CMSC 303 Introduction to Theory of Computation, VCU
Spring 2017, Course Mini-Project
Due: Tuesday, May 2, 2017 in class

Matthew Bowers

Total marks: 26 marks + 3 marks bonus for typing your solutions in LaTeX.

One of the goals of this course is to train you to be able to expand your education independently, given the tools you've learned here. For this reason, your mini-project will consist of the following task. First, please independently read Sections 8.1 and 8.2 on Space Complexity. This will introduce you to the complexity class PSPACE and Savitch's theorem. Next, answer the following questions.

1. [2 marks] In words, which class of problems does PSPACE characterize?
   Because NSPACE $\subset$ SPACE PSPACE characterizes all NP problems.

2. [4 marks] Show that PSPACE is closed under the concatenation and star operations.
   Devise a nondeterministic TM that nondeterministically guesses the end of one language and the beginning of the next. The machine will accept only if the accepting branches are in the correct order. We know that each $L \in$ PSPACE so our machine will be NSPACE($f(n)$) where $f(n)$ is some polynomial function which describes the space used by the worst case branch of our TM. This non-deterministic machine can be simulated on a deterministic machine in $O(f^2(n))$ space.

   The case for star is the same except that each language is the same so we have only the worst case.

3. [4 marks] Prove that $\mathrm{NP} \subseteq \mathrm{PSPACE}$ by giving a polynomial-space algorithm for simulating an NP machine (use Theorem 7.20 for this, which says a language is in NP iff it is decided by a non-deterministic polynomial time Turing machine).

   Claim: SAT is in PSPACE:
   By brute force we can enumerate all possible assignments of the variables in ϕ we can then design a nondeterministic TM to run test every possible assignment. This can be done in polytime and each branch of the TM will use polyspace. It is already shown that such an arrangement will fall int PSPACE.

   Further we know that all NP problems can be reduced to SAT in polytime though they may expand they will not expand beyond polyspace because it would be impossible to carry out such an expansion in polytime. The algorithm is as follows:
   1. Reduce arbitrary NP problem Y to SAT by the poly algorithm that must exist 2. Use CANYIELD to simulate our encoding of SAT.

4. [2 marks] Why is Savitch's theorem surprising, given our study of P versus NP?
   NP problems are vastly more complicated than P problems. In fact P problems are generally trivial compared with NP problems. On the surface this makes that idea of NP and P problems being at all the same surprising. However, I only need one bed to sleep every night, and a family of 6 can get by with but a single toilet if you are willing to hold it for a bit.

5. [14 marks] This question studies the proof of Savitch's theorem. You may assume the TM M in the proof can compute function $f(n)$ in the proof in $O(f(n))$ space.

   (a) [2 marks] In words, give a high-level overview of how the proof of Savitch's theorem works.

   Check if N accepts s or not. This process will be done by repeatedly asking if the machine can path from the start configuration to the end configuration in less than the maximum possible steps by attempting to path through every other possible configuration using half as many steps. By allowing each successive attempt half as many steps we limit the overall number of recursive calls which can be made.

   (b) [6 marks] The CANYIELD procedure in the proof has 6 steps: Describe in a sentence or two the purpose of each step.

   Step 1: is c_1 an accept state or is it 1 step away from the accept state.
   Step 2: It is an enhanced for loop to try every configuration Step 3: try to get from the starting configuration to the current target configuration using a finite and diminishing number of steps. Step 4: Try to get from the current target configuration to the accept configuration using a finite and diminishing number of steps. Step 5: Check if you can complete the current path in the requisite steps and return to the previous call. Step 6: If pathing between every intermediate configuration failed reject.

   (c) [2 marks] Why does the machine M in the proof correctly simulate the NTM N?
   All M does is reframe the question in a clever way. Asking if a Language is decided is equivilant to asking if you can get from the starting configuration to the accepting configuration especially considering that TM's have only 1 accepting configuration unlike NFA's.

   (d) [4 marks] Why does M use $O(f^2(n))$ space?
   We are told that each level of the recurrsion takes $O(f(n))$ space. So the question falls to how deep the recurrsion goes. N was designed to have at most $2^{df(n)}$ possible configuations so in the worst case, to visit every possible configuration will take that many steps. Because the algorithm employs a divide and conquer strategy the depth of the tree will be $\log(\text{max steps})$ which comes out to $O(f(n))$. In total then we have $O(f(n))$ layers each taking $O(f(n))$ space.