COS 226	Algorithms and Data Structures	Spring 2004

Final

This test has 11 questions worth a total of 84 points. You have 150 minutes. The exam is closed book, except that you are allowed to use a one page cheatsheet. 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	
Sub 1	

Problem	Score
6	
7	
8	
9	
10	
Sub 2	

Total	

Name:

Login ID:

Precept: 1 12:30
2 1:30
4 3:30

0. **Miscellaneous. (2 points)**

   (a) Write your name and arizona login in the space provided on the front of the exam, and circle your precept number.

   (b) *Write* and sign the honor code on the front of the exam.

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

   In this course we have encountered several different types of algorithmic analyses. For each type, list *two* algorithms for which that style of analysis was used in lecture or in the textbook. Be specific, e.g., don't just say quicksort, say quicksort with partitioning on a random element.

   (a) Amortized.

   (b) Worst-case.

   (c) Average case.

   (d) Randomized.

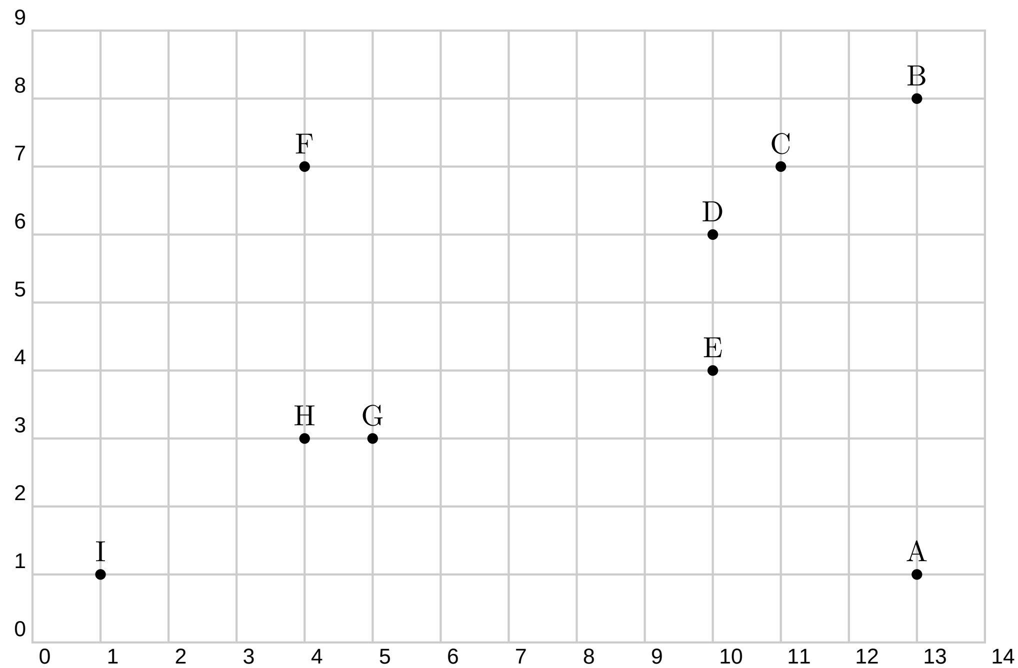
2. **String searching and pattern matching. (8 points)**

   (a) Give a DFA that recognizes the set of all strings (over the two letter alphabet) that contain `aaba`.

   (b) Give an NFA that recognizes the same language that the regular expression `(ab*a | a*b)` describes.

3. **Convex hull. (8 points)**

   Run the Graham scan algorithm to compute the convex hull of points below, starting from A. Give the points that appear on the trial hull (after each iteration) in the order that they appear. Circle your answer.



4. **Discretized Voronoi diagram. (10 points)**

Suppose you want to compute a Voronoi data structure for a set of pixels drawn from an R-by-R grid. Devise an efficient scheme to support the following three operations.

- $create(R)$: create an empty data structure to handle an R-by-R grid.
- $insert(i, j)$: insert the pixel (i, j) into the data structure, where i and j are integers between 0 and R-1.
- $find(x, y)$: return the inserted pixel which is closest to (x, y) in Euclidean distance, where x and y are integers between 0 and R-1.

The create and insert operation should take $O(R^2)$ time; the find operation should take $O(1)$ time. For partial credit, give an algorithm that takes $O(NR^2)$ time for insert, where N is the number of points already inserted into the data structure. *Hint:* think simple.

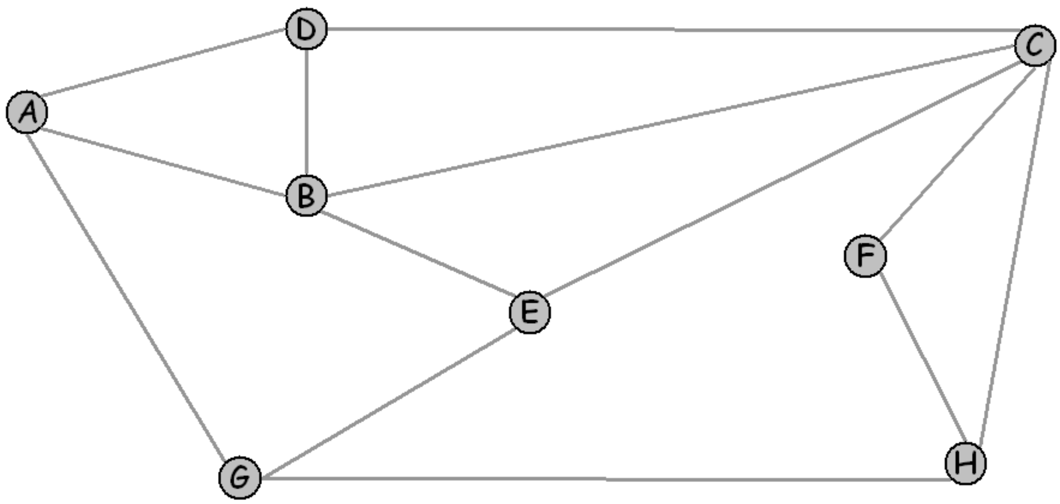
(a) Describe how to implement $create(R)$. Indicate what is being stored.

(b) Describe how to implement $find(i, j)$.

(c) Describe how to implement $insert(i, j)$.

5. **Undirected graphs. (8 points)**

Consider the following undirected network. Run depth first search, starting at vertex A. Assume the adjacency lists are in lexicographic order, e.g., when exploring vertex A, use A-B before A-H. List the vertices in the order they are visited with DFS according to preorder and postorder. Circle your answers.

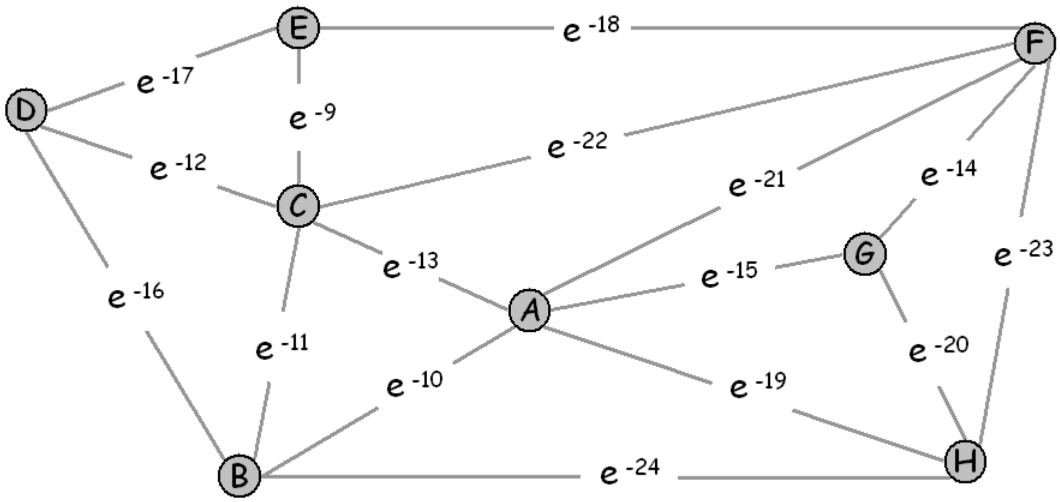


(a) Preorder.

(b) Postorder.

6. **Minimum spanning tree. (8 points)**

   Consider the following undirected network with edge weights as shown.

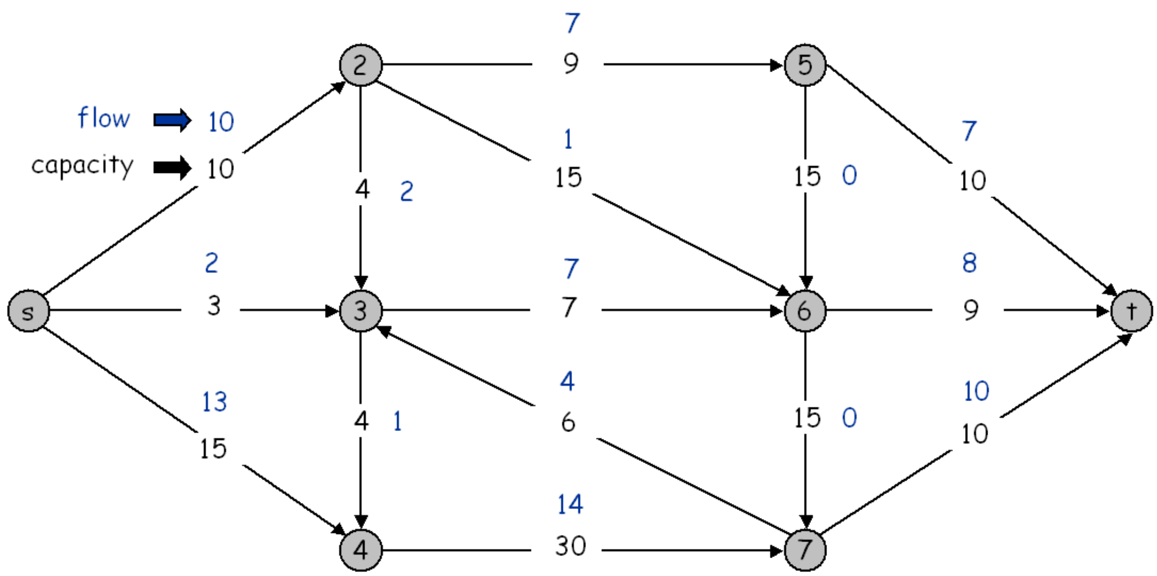


   (a) Give the list of edges in the minimum spanning tree and circle your answer. You may use the fact that e^{-1} is approximately 0.36787944117144423215955238 and e^{-2} is approximately 0.13533528323661269189399995.

   (b) Suppose that the weight of edge B-D is changed to x instead of e^{-16}. For which values of x is B-D an edge in a MST.

7. **Max flow, min cut. (8 points)**

Starting from the following flow (printed above or to the right of the capacities), perform one iteration of the Ford-Fulkerson algorithm. Choose a *shortest* augmenting path, i.e., the path with the fewest number of arcs.



(a) Write down your shortest augmenting path.

(b) Perform the augmentation. What is the *value* of the resulting flow?

(c) Is the resulting flow optimal? If so, give a min cut whose capacity is equal to the value of the flow. If not, give a shortest augmenting path.

8. **Data compression. (8 points)**

(a) What is the Burrows-Wheeler transform of "`baracadabra`"? Circle your final answer.

0											
1											
2											
3											
4											
5	b	a	r	a	c	a	d	a	b	r	a
6											
7											
8											
9											
10											

(b) What is the *inverse* Burrows-Wheeler transform of "`7 rdrbcaaaaab`". Circle your final answer. Show your work for partial credit.

												next
0											r	
1											d	
2											r	
3											b	
4											c	
5											a	
6											a	
7											a	
8											a	
9											a	
10											b	

9. **Linear programming. (8 points)** (Hillier and Lieberman, 3.4-9)

You are the pilot of a cargo plane. The plane has three compartments (front, center, and back) with the following absolute limits on how much weight and volume you can store in each.

	Weight (Tons)	Space (Cubic Feet)
Front	12	7,000
Center	18	9,000
Back	10	5,000

To balance the plane, the amount of weight in each compartment must be exactly equal. You can ship any (or all) of the following four types of cargo, in whole or fractional units.

	Volume (Cubic Feet / Ton)	Profit (Dollars / Ton)
1	500	320
2	700	400
3	600	360
4	400	290

You may ship at most 20, 16, 25, and 23 tons of cargo 1, 2, 3, and 4, respectively. Formulate (but do not solve) a linear program to determine how many tons of each cargo you should ship so as to maximize profit. Use only the following 12 decision variables. It's not necessary to put it into standard form.

$$\begin{aligned} F_i &= \text{tons of cargo } i \text{ to assign to the front compartment} \\ C_i &= \text{tons of cargo } i \text{ to assign to the center compartment} \\ B_i &= \text{tons of cargo } i \text{ to assign to the back compartment} \end{aligned}$$

Put your answer to on the facing page and circle your final answer.

Put your answer to question 9 here. Don't forget to do question number 10.

10. **Reductions. (8 points)**

Given an *undirected* graph with V vertices, E edges, two distinguished vertices s and t, and nonnegative weights on the remaining *vertices*, find a path from s to t that minimizes the sum of the *vertex* weights. Devise an $O(E \log V)$ algorithm for this problem by reducing it to standard shortest path problem on *directed* graphs.

To demonstrate your reduction, draw the shortest path network (directed graph + edge weights) associated with the following input. Assuming it is clear from the picture how your reduction extends to arbitrary graphs, you need not describe it further.