Assignment 6

csci2200, Algorithms

Instructions:

- Honor code: Work on this assignment with at most one partner. Between different teams, Collaboration is at level 1 [verbal collaboration only]
- Write each problem on a separate page; If a problem has multiple parts, you can write all parts on the same page, as long as you leave space in between them.

---

1. **String shuffling:** A *shuffle* of two strings A and B is formed by interspersing the characters into a new string, keeping the characters from A and B in the same order.

   For example, the string BANANAANANAS is a shuffle of the string BANANA and ANANAS (in several different ways, actually: BANANAANANAS, BANANAANANAS and also BANANAANANAS). Similarly, the strings ANEVEGARIN and ANEGAVERIN are both shuffles of NEVER and AGAIN.

   The problem: Given three strings $A[1..m]$, $B[1..n]$ and $C[1..m+n]$, come up with an efficient algorithm to determine whether C is a shuffle of A and B.

   (a) Define your subproblem. Clearly state what function you will compute, what value it should return, what the arguments represent.

   (b) Argue optimal substructure and give a recursive definition of the subproblem.

   (c) Imagine you write a function to compute the subproblem using the formula above (without dynamic programming). Briefly argue what the running time would be.

   (d) Give pseudocode for a recursive, dynamic programming approach and analyze its running time.

(e) **Optional/extra credit:** Translate your efficient dynamic programming algorithm above into either Java or Python code. Your code should be able to take 3 arguments (if Java: preferably on the command line), which represent the three strings, and report whether the third string is a shuffle of the first two. Test your code on a couple of sequences and include a screenshot. Specifically test it on `shuffle (aa, ba, aaba)` ; `shuffle (ba, aa, aaba)` and `shuffle (a, ba, aab)`.

2. **Party problem:** Suppose you are in charge of planning a party for an institution (for e.g., Bowdoin College). The institution has a hierarchical structure, which forms a tree rooted at the President. Linked directly under the president are the people who report directly to the president, and thir "children" are the people who report directly to them, and so on. Part of this hierarchy are the faculty, grouped by department; all faculty in the same department have the department chair as their parent. Each faculty has as children all the student advisees. Assume that every person is listed at the highest possible position in the tree and there are no double affiliations: everybody has one and only one supervisor in this hierarchy and every student has only one advisor.

   You have access to a secret database which ranks each faculty/staff/student with a conviviality rating (a real number, which can be negative if the person is really grumpy or boring). As the party organizer, your goal is to maximize the fun, so you decide to select the guest list such that: for any guest, his/her immediate supervisor is not also a guest. Note that this means that the President may not be invited, but it is what it is. You're doing your job.

   You are given a tree that describes the strucure of the institution. Each node has a (down) pointer to its left-most child, and a (right) pointer to its next sibling Each node also holds a name and a conviviality ranking. Describe an algorithm to make up a guest list that maximizes the sum of the conviviality rankings of the guests. Analyze the running time of your algorithm. Assume the size of the tree (number of nodes) is n.

   Hint: Aim for $O(n)$ time.

   [*We expect: Explanation of your choice of subproblem and its parameters (i.e. what function you will compute, what it returns and what are its parameters); justification of optimal substructure; pseudocode and an English description of your algorithm, as well as analysis of its running time.*]