Solutions for Assignment 1

October 22, 2003

1. Answers

a) False
We will show by contradiction that $f \notin O(g)$. Suppose $f \in O(g)$, then there exist constants $c > 0$ and $a > 0$ such that $\frac{f(n)}{g(n)} \leq cn$ for all $n \geq a$. Thus

$$ \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 $$

which contradicts the previous statement.

b) False
Take $f(n) = 1$ and $g(n) = n$. Then $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$, and

$$ f(n) = \Theta(g(n)) $$

since $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{1}{n} \to 0$.

c) True
Suppose $f \notin O(g)$. Then we know there exists a constant $c$ such that $f(n) \neq O(n)$ for all $n \geq a$. Thus $f(n) < g(n)$ for all $n \geq a$; in other words a constant $d > 1$ such that $g(n) \geq f(n)$ for all $n \geq a$, which implies $g \in O(f)$.

d) False
Counterexample: $f(n) = 1$, $g(n) = n$. In this case $f \notin O(g)$ since for $c - 1$ and $n - 1$, $f(n) = 1 = \frac{f(n)}{g(n)}$ for all $n \geq 1$. However $g \notin O(f)$ since $\lim_{n \to \infty} \frac{f(n)}{g(n)} = \frac{1}{n} \to 0$, which is not below any constant.

2.1

2.2

• Step 1 (state q1 and q2). Go over tape 1 writing its contents to tape 2.

• Step 2 (state q3). Rewind tape 2 while keeping tape 1 in its last input symbol.

• Step 3 (state q4). Go over both tapes, tape 1 from right to left and tape 2 from left to right, comparing symbols. If symbols are equal at any point, reject. Stop when reaching detriment point of tape 1.

Let $T(n)$ be the worst case running time of $4k$, Step 1, 2, and 3 each consists of a single scanning of the tape so $T(n) = O(n)$

2.3

• Step 2. Scan the tape again crossing the first symbol that is not on $X$ and the last symbol that is not in $X$. If original symbols were equal then reject. If no symbol is found then accept.

• Step 3. Go to step 2.

The first step can be done with a single scan of the tape, so it takes $O(n)$ steps. Normally step 2 takes $O(n)$ steps. Step 2 is repeated at most 4 times, since in each step two symbols are transferred into $X$. Therefore the total running time is in $O(n^5)$

2.4 RAM Program
We will use registers:

$r_0$ - original
$r_1$ - next index of element to examine
$r_2$ - next index of element to examine

1 LOAD 1
3 LOAD 1
4 STORE 2
5 LOAD 1
6 ADD 2
7 STORE 1
8 LOAD 2
9 STORE 1
11 ADD 1
12 LOAD 1
13 SUB 1
14 STORE 1
15 JMP 3 //Jmp 3-5 reads a, ..., a into r2, ..., r4
16 LOAD 1
17 SUB -1
18 STORE 2 //r2 = n + 2
19 LOAD -3
20 STORE 1 //r1 = 3
21 LOAD 2
22 SUB 1

2.4 Description and analysis

- Step 1. Scan the tape crossing the first and last symbols. If symbols were equal then reject.

- Step 2. Scan the tape again crossing the first symbol that is not on $X$ and the last symbol that is not in $X$. If original symbols were equal then reject. If no symbol is found then accept.

- Step 3. Go to step 2.
23. \textsc{check} \( \epsilon \) \( < r_1 \) then go to accept
24. \textsc{load} 1
25. \textsc{sub} 1 2
26. \textsc{if} \((\text{false symbol} = \text{sum}) \) then reject
27. \textsc{load} 1
28. \textsc{add} \(-1\)
29. \textsc{store} 1
30. \textsc{load} 2
31. \textsc{sub} \(-1\)
32. \textsc{store} 2
33. \textsc{jump} 21
34. \textsc{load} \(-1\)
35. \textsc{halt}
36. \textsc{load} \(-8\)
37. \textsc{halt}

2.6. Running Time
The first loop (line 3 to line 15) simply reads the input which takes time \( O(n) \). The second loop (line 21 to line 31) runs for \( n^2 \) iterations, so it takes time \( O(n^3) \). The total program runs in time \( O(n^3) \).

3. Let \( A \subseteq P \) \( \neq \emptyset \) and \( A \neq \epsilon \). Since \( A \in P \), we know \( A \in \text{NP,} \) it remains to show \( A \in \text{NP-complete} \).
We need to show that \( A \subseteq \text{NP} \) for all \( L \in \text{NP} \).
Let \( L \in \text{NP} \) be an arbitrary language in \( \text{NP} \). Since \( A \subseteq P \), we conclude \( L \subseteq A \) and so there exists a polynomial time algorithm \( B \) that decides \( L \). We will build a reduction algorithm \( F \) (reduction from \( L \) to \( A \)) in the following way:

Algorithm \( F(L) \):

1. Let \( \epsilon \) be a string in \( A \).
2. Let \( b \) be a string in \( L \).
3. If \( \epsilon \in L \) then \( F(L) = 1 \) and \( F(L) = -1 \)
4. Else \( F(L) = 0 \)

Moreover, since \( B \) runs in polynomial time and \( F \) simply calls \( B \) (polynomial time) and does a constant number of steps, \( F \) runs in polynomial time.

5. Proof:
Step 1: \textsc{dedsat} \( \in \text{NP} \)

Verification: Two truth assignments

Algorithm: \( F(\phi), \phi \neq \phi' \)

- Evaluate formula \( \phi \) using truth assignment given in \( \phi' \). If \( \phi \) is a satisfying assignment then return \( \phi' \).
- Evaluate formula \( \phi \) using truth assignment given in \( \phi' \). If \( \phi \) is a satisfying assignment then return \( \phi' \).
- If \( \phi \neq \phi' \) then return \( \phi' \).

A \( \phi \) is satisfiable \( \iff \) there exist two distinct truth assignments that satisfy \( \phi \) and there exists input \( \phi' \) for \( A \) that causes \( A \) to return 1.
Algorithm \( A \) runs in polynomial time since the formula evaluation can be done in linear time with the size of \( \phi \). Also, \( \phi \) and \( \phi' \) has at most \( n \) variables in \( \phi \), so time complexity is linear time.

Step 2: We will prove \( \text{SAT} \subseteq \text{dedsat} \)
Step 3: We will describe the reduction algorithm \( F \).

Algorithm \( F(\phi) \):

1. Let \( x_1, x_2, \ldots, x_n \) be the variable in \( \phi \)
2. Build another formula \( \phi_0 \) equivalent to \( \phi \) but substituting variables \( x_1, x_2, \ldots, x_n \) by \( \phi_1, \phi_2, \ldots, \phi_n \).
3. Create a formula \( \phi \) as the conjunction of \( \phi \) and \( \phi_0 \) where \( \phi \) has \( \phi_1 \) and \( \phi_0 \).
4. \( \phi \) is satisfiable.