Print Name: _____ **ID:** _____

EC 504 Sample Final

Instructions: Put your name and BU ID on this page. I stapler will allow you to stable together at the end of the exam.. This exam is closed book and no notes – except for 2 pages of personal notes to be passed in the end. No use of laptops or internet. You have one hour and fifty minutes. Do the easy ones first.

1. 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. **Note if you give a good short explanation of your choice you will receive partial credit even if the answer is wrong!**

   (a) Inserting numbers $1, \ldots, n$ into a binary min-heap in that order will take $\Theta(n)$ time.
   Solution: True. Note that, since we are inserting them in increasing order, there will be no swaps, as inserting at the end means the parent will have a larger value.

   (b) Suppose you have an unsorted array of $2n + 1$ elements. One can determine the n smallest elements of the array in $O(n)$ time.
   Solution: True. You can do one pass to determine the $n + 1$-th median using the 5 columns trick. Then you can do a single quicksort pivot on this element to partition the list ot the left for the n smallest elements.

   (c) The second smallest element in a binary min-heap with all elements with distinct values will always be a child of the root.
   Solution: True. The children of the root must be smaller than any of their children.

   (d) Any sorting algorithm on a set of 32 bit positive integers must have a run time complexity $f(N) \in \Omega(N \log(N))$.
   Solution: False. With a fixed range $\leq M$ (precision) there are more an repeat so you can use Bin sort histograming them into separate columns with a single pass $O(N)$ and then write the out $O(N)$ in sorted order.

   (e) There are instances of the integer knapsack problem which can be solved in polynomial time by a greedy algorithm.
   Solution: True. Simplest examples is when all of the tasks have unit capacity requirement.

   (f) Consider a binary search tree with n keys. Finding whether a key value is already in the tree can be done in $O(\log(n))$. **Solution:** False. The tree may not be balanced, so it is $O(n)$.

   (g) 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 $O(\log(n))$.
   Solution: True. This is for balanced trees.

(h) 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.

Solution: False. When $\gamma = k$ the leading solution is $\Theta(n^k \log(n)$. The solution n^k is subdominant.

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

Solution: False. It is not symmetric for directed graphs.

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

Solution: False. Dictionary to get average $\ln \ln(n)$ makes the assumption of approximate linear sequence of sorted number. If for example, you assume as sequence like $a[i] \sim 2^i$ for $i = 0, i, \cdots n-1$ it doesn't wor well – more like $\Omega(n)$.

(k) 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.

Solution: False. If either $d(i, k)$ or $d(k, j)$ was not a minimun, just replace it with the min and $d(i, j) = d(i, k) + d(k, j)$ is smaller!

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

Solution: False. As explained inclass there is not way to have both finds and joins $O(n)$ and $O(m)$. The theory says one mus have very small extra factor of the $\ln^*[n]$ function, which is number of time repeated application of $\log' s$ converges to less than 1

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

Solution: True. There are 2 with 1 as root, 2 with 3 as root, but only 1 with 2 as root.

(n) 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.

Solution: False. It can not remove negative cycles. If there are no negative cycles it removes all negative bonds.

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

Solution: False. This is typical greedy algorithm solutions method.

2. 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) Merge sort
   (b) Insertion sort
   (c) Quicksort
   (d) Radix Sort

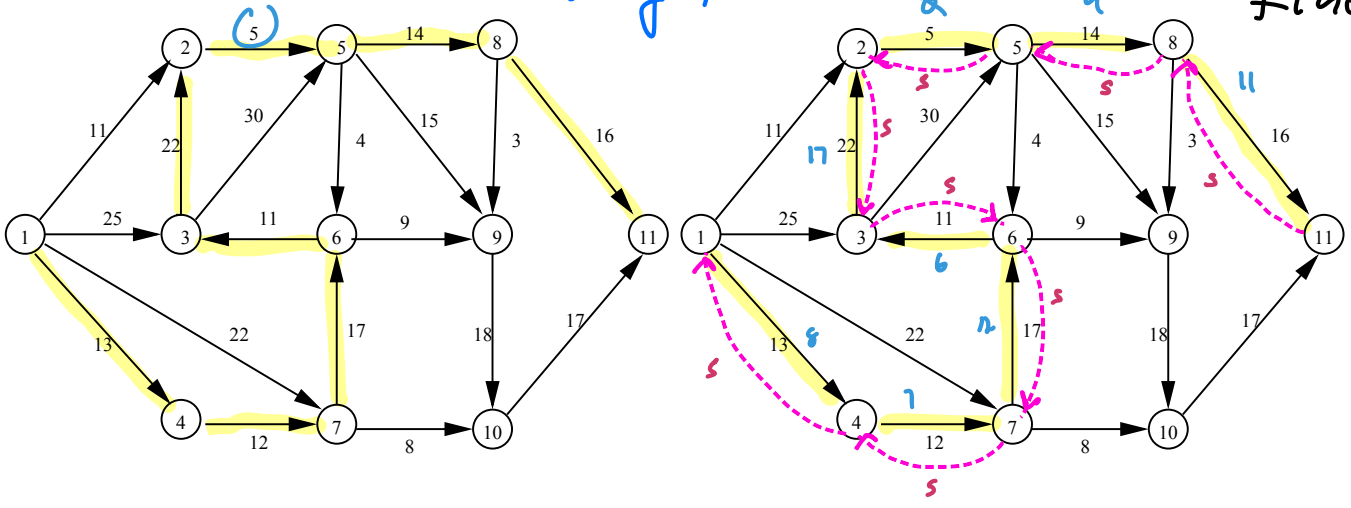
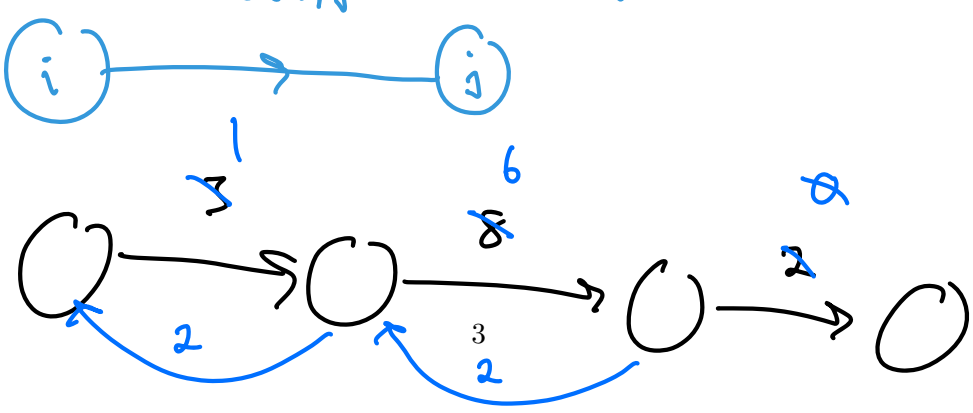


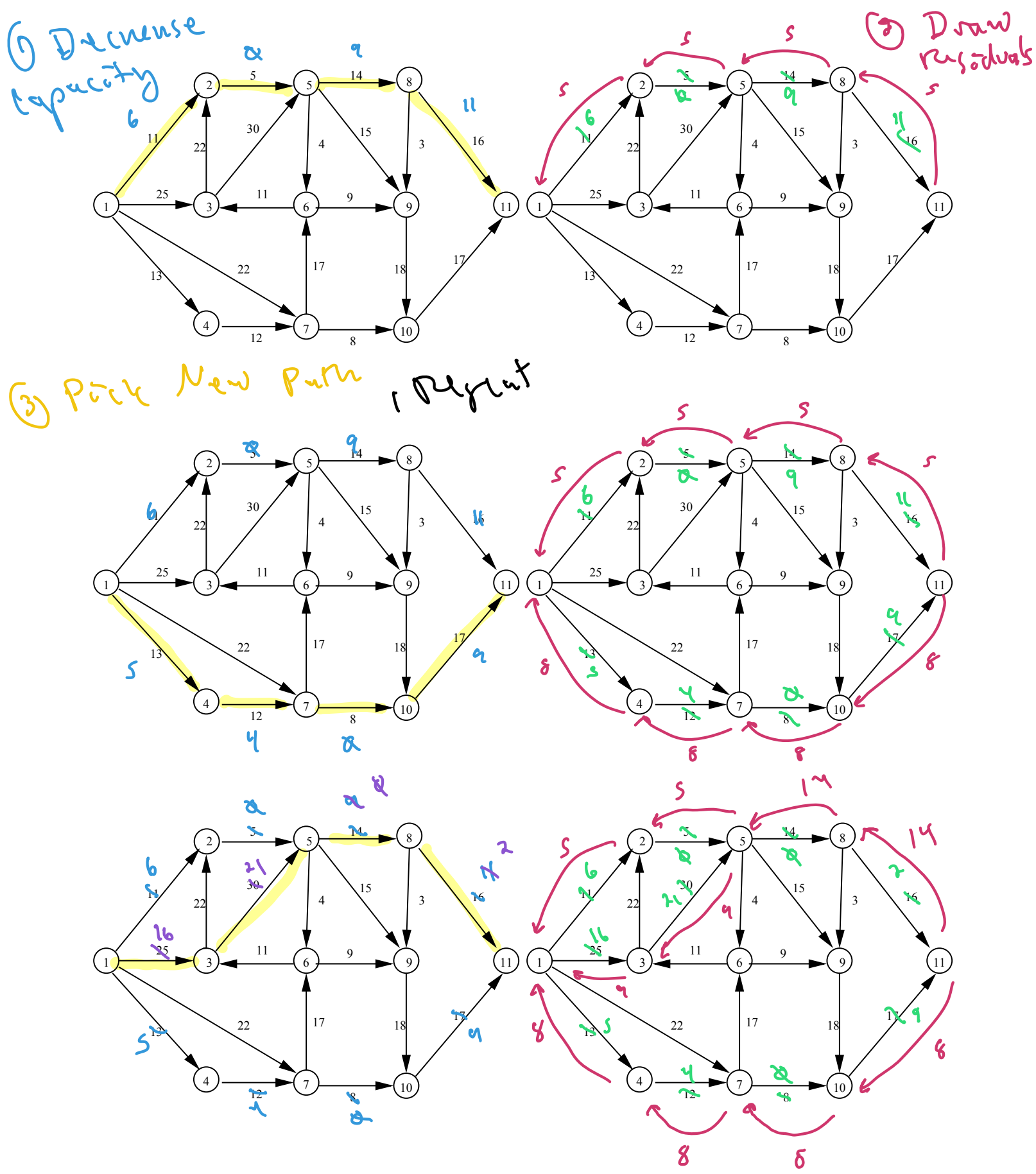
Figure 1:

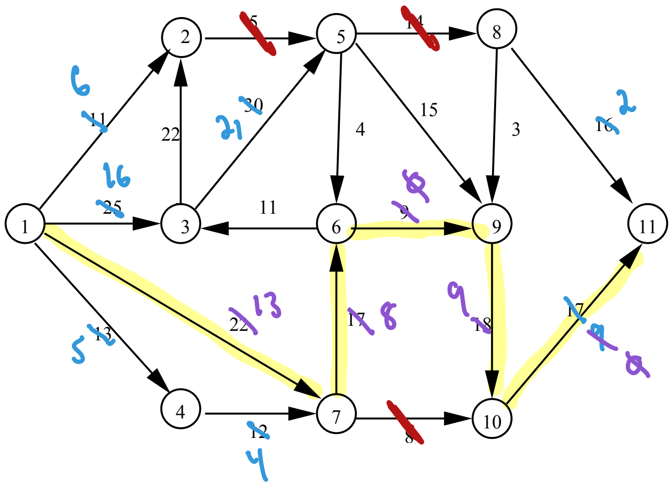
3. (15 pts) Consider Fig. 1 as a directed, capacitated graph, where the numbers on an arc now indicate an arc's capacity to carry flow from node 1 to node 11. In the max-flow algorithm of Ford and Fulkerson, the key step is, once a path has been found, to augment the flow and construct the residual graph for the next iteration.

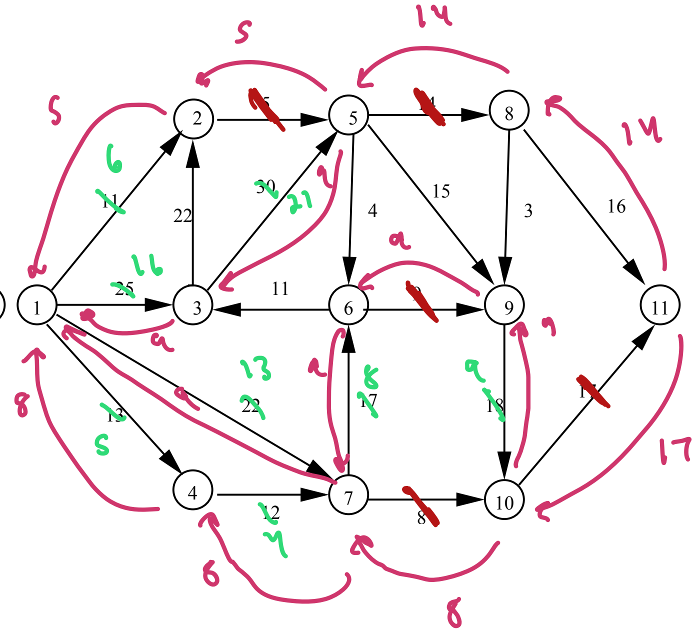
   (a) Suppose your program picks the first augmentation path to be $1 \to 4 \to 7 \to 6 \to 3 \to 2 \to 5 \to 8 \to 11$. Draw the augmented flow path on the left graph above and the residual graph above in the right side. What is the capacity of this path?

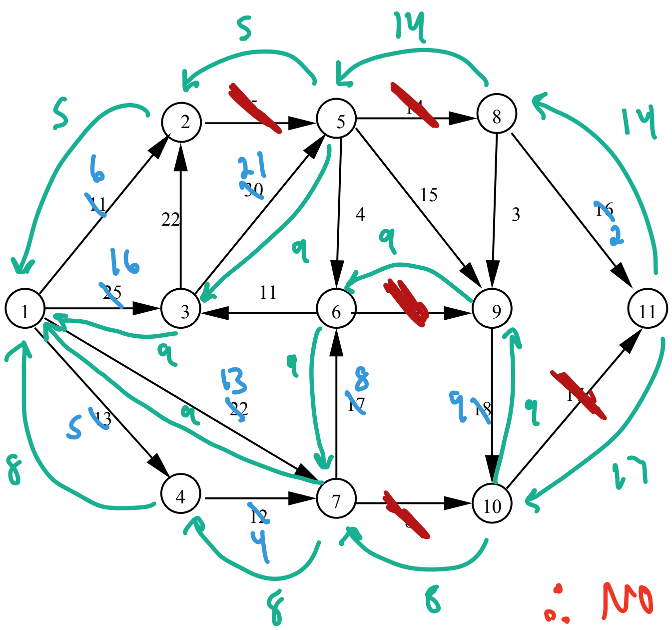
   (b) Now be smarter, starting with shorter paths, beginning with a min hop $1 \to 2 \to 5 \to 8 \to 11$ for the first path and $1 \to 4 \to 7 \to 10 \to 11$ for the second path. Proceeding to usng short augmentation paths to bring you to maximun flow. Draw **all** flow graphs and **all** the residual and after **each** augmentation. What is the maximum flow? Draw the minimum cut to show this right.

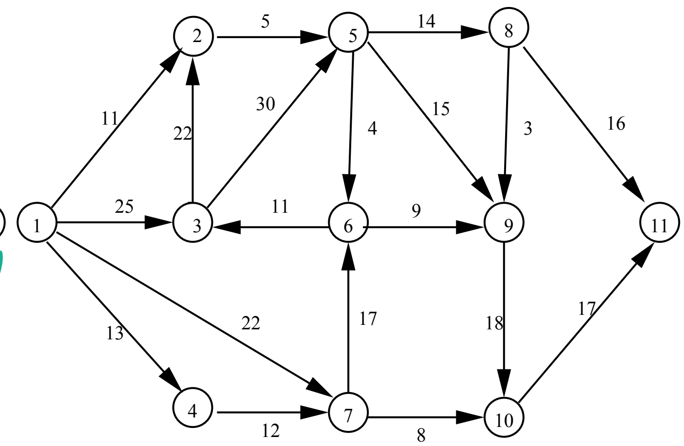



1 Decrease capacity
2 Draw residuals
3 Pick New Path
1 Right

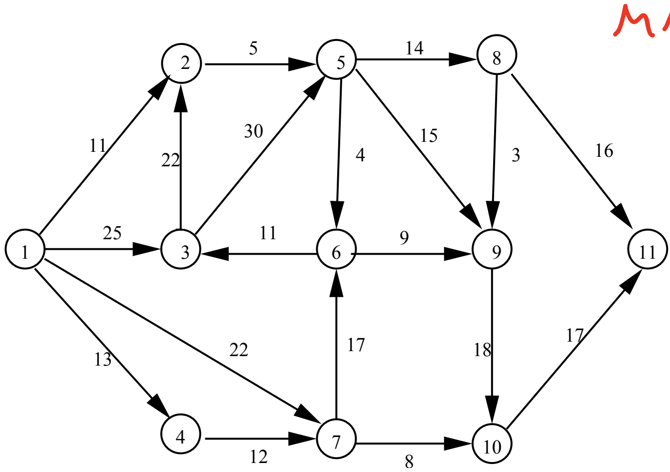


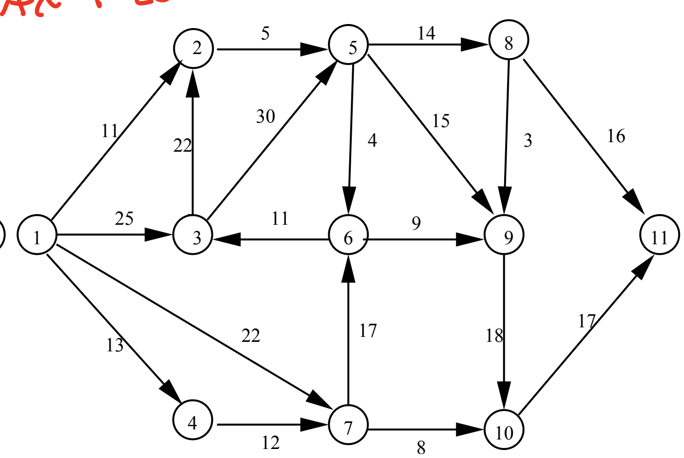






∴ NO ADDITIONAL PATHS —
MAX FLOW = 14 + 17 = 31





4. 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]

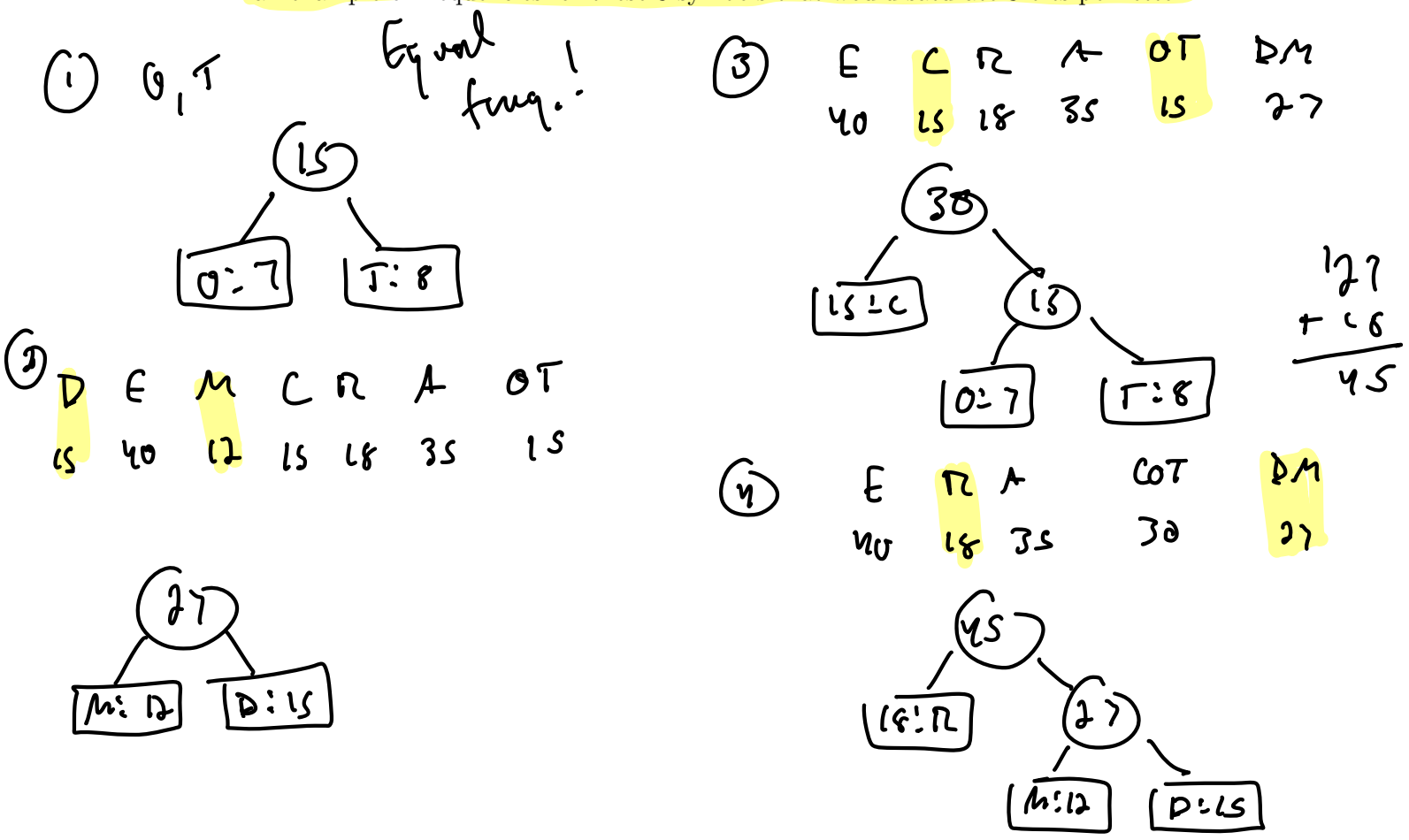
NEED TO BREAK TIES

alphabet:	D	E	M	O	C	R	A	T
frequency:	15	40	12	7	15	18	35	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!) **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] From the ancient Greek prefex: demotic =*common people* + suffex: -kratia = *strength* . In late 18th century (originally denoting an opponent of the *aristocrats* in the French Revolution of 1790): from French *democrate*, on the pattern of aristocrate *aristocrat*.

(5)

E	A	COT	RDM
40	35	30	45

(6)

E	ACOT	RDM
40	65	45

(7)

ACOT	ERDM
65	85

Symbol	Encoding	BPS	Freq
D	1111	4	15/150
E	01	2	40/150
M	1110	4	12/150
O	0010	4	7/150
C	000	3	15/150
R	011	3	18/150
A	10	2	35/150
T	1100	4	8/150

$$ABPS = \frac{1}{150}\left[4(15) + 2(40) + 4(12) + (2)(7) + 3(15) + 3(18) + 2(35) + 4(8)\right]$$

$$= \frac{1}{150}\left[60 + 80 + 48 + 28 + 45 + 54 + 70 + 32\right]$$

$$= \frac{1}{150}\left[210 + 80 + 45 + 101\right]$$

$$= \frac{1}{150}\left[276 + 141\right] = \frac{417}{150} = 2.78$$

Correction to Error on Next Page

5. 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:

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

Val/Sz 10 11 6 4.33 3.5 5

(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.

a) Obj: 2 & 1 & 3 & 4 & 6 & 5*

Value: 11 + 10 + 12 + 13 + 15 + $\frac{1}{2}$(14) = 21 + 25 + 15 + 3.5 = 64.5

Size: 1 + 1 + 2 + 3 + 3 + 1

b) Int Knapsack: 2 & 1 & 3 & 5 & 6 => 1 + 1 + 2 + 3 + 4 = 11

Val - 11 + 10 + 12 + 14 + 15 = 21 + 26 + 15 = 62

6. (15 pts) Consider searching in the "text" $T(1 : N)$ of length $N = 25$

5. 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:

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

Val/Sz 10 11 6 4.33 3.5 5

(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.

a) Obj: 2 & 1 & 3 & 4 & 6 & 5*

Value: 11 + 10 + 6 + 13 + 15 + $\frac{1}{4}$(14) = 45 + 3.5 = 48.5

Size: 1 + 1 + 2 + 3 + 3 + 1 = 11

b) Int Knapsack: 2 & 1 & 3 & 4 & 6 => 1 + 1 + 2 + 3 + 3 = 10

Val = 45

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
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

q=2 q=3 q=1 q=1 q=π[3]=1

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

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!)

[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$P(1:k) = P(1+q-k:q)\}$.**

(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. I don't think there are any ...?

a)

i	1	2	3	4	5	6	7	8
P[i]	B	A	C	B	A	B	A	C
π[i]	0	0	0	1	2	1	2	3

$\pi[q] = \max\{k < q : P[1:k] = P[1+q-k : q]\}$

$\pi[1] = \max\{k < 1 : P[1:k] = P[2-k:1]\}$

$k = 0$

$\pi[2] = \max\{k < 2 : P[1:k] = P[3-k:2]\}$

$k = 0, 1$

$\pi[3] = \max\{k < 3 : P[1:k] = P[4-k:3]\}$

$k = 0, 1, 2$

$\pi[4] = \max\{k < 4 : P[1:k] = P[5-k:4]\}$

$k = 0, 1, 2, 3$

$\pi[5] = \max\{k < 5 : P[1:k] = P[6-k:5]\}$

$k = 0, 1, 2, 3, 4$

$\pi[6] = \max\{k < 6 : P[1:k] = P[7-k:6]\}$

$k = 0, 1, 2, 3, 4, 5$

π(matching i) = max # of matches after shifts

Shift $i - \pi[i]$

$$\pi[7] = \max\{k < 7 : P[1:k] = P[8-k:7]\}$$

✓ ✗ ✓ ✗ ✗ ✗ ✗

$k = 0, 1, 2, 3, 4, 5, 6$

$$\pi[8] = \max\{k < 8 : P[1:k] = P[9-k:8]\}$$

✓ ✗ ✗ ✓ ✗ ✗ ✗ ✗

$k = 0, 1, 2, 3, 4, 5, 6, 7$

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	~~0~~	~~0~~	1	2	3	~~0~~	1

$\pi[q] = \max\{k < q : P[1:k] = P[1+q-k : q]\}$

$\pi[1] = \max\{k < 1$

$\pi[2] = \max\{k < 2 : P[1:k] = P[3-k : 2]\}$

$\pi[3] = \max\{k < 3 : P[1:k] = P[4-k : 3]\}$

$k = 0, 1, 2$

$\pi[4] = \max\{k < 4 : P[1:k] = P[5-k : 4]\}$

$k = 0, 1, 2, 3$

$\pi[5] = \max\{k < 5 : P[1:k] = P[6-k : 5]\}$

✓ ✓ ✗ ✓ ✗

$k = 0, 1, 2, 3, 4$

$\pi[6] = \max\{k < 6 : P[1:k] = P[7-k : 6]\}$

✓ ✗ ✗ ✗ ✗

$k = 0, 1, 2, 3, 4, 5$

$\pi[7] = \max\{k < 7 : P[1:k] = P[8-k : 7]\}$

Example from book!

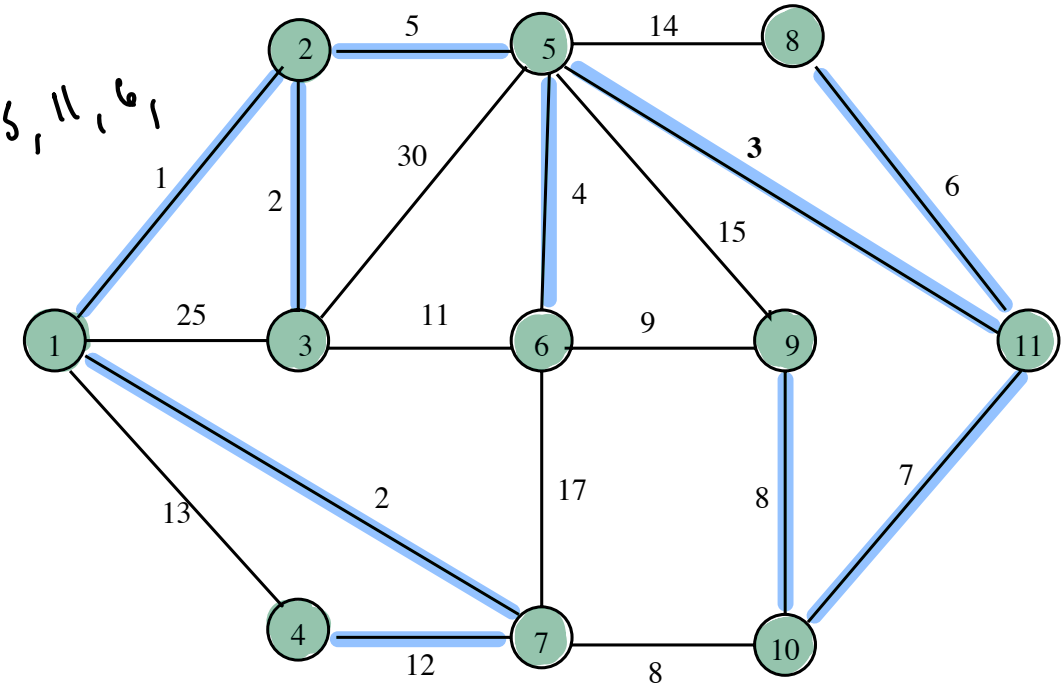


Figure 2:

7. 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.

   (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

[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!**

Dijkstra's

distance / parent

(1,2) d=1 ✓

(1,3) d=25 ✗

(1,7) d=2 ✓

(1,11) d=13 ✓

(2,3) d=3 ✓

(2,5) d=6 ✓

(7,6) d=14 ✗

(7,10) d=10 ✓

(7,14) d=14 ✗

(3,5) d=33 ✗

(3,6) d=14 ✗

(5,6) d=10 ✓

(5,8) d=20 ✗

(5,9) d=21 ✗

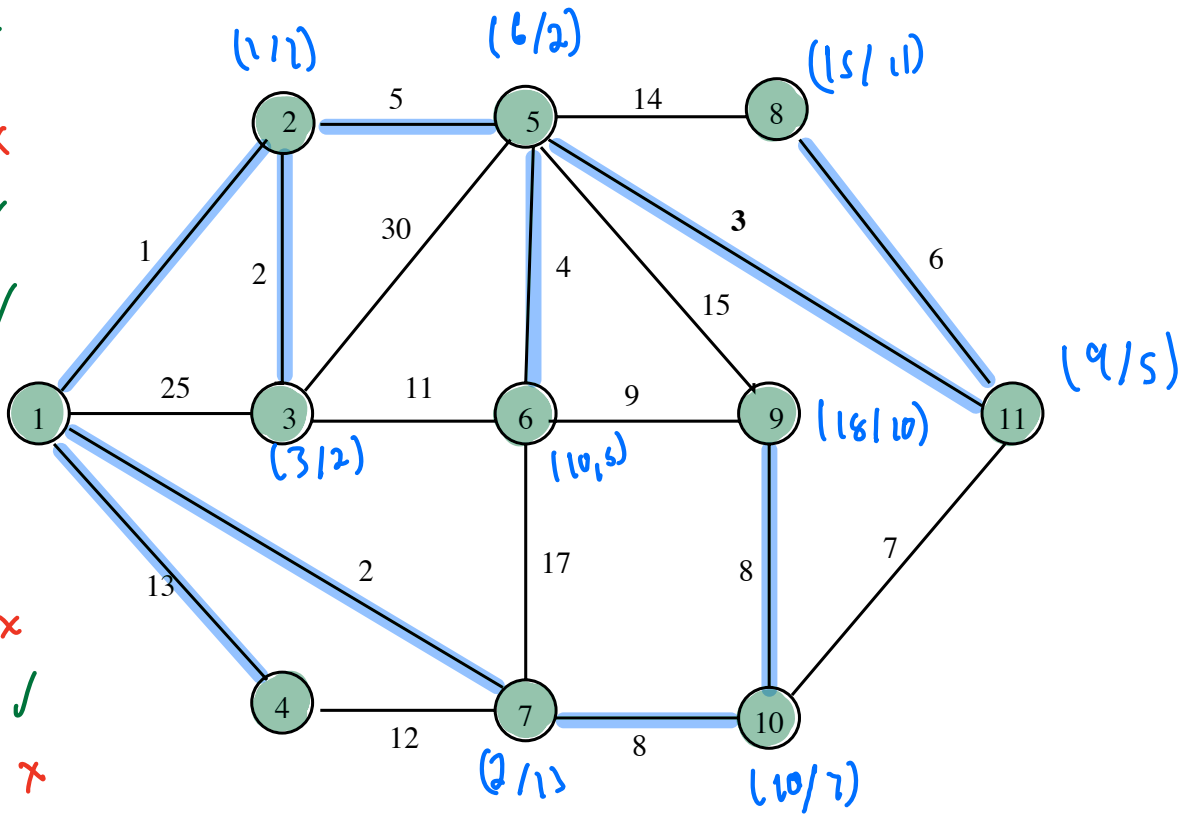
(5,11) d=9 ✓

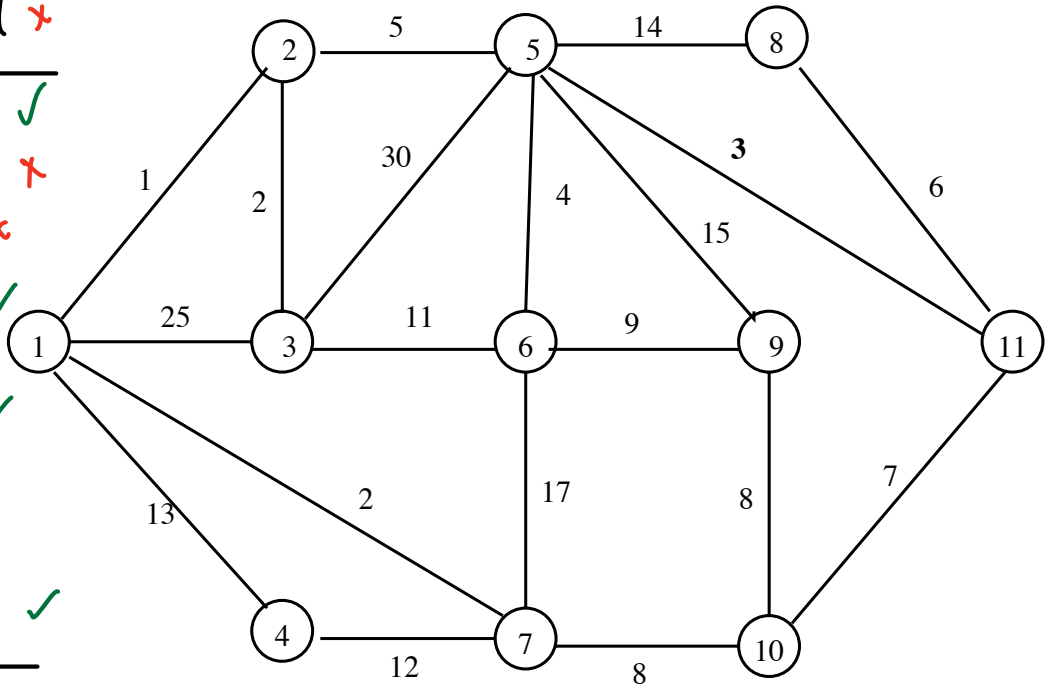
(11,8) d=15 ✓

(11,10) d=16 ✗

(10,9) d=18 ✓

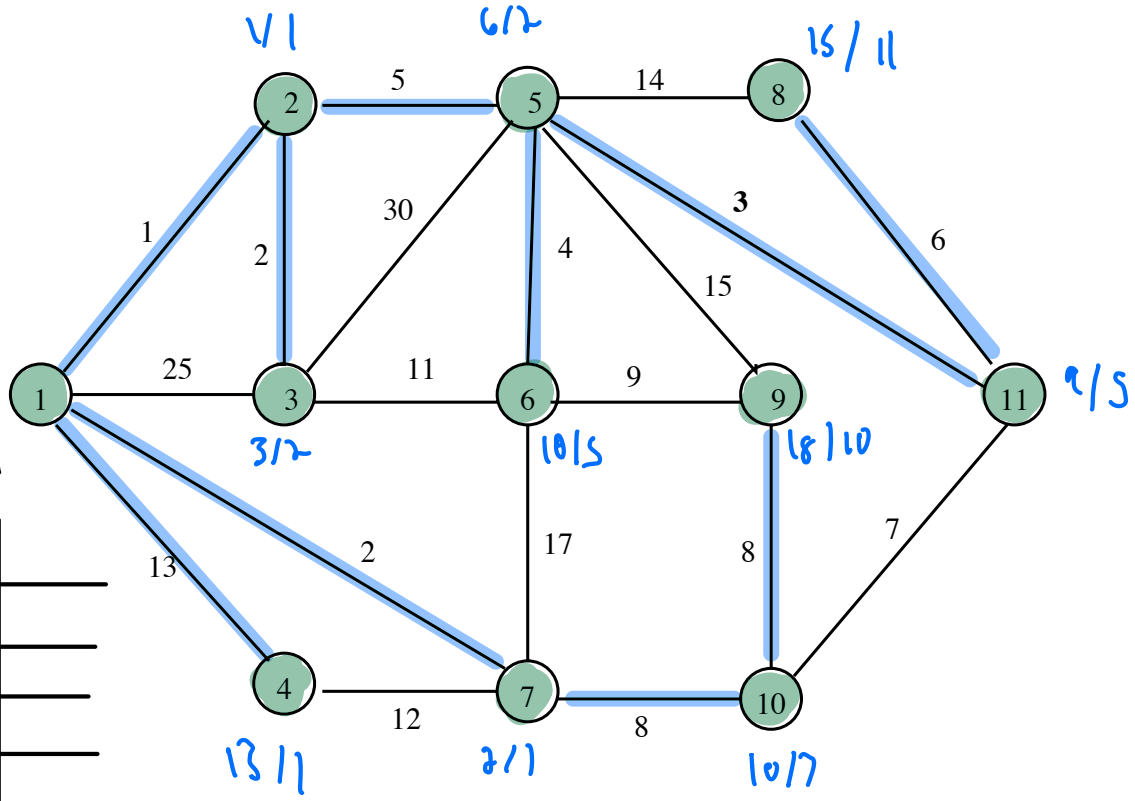
(6,9) d=19 ✗





Bellman-Ford

distance / parent



iteration

N	1	2	3	4
1	∞	∞	∞	∞
2	1	1	1	1
3	25	3	3	3
4	13	13	13	13
5	∞	6	6	6
6	∞	19	10	10
7	2	2	2	2
8	∞	∞	20	15
9	∞	∞	18	14
10	∞	10	10	10
11	∞	∞	9	9