600.271 Key to HW8
Sp '13

(1) First we present a general algo that computes all the vertices that are reachable (i.e. exist paths to) from any given vertex v. (this was discussed in the class). ~~start~~ create a set $S = \{v\}$. From each vertex v in S compute $\{w \mid (v, w)$ is an edge$\}$. Add this set to S. If the size of S increases then repeat the above procedure. We called it the "contigion method" or "breadth first search". Since the number of vertices is n & since repetition happens only after the size of S increases by at least 1 there are at most $n-1$ iterations. The overall speed is $O(nm)$. A more careful way results in $O(m)$ steps.

Now test that all vertices are reachable from the given vertex. This takes $O(nm)$ steps. Then test that from each of the other vertices u is reachable. This takes $O(n^2m)$ steps. The overall speed is $O(n^2m)$. So the algo. is in P.

After testing that there is a path from u to every vertex, the rest of the computation cs be speeded up by first computing

$G^R = \{(u,v) \mid (v,u)$ is an edge of $G\}$ in $O(n^2)$

steps. Then test whether every vertex is reachable from u in G^R.

You don't need to worry about this refinement.

IV 4) Given an nfa M and a string x, is $x \in L(M)$?

We cannot assume that the length of $[M]$ is far smaller than $|x|$. So we cannot convert the nfa M to a dfa. The size of the dfa can be exponentially larger than $|[M]|$.

Let $x = a_1 \ldots a_m$

For each $0 \le i \le m$ we compute

$S_i = \{q \mid$ nfa M can end up in state q on the input $a_1 \ldots a_i\}$

$S_0 = \{q_0\}$ q_0 is the initial state of M

Computation of S_i from S_{i-1} $\underbrace{a_1 \ldots a_{i-1}}_{q} \, a_i \quad q \to q'$

$S_i = \{q' \mid q \in S_{i-1}$ and $q \xrightarrow{a_i} q'$ is a transition of $M\}$

After computing S_m, we check whether one of the final states of M is in S_m.

We will grade this as a BONUS PROBLEM since you are not required to know any aspects of the dynamic programming technique.

IV 6. Algo: Let the elements be $x_1, \ldots, x_n$.

Compute $W = \sum_{i=1}^{n} x_i$.

Check whether any $x_i > 2\frac{W}{3}$. If so, output 'yes' else output 'no' and halt.

Clearly the algo runs in $O(n)$ steps.

Now we argue that the algo is correct.

If some $x_i > \frac{2W}{3}$, then in any partition, the set that doesn't contain this x_i ~~is~~ adds upto $< \frac{W}{3}$.

If no $x_i > \frac{2W}{3}$; then we argue that ~~a~~ the required partition exists.

Case 1: If there exists an x_k s.t. $\frac{W}{3} \le x_k \le \frac{2W}{3}$, then $\{x_k\}, \{x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n\}$ ~~is~~ is OK.

Case 2: If every $x_k < \frac{W}{3}$:

Start with $S = \{x_1\}$.

~~Greedily~~ Repeatedly & greedily include the next element int S until the sum of the elements is $\ge \frac{W}{3}$. This sum must be $< \frac{2W}{3}$ since each element is $< \frac{W}{3}$. Now the remaining elements form block 2.

8. CFG G, is $L(G) = \phi$?

we first compute $\Delta = \{A \mid (\exists x \in \Sigma^*) (A \overset{*}{\Rightarrow} x)\}$.

Then we check whether $S \in \Delta$. If $S \notin \Delta$ then 'yes'.

<u>Computation of Δ</u> (For simplicity, we assume G in normal form).

If $A \to a$, $a \in \Sigma$, is a production then include A in G.

Repeat the following procedure as long as the size of Δ increases

For every production $A \to BC$, if $B, C \in \Delta$ then add A to Δ.

Since the size of Δ is bounded by the number of nonterminals, if there are t nonterminals, the number of repetitions is $\leq t-1$.

~~Argue that it runs in O(~~

If the size of G is n, then the number of nonterminals is $\leq n$ & the number of productions is $\leq n$

Each repetition takes $O(n)$ steps. Hence the overall algorithm runs in $O(n^2)$ steps.

V 1. digraph G, k

$k \le l_1, l_2 \le n$.

nondet Guess $u_1, u_2, \ldots, u_{l_1}$
$v_1, v_2, \ldots v_{l_2}$ s.t. ~~$l_1, l_2 \le k$~~.

check that $u_1, u_2, \ldots, u_{l_1}, v_1, v_2, \ldots, v_{l_2}$ are distinct.

check that $(u_1, u_2), (u_2, u_3), \ldots, (u_{l_1 - 1}, u_{l_1}), (u_{l_1}, u_1)$,
$(v_1, v_2), (v_2, v_3), \ldots (v_{l_2 - 1}, v_{l_2}) (v_{l_2}, v_1)$ are

edges.
If all the tests succeed, output 'yes' else output 'no' & halt.
Argue the correctness.

Verifier method: Proof needed is
$u_1, u_2, \ldots, u_{l_1}, v_1, v_2, \ldots, v_{l_2}$. The rest of the steps from the det. verifier.

5. $x = a_1 a_2 \cdots a_n$, is x not a prime.
Nondet guess $y = b_1 b_2 \cdots b_m$, $z = c_1 \cdots c_l$, $m, l \le n$.
Then check $x = yz$. If so, answer 'yes', else answer 'no'.

<u>Verifier method</u>: Proof needed is y, z.
Then the verifier is as above.

(Of course, in both cases y suffices. Then we divide x by y & test that there is no remainder.)