*6,7,8

Matthew Chin

1 2 3 4 5 6 7 8 ?

CMSC 351 Summer 2025 Homework 8

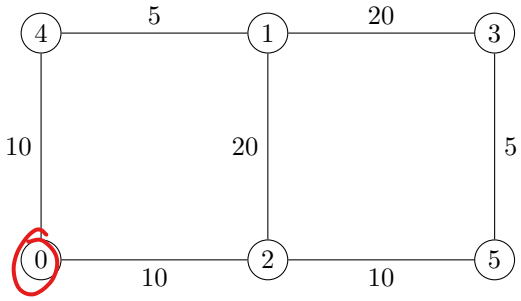
Due Tuesday 12 August 2025 by 23:59 on Gradescope.

Directions:

- Homework must be done on printouts of these sheets and then scanned properly, or via Latex, or by downloading, writing on the PDF, and uploading. If you use Latex please do not change the Latex formatting.
- Do not use your own blank paper!
- The reason for this is that Gradescope will be following this template to locate the answers to the problems so if your answers are organized differently they will not be recognized.
- Tagging is automatic, you will not be able to manually tag.

---

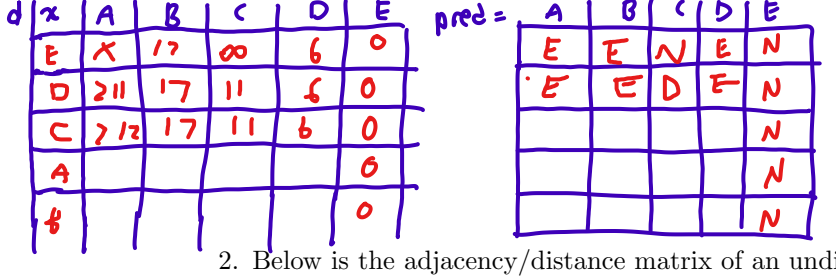
1. We run Dijkstra's Algorithm on the following graph starting at vertex 0. [20 pts]



Show the DIST and PRED lists after each iteration of Dijkstra's Algorithm. In DIST use ∞ for infinity and in PRED use N for $NULL$.

If you find you can choose between several vertices always choose the one with smallest index.

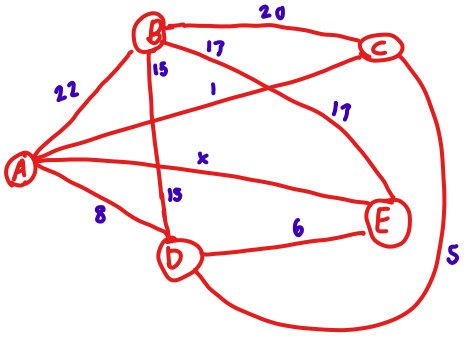
	Iter	DIST 0	DIST 1	DIST 2	DIST 3	DIST 4	DIST 5	PRED 0	PRED 1	PRED 2	PRED 3	PRED 4	PRED 5	
pick 0 →	1	0	∞	10	∞	10	∞	N	N	0	N	0	N	S: { }
pick 2 →	2	0	30	10	∞	10	20	N	2	0	N	0	2	S = {0}
pick 4 →	3	0	15	10	∞	10	20	N	4	0	N	0	2	S = {0,2}
pick 1 →	4	0	15	10	35	10	20	N	4	0	1	0	2	S = {0,2,4}
pick 5 →	5	0	15	10	25	10	20	N	4	0	5	0	2	S = {0,1,2,4}
pick 3	6	0	15	10	25	10	20	N	4	0	5	0	2	S = {0,1,2,4,5}



d[D] + w(D,C) < d[C]?
6 + 5 = 11<inf? yes so d[A] = x > 11
d[A] = min(x,14)
d[D] + w(D,A)

examine C: d[A] = d[C] + w(C,A)<d[A]
11 + 1 = 12 < d[A]
12 < x > 11 so x > 12

2. Below is the adjacency/distance matrix of an undirected weighted graph. The variable x is a positive integer and all the weights are different from one another:



	A	B	C	D	E
A	0	22	1	8	x
B	22	0	20	15	17
C	1	20	0	5	∞
D	8	15	5	0	6
E	x	17	∞	6	0

E - 6 - D
E - 17 - B
E - x - A
E - inf - C

E?C->A

After running Dijkstra's algorithm starting at vertex E, we have come up with the following:

Fact: When going from E to A, the predecessor of A is C.

Answer the following - you don't need to do (a) before (b); it might be better to work them out at the same time.

(a) What can you say about x? Your answer could be an inequality, set of inequalities, etc. Be as precise as possible. [5 pts]

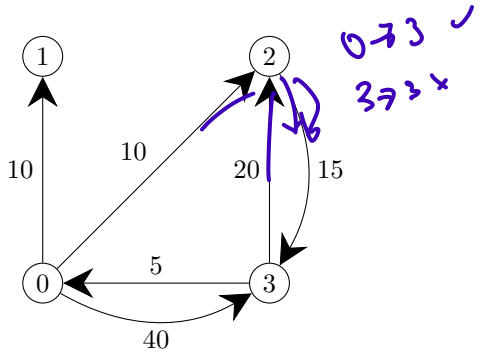
What is true of x?	X > 12

(b) In order, after E, what vertices were chosen second, third, fourth, and fifth? [5 pts]

First	Second	Third	Fourth	Fifth
E	D	C	A	B

Scratch work; not graded:

3. We run Floyd's Algorithm on the following graph starting at vertex 0. [16 pts]



Show the DIST and PRED matrices after the pass-by 0 and 2 of Floyd's Algorithm. In DIST use ∞ for infinity and in PRED use N for NULL.

Pass by 0: allow 0 as intermediate vertex:

DIST				
	0	1	2	3
0	0	10	10	40
1	∞	0	∞	∞
2	∞	∞	0	15
3	5	15	15	0

PRED				
	0	1	2	3
0	0	0	0	0
1	N	1	N	N
2	N	N	2	2
3	3	0	0	3

Pass by 2: allow ② as an intermediate vertex (Pass by 1 makes no changes)

DIST				
	0	1	2	3
0	0	10	10	25
1	∞	0	∞	∞
2	∞	∞	0	15
3	5	15	15	0

PRED				
	0	1	2	3
0	0	0	0	2
1	N	1	N	N
2	N	N	2	2
3	3	0	0	3

4. Suppose a graph has four vertices indexed 1 to 4 and we run Floyd's algorithm on it. After the Pass By 2 step the distance matrix is on the left below. On the right, fill in the specified values. [10 pts]

Pass By 2 Distance Matrix				
	1	2	3	4
1	0	10	15	30
2	∞	0	5	20
3	∞	10	0	5
4	∞	∞	15	0

Calculate the Following Distances	
After the Pass By 3 step, $D[1,2] =$	10
After the Pass By 3 step, $D[1,4] =$	20
After the Pass By 3 step, $D[4,2] =$	25

Put scratch work below; Scratch work is not graded:

Pass by 3: check d[1,2]

d[1,3] + d[3,2] < d[1,2] ?

15 + 10 <

25 ≮ 10 ✗ no update

check d[1,4]

d[1,3] + d[3,4] < d[1,4] ?

15 + 5 < 30 ?

20 < 30 ✓ update d[1,4] to 20

check d[4,2]:

d[4,3] + d[3,2] < d[4,2] ?

15 + 10 < ∞ ?

25 < ∞ ✓ update d[4,2] to 25

via piazza@279

5. Suppose that we modified Floyd's algorithm as follows:

pred[i][j] = k

We replace `pred[i][j]=pred[k][j]` with `pred[i][j]=pred[i,k]`

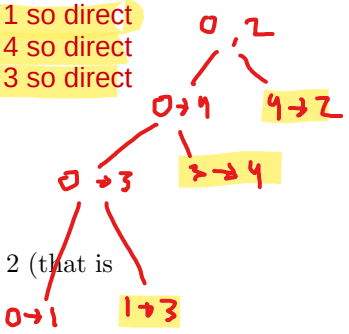
Note that `pred[i][j]` no longer gives the predecessor of `j` on the shortest path from `i` to `j`.

(a) We run this modified Floyd's Algorithm on a graph with five vertices and the final resulting `pred` matrix is as follows: [5 pts]

	0	1	2	3	4
0	0	0	4	1	3
1	4	1	4	1	3
2	N	N	2	N	N
3	4	3	4	3	3
4	4	0	4	1	4

i→k→j

p[0,2] = 4 0->4, 4->2
p[0,4] = 3 0->3, 3->4
p[0,3] = 1 0->1, 1->3
p[1,3] = 1 so direct
p[4,2] = 4 so direct
p[3,4] = 3 so direct



What is the shortest path from 0 to 2? Answer as a list of vertices such as $0, 1, 2$ (that is not the correct answer).

Shortest Path	0,1,3,4,2

(b) Suppose we run this modified Floyd's algorithm on a graph with four vertices. Explain why it is not possible for the final resulting `pred` to be this matrix, where the ? are unknown: [10 pts]

	0	1	2	3
0	0	2	3	?
1	?	1	?	?
2	?	3	2	?
3	?	?	?	3

Why Not?

not possible because it contradicts the rule
w/ the modified rule, the value at p[i][j] after the entire algorithm contains the greatest optimal-index vertex w/ respect to d[i][j] that made the path i->j more optimal (a shorter path)
if a later vertex that is processed improves some path, it will also optimize the whole larger path that uses that part of a path, so both should get their pred[][] updated to the vertex that is currently being examined.

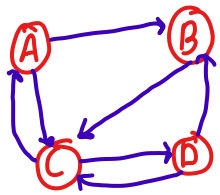
So in this example, pred[0][2] = 3 which means that the vertex 3 improves the path 0->2 at the last step.
but the path 0->1 had 2 as an intermediate vertex so if examining the vertex 3 improved the 0->2 path, 0->1 should also be improved, so pred[0][1] should be 3 and not 1

6. We apply Floyd's Algorithm to a simple, directed graph and we find the following predecessor matrix with some missing entries. [10 pts]

Suppose we know for sure that there is only one shortest path from any vertex to any other vertex. Fill in the missing entries.

PRED				
	A	B	C	D
A	A	A	B	C
B	C	B	B	C
C	C	D	C	C
D	C	D	B	D

Scratch Work; Scratch Work is Not Graded:



Here i have drawn the shortest paths, and the problem says that there is only one shortest path from one vertex to any other vertex.
So i can fill in the rest of the entries in the PRED list using these shortest paths in purple?

A?C->D

A->?->B->C->D
A->B->C->D

A→C→D
A→B→C
B→C→A
D→C→A
C→D→B
D→B→C

Q: blue entries are by deduction. is this deduction possible b/c of the "there is only one shortest path..."?

can fill in diagonal rows

B?C->A

C->A

C?D->B

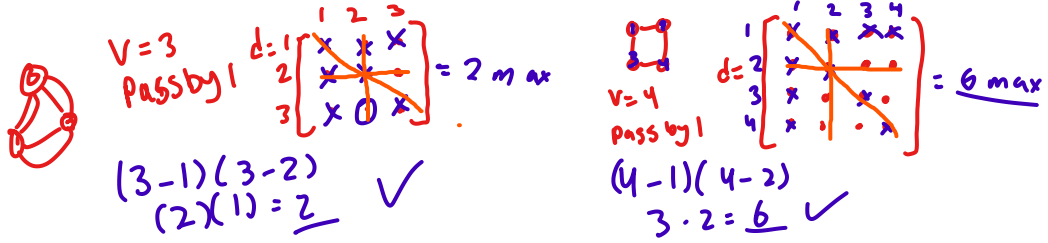
C?D->B

D?C->A

D?B->C

= D->B->C->A

new sol: trace from graph assuming pred[i][j] means path is i->???->pred[i][j]->j



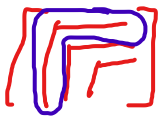
$v^2 - 2v - 1$
$= (v-1)^2$

7. Suppose we apply Floyd's Algorithm to a graph with $V \geq 2$ vertices.

   (a) Without knowing anything about the graph, what is the maximum number of predecessor matrix entries which could possibly be updated during one pass-by iteration? [3 pts]

Maximum Number =	2v-1

2v-1



   (b) Explain your answer to (a). [6 pts]

   Solution:

in floyds algorithm you have max V^2 entries in the pred matrix.
so there are max v^2 entries able to updated, but you have to subtract certain entries.
- You can never update the entries in row n because those already have predecessor of n
- You can also never update entries in column n because adding n as an intermediate vertex does not change the shortest path between pairs of vertices that already lead towards n
  -> this is 2V - 1 (minus 1 to avoid double counting the value at pred[n][n] which gets 2v-1

one of the TA's told me to disregard the fact that diagonal entries can never be updated, because I originally counted those in the total. However, if we count those in, that is another V-1 entries (minus 1 again to avoid triple counting pred[n][n]). This would yield a total of
v^2 - v - 1 - 2v - 1 = v^2 - 3v - 2 = (v-1)(v-2)
so i think that the actual total would be (v-1)(v-2) but if the problem doesn't want us to count the diagonal entries then this would just be 2v-1 i think?

8. The following line in Dijkstra's Algorithm may involve making a choice when there are many vertices with smallest distance:

```
x = vertex in G-S with smallest distance
```

in G and Not in S{}

Suppose a graph has V vertices.

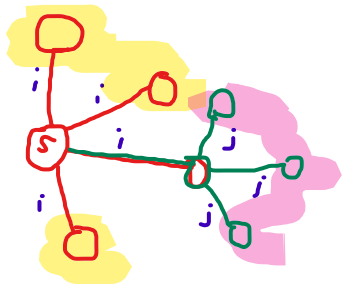
The first time this line is encountered there is one choise, the starting vertex. Suppose the second time this line is encountered there are k choices where $2 \leq k \leq V - 1$.

   (a) How many choices will there be the third time this line is encountered? [3 pts]

How Many?	K-1

   (b) Explain your answer to (a). [7 pts]

   Solution:



I suppose that the idea here is that you would have k vertices which are the same distance from s. So you would want to choose one of these k vertices this happens at the second iteration.
So at the third iteration, we have the 3 original i minimum distance (arbitrary value) and the possibly new nodes in play which are differently j distance away, but the choices that you have to choose the new x is from the same set from before of the i distance nodes minus the one that you just examined in the previous 2nd iteration so that would be k-1