University of Nevada, Las Vegas Computer Science 477/677 Fall 2021

Assignment 6: Due Monday November 1, 2021 11:59 pm

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. Turn in the completed assignment on canvas, using instructions given to you by the grader, Mr. Heerdt, by 11:59 PM November 1.

"Pseudocode" means "false code." It is supposed to represent a program in such a way to be humanly understandable, but need not follow any particular rules. Here is pseudocode for selection sort.

```
    // Selection sort on an array A of size n.  Indices run from 1 to n.
For all i from 1 to n-1:   // select the i^th smallest item
                           // place it in A[i]
  For all j from i+1 to n: // replace A[i] by A[j] if it is smaller
    If(A[j] < A[i])
      Swap(A[i],A[j])      // we assume the code for Swap is available
```

In Problems we assume that G is a weighted directed graph with vertices numbered 0 through n-1. We let $E[i, j]$ be the weight of the arc from vertex i to vertex j. If no such arc exists, $E[i, j] = \infty$ by default.

1. Let G be defined by the following array of out-neighbor lists. If list i contains the entry (j,w), there is an arc from i to j of weight w.

0 :	$(1, 5)(3, -1)$
1 :	$(0, 1)(2, 4)$
2 :	$(0, 2)(3, 5)$
3 :	$(5, 3)(4, 1)$
4 :	$(5, 1)$
5 :	$(4, 2)$

Fill in the array of in-neighbor lists of G.

0 :	$(1, 1), (2, 2)$
1 :	$(0, 5)$
2 :	$(1, 4)$
3 :	$(0, -1), (2, 5)$
4 :	$(3, 1), (5, 2)$
5 :	$(3, 3)(4, 1)$

Identify the strong components of G.

$\{0, 1, 2\}$, $\{3\}$, $\{4, 5\}$

2. Write pseudocode for the Bellman-Ford algorithm to solve the single source minpath problem for G, where vertex 0 is the source. Your output should be two arrays of size n:

   1. distance[i], the shortest path distance from the source to vertex i. If G has negative edges, the distance array could have negative entries. If no path from 0 to i exists, distance[i] is infinity.
   2. back[i], the next-to-the last vertex on the path of shortest length from the source to vertex i. Of course, back[0] is undefined, and back[i] is undefined if there is no path from the source to vertex i.

   No minpath algorithm works if the directed graph has a "negative cyle," that is, a cycle of negative weight. But, if the code is written properly, the Bellman-Ford algorithm can halt with an error message if G has a negative cycle. Try to do this in your code. If there is no negative cycle, the time complexity of your code should be $O(m\ell)$, where m is the number of arcs of G and ℓ is the maximum number of edges in any shortest length path.

   Let the arcs be numbered from 1 to m. For any $1 \leq k \leq m$, let the kth arc be from $i[k]$ to $j[k]$ and have weight $w[k]$. Here is my code:

```
for(i = 0 to n-1)
 {
  back[i] = undefined;
  dist[i] = infinity;
 }
changed = true; //changed is a Boolean variable, indicating that at least
                  one value of dist has decreased
for(t = 1; t < n and changed; t++) // Iteration t tries to find new shortest
                                      paths with t edges.  If not changed,
                                      there is no need for a new iteration of
                                      the outer loop.
  for(k = 1; k <= m; k++)
    {
     changed = false;
     i = i[k];
     j = j[k];
     w = w[k];
     // The kth arc is from i to j and has weight w
     temp = dist[i] + w;
     if(temp < dist[j])
       {
        dist[j] = temp;
        back[j] = i;
        changed = true;
       }
    }
if(changed) cout << "There is a negative cycle" << endl;
// if some value of dist is changed on the nth iteration of the outer loop,
   we have found a path of length n, which must contain a negative cycle.
```

3. Write pseudocode for the Floyd-Warshall algorithm to solve the all-pairs minpath problem for G. Assume there G has no negative cycle. The output of the algorithm is two square matrices:
   1. distance[i][j], the least weight of any path from i to j.
   2. forward[i][j], the second vertex of the least weight path from i to j. Note that forward[i][i] is undefined.

   We assume that a square array W is given, where $W[i,j]$ is the weight of the arc from i to j. If there is no such arc, $W[i,j] = \infty$. Here is the code:

```
for(i = 1 to n)
 {
  // initialization loop
  for(j = 1 to n)
   {
    distance[i,j] = W[i,j];
    forward[i,j] = j;
   }
  distance[i,i] = 0;
  forward[i,i] = undefined;
 }
for(j = 1 to n)
 {
  // main loop
  for(i = 1 to n)
   for(k = 1 to n)
    {
     temp = distance[i,j]+distance[j,k];
     if(temp < distance[i,k])
      {
       distance[i,k] = temp;
       forward[i,k] = forward[i,j];
      }
    }
 }
```

4. Walk through Dijkstra's algorithm for the single source minpath problem for the directed graph illustrated below. Instead of numbering the vertices 0 through 19, I have assigned them letters from A to T. The source vertex is S.

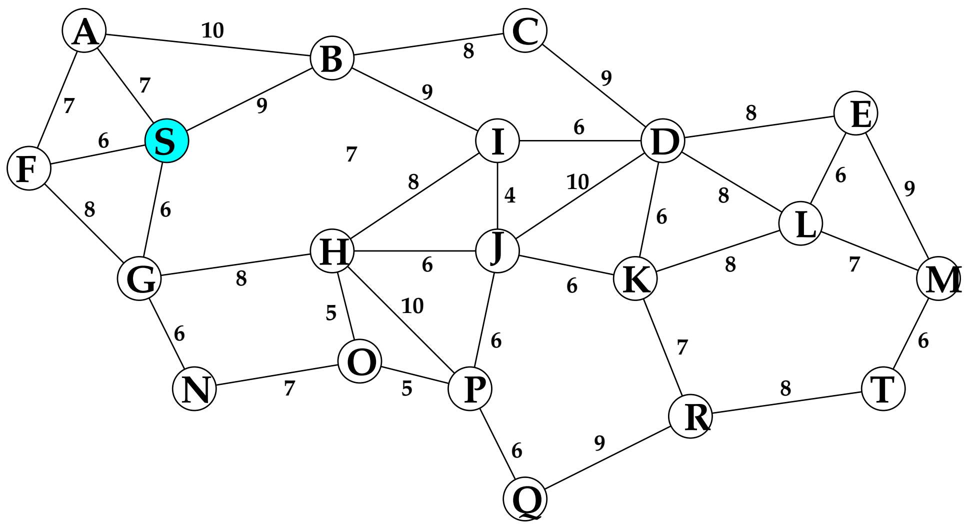
   After each iteration of the main loop, show

   1. The array dist, where dist[x] is the smallest length of any path found so far from S to x. (Initially, dist[x] = ∞ for most x.)

   2. The array back, where back[x] is the next-to-the last vertex on the path of smallest weight found so far from S to x.

   3 The contents of heap. Do not try to show the structure of the heap, simply list its members.

   Attach an extra sheet of paper if necessary.



Heap: S

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0																			
back	$\star$																			

Heap: FGAB

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9				6	6												
back	$\star$	S	S				S	S												

Heap: ABNH

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9				6	6	14		12				12					
back	$\star$	S	S				S	S	G		G				G					

Heap: BNH

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9				6	6	14						12					
back	$\star$	S	S				S	S	G						G					

Heap: NHCI

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17			6	6	14	18					12					
back	$*$	S	S	B			S	S	G	B					G					

Heap: HCIO

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17			6	6	14	18					12	19				
back	∗	S	S	B			S	S	G	B					G	N				

Heap: CIOJP

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17			6	6	14	18	20				12	19	24			
back	∗	S	S	B			S	S	G	B	H				G	N	H			

Heap: IOJPD

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	26		6	6	14	18	20				12	19	24			
back	∗	S	S	B	C		S	S	G	B	H				G	N	H			

Heap: OJPD

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24		6	6	14	18	20				12	19	24			
back	∗	S	S	B	I		S	S	G	B	H				G	N	H			

dist[D] and back[D] have been updated.

Heap: JPD

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24		6	6	14	18	20				12	19	24			
back	∗	S	S	B	I		S	S	G	B	H				G	N	H			

Heap: PDK

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24		6	6	14	18	20	26			12	19	24			
back	∗	S	S	B	I		S	S	G	B	H	J			G	N	H			

Heap: KQLE

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30		
back	∗	S	S	B	I	D	S	S	G	B	H	J	D		G	N	H	P		

Since dist[P] = dist[D], we can deletemin P and D in the same step.

Heap: QLER

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30	33	
back	∗	S	S	B	I	D	S	S	G	B	H	J	D		G	N	H	P	Q	

Heap: LER

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32		12	19	24	30	33	
back	∗	S	S	B	I	D	S	S	G	B	H	J	D		G	N	H	P	Q	

Heap: RM

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	
back	∗	S	S	B	I	D	S	S	G	B	H	J	D	L	G	N	H	P	Q	

We can dequeue L and E in the same step.

Heap: MT

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	41
back	$*$	S	S	B	I	D	S	S	G	B	H	J	D	L	G	N	H	P	Q	R

Heap: T

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	41
back	$*$	S	S	B	I	D	S	S	G	B	H	J	D	L	G	N	H	P	Q	R

It is prudent to continue the algorithm until all vertices are fully processed, despite the fact that we all the values of the arrays are correct.

Heap:

	S	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	T
dist	0	7	9	17	24	32	6	6	14	18	20	26	32	39	12	19	24	30	33	41
back	$*$	S	S	B	I	D	S	S	G	B	H	J	D	L	G	N	H	P	Q	R