COS 226	Algorithms and Data Structures	Fall 2012

Final Exam

This test has 16 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 out and sign the Honor Code pledge before turning in the test.**

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

Problem	Score
0	
1	
2	
3	
4	
5	
6	
7	
Sub 1	

Problem	Score
8	
9	
10	
11	
12	
13	
14	
15	
Sub 2	

Total	

Name:

Login:

Room:

Precept:

P01	F 11	Maia Ginsburg
P02	F 12:30	Diego Perez Botero
P03	F 1:30	Diego Perez Botero
P03B	F 1:30	Dushyant Arora
P04	Th 2:30	Maia Ginsburg
P04A	Th 2:30	Dan Larkin

Jan 22: `653e 10f3 8823`

0. **Initialization. (1 point)**

   Write your name and Princeton NetID in the space provided on the front of the exam; circle your precept number; and write and sign the honor code.

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

   (a) Suppose that you observe the following running times for a program with an input of size N.

N	time
5,000	0.2 seconds
10,000	1.2 seconds
20,000	3.9 seconds
40,000	16.0 seconds
80,000	63.9 seconds

   Estimate the running time of the program (in seconds) on an input of size N =200,000.

   (b) How many bytes of memory does a `KMP` object consume as a function of the length of the pattern M and the size of the alphabet R? Use tilde notation to simplify your answer.

```
public class KMP {
    private int[][] dfa;
    private char[] pat;

    public KMP(String pattern, int R) {
        int M = pattern.length();
        dfa = new int[R][M];
        pat = new char[M];
        ...
    }
    ...
}
```

2. **Graphs. (5 points)**

Consider the following Java class. Assume that digraph G has no parallel edges.

```
public class Mystery {
    private boolean[] marked;

    public Mystery(Digraph G, int s) {
        marked = new boolean[G.V()];
        mystery(G, s);
    }

    private void mystery(Digraph G, int v) {
        marked[v] = true;
        for (int w : G.adj(v))
            if (!marked[w]) mystery(G, w);
    }

    public boolean marked(int v) {
        return marked[v];
    }
}
```

(a) Describe in one sentence what the method `marked(v)` returns for vertex v after calling the constructor with a digraph G and a vertex s.

(b) Suppose that a `Digraph` is represented using the *adjacency-lists* representation. What is the order of growth of the running time of the constructor in the *worst case*?

1 $\quad V$ $\quad E$ $\quad E+V$ $\quad V^2$ $\quad EV$ $\quad E^2$

(c) Suppose that a `Digraph` is represented using the *adjacency-lists* representation. What is the order of growth of the running time of the constructor in the *best case*?

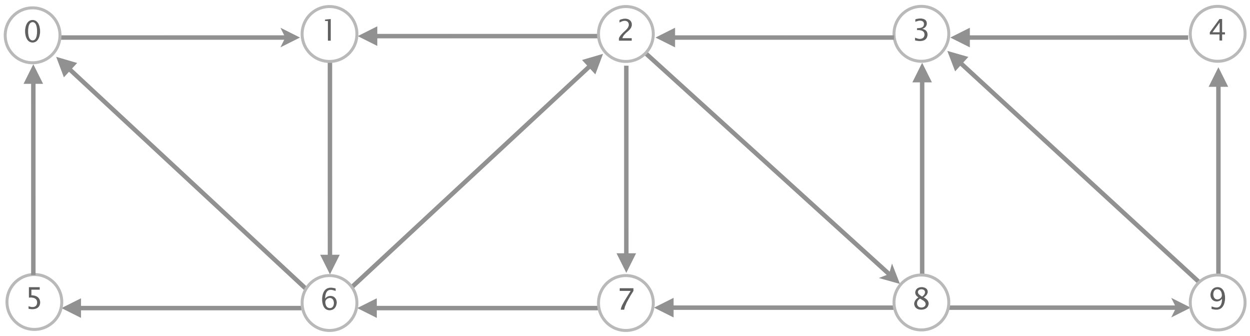
1 $\quad V$ $\quad E$ $\quad E+V$ $\quad V^2$ $\quad EV$ $\quad E^2$

(d) Suppose that a `Digraph` is represented using the *adjacency-matrix* representation. What is the order of growth of the running time of the constructor in the *worst case*?

1 $\quad V$ $\quad E$ $\quad E+V$ $\quad V^2$ $\quad EV$ $\quad E^2$

3. **Graph search. (6 points)**

Consider the following digraph. Assume the adjacency lists are in sorted order: for example, when iterating through the edges pointing from 2, consider the edge $2 \to 7$ before $2 \to 8$.



Run depth-first search on the digraph, starting from vertex 0.

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

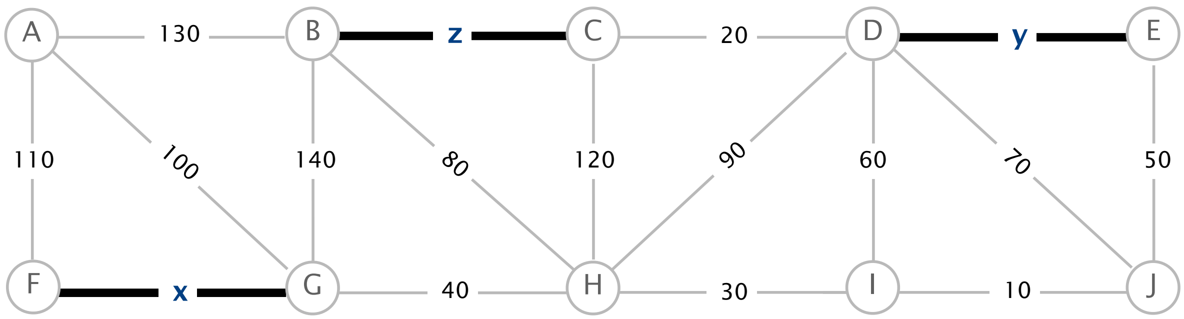
0
--- --- --- --- --- --- --- --- --- ---

(b) List the vertices in *preorder*.

0
--- --- --- --- --- --- --- --- --- ---

4. **Minimum spanning trees. (8 points)**

Suppose that a MST of the following edge-weighted graph contains the edges with weights x, y, and z.



(a) List the weights of the other edges in the MST in ascending order of weight.

```
 10
----   ----   ----   ----   ----   ----
```

(b) Circle which one or more of the following can be the value of x?

5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

(c) Circle which one or more of the following can be the value of y?

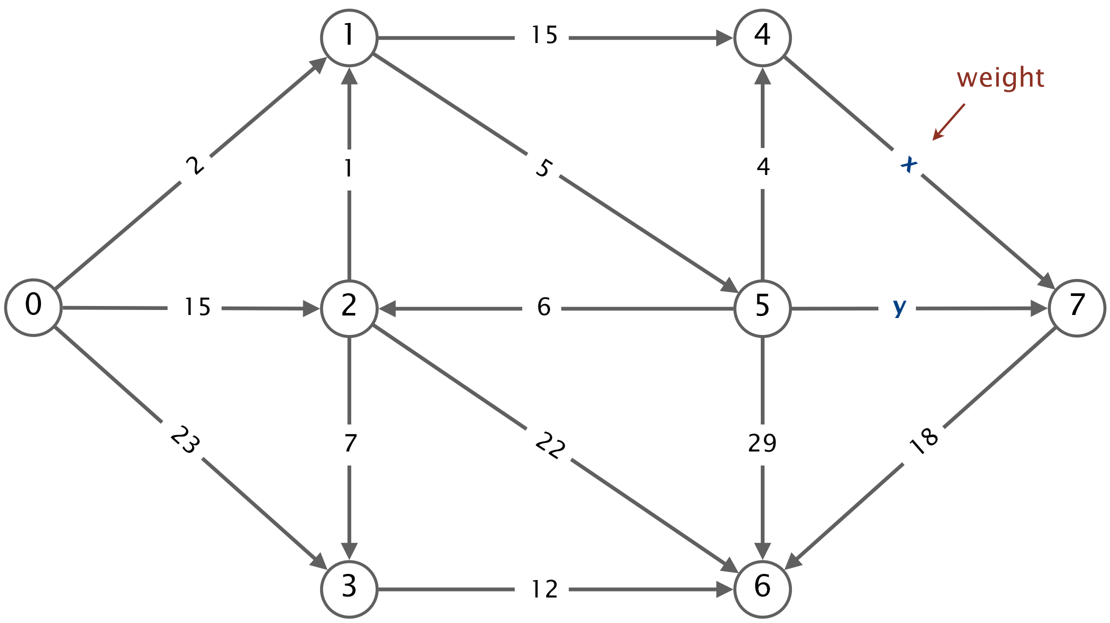
5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

(d) Circle which one or more of the following can be the value of z?

5 15 25 35 45 55 65 75 85 95 105 115 125 135 145

5. **Shortest paths. (8 points)**

Suppose that you are running Dijkstra's algorithm on the edge-weighted digraph below, starting from vertex 0.



The table below gives the `edgeTo[]` and `distTo[]` values immediately after vertex 4 has been deleted from the priority queue and relaxed.

v	distTo[]	edgeTo[]
0	0.0	*null*
1	2.0	$0 \to 1$
2	13.0	$5 \to 2$
3	23.0	$0 \to 3$
4	11.0	$5 \to 4$
5	7.0	$1 \to 5$
6	36.0	$5 \to 6$
7	19.0	$4 \to 7$

(a) Give the order in which the first 4 vertices were deleted from the priority queue and relaxed.

			4

(b) What are all possible values of the weight of the edge x?

(c) What are all possible values of the weight of the edge y?

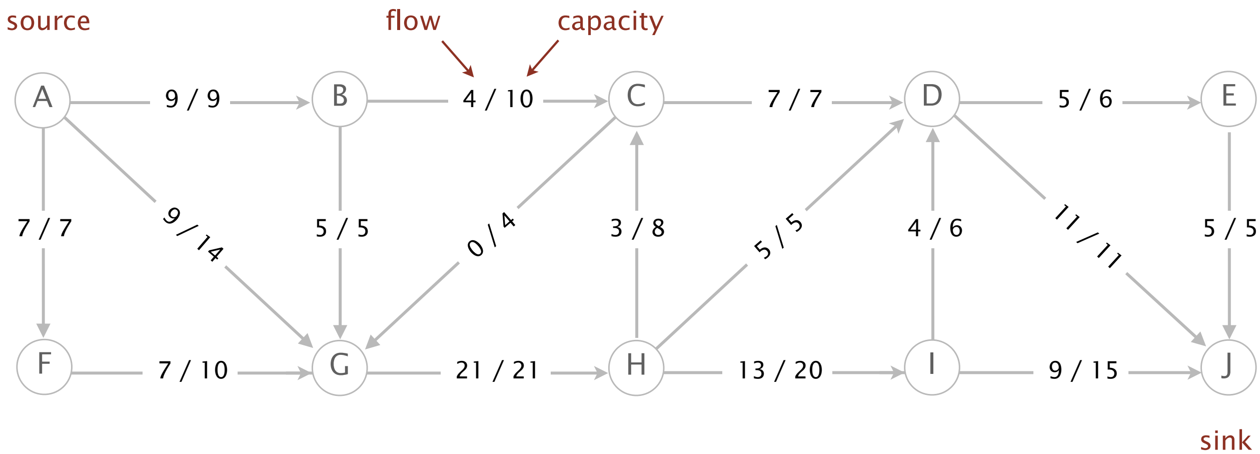
(d) Which is the next vertex to be deleted from the priority queue and relaxed?

(e) In the table below, fill in those entries (*and only those entries*) in the `edgeTo[]` and `distTo[]` arrays that change (from the corresponding entries on the facing page) when the *next* vertex is deleted from the priority queue and relaxed.

v	`distTo[]`	`edgeTo[]`
0		
1		
2		
3		
4		
5		
6		
7		

6. **Maximum flow. (8 points)**

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



(a) What is the value of the flow f?

(b) Starting from the flow f given above, perform one iteration of the Ford-Fulkerson algorithm. List the sequence of vertices on the augmenting path.

(c) What is the value of the maximum flow?

(d) List the vertices on the source side of the minimum cut in alphabetical order.

(e) What is the capacity of the minimum cut?

7. **String sorting algorithms. (7 points)**

   The column on the left is the original input of strings to be sorted; the column on the right are the strings in sorted order; the other columns are the contents at some intermediate step during one of the algorithms listed below. Match up each algorithm by writing its number under the corresponding column. You may use a number more than once.

KISS	ABBA	ENYA	ABBA	ENYA	ACDC	SOAD	SADE	ABBA
ENYA	ACDC	INXS	ACDC	ABBA	ABBA	WHAM	CAKE	ACDC
INXS	AQUA	DIDO	AQUA	AQUA	AQUA	ABBA	CARS	AQUA
STYX	BECK	CARS	BECK	ACDC	BUSH	MOBY	JAYZ	BECK
SOAD	BLUR	ACDC	BLUR	SOAD	BLUR	BECK	ABBA	BLUR
ACDC	BUSH	FUEL	BUSH	CAKE	BECK	ACDC	ACDC	BUSH
KORN	CAKE	BUSH	CAKE	MUSE	CAKE	SADE	BECK	CAKE
FUEL	CARS	ABBA	CARS	HOLE	CARS	DIDO	WHAM	CARS
BUSH	DIDO	AQUA	DIDO	SADE	DIDO	FUEL	DIDO	DIDO
ABBA	ENYA	CAKE	ENYA	BUSH	ENYA	CAKE	KISS	ENYA
WHAM	FUEL	BLUR	FUEL	RUSH	FUEL	HOLE	BLUR	FUEL
CAKE	HOLE	JAYZ	HOLE	BECK	HOLE	TSOL	INXS	HOLE
BLUR	INXS	BECK	INXS	FUEL	INXS	KORN	ENYA	INXS
MUSE	JAYZ	HOLE	JAYZ	TSOL	JAYZ	CARS	SOAD	JAYZ
BECK	KISS	KORN	KISS	WHAM	KISS	MUSE	MOBY	KISS
MOBY	KORN	KISS	KORN	KORN	KORN	BUSH	HOLE	KORN
HOLE	MUSE	TSOL	TSOL	DIDO	MUSE	RUSH	KORN	MOBY
TSOL	MOBY	MOBY	MOBY	BLUR	MOBY	KISS	AQUA	MUSE
JAYZ	RUSH	MUSE	MUSE	KISS	RUSH	AQUA	TSOL	RUSH
AQUA	STYX	SADE	SADE	INXS	STYX	BLUR	STYX	SADE
SADE	SOAD	WHAM	WHAM	CARS	SOAD	INXS	FUEL	SOAD
CARS	SADE	SOAD	SOAD	STYX	SADE	ENYA	MUSE	STYX
DIDO	TSOL	RUSH	RUSH	MOBY	TSOL	STYX	BUSH	TSOL
RUSH	WHAM	STYX	STYX	JAYZ	WHAM	JAYZ	RUSH	WHAM
----	----	----	----	----	----	----	----	----
0								1

(0) Original input

(1) Sorted

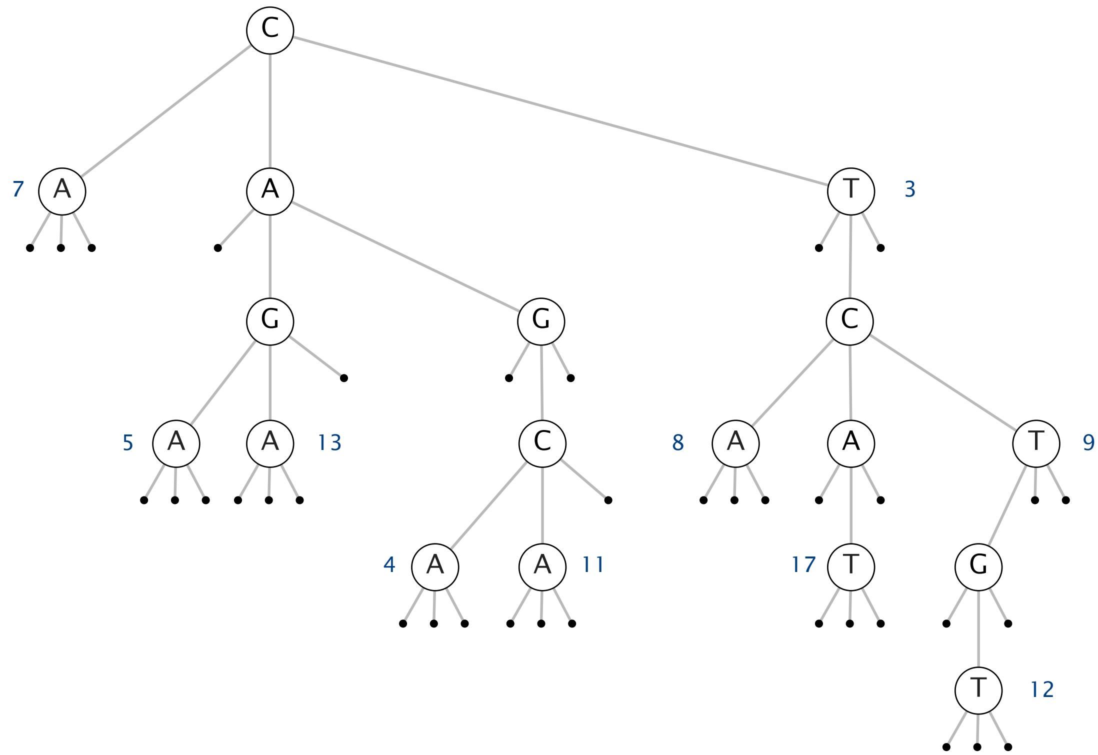
(2) LSD radix sort

(3) MSD radix sort

(4) 3-way string quicksort (no shuffle)

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

Consider the following ternary search trie, where the values are shown next to the nodes of the corresponding string keys.



(a) Circle which one or more of the following strings are keys in the TST?

`A`	`AGA`	`CA`	`CAA`	`CACA`	`CAT`	`CGA`
`CGCA`	`TA`	`TC`	`TCA`	`TGT`	`TT`	`TTT`

(b) Insert the two strings `CGTT` and `TGA` into the TST with the associated values 0 and 99, respectively; update the figure above to reflect the changes.

9. **Knuth-Morris-Pratt substring search. (5 points)**

   Below is a partially-completed Knuth-Morris-Pratt DFA for a string s of length 12 over the alphabet { A, B, C }. Reconstruct the string s in the space below. (You need not fill in the first three rows of the table, but they may be used to award partial credit.)

	0	1	2	3	4	5	6	7	8	9	10	11
A		1									11	12
B											0	5
C						3		3			0	0
s											A	A

10. **Boyer-Moore substring search. (5 points)**

    Suppose that you run the Boyer-Moore algorithm (the basic version considered in the textbook and lecture) to search for the pattern

    `I D O F T H E`

    in the text

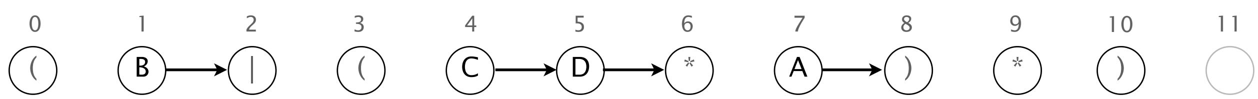
    `M E N D E R O F R O A D S W I T H T H E A I D O F T H E`

    Give the trace of the algorithm in the grid below, circling the characters in the pattern that get compared with the text.

M	E	N	D	E	R	O	F	R	O	A	D	S	W	I	T	H	T	H	E	A	I	D	O	F	T	H	E
I	D	O	F	T	H	E																					

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

Suppose that we run the RE-to-NFA construction algorithm from the lecture and textbook on the regular expression `( B | ( C D * A ) * )`. The match transitions are shown below.



Circle which one or more of the following edges are in the ϵ-transition digraph.

$0 \to 2$	$0 \to 3$	$0 \to 4$	$0 \to 8$
$2 \to 8$	$2 \to 9$	$2 \to 10$	$2 \to 11$
$3 \to 4$	$3 \to 6$	$3 \to 8$	$3 \to 9$
$5 \to 6$	$5 \to 7$	$6 \to 5$	$6 \to 7$
$8 \to 10$	$9 \to 2$	$9 \to 3$	$9 \to 8$

12. **Huffman codes. (5 points)**

(a) Draw the Huffman trie corresponding to the encoding table below.

char	freq	encoding
B	2	01111
F	1	01110
H	3	0110
I	?	00
L	5	010
M	15	10
S	15	11

(b) Circle which one or more of the following are possible values for the frequency of the character I.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

13. **Data compression. (6 points)**

What is the compression ratio achieved by the following algorithms and inputs? Write the best-matching letter from the right-hand column in the space provided. For Huffman and LZW, assume that the input is a sequence of 8-bit characters ($R = 256$).

Recall, the *compression ratio* is the number of bits in the compressed message divided by the number of bits in the original message.

_____ Run-length coding with 8-bit counts for best-case inputs of N bits.

_____ Run-length coding with 8-bit counts for worst-case inputs of N bits.

_____ Huffman coding for best-case inputs of N characters.

_____ Huffman coding for worst-case inputs of N characters.

_____ LZW coding for best-case inputs of N characters using 12-bit codewords. Recall: no new codewords are added to the table if the table already has $2^{12} = 4096$ entries.

_____ LZW coding for worst-case inputs of N characters using with 12-bit codewords. Recall: no new codewords are added to the table if the table already has $2^{12} = 4096$ entries.

A. $\sim 1/4096$
B. $\sim 1/3840$
C. $\sim 1/2731$
D. $\sim 1/2560$
E. $\sim 1/320$
F. $\sim 1/256$
G. $\sim 1/255$
H. $\sim 1/128$
I. $\sim 1/127$
J. $\sim 1/32$
K. $\sim 8/255$
L. $\sim 1/16$
M. $\sim 1/8$
N. $\sim 1/7$
O. $\sim 1/4$
P. $\sim 1/2$
Q. $\sim 2/3$
R. ~ 1
S. $\sim 3/2$
T. ~ 2
U. ~ 3
V. ~ 4
W. ~ 7
X. ~ 8

14. **Algorithm design. (8 points)**

Two strings s and t are *cyclic rotations* of one another if they have the same length and s consists of a suffix of t followed by a prefix of t. For example, `"suffixsort"` and `"sortsuffix"` are cyclic rotations.

Given N distinct strings, each of length L, design an algorithm to determine whether there exists a pair of distinct strings that are cyclic rotations of one another. For example, the following list of $N = 12$ strings of length $L = 10$ contains exactly one pair of strings (`"suffixsort"` and `"sortsuffix"`) that are cyclic rotations of one another.

```
algorithms    polynomial    sortsuffix    boyermoore
structures    minimumcut    suffixsort    stackstack
binaryheap    digraphdfs    stringsort    digraphbfs
```

For full credit, the order of growth of the running time should be NL^2 (or better) in the worst case. You may assume that the alphabet size R is a small constant. Your answer will be graded on correctness, efficiency, clarity, and succinctness.

(a) Describe your algorithm in the space below.

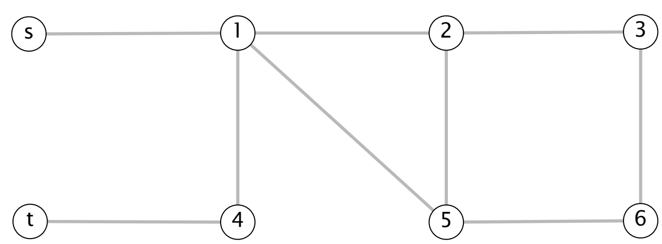
(b) What is the order of growth of the running time of your algorithm (in the worst case) as a function of both N and L?

15. **Reductions. (8 points)**

Consider the following two graph problems:

- LONGESTPATH. Given an undirected graph G and two distinct vertices s and t, find a simple path (no repeated vertices) between s and t with the most edges.
- LONGESTCYCLE. Given an undirected graph G', find a simple cycle (no repeated vertices or edges except the first and last vertex) with the most edges.

(a) Show that LONGESTPATH linear-time reduces to LONGESTCYCLE. Give a brief description of your reduction. To illustrate your reduction, superimpose the LONGESTCYCLE instance G' that it constructs in order to solve the following LONGESTPATH instance G:



(b) Circle which one or more of the following that can you infer from the facts that LONGESTPATH is NP-complete and that LONGESTPATH linear-time reduces to LONGESTCYCLE.

i. If there exists an N^3 algorithm for LONGESTCYCLE, then $P = NP$.

ii. If there does not exist an N^3 algorithm for LONGESTCYCLE, then $P \neq NP$.

iii. If there exists an N^3 algorithm for LONGESTCYCLE, then there exists an N^3 algorithm for LONGESTPATH.

iv. If there exists an N^3 algorithm for LONGESTPATH, then there exists an N^3 algorithm for LONGESTCYCLE.