COS 226	Algorithms and Data Structures	Fall 2017

Final

This exam has 16 questions worth a total of 100 points. You have 180 minutes. This exam is preprocessed by a computer, so please **write darkly** and **write your answers inside the designated spaces.**

Policies. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11 paper, two sides, in your own handwriting). No electronic devices are permitted.

Discussing this exam. Discussing the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. Do not remove this exam from this room. In the space below, write your name and NetID; mark your precept number; and write and sign the Honor Code pledge. You may fill in this information now.

Name:

NetID:

Exam room: McCosh 10 ● Other ○

Precept: P01 ○ P02 ○ P03 ○ P03A ○ P04 ○ P05 ○ P06 ○

"I pledge my honor that I will not violate the Honor Code during this examination."

Signature

1. **Initialization. (2 point)**

   In the space provided on the front of the exam, write your name and NetID; mark your precept number; and write and sign the Honor Code pledge.

2. **Memory. (5 points)**

   Consider the following representation for a ternary search trie for LZW compression with string keys and integer values:

```
public class TernarySearchTrie {
    private int n;            // number of key-value pairs
    private Node root;        // root node

    private static class Node {
        private char c;       // character
        private int value;    // value of key-value pair
        private Node left;    // left sub-trie
        private Node mid;     // middle sub-trie
        private Node right;   // right sub-trie
    }
    ...
}
```

   Using the 64-bit memory cost model from lecture and the textbook, how much memory does a `TernarySearchTrie` object use as a function of the number of key–value pairs n. Use tilde notation to simplify your answer.

   ~ [ ] bytes

   Hint 1: For LZW compression, the number of TST nodes equals the number of key–value pairs (because every prefix of a key is also a key).

   Hint 2: There is no 8-byte inner-class overhead for static nested classes.

3. **Running time. (6 points)**

Let `x` be a `StringBuilder` object of length `n`. For each code fragment at left, write the letter corresponding to the order of growth of the running time as a function of n.

Assume that Java's `StringBuilder` data type represents a string of length n using a resizing array of characters (with doubling and halving), with the first character in the string at index 0 and the last character in the string at index $n-1$.

A. 1

B. $\log n$

C. $n \log n$

D. n

E. n^2

☐
```
// converts x to a String
String s = "";
for (int i = 0; i < n; i++)
    s += x.charAt(i);
```

☐
```
// creates a copy of x
StringBuilder y = new StringBuilder();
for (int i = 0; i < n; i++)
    y.append(x.charAt(i));
```

☐
```
// reverses x
for (int i = 0; i < n/2; i++) {
    char c1 = x.charAt(i);
    char c2 = x.charAt(n - i - 1);
    x.setCharAt(i, c2);
    x.setCharAt(n - i - 1, c1);
}
```

☐
```
// concatenates x with itself
for (int i = 0; i < n; i++)
    x.append(x.charAt(i));
```

☐
```
// removes the last n/2 characters of x
for (int i = 0; i < n/2; i++)
    x.deleteCharAt(x.length() - 1);
```

☐
```
// removes the first n/2 characters of x
for (int i = 0; i < n/2; i++)
    x.deleteCharAt(0);
```

4. **String sorts. (5 points)**

The column on the left contains the original input of 24 strings to be sorted; the column on the right contains the strings in sorted order; the other 5 columns contain the contents at some intermediate step during one of the 3 radix-sorting algorithms listed below. Match each algorithm by writing its letter in the box under the corresponding column.

0	null	byte	cost	byte	java	byte	byte
1	tree	cost	lifo	cost	load	cost	cost
2	lifo	edge	list	edge	find	miss	edge
3	list	find	miss	flip	tree	hash	find
4	miss	flip	hash	find	byte	java	flip
5	hash	hash	java	hash	edge	load	hash
6	java	java	load	java	trie	leaf	java
7	next	lifo	leaf	lifo	type	flip	lazy
8	load	list	flip	list	leaf	link	leaf
9	leaf	load	link	load	hash	list	left
10	flip	leaf	byte	leaf	path	edge	lifo
11	path	lazy	edge	lazy	sink	lazy	link
12	byte	left	lazy	left	link	left	list
13	edge	link	left	link	rank	find	load
14	lazy	miss	find	miss	null	lifo	miss
15	trie	null	next	null	lifo	next	next
16	find	next	null	next	flip	null	null
17	left	path	type	path	swap	type	path
18	type	rank	sink	rank	miss	sink	rank
19	sink	sink	trie	sink	list	trie	sink
20	link	swap	swap	swap	next	swap	swap
21	swap	tree	path	tree	left	path	tree
22	cost	trie	rank	trie	cost	rank	trie
23	rank	type	tree	type	lazy	tree	type
	A						E

A. Original input

B. LSD radix sort

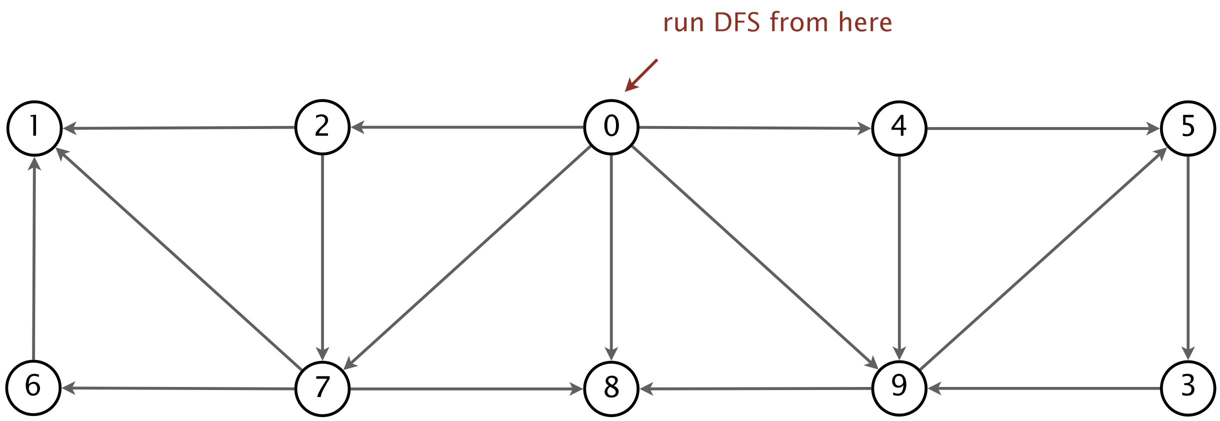
C. MSD radix sort

D. 3-way radix quicksort (*no shuffle*)

E. Sorted

5. **Depth-first search. (6 points)**

Run depth-first search on the following digraph, starting from vertex 0. Assume the adjacency lists are in sorted order: for example, when iterating over the edges pointing from 7, consider the edge 7 → 1 before either 7 → 6 or 7 → 8.



(a) List the 10 vertices in *preorder*.

0 ___ ___ ___ ___ ___ ___ ___ ___ ___

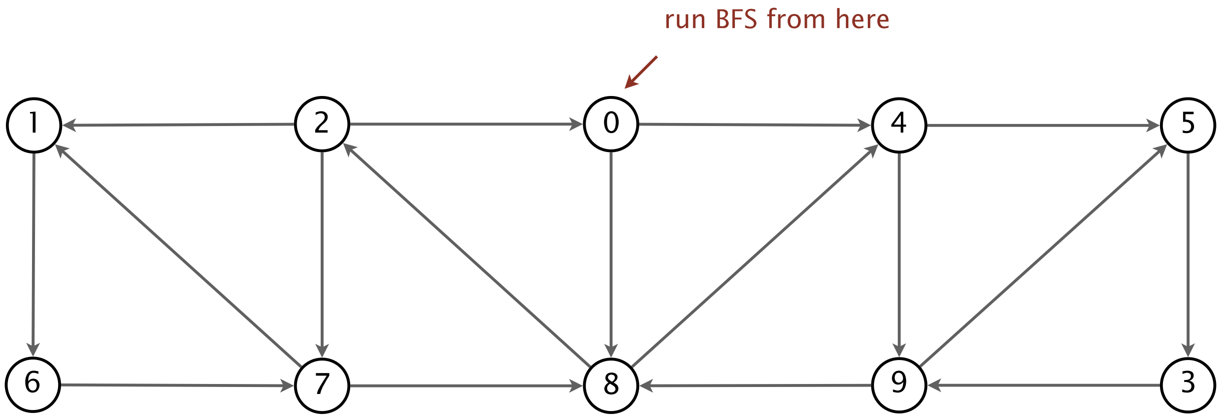
(b) List the 10 vertices in *postorder*.

___ ___ ___ ___ ___ ___ ___ ___ ___ 0

(c) Does this digraph have a topological order? If yes, write one in the box below; if no, succinctly explain why not.

6. **Breadth-first search. (4 points)**

Run breadth-first search on the following digraph, starting from vertex 0. Assume the adjacency lists are in sorted order: for example, when iterating over the edges pointing from 7, consider the edge $7 \to 1$ before either $7 \to 6$ or $7 \to 8$.

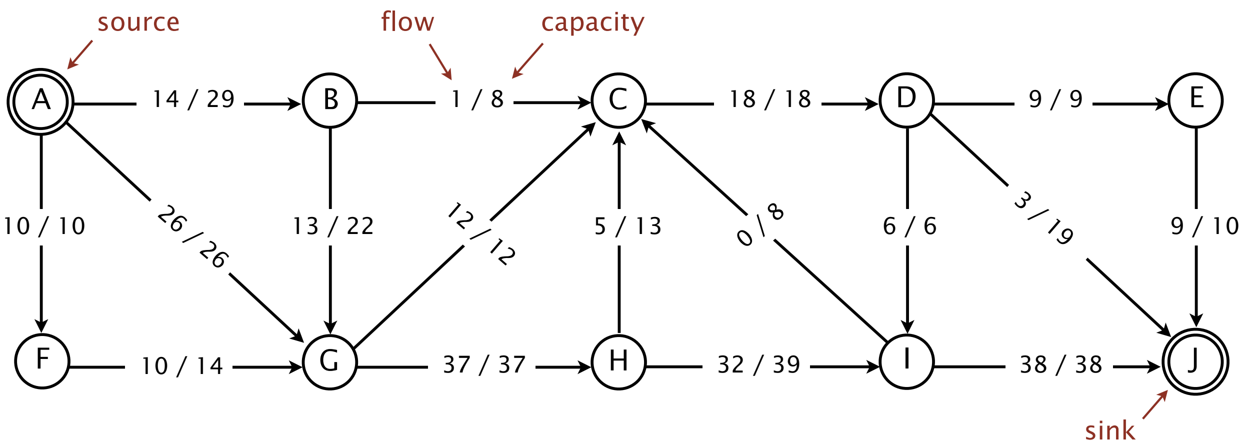


List the 10 vertices in the order in which they are added to the queue.

0
___ ___ ___ ___ ___ ___ ___ ___ ___ ___

7. **Maximum flow. (10 points)**

Consider the following flow network and flow f from the source vertex A to sink vertex J.



(a) What is the value of the flow f? Mark the correct answer.

47 ○ 50 ○ 51 ○ 52 ○ 53 ○ 54 ○ 55 ○ 65 ○ 70 ○ 79 ○

(b) What is the capacity of the cut $\{A, F, G\}$? Mark the correct answer.

41 ○ 46 ○ 49 ○ 50 ○ 56 ○ 63 ○ 65 ○ 74 ○ 78 ○ 100 ○

(c) Starting from the flow f, perform one iteration of the Ford–Fulkerson algorithm. Which vertices are on the (unique) augmenting path? Mark all that apply.

A □ B □ C □ D □ E □ F □ G □ H □ I □ J □

(d) What is the bottleneck capacity of the augmenting path? Mark the correct answer.

0 ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○

(e) Which vertices are on the source side of the (unique) minimum cut? Mark all that apply.

A □ B □ C □ D □ E □ F □ G □ H □ I □ J □

8. **LZW compression. (6 points)**

Expand the following LZW-encoded sequence of 8 hexadecimal integers.

`43 41 41 81 42 84 41 80`

Assume the original encoding table consists of all 7-bit ASCII characters and uses 8-bit codewords. Recall that codeword 80 is reserved to signify end of file.

(a) What was the encoded message?

C									

(b) Which of the following strings are in the LZW dictionary upon termination of the algorithm? Mark all that apply.

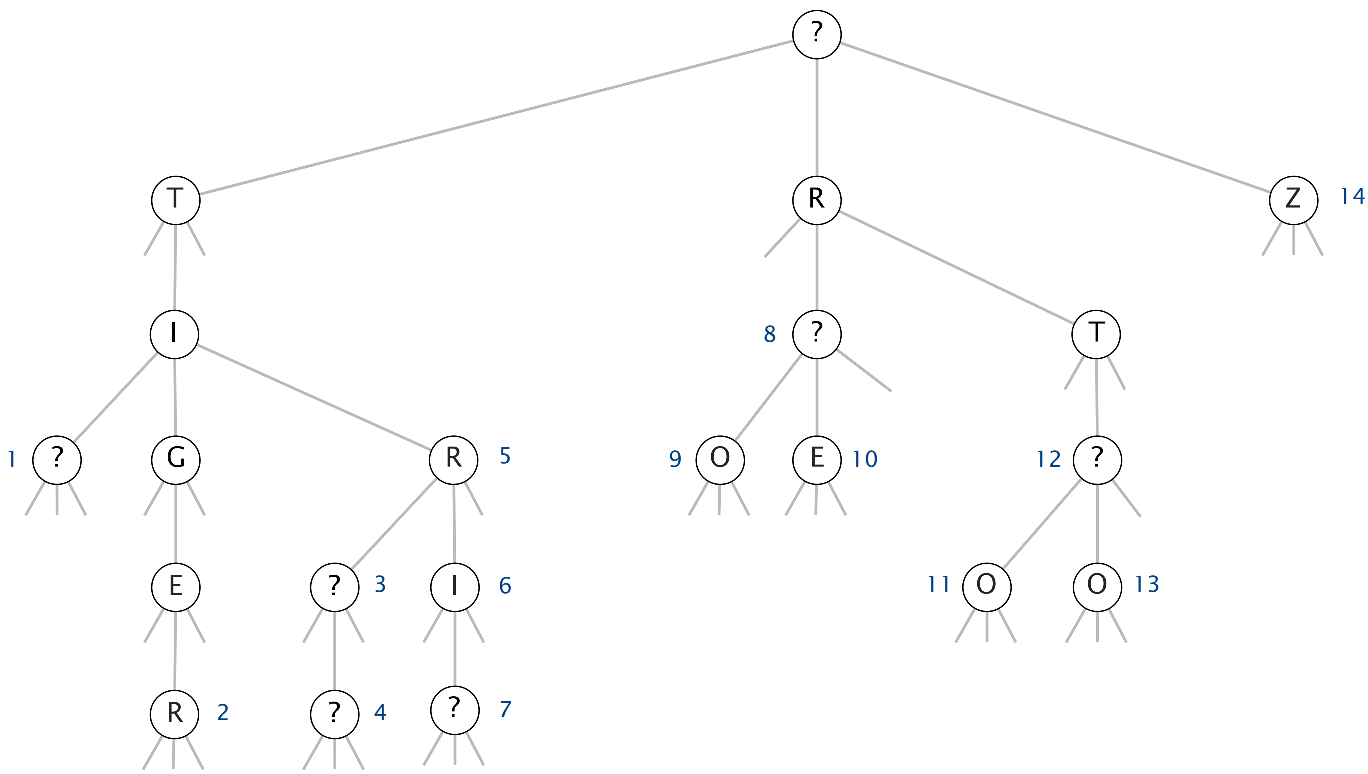
AA	AB	ABA	AC	ACA	BC	CA	CAA	CAB	CABA	CABC
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SP	!	"	#	$	%	&	‘	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

For reference, this is the hexadecimal-to-ASCII conversion table from the textbook.

9. **Ternary search tries. (6 points)**

Consider the following TST, where the values are shown next to the nodes of the corresponding string keys. Each node labeled with a `?` contains some uppercase letter (possibly different for each node).



Which of the following string keys are (or could be) in the TST? Mark all that apply.

TIGER	TILE	TO	TOO	TREE	TRIE	TRUE	TWO	URGE
■	☐	☐	☐	☐	☐	☐	☐	☐

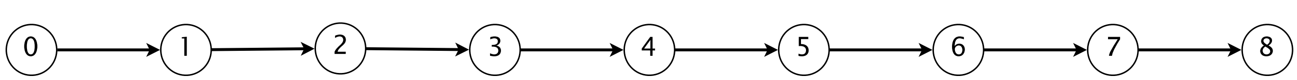
10. **Knuth–Morris–Pratt substring search. (6 points)**

Below is a partially-completed Knuth–Morris–Pratt DFA for the string

`C C A C C A C B`

over the alphabet { `A`, `B`, `C` }. Complete the third row of the table.

	0	1	2	3	4	5	6	7
`A`	0	0	3	0	0	6	0	0
`B`	0	0	0	0	0	0	0	8
`C`								
s	C	C	A	C	C	A	C	B



Feel free to use this diagram for scratch work.

11. **Programming assignments. (12 points)**

Answer the following questions related to COS 226 programming assignments.

(a) Suppose that in the *WordNet* assignment, you needed to check whether a digraph G is a rooted tree (instead of a rooted DAG). A *rooted tree* is a digraph that contains a root vertex r such that there is exactly one directed path from every vertex to r.

Which of the following properties hold for all rooted trees? Mark all that apply.

- ☐ There is exactly one vertex of outdegree 0.
- ☐ There is exactly one vertex of indegree 0.
- ☐ There are no directed cycles.
- ☐ There is a directed path between every pair of vertices.
- ☐ There are $V - 1$ edges, where V is the number of vertices.
- ☐ There are $E - 1$ vertices, where E is the number of edges.

(b) In the *Seam Carving* assignment, what is the worst-case running time of an efficient algorithm for finding a *horizontal seam of minimum total energy* in a picture of width W and height H? Mark the best answer.

W	H	$W + H$	WH	WH^2	W^2H
○	○	○	○	○	○

(c) Suppose that you compress the text of *Algorithms, 4th edition* using one of the following sequences of transformations:

A. Huffman coding
B. Burrows–Wheeler transform
C. Burrows–Wheeler transform ⟶ Huffman coding
D. Burrows–Wheeler transform ⟶ move-to-front coding ⟶ Huffman coding.
E. Huffman coding ⟶ Burrows–Wheeler transform.

Which of the following can you infer? Mark all that apply.

- ☐ A achieves a better compression ratio than B.
- ☐ C achieves a better compression ratio than A.
- ☐ E achieves a better compression ratio than A.
- ☐ D achieves the best compression ratio among A–E.

(d) In which of the following programming assignments was the *super-source trick* (implicitly or explicitly adding a source vertex to convert a graph or digraph with multiple sources into one with a single source) a key component in improving the order of growth of the running time? Mark all that apply.

- ☐ Assignment 1 (*Percolation*)
- ☐ Assignment 2 (*Deques and Randomized Queues*)
- ☐ Assignment 3 (*Autocomplete*)
- ☐ Assignment 4 (*8-Puzzle*)
- ☐ Assignment 5 (*Kd-Trees*)
- ☐ Assignment 6 (*WordNet*)
- ☐ Assignment 7 (*Seam Carving*)
- ☐ Assignment 8 (*Burrows–Wheeler*)

12. **Properties of minimum spanning trees. (5 points)**

Let G be a connected graph with distinct edge weights. Let S be a cut that contains exactly 4 crossing edges e_1, e_2, e_3, and e_4 such that $weight(e_1) < weight(e_2) < weight(e_3) < weight(e_4)$. For each statement at left, write the letter corresponding to the best-matching description at right.

	Statement
☐	Kruskal's algorithm adds edge e_1 to the MST.
☐	Prim's algorithm adds edge e_4 to the MST.
☐	If Kruskal's algorithms adds edges e_1, e_2, and e_4 to the MST, then it also adds e_3.
☐	If edges e_1 and e_2 are both in the MST, then Kruskal's algorithm adds e_1 to the MST before e_2.
☐	If edges e_1 and e_2 are both in the MST, then Prim's algorithm adds e_1 to the MST before e_2.

A. *True* for every such edge-weighted graph G and every such cut S.

B. *False* for every such edge-weighted graph G and every such cut S.

C. *Neither A* nor *B*.

13. **Properties of shortest paths. (5 points)**

Let G be any DAG with positive edge weights and assume all vertices are reachable from the source vertex s. For each statement at left, identify whether it is a property of Dijkstra's algorithm and/or the topological sort algorithm by writing the letter corresponding to the best-matching term at right.

- [ ] If G contains the edge $v \to w$, then vertex v is relaxed before vertex w.
- [ ] Each vertex is relaxed at most once.
- [ ] If the length of the shortest path from s to v is less than the length of the shortest path from s to w, then vertex v is relaxed before vertex w.
- [ ] Immediately after relaxing any edge $v \to w$, `distTo[w]` is the length of the shortest path from s to w.
- [ ] During each edge relaxation, for each vertex v, `distTo[v]` either remains unchanged or decreases.

A. Dijkstra's algorithm.

B. Topological sort algorithm.

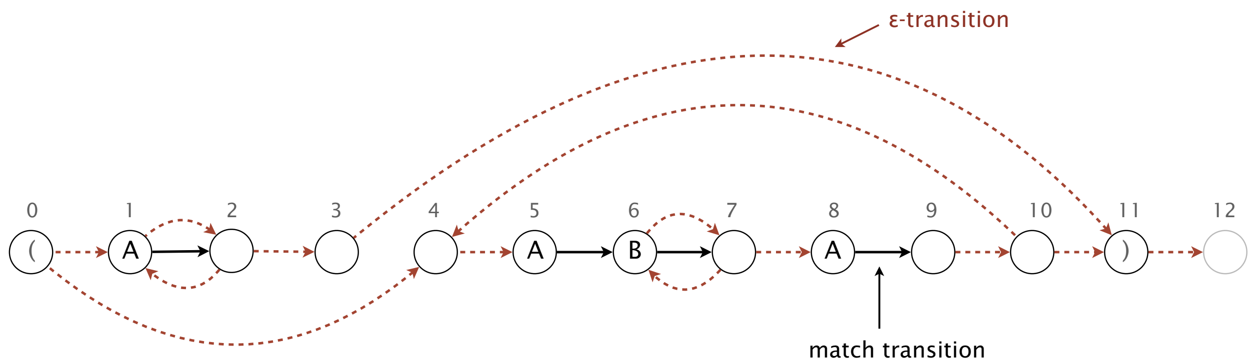
C. *Both* A and B.

D. *Neither* A nor B.

Recall that relaxing a vertex v means relaxing every edge pointing from v.

14. **Regular expressions. (6 points)**

Consider the following NFA, where 0 is the start state and 12 is the accept state:



(a) Complete the regular expression below so that it matches the same set of strings as the NFA by writing one of the following symbols in each box:

( ) * + |

(	A				A	B		A			)

(b) Suppose that you simulate the NFA with the following input:

A A A A A

In which state(s) could the NFA be after reading the entire input? Mark all that apply.

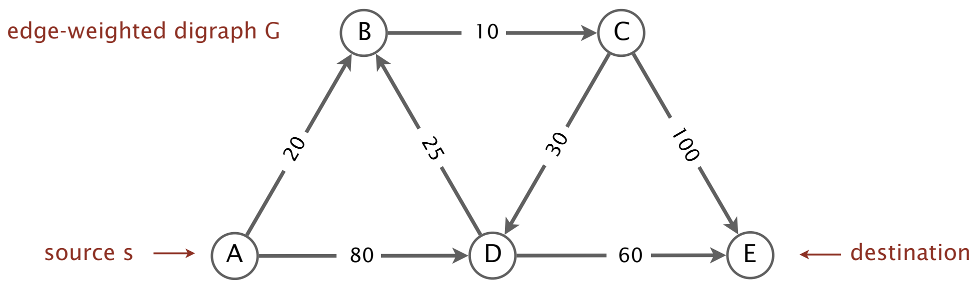
0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10 ☐ 11 ☐ 12 ☐

15. **Shortest discount path. (8 points)**

Consider the following variant of the shortest path problem.

SHORTEST-DISCOUNT-PATH. Given an edge-weighted digraph G with positive edge weights, a source vertex s, and a destination vertex $t \neq s$, find the weight of the *shortest discount path* from s to t, where the weight of a *discount path* is the sum of the weights of the edges in the path, but with the largest weight in the path discounted by 50%.

For example, in the SHORTEST-DISCOUNT-PATH instance below, the shortest path from A to E is $A \to B \to C \to D \to E$ (with weight $120 = 20 + 10 + 30 + 60$) but the the shortest discount path is $A \to B \to C \to E$ (with weight $80 = 20 + 10 + \frac{100}{2}$).



Design an efficient algorithm for solving the SHORTEST-DISCOUNT-PATH problem by solving a traditional shortest path problem on a related edge-weighted digraph G' with positive weights. To demonstrate your algorithm, draw G' for this SHORTEST-DISCOUNT-PATH instance in the space provided the facing page.

Draw G' here. Be sure to specify the weight of each edge and label the source and destination.

Hint: you shouldn't need more than 10 vertices or 21 edges.

In general, how many vertices and edges does G' have as a function of V and E?
(where V and E denote the number of vertices and edges in G, respectively)

number of vertices in G'

number of edges in G'

16. **Substring of a circular string. (8 points)**

Design an algorithm to determine whether a string s is a substring of a *circular* string t. Let m denote the length of s and let n denote the length t. Assume the binary alphabet.

For reference, the following table shows a few examples:

string s	*circular string t*	*substring*
ABBA	BBBBBBABBABBBBB	*yes*
ABBA	BABBBBBABBBBBAB	*yes*
BBAABBAABBAABB	ABBA	*yes*
ABBA	BBBBBBBABABBBB	*no*
BAABAAB	ABBA	*no*

Give a crisp and concise English description of your algorithm in the space below.

Your answer will be graded for correctness, efficiency, and clarity. For full credit, the order of growth of the worst-case running time must be $m + n$.

This page is provided as scratch paper. If you tear it out, please write your name, NetID, and precept number in the space provided and return it inside your exam.

Name: ______________________ **NetID:** ____________ **Precept:** _______