University of Illinois at Urbana-Champaign
Department of Computer Science

Qualifying Examination—Part I

Theoretical Computer Science

9 a.m.–12 noon, Tuesday, November 12, 2001

Do not write your name anywhere on this examination, so that the exam can be graded without knowledge of your identity. Write your ID number legibly in the box below and at the upper right corner of every page. Also, in the box below, write a pseudonym of your choice (not your name or ID) to be used in posting the results of this exam.

ID Number:
Pseudonym:

Do all four problems in this booklet. The remaining four problems will be in Part II this afternoon. All eight problems are equally weighted, so do not spend too much time on any one question. Blank pages follow each problem for extra workspace.

Question	Points	Score	Grader
1	50		
2	50		
3	50		
4	50		
5	50		
6	50		
7	50		
8	50		
Total	400		

1. **Bin Packing.**

   The **Bin-Packing** problem is defined as follows. You are given an infinite supply of bins of unit size and a set of n items to be packed into these bins. The goal is to pack the given set of items into as few bins as possible.

   (a) The **Exponential-Packing** problem is the **Bin-Packing** problem with the additional restriction that the size any item is one of 1/2, 1/4, 1/8, 1/16, ... ,$1/(2^k)$,. Either show that **Exponential-Packing** is NP-complete or give a polynomial time algorithm for solving it.

   (b) The **Harmonic-Packing** problem is the **Bin-Packing** problem with the additional restriction that the size of any item equals either 1/2, 1/3 or 1/4. Either show that the **Harmonic-Packing** problem is NP-Complete or give a polynomial time algorithm for solving it.

2. **Median Oracle.**

   You are given an unsorted array of n distinct items from some totally ordered universe (say, the integers). For whatever reason, the Powers That Be have made it impossible for you to directly compare items in this list. Instead, you have access to a `Median` oracle, which tells you in $O(1)$ time which of three arbitrary items from the array is between the other two.

   (a) Prove that it is impossible to find the minimum element of the array using only the `Median` oracle.

   (b) Describe and analyze an efficient algorithm to find the minimum and maximum elements of the array using the `Median` oracle.

   (c) Describe and analyze an efficient randomized algorithm to find the median of the entire array using the `Median` oracle. For extra credit, describe a deterministic algorithm!

   (d) Describe and analyze an efficient randomized algorithm to sort the array using the `Median` oracle, in either increasing or decreasing order. For extra credit, describe a deterministic algorithm!

   (e) Prove that your sorting algorithm is optimal.

3. **Connected Components.** Given an undirected graph G with n vertices, and two vertices s and t, describe a *deterministic* algorithm that decides if s and t are in the same connected component of G (i.e., Is there a path between s and t in G?) using only $O(\log n)$ space (in the unit-cost model, or $O(\log^2 n)$ space in the ram (log cost) model.

4. **Iteration Lemmas.**

Ogden's Lemma is often not powerful enough to prove non-membership of a language in the class of context free languages. In this exercise we will prove a stronger condition due to Bader and Moura.

I Consider the language $L = b^* \cup aa^+b^* \cup \{ab^p | p \text{ prime}\}$. Show that Ogden's lemma cannot be used to prove that L is not context-free.

Bader-Moura's Lemma states the following: For a context-free language L there is an integer N such that for any word w and for any choice of d distinguished positions and e excluded positions in w with $d > N^{1+e}$, there are words x, u, y, v, z such that

(a) $w = xuyvz$, with $uv \neq \epsilon$ (the empty string),

(b) either x, u, y each contain at least one distinguished position, or y, v, z each contain at least one distinguished position,

(c) the word uv contains no excluded position,

(d) if uyv contains r distinguished and s excluded positions, then $r \leq N^{1+s}$, and

(e) $xu^nyv^nz \in L$, for all n.

We will now prove this lemma by solving the following subparts. For a grammar $G = (V, T, P, S)$, let t be the maximal length of the right-hand side of the productions, k be the number of non-terminals (i.e., $|V| = k$), and let $N = t^{2k+6}$ (this is the N in the lemma). Consider a word w with e excluded and d distinugished positions, as in the lemma. A node in the derivation tree for w will be called a *branch point* if it has at least two children which have distinguished descendents.

II Show that there is a path P in the derivation tree that has at least $2(k+3)(e+1)$ branch points.

A branch point in P is called a left branch point if the node has a child with a distinguished descendent to the left of P; a right branch point is defined analogously. Note, a branch point maybe both a left and a right branch point.

Clearly, P has at least $(k+3)(e+1)$ left branch points or $(k+3)(e+1)$ right branch points. Assume that P has $(k+3)(e+1)$ left branch points.

III Show that in P there are $e+1$ *pairs* of branch points, such that the nodes in each pair are labelled by the same non-terminal.

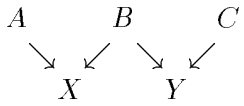
Let $X_1, \ldots X_{e+1}$ be the labels of these pairs of left branch points. Thus there exist $e+1$ pairs of words $u_i, v_i \in T^*$ such that $X_i \Rightarrow^* u_i X_i v_i$. More over since these are obtained from left branch points each u_i has a distinguished position. Now, since there are $e+1$ pairs of words (u_i, v_i), and only e excluded positions, there is one pair (u_i, v_i) which has no excluded position.

IV Prove the lemma.

V Use the lemma to show that the language defined in part I is not context-free. You may assume that the lemma is right, even if you could not prove it.

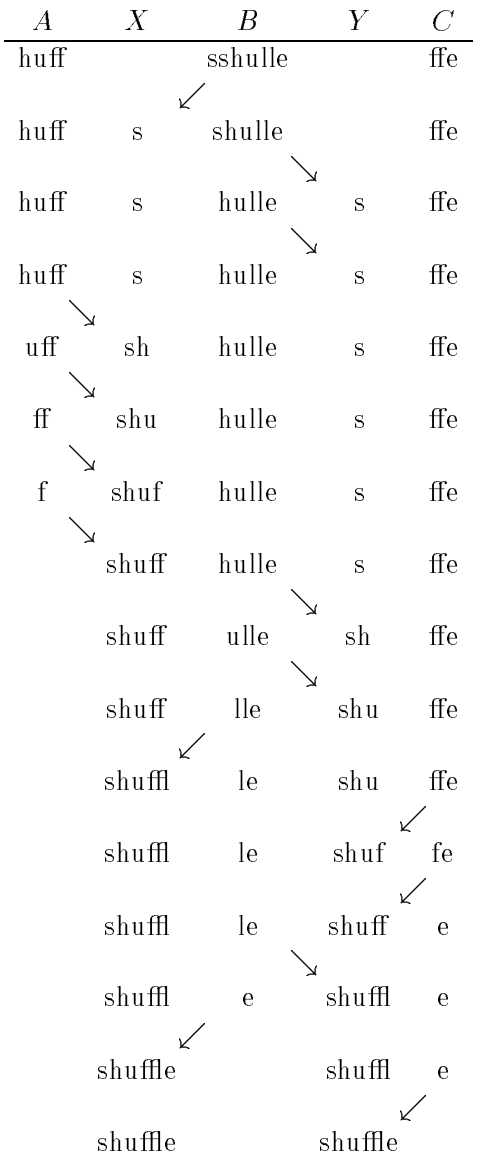
5. **Double Shuffle.**

Consider the following puzzle game, called "Double Shuffle". You are given three input strings A, B and C. During the course of the puzzle, you will create two output strings X and Y. In each move, you remove the first character from one of the input strings and add it to the end of the one of the output strings. You are not allowed to move a character from A to Y or from C to X; the only legal moves look like this:



The game ends when every character has been moved to the output strings. The puzzle is *solved* if the final output strings X and Y are the same. The following page gives a solution for A = "huff", B = "sshulle", and C = "ffe".

Here is a solution for a sample puzzle:



Describe and analyze an efficient algorithm that either outputs a solution to any given instance of Double Shuffle or correctly reports that no solution is possible.

6. **Tree Separators.**

$T = (V, E)$ is a tree, a node $v \in T$ is a separator of T, if v satisfies the following condition:

$$\forall f \in F \qquad |f| \leq \frac{2}{3}n,$$

where $F = T \setminus \{v\}$ is the forest resulting from T by removing v and all the edges attached to v from T, $n = |T|$, and $|T|$ denotes the number of nodes in a tree T. Intuitively, a separator is a good vertex to use in breaking a tree into subtrees.

(a) Prove that there exists a separator in any tree.

(b) Show a linear time algorithm to find a separator of a tree.

(c) We assign a weight $w(e)$ to each edge e of the tree. Design a data structure so that the tree built using the data structure can answer the query "Which is the edge with the smallest weight on the unique path between the nodes a and b?" in $O(\log n)$ time. The data-structure should use $O(n \log n)$ space and preprocessing.

(d) (* hard) Show a data-structure to answer the query in $O(\log \log n)$ time, that uses $O(n \log \log n)$ space and preprocessing.

(e) (** really hard) Show that one can build a data-structure that answer the query in $O(1)$ time with $O(n \log n)$ space.

7. **Randomized Computation.**

   (a) Prove that if L $\in$ RP, then for each n there is a circuit $C_{L,n}$ of size at most $p(n)$ for some polynomial p, such that for all strings w of length n, $C_{L,n}(w) = 1$ iff $w \in L$. *Hint: a sequence s of bits flipped witnesses that $w \in L$ if the RP machine accepts w when the coin flips agree with s. Use a counting argument to show that there is a small (poly) sized set S of witnesses such that for any $w \in L$, one of the elements of S is a witness for w.* Why doesn't this prove that P = NP?

   (b) Show that there are undecidable languages L such that L has poly-sized circuits as above (i.e., for each n there is a circuit $C_{L,n}$ accepting exactly the elements of $L \cap \Sigma^n$.

   (c) Prove that if NP $\subseteq$ BPP, then in fact NP = RP.

8. **Fixed Parameter Complexity.**

   Recall vertex cover is the problem of determining given a graph G and number k whether or not there is a subset of vertices of cardinality k or less such that each edge is incident to at least one of the vertices. Give an algorithm for vertex cover that runs in time $f(k)p(|G|)$, where p is a polynomial, and f is any function. How does this compare with the obvious $O(|G|^k)$ algorithm?