searching in the "text" $T(1:N)$ of length $N = 25$ using the KMP algorithm.

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
b	a	c	b	b	a	c	b	a	b	a	c	b	a	b	a	c	b	a	c	b	a	b	a	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	c	b	a	b	a	c

(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 prefix 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 prefix function looks at the first q values of $P(1:q)$ and ask how far you can shift to match to have largest prefix match the suffix in these q terms.**

$$\pi(q) = MAX\{k < q \textbf{ such that } P(1:k) = P(1+q-k:q)\}.$$

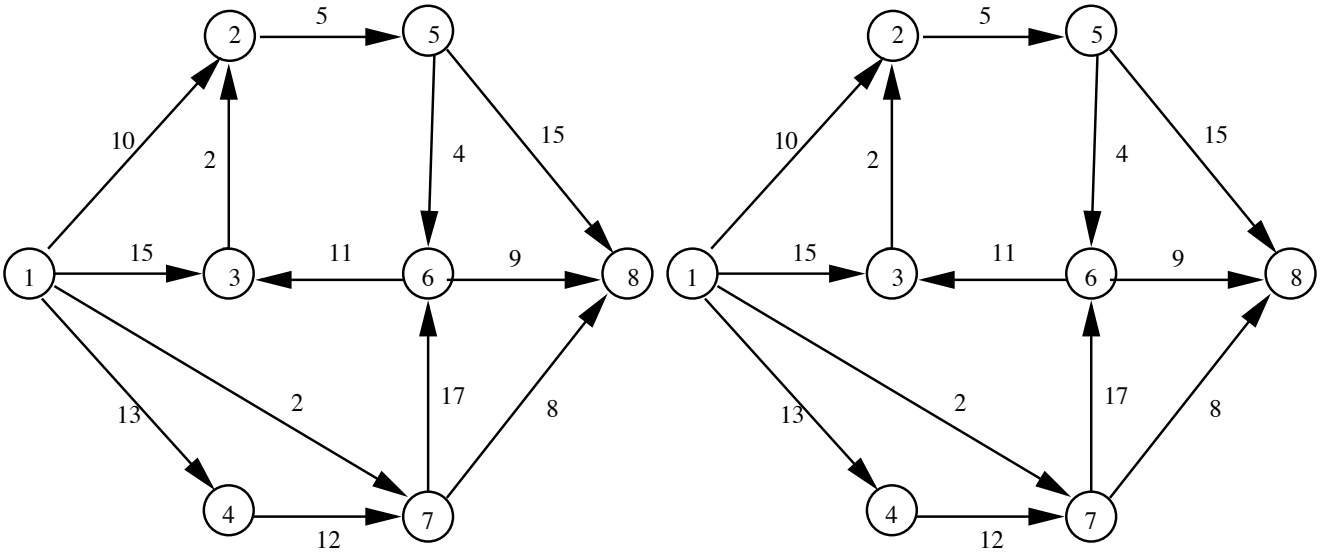
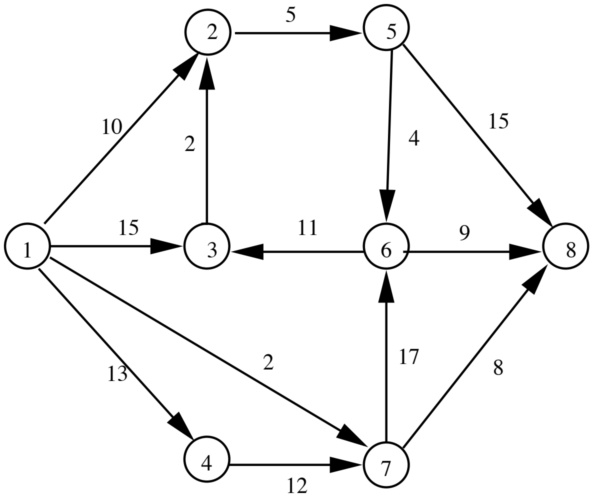


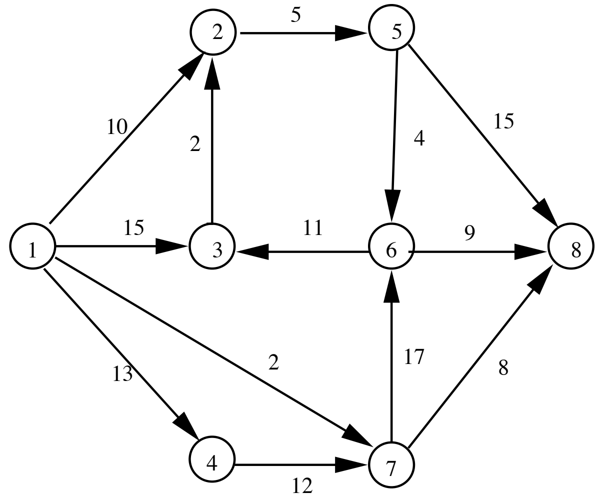
Figure 4:

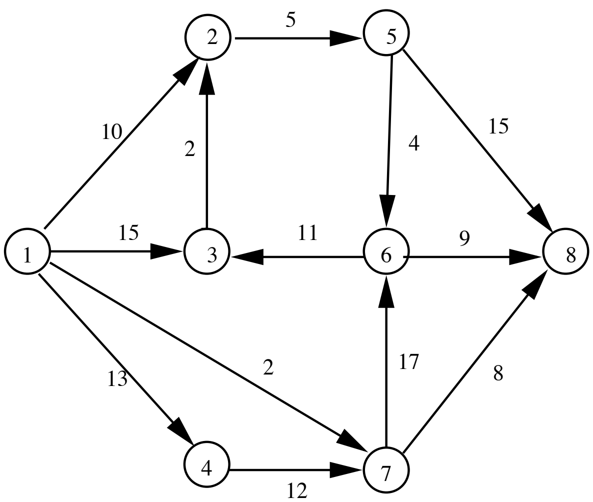
8. (15 pts) Consider Fig. 4 as a directed, capacitated graph, where the numbers on an arc now indicate an arc's capacity to carry flow from node 1 to node 8. In the max-flow algorithm of Ford and Fulkerson, the key step is, once a path has been found, to augment the flow and construct the residual graph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

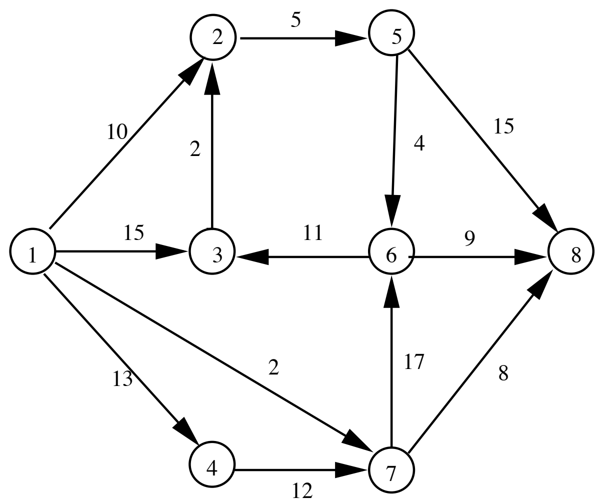
   (a) Suppose your program picks the first augmentation path to be $1 \to 7 \to 6 \to 3 \to 2 \to 5 \to 8$. Draw the augmented flow path on the left graph above and the residual graph above in the right side. What is the capacity of this path?

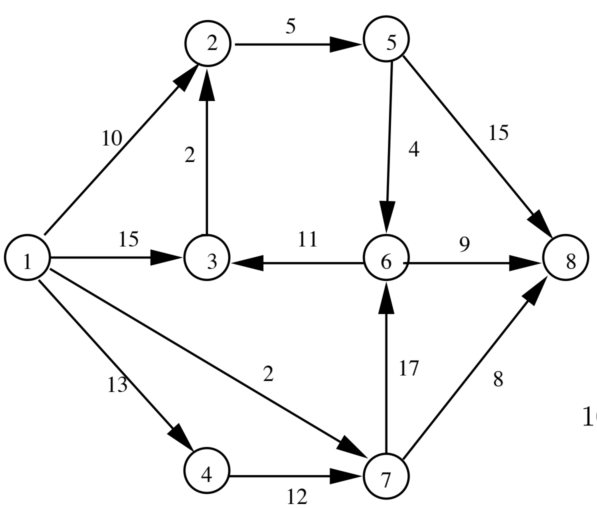
   (b) Now be smarter and beginning with a min h op $1 \to 2 \to 5 \to 8$ for the first path enumerate the additional paths that bring you to maximun flow. Draw **all** flow graphs and **all** the residual and after **each** augmentation. What is the maximum flow? Draw the minimum cut to show this right.

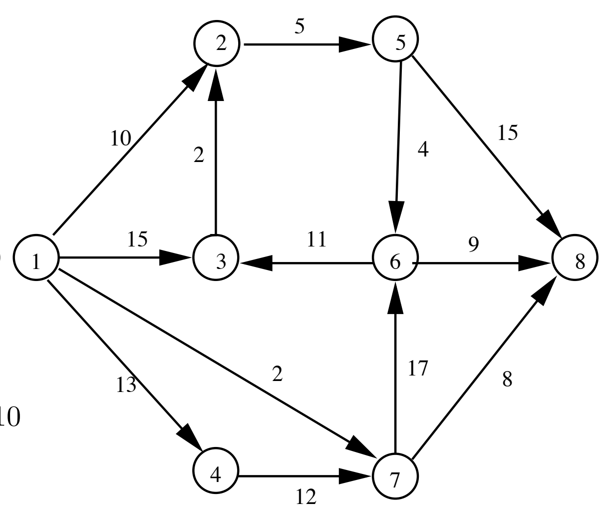

2
5
5
10
2
4
15
1
15
3
11
6
9
8
13
2
17
8
4
7
12


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


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


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


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


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