PROBLEMS

The sets described below shall be used in many of the problems for this chapter. As usual, w represents a string of 0's and 1's and the superscript R stands for *string reversal*.

$A = 0^n1^n$	$B = 0^n1^n0^n$	$C = 0^n1^m0^n1^m$
$D = w\#w^R$	$E = w\#w$	$F = ww$

Measures and Resource Bounds

1. Present the least time consuming algorithms you are able for deciding membership in sets A, D, and F above on multitape Turing machines. Analyze the space required for these algorithms.

2. Develop the most space efficient algorithms you can for membership in the sets A, D, and F above. What time is needed for each?

3. How many times a Turing machine reverses the direction of its tape heads can be a measure of complexity too. Try to compute membership in the sets A, D, and F above with the fewest number of tape head reversals. Describe your algorithms.

4. Assume that one unit of ink is used to print a symbol on a Turing machine tape square. How much ink is used to decide membership in (as usual) sets A, D, and F above?

5. Try time as a complexity measure on one tape (rather than multitape) Turing machines. Now how much time do algorithms for deciding A, D, and F take?

6. Let T(n), S(n), R(n), and I(n) denote the time, space, reversals, and ink needed for Turing machine M to process inputs of length n. Why is it obvious that a machine always uses more time than space? This indicates that $S(n) \leq T(n)$. What are the relationships between the above four measures of complexity.

7. Define a measure of complexity which you feel properly reflects the actual cost of computation. Determine the complexity of deciding membership in the sets A, D, and F above.

8. Using multitape Turing machines as a computational model, precisely prove that linear speedup in time is possible.

9. Let us use Turing machines with a read-only input tape and a single work tape as our computational model. In addition, let us only use 0, 1, and blank as tape symbols. Is there a difference in space complexity when multitrack work tapes are used as opposed to one track work tapes? (That is, if a task requires m tape squares on a k track machine, how much space is needed on a one track model?)

10. Solve exercise 9 for time instead of space.

11 Show that the simulation of a k tape Turing machine's computation by a one tape Turing machine can be done in the square of the original time.

12. If $\inf_{n\to\infty} f(n)/g(n) = 0$, then what is the relationship between the functions f and g? Does f = O(g)? Does g = O(f)?

13. If $\inf_{n\to\infty} f(n)/g(n) = k$, for some constant k, then what is the relationship between the functions f and g? Again, does f = O(g)? Does g = O(f)?

14. What are the time and space requirements for LL parsing (from the presentation on languages)?

15. Using the speedup theorem, prove the bizarre corollary concerning supercomputers and pocket calculators.

Complexity Classes

1. Show that n^2 and 2^n are time functions.

2. Demonstrate that $(\log_2 n)^2$ is a space function.

3. Prove that for every recursive function f(n) there exists a time function t(n) and a space function s(n) which exceed f(n) for all values of n.

4. Show that there are recursive functions which cannot be time or space functions.

5. Verify that a set is a member of *DSPACE(n)* if its members can be recognized within O(n) space for all but a finite number of n.

6. Prove that if a set can be *accepted* by some deterministic Turing machine, then its membership problem can be *decided* in the same amount of space or time.

7. Show that the family of *DSPACE* classes is closed under union, intersection, and complement.

8. Prove that the family of *NSPACE* classes is closed under concatenation and Kleene star.

9. Design a nondeterministic Turing machine which accepts members of the above sets A and C. (That is, $A \cup C$.) Is this quicker than doing it in a deterministic manner?

10. Present a nondeterministic Turing machine which finds roots of polynomials. (That is, given a sequence of integers a_0 , ... , a_n it figures out a value of x such that: $a_0 + a_1x + a_2x^2 + ... + a_nx^n = \delta$ for some specified error bound δ.)

11. Describe how in a nondeterministic manner one could accept pairs of finite automata which do not accept the same set.

12. Show that if a space function is at least O(n) then it is efficiently space computable.

13. How much space is needed to compute $\log_2 n$?

14. Prove the time hierarchy theorem using $r(n)^2$ rather than $r(n)\log_2 r(n)$.

Reducibilities and Completeness

1. Show that the set A above is reducible to the sets D and F by mappings which can be computed by finite automata.

2. Provide a mapping from set D above to set E. Analyze the space requirements of the mapping.

3. Show that set A above is reducible to set C. Take a Turing machine M_C which decides membership in C, combine it with the machine which maps A into C, and produce a machine which decides membership in A. Further, do this so that the space requirements are only $O(\log_2 n)$.

4. Precisely state and prove theorems 1 and 2 for time complexity.

5. Show that if the set A is a member of the class *DTIME(B)* then *DTIME(A)* is a subclass of *DTIME(B)*. Be sure to consider the cases where the sets A and B have speedup.

6. Prove theorem 4 for nondeterministic time classes.

7. Prove that complexity classes named by sets with speedup cannot be named by recursive functions. (If S has speedup then there is no recursive f such that *DSPACE(A) = DSPACE(f)*.)

The Classes $\mathcal{P}$ and $\mathcal{NP}$

1. Estimate the time (in seconds, years, etc.) that it would take to solve a problem which takes $O(2^n)$ steps on a typical computer. Do this for various values of n.

2. Prove that any set in $\mathcal{P}$ can be reduced to any other set in $\mathcal{P}$ via some polynomial time mapping.

3. Verify theorem 2.

4. Show that the existence of a set in $\mathcal{NP}$ whose complement was not also in $\mathcal{NP}$ would lead to a proof that $\mathcal{P} \neq \mathcal{NP}$.

5. Demonstrate that the entire satisfiability problem is indeed in $\mathcal{NP}$.

6. Present an algorithm for converting a Turing machine instruction into the set of clauses needed in the proof that SAT is $\mathcal{NP}$-complete.

Intractable Problems

1. Convert the set of clauses $\{(v_1), (v_2,\overline{v_3}), (v_1,\overline{v_2},v_3)\}$ into proper 3-SAT format.

2. Convert the set of clauses $\{(v_1,\overline{v_2}), (v_1, v_2, v_3, v_4)\}$ into proper 3-SAT format.

3. How much time does it take a nondeterministic Turing machine to recognize a member of 3-SAT? Assume that there are n clauses.

4. Show precisely that 0-1INT is in $\mathcal{NP}$.

5. What time bound is involved in recognizing cliques in graphs? Before solving this, define the data structure used as input. Might different data structures (adjacency matrix or edge list) make a difference in complexity?

6. Verify that the transformation from 3-SAT to CLIQUE can be done in polynomial time.

7. Prove that the vertex cover problem is $\mathcal{NP}$-complete.

8. Show how to construct in polynomial time the graph used in the proof that COLOR is $\mathcal{NP}$-complete

9. Suppose you wanted to entertain n people. Unfortunately some of them do not get along with each other. What is the complexity of deciding how many parties you must have to entertain all of the people on your list?