CCCP – Test 1 (Due: 30 May 2021 11:59)

Aakash, 2018B4A70887

May 29, 2021

1. [20] Let us define a multiplication operations on three symbols a, b, c according to the following table:

	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

thus $ab = b, ba = c$, and so on. Note that the multiplication operation defined by the table is neither associative nor commutative.

Find an efficient algorithm that examines a string of these symbols, say *bbbbac* and decides whether or not it is possible to paranthesize the string in such a way that the value of the resulting expression is a. For example, on input *bbbbac*, your algorithm should return yes because $((b(bb))(ba))c = a$.

You have to submit the following:

(a) Derivation of pseudo code that you are going to write (should consist of main steps, can ignorre corner cases here)

(b) Complete pseudo code/C++ code on how to solve this problem, given a string input s consisting of only characters a,b and c.

(c) Running time of your algorithm.

2. [(3+7)*3=30] Consider the following struct representing node of a tree:

```
struct TreeNode
{
    int elem;
    TreeNode* left;
    TreeNode* right;
};
```

Write the following functions in pseudo code/C++ by first deriving/proving/mentioning how/why your algorithm works

```
// This function returns the number of times 'num' appears in the tree rooted at 'root'
int occurence(TreeNode* root, int num);
// Returns the pointer array containing rightmost path in the tree.
// Last element of vector should have no childs.
vector<TreeNode*> rightMostPath(TreeNode* root);
// Checks whether the tree rooted at 'root' is max heap or not.
// For a tree to be maxHeap,
// the value at every node should be greater than or equal to its childs.
bool maxHeap(TreeNode* root);
```

Note. Consider each and every corner case. One mistake will result in -3 points.

3. [3 + 7] You are given a string of n characters $s[1 \cdots n]$, which you believe to be a corrupted text document in which all punctuation has vanished (so that it looks something like itwasthebestoftimes...). You wish to reconstruct the document using a dictionary, which is available in the form of a Boolean function $dict()$ for any string w,

$$\text{dict}(w) = \begin{cases} true & \text{if } w \text{ is a valid word} \\ false & \text{otherwise} \end{cases}$$

Give a dynamic programming algorithm that determines whether the string $s[\cdot]$ can be reconstituted as a sequence of valid words. The running time should be at most $O(n^2)$, assuming calls to dict take unit time. (3 for explanation and 7 for C++/pseudo code covering all cases)

4. [5+5] Prove the following:

   (a) If $d = gcd(a, b)$, then there exist integers x and y such that

   $$d = ax + by$$

   (b) If d divides N, then

   $$N = \sum_{d \text{ divides } N} \phi(d)$$

   where ϕ is Euler Phi function.

5. [10] Consider the problem of determining whether a given undirected graph $G = (V; E)$ contains a triangle or cycle of length 3. You can do it by choosing 3 vertices and check whether they form a triangle or not for every possible combination. Can you do better? Yes.

   (a) Using DFS may be tempting, but will it work? (Assume nodes are assigned distinct integers in the graph and DFS always choose that neighbour first which have least integer associated with it)

   (b) If yes, provide an algo using DFS to find a triangle in graph G, and argue why it will find the triangle. If not, produce a counterexample.

6. [20] You are given a connected weighted graph length that stores the road length between E pairs of cities i and j ($1 \leq V \leq 1000$, $0 \leq E \leq 10000$), the price p_i of fuel at each city i, and the fuel tank capacity c of a car ($1 \leq c \leq 100$). Assume that distance between cities is always an integer.

   Using Dijkstra's algorithm, determine the cheapest trip cost from starting city s to ending city e using a car with fuel capacity c. All cars use one unit of fuel per unit of distance and start with an empty fuel tank. (Hint: don't think of applying dijkstra to cities as nodes. You can add/remove any number of nodes or make a new graph as long as number of nodes in your graph is less than or equal to 100000 vertices).