COS 226	Algorithms and Data Structures	Spring 2015

Final

This exam has 14 questions worth a total of 100 points. You have 180 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet (8.5-by-11, both sides, in your own handwriting). No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided. **Write and sign the Honor Code pledge just before turning in the exam.**

This exam is preprocessed by computer: if you use pencil (and eraser), write darkly; write all answers inside the designated rectangles; do not write on the corner marks.

"*I pledge my honor that I have not violated the Honor Code during this examination.*"

Name:

netID:

Room:

Precept: P01 ○ P01A ○ P02 ○ P03 ○ P04 ○ P05 ○ P05A ○ P06 ○ P06A ○ P06B ○ P07 ○

Problem	Score
0	
1	
2	
3	
4	
5	
6	
Sub 1	

Problem	Score
7	
8	
9	
10	
11	
12	
13	
Sub 2	

Total	

P01	Th 11	Andy Guna
P01A	Th 11	Shivam Agarwal
P02	Th 12:30	Andy Guna
P03	Th 1:30	Swati Roy
P04	F 10	Robert MacDavid
P05	F 11	Robert MacDavid
P05A	F 11	Shivam Agarwal
P06	F 2:30	Jérémie Lumbroso
P06A	F 2:30	Josh Wetzel
P06B	F 2:30	Ryan Beckett
P07	F 3:30	Jérémie Lumbroso

0. **Initialization** (1 point)

In the space provided on the front of the exam, write your name and Princeton netID; mark your precept number; write the name of the room in which you are taking the exam; and write and sign the honor code.

1. **Analysis of Algorithms** (8 points)

(a) You observe the following memory usage for a program with an input of size N.

N	*memory*
1,000	2.1 MB
2,000	8.2 MB
4,000	32.4 MB
8,000	128.8 MB

Estimate the memory usage of the program (in megabytes) on an input of size 24,000. Your answer should be accurate to within 5%.

[ ] megabytes

(b) Consider the following implementation of a trie data type:

```
public class TrieST<Value> {
    private static final int R = 256;
    private Node root;          // root of trie
    private int N;              // number of nodes in the trie

    private static class Node {
        private Object val;
        private Node[] next = new Node[R];
    }
    // ...
}
```

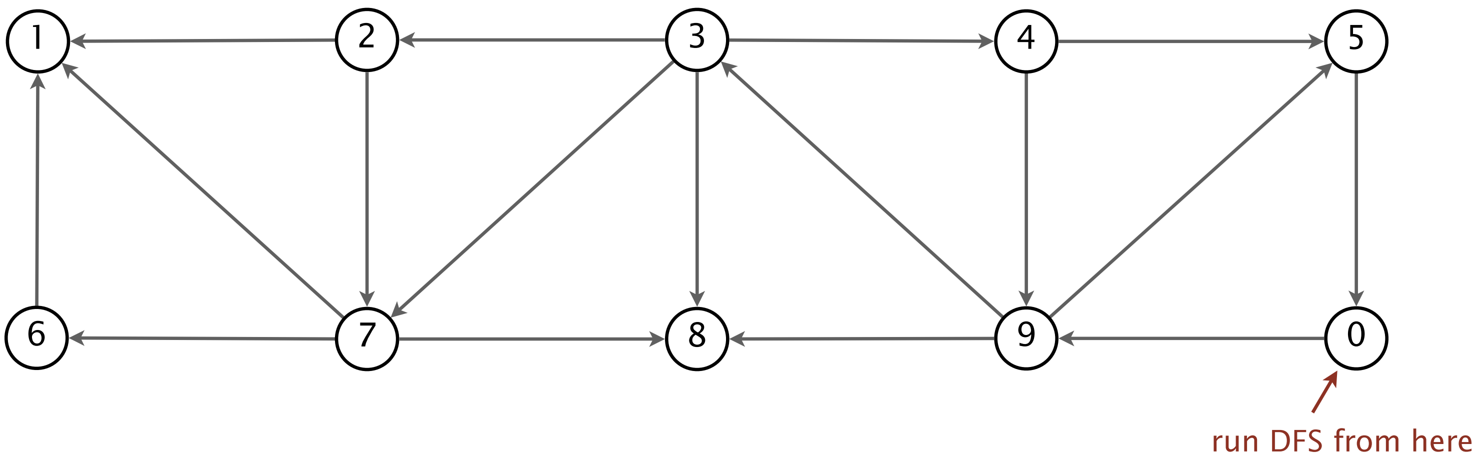
Using the 64-bit memory cost model from lecture and the textbook, how much memory (in bytes) does a `TrieST` object use to store M key-value pairs in N nodes as a function of N and M?

Use tilde notation to simplify your answer. Do not include the memory for the values themselves but do include all other memory (including references to values). Recall that with a static nested class, there is no 8 byte inner class overhead.

$\sim$ [ ] bytes

2. **Graph Search** (6 points)

Perform a depth-first search in the digraph below, starting from vertex 0. Assume the adjacency lists are in sorted order: for example, when iterating over the edges pointing from 3, process the edge $3 \to 2$ before either $3 \to 7$ or $3 \to 8$.



(a) List all vertices in *reverse postorder*.

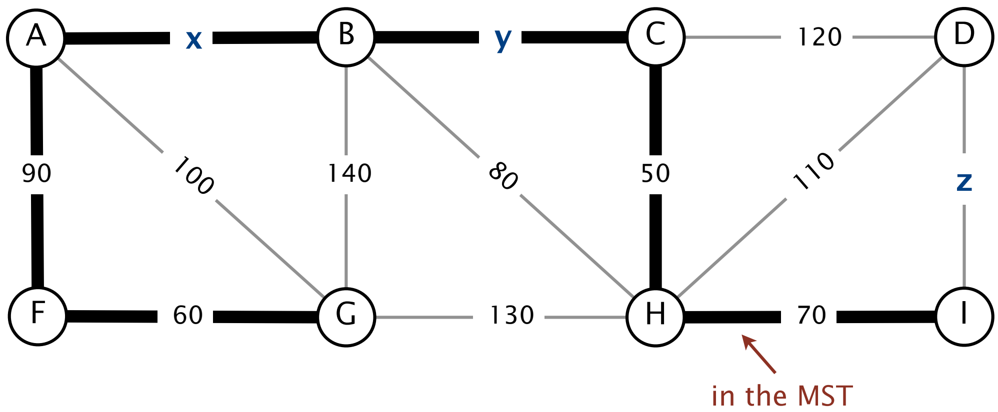
0									

(b) List all vertices in *preorder*.

0									

3. **Minimum Spanning Tree** (8 points)

The following diagram shows the set of edges (in thick black lines) selected at some intermediate step of an MST algorithm.



(a) Which of the following could be the weights of edges x, y, and z, respectively, at some intermediate step of *Kruskal's algorithm*? Mark all that apply.

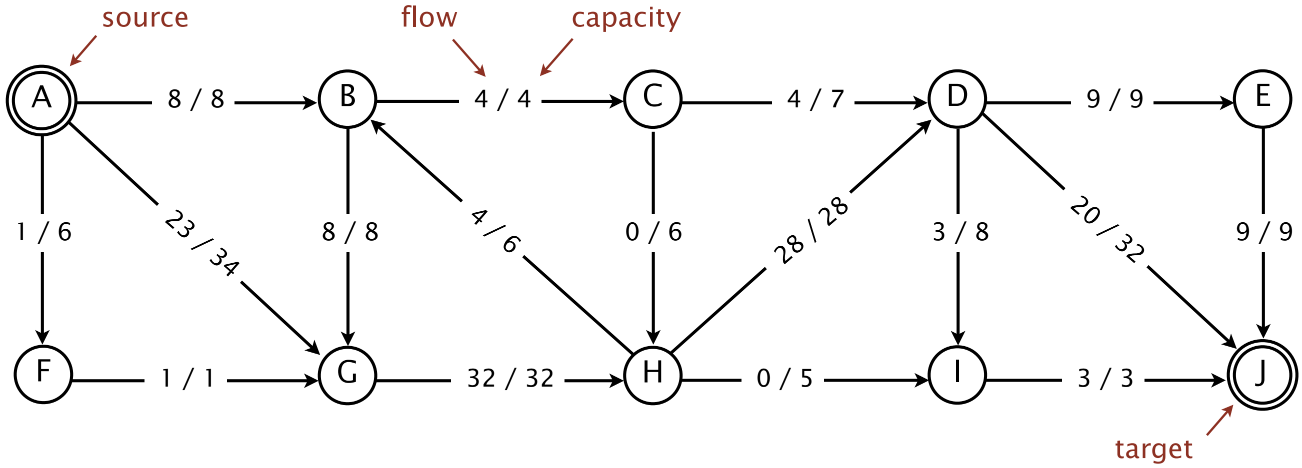
	55	65	75	85	95	105	115	125	135	145
x	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
y	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
z	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

(b) Which of the following could be the weights of edges x, y, and z, respectively, at some intermediate step of *Prim's algorithm, starting from vertex A*? Mark all that apply.

	55	65	75	85	95	105	115	125	135	145
x	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
y	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
z	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

4. **Maximum Flow** (10 points)

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



(a) Mark the value of the flow f.

0 ○ 10 ○ 20 ○ 22 ○ 24 ○ 26 ○ 28 ○ 30 ○ 32 ○ 34 ○ 36 ○ 38 ○ 40 ○

(b) Starting from the flow f, perform one iteration of the Ford-Fulkerson algorithm. Mark all vertices that are on the (unique) augmenting path.

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

(c) Mark the bottleneck capacity of the augmenting path.

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

(d) Mark the vertices on the source side of a minimum cut.

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

(e) Mark the edges below, for which doubling the capacity would increase the value of the maximum flow.

$A \to G$ ☐ $B \to C$ ☐ $D \to I$ ☐ $G \to H$ ☐ $I \to J$ ☐ $H \to D$ ☐

5. **String Sorting Algorithms** (7 points)

The column on the left is the original input of 24 strings to be sorted; the column on the right are the strings in sorted order; the other 7 columns are the contents at some intermediate step during one of the 3 radix sorting algorithms listed below.

Match up each column with the corresponding sorting algorithm. You may use a number more than once.

Hint: think about algorithm invariants. Do not trace code.

0	lust	bole	bone	bole	leaf	bone	lava	cafe	bole
1	rust	bone	buff	fawn	teal	bole	herb	sage	bone
2	fawn	buff	bole	flax	pear	buff	sand	palm	buff
3	pine	cafe	cafe	cafe	flax	cafe	gold	sand	cafe
4	sand	herb	dust	buff	gray	dust	pine	lava	dust
5	cafe	dust	fawn	herb	puce	fawn	cafe	fawn	fawn
6	pear	gray	flax	dust	cafe	flax	puce	leaf	flax
7	puce	flax	gray	gray	buff	gray	sage	teal	gold
8	sage	gold	gold	bone	sage	gold	rose	pear	gray
9	herb	fawn	herb	gold	gold	herb	bone	herb	herb
10	dust	leaf	lust	leaf	bole	lust	lime	lime	lava
11	gray	lava	lava	lava	palm	lava	bole	pine	leaf
12	rose	lime	leaf	lime	lime	leaf	buff	flax	lime
13	gold	lust	lime	lust	sand	lime	leaf	plum	lust
14	bone	rose	pine	rose	pine	pine	teal	gold	palm
15	buff	sage	pear	sage	bone	pear	plum	bole	pear
16	lava	plum	puce	plum	herb	puce	palm	bone	pine
17	plum	puce	plum	puce	rose	plum	fawn	rose	plum
18	leaf	pear	palm	pear	lust	palm	pear	gray	puce
19	lime	sand	rust	sand	rust	rust	lust	puce	rose
20	flax	pine	rose	pine	dust	rose	rust	buff	rust
21	bole	teal	sand	teal	plum	sand	dust	lust	sage
22	teal	palm	sage	palm	lava	sage	flax	rust	sand
23	palm	rust	teal	rust	fawn	teal	gray	dust	teal
	----	----	----	----	----	----	----	----	----

0 4

(0) Original input

(1) LSD radix sort

(2) MSD radix sort

(3) 3-way radix quicksort (no shuffle)

(4) Sorted

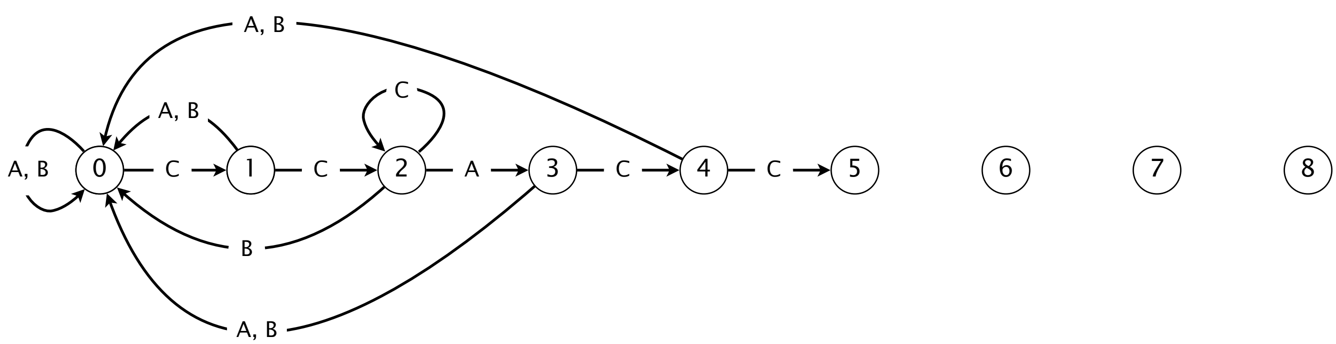
6. **Substring Search** (8 points)

(a) Consider the Knuth-Morris-Pratt DFA for the following string of length 8:

`C C A C C C A B`

Complete the last three columns of this partially-completed DFA table. (Feel free to use the partially-completed DFA diagram below for scratch work.)

	0	1	2	3	4	5	6	7
A	0	0	3	0	0			
B	0	0	0	0	0			
C	1	2	2	4	5			



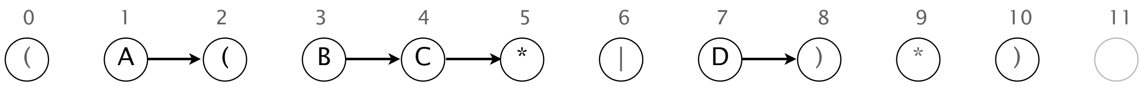
(b) What is the Rabin-Karp hash function of `text[4..11]` over the decimal alphabet with $R = 10$, using the modulus $Q = 157$?

j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
text[j]	6	1	3	2	6	9	?	?	?	7	7	8	4	4	2	9	5	1	9	6

The digits labeled with a question mark (?) are suppressed. Assume that the hash function of `text[3..10]` is 115 and note that 10000000 (mod 157) = 42.

7. **Regular Expressions** (6 points)

Consider the NFA that results from applying the RE-to-NFA construction algorithm from lecture and the textbook to the regular expression `( A ( B C * | D ) * )`. The states and match transitions are shown below, but the ε-transitions are suppressed.



(a) Which of the following are edges in the ε-transition digraph? Mark all that apply.

$0 \to 1$ ☐	$0 \to 8$ ☐	$0 \to 9$ ☐	$0 \to 10$ ☐	$0 \to 11$ ☐
$2 \to 6$ ☐	$2 \to 7$ ☐	$2 \to 8$ ☐	$2 \to 9$ ☐	$2 \to 10$ ☐
$4 \to 2$ ☐	$4 \to 5$ ☐	$4 \to 6$ ☐	$4 \to 7$ ☐	$4 \to 8$ ☐
$5 \to 4$ ☐	$5 \to 5$ ☐	$5 \to 6$ ☐	$5 \to 7$ ☐	$5 \to 9$ ☐
$6 \to 2$ ☐	$6 \to 7$ ☐	$6 \to 8$ ☐	$6 \to 9$ ☐	$6 \to 10$ ☐
$9 \to 2$ ☐	$9 \to 6$ ☐	$9 \to 8$ ☐	$9 \to 10$ ☐	$9 \to 11$ ☐

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

`A B C C C B B C B B B`

In which states could the NFA be? Mark all that apply.

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

8. **LZW Compression** (5 points)

Expand the following LZW-encoded sequence of 10 hexadecimal integers.

`42 42 41 43 81 43 83 85 87 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?

B														

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

☐ AC ☐ ACB ☐ BA ☐ BB ☐ BC ☐ BBA ☐ BBC ☐ BAC ☐ BCA ☐ CA ☐ CAC ☐ CB ☐ CBB

	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, above is the hexadecimal-to-ASCII conversion table from the textbook.

9. **Burrows-Wheeler Transform** (6 points)

(a) What is the Burrows-Wheeler transform of the following?

B A C B C D B A

(b) What is the Burrows-Wheeler inverse transform of the following?

5

B A D A B B D C

Feel free to use both of these grids for scratch work.

10. **Properties of Problems** (9 points)

Mark whether each of the following statements are True or False.

(a) **Reductions.** Suppose that Problem X poly-time reduces to Problem Y.

	True	False
If X can be solved in polynomial time, then so can Y.	○	○
If Y can be solved in quadratic time, then X can be solved in polynomial time.	○	○
If X cannot be solved in quadratic time, Y cannot be solved in polynomial time.	○	○
If Y cannot be solved in polynomial time, then neither can X.	○	○
If Y is NP-complete, then so is X.	○	○

(b) **Minimum spanning trees.** Let G be any simple graph (no self-loops or parallel edges) with positive and distinct edge weights.

	True	False
Any MST of G must include the edge of minimum weight.	○	○
Any MST of G must exclude the edge of maximum weight.	○	○
The MST of G is unique.	○	○
If the weights of all edges incident to any vertex v are increased by 17, then any MST in G is an MST in the modified edge-weighted graph.	○	○
If the weights of all edges in G are increased by 17, then any MST in G is an MST in the modified edge-weighted graph.	○	○

(c) **Shortest Paths.** Let G be any simple digraph (no self-loops or parallel edges) with positive and distinct edge weights.

	True	False
Any shortest path from s to t in G must include the edge of minimum weight.	○	○
Any shortest path from s to t in G must exclude the edge of maximum weight.	○	○
The shortest path from s to t in G is unique.	○	○
If the weights of all edges leaving s are increased by 17, then any shortest path from s to t in G is a shortest path in the modified edge-weighted digraph.	○	○
If the weights of all edges in G are increased by 17, then any shortest path from s to t in G is a shortest path in the modified edge-weighted digraph.	○	○

11. **Properties of Algorithms** (9 points)

(a) Consider the execution of *depth-first search* on a digraph G from vertex s, beginning with the function call `dfs(G, s)`. Suppose that `dfs(G, v)` is called during the depth-first search. Which of the following statements can you infer at the moment when `dfs(G, v)` is called? Mark all that apply.

- ☐ G contains a directed path from s to v.
- ☐ The function-call stack contains a directed path from s to v.
- ☐ The `edgeTo[]` array contains a directed path from s to v.
- ☐ If G includes an edge $v \to w$ for which w has been previously marked, then G has a directed cycle containing v.
- ☐ If G includes an edge $v \to w$ for which w is currently a vertex on the function-call stack, then G has a directed cycle containing v.

(b) Consider the execution of *breadth-first search* on a digraph G, starting from vertex s. Suppose that vertex v is removed from the queue during the breadth-first search. Which of the following statements can you infer at the moment when v is removed from the queue? Mark all that apply.

- ☐ G contains a directed path from s to v.
- ☐ The queue contains a directed path from s to v.
- ☐ The `edgeTo[]` array contains a directed path from s to v.
- ☐ If G includes an edge $v \to w$ for which w has been previously marked, then G has a directed cycle containing v.
- ☐ If G includes an edge $v \to w$ for which w is currently a vertex on the queue, then G has a directed cycle containing v.

(c) Which of the following statements about string-processing algorithms are true?
Mark all that apply.

- ☐ Both MSD radix sort and LSD radix sort are stable sorting algorithms.
- ☐ The shape of an R-way trie depends not only on the keys that were inserted but also on the order in which they were inserted.
- ☐ The shape of a ternary search tree depends not only on the keys that were inserted but also on the order in which they were inserted.
- ☐ Searching for an M-character pattern in an N-character text takes time proportional to M in the best case and $M + N$ in the worst case using the Boyer-Moore algorithm (with the mismatch character heuristic only).
- ☐ Building the NFA corresponding to an M-character regular expression (using the algorithm from the textbook and lecture) takes time proportional to M in the worst case.

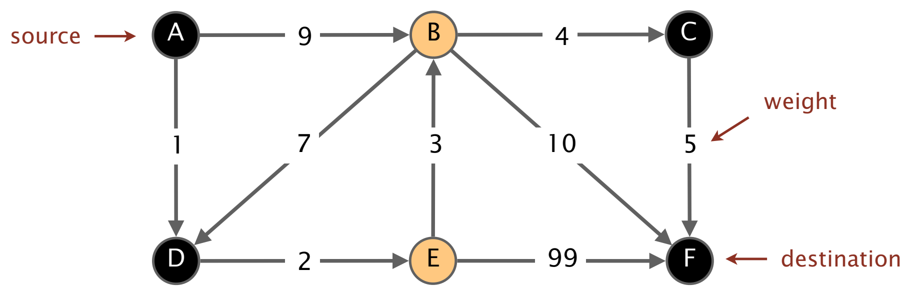
12. **Reductions** (8 points)

Consider the following graph-processing problems:

- SHORTEST-PATH. Given an edge-weighted digraph G with nonnegative edge weights, a source vertex s, and a destination vertex t, find a shortest path from s to t.

- SHORTEST-PRINCETON-PATH. Given an edge-weighted digraph G with nonnegative edge weights, a source vertex s, a destination vertex t, and *with each vertex colored black or orange*, find a shortest path from s to t that uses *at most one orange vertex*. Assume that the source vertex is not orange.

In the edge-weighted digraph below, the shortest path from A to F is $A \to D \to E \to B \to C \to F$ (weight 15) but the the shortest Princeton path is $A \to B \to C \to F$ (weight 18).



(a) Give a linear-time reduction from SHORTEST-PATH to SHORTEST-PRINCETON-PATH. To demonstrate your reduction, draw the edge-weighted digraph (labeling the source and destination vertices and coloring each vertex black or orange) that you would construct to solve the SHORTEST-PATH instance above.

(b) Give a linear-time reduction from SHORTEST-PRINCETON-PATH to SHORTEST-PATH. To demonstrate your reduction, draw the edge-weighted digraph (labeling the source and destination vertices) that you would construct to solve the SHORTEST-PRINCETON-PATH instance on the facing page.

13. **Algorithm Design** (9 points)

(a) Design a data structure that supports the following API:

`public class StreamingSum`		
`public`	`StreamingSum()`	*create an empty data structure*
`public void`	`add(int weight)`	*add the weight to the data structure*
`public void`	`remove()`	*remove the least-recently added weight*
`public int`	`sum()`	*sum of weights in data structure*

Here is an example,

```
StreamingSum ss = new StreamingSum();
ss.add(1);          // 1              ( add 1    )
ss.add(2);          // 1 2            ( add 2    )
ss.add(3);          // 1 2 3          ( add 3    )
ss.sum();           // 1 2 3          ( return 6 )
ss.add(4);          // 1 2 3 4        ( add 4    )
ss.remove();        // 2 3 4          ( remove 1 )
ss.sum();           // 2 3 4          ( return 9 )
```

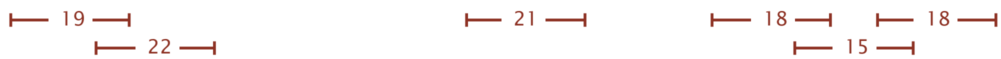
Each operation should take constant time in the worst case.

Declare the instance variables for your `StreamingSum` data type. You may declare nested classes but you may *not* use higher-level data types (such as those in `algs4.jar` or `java.util`).

(b) Given a binary string s with integer weights associated with each character and a query string t, find a minimum weight occurrence of t in s (or report that t does not appear as a substring in s). The weight of an occurrence is equal to the sum of the weights of the corresponding characters in the text.

For example, if $s = AAABABABBABAAABABBBABABABABB$ and $t = ABAB$, and the weights are given as below, then the minimum weight occurrence of t in s starts at index 21.

	0	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	26	27
query string t	A	B	A	B																								
text string s	A	A	A	B	A	B	A	B	B	A	B	A	A	A	B	A	B	B	B	A	B	A	B	A	B	A	B	B
weights	3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3	2	3	8	4	6	2	6	4	3	3	8	3



Your algorithm should run in time proportional to $N + M$, where N and M are the lengths of s and t, respectively. Your answer will be graded on correctness, efficiency, clarity, and conciseness.