Homework 10

CMPSC 465

Kinner Parikh

December 1, 2022

Problem 1:

I did not work with anyone
I did not consult without anyone my group member
I did not consult any non-class materials

Problem 2: *Apply the greedy algorithm for Horn formulas (covered in the lecture) to find the variable assignment that solves the following horn formulas:*

1. $(w \wedge y \wedge z) \Rightarrow x, (x \wedge z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \wedge y) \Rightarrow w, (\bar{w} \vee \bar{x} \vee \bar{y}), (\bar{z})$

To find if this is valid, we first set all the variables to false. Then, we go through the list of clauses and check if the clause is true. If it is true, we move on to the next clause. If it is false, we check if the clause is a horn clause. If it is a horn clause, we set the variable to true and move on to the next clause. If it is not a horn clause, we move on to the next clause. We repeat this process until we have gone through all the clauses. If we have gone through all the clauses and all of them are true, then the formula is valid. If we have gone through all the clauses and at least one of them is false, then the formula is not valid. We set x to true from the $\Rightarrow x$ statement. From this, we know that y is true because of $x \Rightarrow y$. w must also be true because $(x \wedge y) \Rightarrow w$. Plugging these into $(\bar{w} \vee \bar{x} \vee \bar{y})$, it evaluates to false. Even though $\bar{z}$ is true, all complete negative statements do not evaluate to true, so there is no valid variable assignment.

2. $(x \wedge z) \Rightarrow y, z \Rightarrow w, (y \wedge z) \Rightarrow x, \Rightarrow z, (\bar{z} \vee \bar{x}), (\bar{w} \vee \bar{y} \vee \bar{z})$

Using the same rules as defined above, we know that z must be true from $\Rightarrow z$. w must be true as well from $z \Rightarrow w$. All other statements evaluate correctly, so x and y remain false. Plugging these into $(\bar{z} \vee \bar{x})$ and $(\bar{w} \vee \bar{y} \vee \bar{z})$, they evaluate to true. There is a valid variable assignment. The statement that solves the horn formula is $z \Rightarrow w$.

Problem 3: *Linear time algorithm*

```
Algorithm 1: Contiguous Subarray of Maximum Sum
Input: A list of numbers $S = a_1, a_2, \ldots, a_n$
Output: The contiguous subsequence of maximum sum (a subsequence of length zero has
        sum zero)
maxSoFar ← $S[1]$;
currMax ← $S[1]$;
start, end, sFast ← 1, 1, 1;
for $i \leftarrow 2$ to $n$ do
    currMax = $max(S[i], currMax + S[i])$;
    if $maxSoFar < currMax$ then
        maxSoFar ← currMax;
        start ← sFast;
        end ← $i$;
    end if
    if $currMax <$ 0 then
        s ← $i + 1$;
    end if
end for
return subarray $S[start, end + 1]$;
```

maxSubArray($S[1 \ldots i]$) = max($S[i]$, maxSubArray($S[1 \ldots i - 1]$) + $S[i]$)

This recurrence relation takes the list of numbers between 1 and an index i where $i \leq n$, and finds the maximum subarray within that subarray. As we continue, the outer subarray will continue to grow and we can grow to find the overall subarray with the largest sum. Because this algorithm traverses the list of numbers once, the time complexity is $O(n)$.

Problem 4: *Find length of longest common substring in O(mn) time*

Algorithm 2: Length of the longest common substring

```
Input: Two strings $S_1$ and $S_2$
Output: The length of the longest common substring
$n \leftarrow length(S_1)$;
$m \leftarrow length(S_2)$;
$LCS[1 \dots n][1 \dots m] \leftarrow 0$;
for $i \leftarrow 1$ to $n$ do
    for $j \leftarrow 1$ to $m$ do
        if $S_1[i] = S_2[j]$ and $i > 1$ and $j > 1$ then
            $LCS[i][j] \leftarrow LCS[i-1][j-1] + 1$;
        end if
        else
            $LCS[i][j] \leftarrow 0$;
        end if
    end for
end for
return max(LCS);
```

$$LCS(i,j) = \begin{cases} 0 & \text{if } S_1[i] \neq S_2[j] \\ LCS(i-1,j-1)+1 & \text{if } S_1[i] = S_2[j] \end{cases} \tag{1}$$

If two characters in the two strings are the same, then we can assume that it will be the start of the longest substring, so we want to increment the count by one at that index. If it continues to be a substring, then $LCS(i+1, j+1)$ will hold the value $LCS(i,j)+1$. We run through each of the strings once. The time complexity is $O(mn)$.