CS340 => Assignment 7

Nov. 18
2022

Problem 1)

(a)

K -> Known
D -> Distance
P -> Previous

V	K	D	P
A		(0)	
B		∞	
C		∞	
D		∞	
E		∞	
F		∞	
G		∞	

V	K	D	P
A	Y	0	
B		5	A
C		(3)	A
D		∞	
E		∞	
F		∞	
G		∞	

V	K	D	P
A	Y	0	
B		(5)	A
C	Y	3	A
D		10	C
E		10	C
F		∞	
G		∞	

V	K	D	P
A	Y	0	
B	Y	5	A
C	Y	3	A
D		10	C
E		8	E
F		∞	
G		(6)	B

V	K	D	P
A	Y	0	
B	Y	5	A
C	Y	3	A
D		10	C
E		(7)	G
F		∞	
G	Y	6	B

V	K	D	P
A	Y	0	
B	Y	5	A
C	Y	3	A
D		9	E
E	Y	7	G
F		(8)	E
G	Y	6	B

V	K	D	P
A	Y	0	
B	Y	5	A
C	Y	3	A
D		(9)	E
E	Y	7	G
F	Y	8	E
G	Y	6	B

V	K	D	P
A	Y	0	
B	Y	5	A
C	Y	3	A
D	Y	9	E
E	Y	7	G
F	Y	8	E
G	Y	6	B

V	K	D	P
A			
B			
C			
D			
E			
F			
G			

Shortest path from A to Others.

A: [A] -> 0
B: [A,B] -> 5
C: [A,C] -> 3
D: [A,B,E,D] -> 9
E: [A,B,G,E] -> 7
F: [A,B,G,E,F] -> 8
G: [A,B,G] -> 6

(b)

V	dQ	D	P
A		∞	
B		(0)	
C		∞	
D		∞	
E		∞	
F		∞	
G		∞	
q	B		

V	dθ	D	P
A		∞	
B	Y	0	
C		(1)	B
D		∞	
E		1	B
F		∞	
G		1	B
q	C,E,G		

V	dθ	D	P
A		∞	
B	Y	0	
C	Y	1	B
D		2	C
E		(1)	B
F		∞	
G		1	B
q	E,G,D		

V	dθ	D	P
A		∞	
B	Y	0	
C	Y	1	B
D		2	C
E	Y	1	B
F		2	E
G		(1)	B
q	G,D,F		

V	dθ	D	P
A		(∞)	
B	Y	0	
C	Y	1	B
D		(2)	C
E	Y	1	B
F		2	E
G	Y	1	B
q	D,F		

V	dθ	D	P
A		3	D
B	Y	0	
C	Y	1	B
D	Y	2	C
E	Y	1	B
F		(2)	E
G	Y	1	B
q	F,A		

V	dθ	D	P
A		(3)	D
B	Y	0	
C	Y	1	B
D	Y	2	C
E	Y	1	B
F	Y	2	E
G	Y	1	B
q	A		

V	dθ	D	P
A	Y	3	D
B	Y	0	
C	Y	1	B
D	Y	2	C
E	Y	1	B
F	Y	2	E
G	Y	1	B
q			

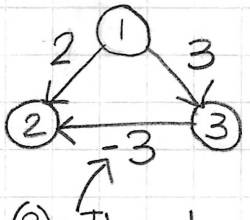
V	dθ	D	P
A			
B			
C			
D			
E			
F			
G			
q			

A: [B, C, D A] -> 3
B: [B] -> 0
C: [B, C] -> 1
D: [B, C, D] -> 2
E: [B, E] -> 1
F: [B, E, F] -> 2
G: [B, G] -> 1

Problem 2)

(1) no cycle with negative cost.
(2) G has an edge with negative cost.
(3) Dijkstra's Algorithm does not work on the example.

Example)



(1) this does not have a cycle, regardless once you get to 2, you cannot go any where - all the paths lead to 2.

(2) This has a negative edge therefore the example works.

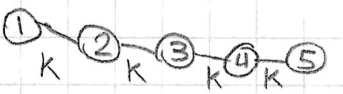
(3)

V	K	D	P
1	Y	0	
2	Y	2	
3		3	

After the first step the algorithm would set "2" to known, this is incorrect since we can see there is a possible path of 0. in [1,3,2] instead on [1,2]

Problem 3)

As I am interpreting this question it is to achieve the given running time. $(O(k \cdot |V| \cdot |E|)$



$5-1=4$
$(V-1)\cdot 5 = 4k.$

We are given that each edge is a integer $\{1,2 \ldots k\}$. The shortest path that can be generated would have at best 0 edges and at most V-1 edges. Therefore the possible lenghts are: 0 to $(V-1)\cdot k_{max}$ or INF.

For storage we use a "bucket idea" by creating a mutidimensional array storing vertices with the respective distances. The buckets would be: $[0, k\cdot(V-1)]$ at the end a sentinel value can be added to sort unconnected vertices such as $(k\cdot(V-1)+1)$.

algorithm:

$O(V)$ (1) Put everything in INF bucket except for start that goes to 0.

$O(k\cdot(V-1))$ (# buckets)

(2) Start with bucket array at bucket of 0.

(3) while bucket < INF bucket AND bucket empty (Arrray getting stuck)

(4) increment to the next bucket in array.

$O(E)$

(5) for each vertex in bucket

(6) for each path out of vertex, if its in a larger bucket

(7) Set bucket to smaller found one. (move it down.)

(8) set current vertex previouse to vertex start bucket.

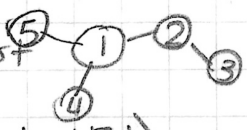
(9) move onto the next bucket.

$= (O(E) + O(k\cdot V)) + O(k)$

$\therefore = O(k\cdot V + E) \Rightarrow \boxed{O(k\cdot|V| + |E|)}$

Problem 4)

(a) The graph is undirected which means that all of the relationships are two-way, and no cycles mean that every vertex only has 1 path. Example



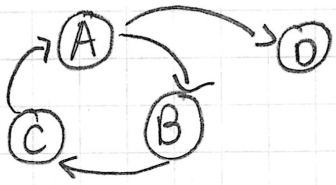
The algorithm we use is a breadth search first with a queue implementation.

Worstcase: $O(|V|+|E|)$

(1) Start with an empty queue, and distance 0. $O(1)$ (+ distance array (key-pairs))
(2) Choose the S, and insert it into queue. $O(1)$
(3) While the queue is not empty $O(|V|+|E|)$
  (4) remove the vertex with the queue
  (5) add all neighbour vertexes, with previous distance plus 1. $O(1)$
  (6) mark vertex visited

$= O(|V|+|E|)$ the total number of runs will be vertex and edges as we see a new vertex we enque while an edge deques.

(b) If direction was important due to cycling we would have to check every possible path, from S to V, even infinite if looping.



[A, B, C, A, D]
[A, D]
[A, B, C, AB, C, D]

} this can keep going forever and hard to check for.

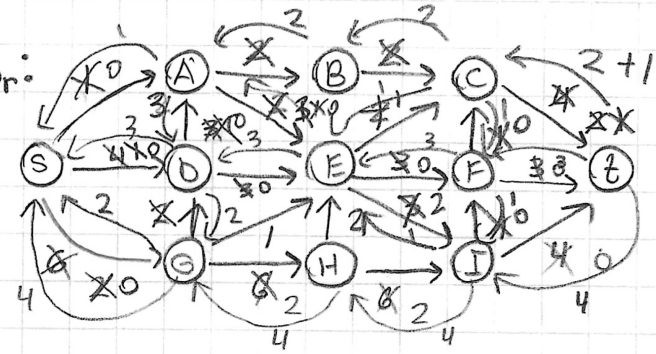
If this was accounted for we would need to check every path combination which is factorial.

↳ $\Omega(|V|!) \rightarrow \Omega(V \cdot (V-1) \cdot (V-2) \ldots)$

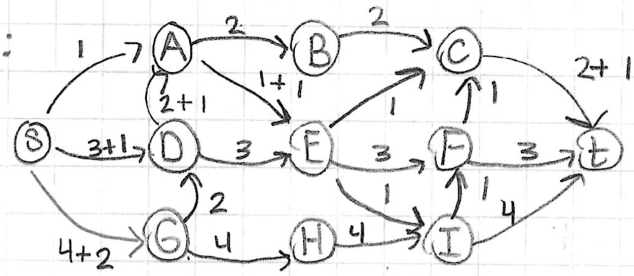
Problem 5)

Working Graphs

G_r:



G_f:

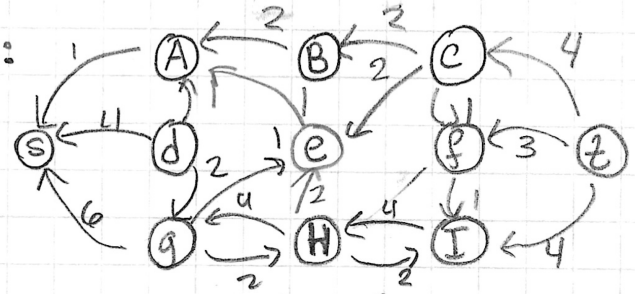


$4+4+3=11$?

Max flow Confirmed. $\Rightarrow 11$

1) flow 4
2) flow 3
3) flow 2
4) flow 1
5) flow 1

G_r:



G_f: