CS 170 Homework 6

Due **3/4/2024, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".
Solution: Lakshya Nagal, SID: 3037935253

2 2-SAT

In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (OR) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value true or false to each of the variables so that all clauses are satisfied – that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee x_4)$$

Recall that $\vee$ is the logical-OR operator and $\wedge$ is the logical-AND operator and $\overline{x}$ denotes the negation of the variable x. This instance has a satisfying assignment: set x_1, x_2, x_3, and x_4 to `true, false, false, and true`, respectively.

The purpose of this problem is to lead you to a way of solving 2SAT efficiently by reducing it to the problem of finding the strongly connected components of a directed graph. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes: one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from from $\overline{\alpha}$ to β, and one from the $\overline{\beta}$ to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to each of the implications $\overline{\alpha} \implies \beta$ and $\overline{\beta} \implies \alpha$. In this sense, G_I records all implications in I.

(a) Show that if G_I has a strongly connected component containing both x and $\overline{x}$ for some variable x, then I has no satisfying assignment.
Solution: If G_I has an SCC that contains both a literal and its negation, by the definition of SCC there is a path from the literal to its negation, leading to a contraction. consider the following SCC:

$$\begin{tikzcd}
x_1 \arrow[dd] & & x_2 \arrow[ll] \\
& \overline{x_1} \arrow[dr] & \\
x_4 \arrow[ur] \arrow[rr] & & x_3 \arrow[uu]
\end{tikzcd}$$

If $x_1 : T, x_2 : T, x_4 : T, x_3 : T$ we have a $(x_4 \to \overline{x_1}) \leftrightarrow (T \to F)$ but this is a contradiction. If $x_1 : T, x_4 : F$ we have $F \to F$ which is true but now we have $(x_1 \to x_4) \leftrightarrow (T \to F)$ which is a contraction. It can be shown similarly that any assignment leads to a contraction.

(b) Now show the converse of (a): namely, that if none of G_I's strongly connected components contain both a literal and its negation, then the instance I must be satisfiable.

Hint: Pick a sink SCC of G_I. Assign variable values so that all literals in the sink are True. Why are we allowed to do this, and why doesn't it break any implications?
Solution: A sink by definition has no outgoing edges, we pick a sink SCC because we know that if there are only incoming edges, and all the variables are true, then we cant possibly have a contradiction from an implication coming from another SCC as $F \to T$ is true and $T \to T$ is a also true. Within the SCC any an assignment of true for all the variables also doesn't leave room for contradictions.

$$\begin{tikzcd}
& \overline{x_3} \arrow[dr] & \\
t/f \arrow[ur, bend left] & x_1 \arrow[u] & x_2 \arrow[l]
\end{tikzcd}$$

Possible assignment to make all x_i in SCC above true $x_1 : T, x_2 : T, x_3 : F$

(c) Conclude that there is a linear-time algorithm for solving 2SAT. Provide the algorithm description and runtime analysis; proof of correctness is not required.
Solution: Construct a graph like the one given in the description of the problem. Using Kosaraju's algorithm we can find all the strongly connected components in $O(2n + 2m)$. We can then iterate over all the strongly connected components and check if we have a literal and its negation in the same SCC. If we do return "not satisfiable" otherwise assign values to all the variables in the SCC such that they are true. The total runtime of this algorithm is $O(n + m)$.

3 Copper Pipes

Bubbles has a copper pipe of length n inches and an array of nonnegative integers that contains prices of all pieces of size at most n. He wants to find the maximum value he can make by cutting up the pipe and selling the pieces. For example, if length of the pipe is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6).

length	1	2	3	4	5	6	7	8
price	1	5	8	9	10	17	17	20

Give a dynamic programming algorithm so Bubbles can find the maximum obtainable value given any pipe length and set of prices. Clearly describe your algorithm and analyze its runtime (proof of correctness not required).

Solution:

Let $f(i)$ be the maximum value we can get by cutting the pipe into pieces from 1 to i.

$$\begin{aligned}
f(0) &= 0 \\
f(1) &= price[1] \\
f(2) &= \max\{f(1), price[2]\} \\
f(3) &= \max\{price[1] + f(2), price[2] + f(1), price[3] + f(0)\} \\
f(4) &= \max\{price[1] + f(3), price[2] + f(2), price[3] + f(1), price[4] + f(0)\} \\
&\vdots \\
f(i) &= \max_{1 \le j \le i}\{price[j] + f(i-j)\}
\end{aligned}$$

Algorithm:

```
1: function COPPER_PIPES($n$, prices)
2:     f$[n+1] \leftarrow 0$                          ▷ Initialize array to have $n+1$ zeros
3:     for $i \leftarrow 1$ to $n$ do
4:         for $j \leftarrow 1$ to $i$ do
5:             f$[i] \leftarrow \max($f$[i]$, prices$[j]$ + f$[i-j])$    ▷ Apply recurrence from above
6:     return f$[n]$
```

Runtime Analysis:

The outer loop goes from 1 to n and the inner loop iterates per outer loop $1, 2, 3, 4, \dots, n$ respectively. For a total runtime of $O(n^2)$. The space complexity is $O(n)$.

4 Mechanical DP (Optional, Ungraded)

(a) In the longest common substring problem, you are given two strings, $a = a_1a_2\cdots a_n$ and $b = b_1b_2\cdots b_m$. You want to find the largest k for which there are indices i and j with $a_ia_{i+1}\cdots a_{i+k-1} = a_ja_{j+1}\cdots a_{j+k-1}$.

(i) For strings $a =$ "compsci" and $b =$ "pompous", fill out remainder of the DP table below.

	c	o	m	p	s	c	i
p	0	0	0	1	0	0	0
o	0	1	0	0	0	0	0
m	0	0	1	0	0	0	0
p	0	0	0	1	0	0	0
o	0	1	0	0	0	0	0
u	0	0	0	0	0	0	0
s	0	0	0	0	1	0	0

(ii) What's the longest common substring?
Solution: Longest common prefix can be found on the diagonal, "omp".

(b) In the longest common subsequence problem, you are given two strings, $a = a_1a_2\cdots a_n$ and $b = b_1b_2\cdots b_m$. You want to find the largest k for which there are indices $i_1 < i_2 < \cdots < i_k$ and $j_1 < j_2 < \cdots j_k$ with $a_{i_1}a_{i_2}\ldots a_{i_k} = b_{j_1}b_{j_2}\ldots b_{j_k}$.

(i) For strings $a = algorithm$ and $b = lithium$, fill out the remainder of the DP table below.

	a	l	g	o	r	i	t	h	m
l	0	1	1	1					
i	0	1	1	1					
t	0	1	1	1					
h	0	1	1	1					
i	0	1	1	1					
u	0	1	1	1					
m	0	1	1	1					

(ii) What's the longest common subsequence?

(iii) The table below is a DP table for the longest common subsequence for string X and string Y. What indices of X form the longest common subsequence? List all possibilities.

	$X[0]$	$X[1]$	$X[2]$	$X[3]$	$X[4]$
$Y[0]$	0	0	0	0	0
$Y[1]$	0	1	1	1	1
$Y[2]$	0	1	1	2	2
$Y[3]$	0	1	1	2	3

(iv) Is the following sub-table possible in the longest common subsequence problem? Explain.

	$X[0]$	$X[1]$
$Y[3]$	2	2
$Y[4]$	3	3

(v) Is the following sub-table possible in the longest common subsequence problem? Explain.

	$X[0]$	$X[1]$
$Y[0]$	0	1
$Y[1]$	1	2