FACULTY OF SCIENCE

FINAL EXAMINATION

COMPUTER SCIENCE 308-251B
DATA STRUCTURES AND ALGORITHMS

Examiner: Luc Devroye — Wednesday, April 24, 1996

Associate Examiners: Nick Roy, Raul Silvera — 9:00–12:00

Instructions: This is a closed-book examination. However, each student is allowed to bring one colored regular-sized (8 by 11) sheet with any amount of personal handwriting on front and back, as well as a magnifying glass. Answer on the question sheets.

Identification.

YOUR NAME	STUDENT ID NUMBER

Marks.

QUESTION	WEIGHT	EST. TIME	YOUR SCORE
1	9%	15 min.	
2	9%	15 min.	
3	6%	15 min.	
4	15%	20 min.	
5	10%	10 min.	
6	15%	20 min.	
7	12%	20 min.	
8	6%	20 min.	
9	8%	10 min.	
10	10%	15 min.	
TOTAL	100%	160 min.	

Question 1. Lempel-Ziv coding. A given primitive language has just two letters, `b` and `o`. A message of 12 consecutive letters without blanks is given to us: `booboobooooo`). We store this message as a standard USASCII-encoded text file on our UNIX-based system (b is hexadecimal 62, o is hexadecimal 7F).

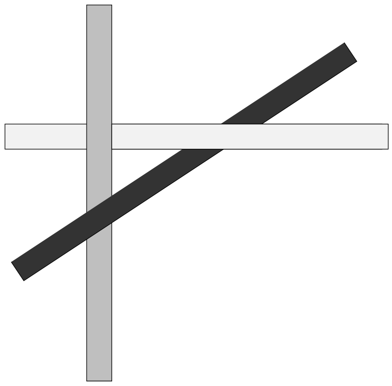
A. If we use USASCII encoding, how many bits are taken by this message?

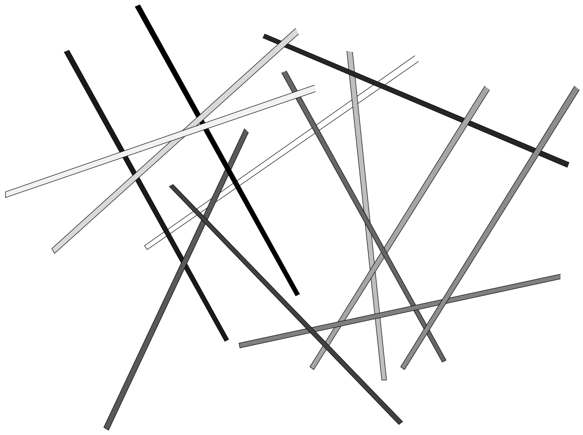
B. We compress the message by the LZ78 method (the Lempel-Ziv method seen in class). Describe the bits of the compressed message. This is best done by showing the compressed sequence, and using arrows and boxes to show what means what.

Question 2. Applications. Assume that I want to keep a data base of pictures. Each picture consists of 1000×1000 bits that represent black or white pixels. There are about 8000 pictures in my collection. As new pictures arrive, I would like to quickly determine if I already have them in my data base. How would you deal with this problem? Please be specific in terms of the sizes of various things!

Question 3. Linear time selection. Give the recurrence for the worst-case time of selection based upon finding the median of medians-of-3 (instead of medians-of-5 as is standard). What is the solution of the recurrence and how did you get it (short answer, please).

Question 4. Graphs. Assume that we are given a pile of n sticks. A stick can be removed from this pile by a crane that lifts it vertically, provided that no other stick obstructs its path; we say that an obstructing stick is "on top". Convince yourself that some piles may easily be untangled (i.e., all sticks can be removed one by one), and some cannot be untangled. For each stick `s`, we are given a list of sticks `s'` that are "on top" of it. The figure on the left shows three sticks that cannot be untangled. The pile on the right can be untangled however.





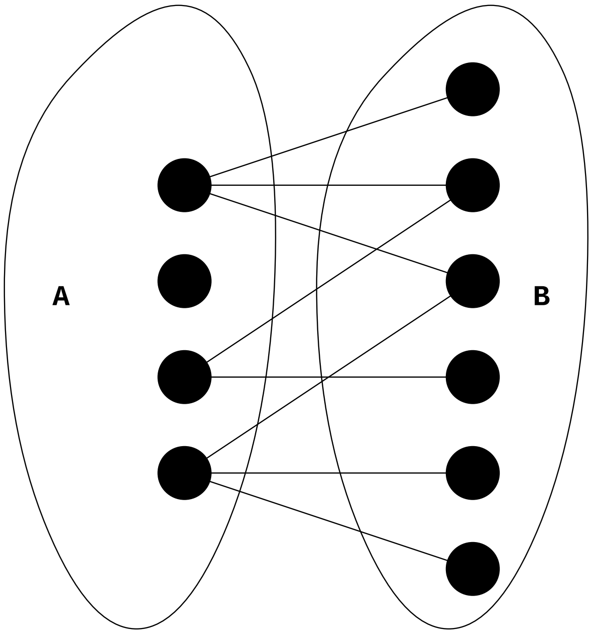
A. State this problem in graph theoretical lingo.
B. Give an algorithm for determining if a pile can be untangled, and if so, give the order of stick removal.
C. What is the complexity of your algorithm?

Question 5. Huffman trees. This question is about a Huffman code for sending a coded video image. Each image consists of a sequence of gray-level pixels, where a gray-level pixel takes a value between 0 and 15 (there are 16 gray levels). If you send the raw pixels, it takes 4 bits per pixel. However, after looking at many data, we have found that pixels occur with the following frequencies in typical images:

pixel value	probability
0 and 15	1/4
1 and 14	1/8
2 and 13	1/16
3 through 12	1/80

Show a possible Huffman code for the pixels and compute the expected length of the code per pixel (unfinished summations are accepted).

Question 6. Creative question on graphs: recognizing bipartite graphs. A bipartite graph is a graph that is such that one can split the nodes in two sets A and B such that all edges flow from A to B and no edge is between two nodes in A or two nodes in B. In other words, A and B are islands without roads, and there only exist bridges between A and B. A graph (V, E) with $|V|$ nodes, numbered 1 through $|V|$ is given in adjacency list format. The figure below illustrates the structure of any bipartite graph:



Part A. Give a PASCAL or C type definition for storing a graph in adjacency list format.

Part B. Give an algorithm that takes an arbitrary graph and verifies in time $O(|E|+|V|)$ that this graph is bipartite.

Part C. Indicate why your algorithm takes linear time.

Question 7. The ADT median-priority-queue (or MPQ for short). The ADT median-priority-queue operates on a set of elements with different key values, and involves the following operations:

1. `makenull`
2. `insert`
3. `delete-median`
4. `makeMPQ`

It is known that the number of elements never exceeds a given number n. The operation `makeMPQ` takes a given number of elements and inserts these globally (not one by one)—recall `buildheap` for a similar situation. For `delete-median`, note that the median of a set of $2n$ elements is the n-th smallest, and that the median of a set of $2n+1$ elements is the $n+1$-st smallest element. Suggest, adapt or invent a data structure so that the following can be achieved:

A. Each `insert` and `delete-median` takes $O(\log n)$ worst-case time.

B. `makeMPQ` takes $O(n)$ time in the worst case, where n is the number of elements involved.

Your answer should be such that a good programmer would know how to implement your data structure. You must tell us therefore in words how `makeMPQ`, `insert` and `delete-median` are to be implemented, and you must draw the data structure.

Why is it impossible to perform a `makeMPQ` on n elements followed by n `delete-median`s using less than $(n \log_2 n)/2$ comparisons?

Question 8. Theory. Prove that the height of a red-black tree with n external nodes does not exceed $2 \log_2 n$.

Question 9. Pixel quadtrees. In a pixel quadtree, we observe 3001 final regions (each region corresponds to a leaf in the pixel quadtree). How many internal nodes are there?

Question 10. Complete binary trees. A complete binary tree is implemented with the help of left and right pointers, and not as an array as is common practice. Nodes in a complete binary tree have a standard numbering system, where the root has number 1, its children numbers 2 and 3 and so forth. The operation `locate (n,T)` locates the node with number n in a tree `T`, where `T` may be considered as a pointer to the root. This must be done in $O(\log n)$ time using only $O(\log n)$ extra space. The operations log and exp are not available, but +, -, mod and idiv (integer division) are. Give your algorithm below.