Topics for the Final Examination

- Design of automata and grammars; regular expressions
- Computability problems
- Decision problems
- Undecidability via reduction
- Recursive and Recursively Enumerable sets
- P and NP algorithms
- NP-completeness via Polynomial Time Computable Reduction
- No pumping lemmas, no congruence lemmas, and no diagonalizations. You will be asked to prove undecidability via reduction and NP-completeness via polynomial time reduction.
III. Prove that the following problem is computable.

\[ f(x, y) = \begin{cases} 
1 & \text{if there exist TMs } M_1, M_2 \text{ s.t. } x = [M_1], \ y = [M_2], \ \text{and } L(M_1) \cap L(M_2) \neq \emptyset \\
\text{undefined} & \text{otherwise}
\end{cases} \]

- Check whether \( x \neq y \) are codes of TMs. If not go into a loop, i.e., \( f(x, y) \) is undefined.
- If no, let \( x = [M_1], \ y = [M_2] \).
- At any stage keep track of the sets of strings accepted by \( M_1 \) and \( M_2 \) up to that instant. Let \( L_i \) = \( 2^i \). Initially, \( L_0 = L_1 = \emptyset \).
- At any stage \( i \), being in simulation of \( M_1, M_2 \) on input \( x \) (as a binary string). Simulate all the pending processes by one additional step. If an \( M_j \) accepts on input \( j \), include \( j \) in \( L_i \). Remove \((M_j, j)\) simulation process. Then check whether \( j \) is in the current \( L_{i+1} \). If so, output 1 and halt. Else, go to the next stage.

IV. Given TMs \( M_1, M_2, \ldots, M_n \), prove that the following set is recursively enumerable:

\[ \{ x \mid \text{every } M_j \text{ halts on input } x \} \]

Function \( f \) keeps track of the number of TMs that have so far halted on any particular input. For example, if \( f(i) = k \) then all that stage \( k \) of the \( n \) TMs have halted on input \( x \).

- At any stage \( i \), start simulations of \( M_1, M_2, \ldots, M_n \) on input \( i \) as separate processes. Set \( f(i) = 0 \). Simulate all the running processes by one more step. If any \( M_j \) illegally halts on input \( i \), then remove this process \( x \) increment \( f(i) \) by 1. Then check whether \( f(i) = n \). If so, output 1. Go to stage \( i+1 \).
VII. Design a P algorithm for the following problem. Also, estimate the speed of the algorithm.

Given a digraph \( G \), a vertex \( u \), and a value \( k \), are there \( k \) distinct vertices \( v_1, v_2, \ldots, v_k \) s.t. for every \( i \in \{1, \ldots, k\} \), there is a path from \( u \) to \( v_i \) and there is a path from \( v_i \) to \( u \)?

A similar problem is done in class this year (2013).

VIII. Design an NP algorithm for the following problem. Prove its correctness and estimate its speed.

Given \( n \) digraphs, \( G_1, G_2, \ldots, G_n \), each having \( n \) vertices, does there exist a \( k \geq n/2 \) s.t. at least \( n/2 \) of the digraphs contain a simple cycle of length \( k \)?

\[
\text{Guess } i_1, i_2, \ldots, i_k. \quad \text{Verify } i_1 < i_2 < \ldots < i_k \text{ and } k \geq n/2.
\]

For each \( ij \): guess \( k \) vertices \( v_{i_1}, \ldots, v_{i_k} \). Verify that 
\( v_{i_1}, \ldots, v_{i_k} \) form a simple cycle in \( G_{ij} \).

Correctness: easy.

Speed: Testing the distinctness of \( v_i \), \( v_j \) quickly requires \( O(k^2) \) steps; overall \( O(n^3) \).
All the subproblems carry equal weight.

I. Design a nondeterministic pda for the language:
\[ \{xyycz | x, y, z \in \{a, b\}^*, y = x^R \text{ or } (z \neq y^R \text{ and } \#_awz \text{ is an odd integer})\}. \]
III. Prove that the following problem is decidable.

1. Given $[M_1]$ and $[M_2]$, $M_1$ a dfa and $M_2$ a dlba, is $L(M_1)$ a finite set and $L(M_1) \cap L(M_2) \neq \emptyset$?

Testing $L(M)$ is finite set.

Minimizing the number of states of $M$. The only property we need is that there are no useless states.

Let $M$ have $s$ states.

Claim: $L(M)$ is finite if $M$ doesn't accept any string of length in the range $[1, 2^s]$.

Now generate all strings of length 1, then $x1$, then $x2$, ..., then $x2^s$ to check whether at least one of them is accepted by $M$.

So $L(M)$ is not finite. If not, $L(M)$ is finite.

Knowing $L(M)$ is finite, to test $L(M) \cap L(M_2) \neq \emptyset$.

Let $M_2$ have $t$ states and $t$ tape symbols. We have shown that if $M_2$ accepts a string of length $n$, then it must accept within $3n^2$ steps. So any given string can be tested for acceptance by $M_2$.

Now we generate all strings of length 1, then length 2, then length 3, ..., then length $2^s$. For each string, test whether it is accepted by both $M_1$ and $M_2$. If there is such a string output 'yes', else output 'no' at halt.
V. Prove the undecidability of the following problem.

1. Given $[M_1], [M_2], [M_3]$, do there exist $x_1, x_2, x_3$ such that $x_1 < x_2 < x_3$ and for $i = 1, 2, 3$, Turing machine $M_i$ halts on input $x_i$? (Hint: Make use of the undecidability of the blank tape halting problem.)

BTTP $\leq_m$ this problem

Typically: $(M), [M_1], [M_2], [M_3]$

Given $(M)$, we transform it to $(M), [M_1], [M_2], [M_3]$ by:

- TM $M$ halts on $B$ if there exist $x_1, x_2, x_3$ s.t. $x_1 < x_2 < x_3$
- $M_i$ halts on $x_i$
- We choose $x_1 = 1, x_2 = 11, x_3 = 111$

- TM $M_1$: halts on input 1, $\not{\text{halts}}$ on any other input.
- TM $M_2$: halts on input 11 and $\not{\text{halts}}$ on any other input.
- TM $M_3$: $\not{\text{halts}}$ on any input other than 111. On input 111, it erases the input & simulates $M$ on blank tape.

Note: that our goal is achieved.

Also, the transformation from $(M)$ to $[M_1], [M_2], [M_3]$ is computable.
VII. Design a P algorithm for one of the following problems. Estimate its speed.

1. Given a directed graph $G$ with $n$ vertices, is there an ordering of its vertices into $v_{k_1}, v_{k_2}, \ldots, v_{k_n}$ such that for every $1 \leq i < j \leq n$ there is an edge from $v_{k_i}$ to $v_{k_j}$?

2. Given an nfa $M$ and a string $x$, is $x \in L(M)$? (Conversion of $M$ into dfa is not possible since this can take an exponential number of steps.)

1. Let the adjacency matrix be $A$.
   Find an $i$ s.t. the $i$th row is all 1's except $A(i,i)$.
   Make $i$ to be $k_1$. Eliminate ith row and ith column.
   Repeat the process to find $k_2, \ldots, k_n$.
   Each iteration requires scanning all the entries of $A$ once, i.e. $O(n^2)$ steps. Thus, done $O(n)$ times, overall also: $O(n^3)$ steps.

2. Let $x = a_1 a_2 \cdots a_n$.
   For each $i$, we compute $S_i$: the set of states $M$ can reach on input $a_1 a_2 \cdots a_i$.
   Start with $S_0 = \{q_0\}$.
   Computation of $S_{i+1}$: For each $q \in S_i$, place all states from $q$ on input $a_{i+1}$.
   Remove duplicates.
   At the end, check whether $S_n$ contains a final state.
   Speed: $|S_i| \leq n$. From each state in $S_i$ on input $a_{i+1}$, $M$ can reach at most $n$ states. So the number of steps needed to compute $S_{i+1} = O(n^3)$.
   Overall $O(n^3)$. 

7