Comp 251: Practice problems

Instructor: Jérôme Waldispühl

This is a collection of problems that you can use to prepare the COMP 251 final exam. This selection of problems covers the second part of the class (i.e. after the mid-term exam). However, the final exam will also cover the material of the first part of the class. Please, refer to the practice problems for the mid-term, and solution of the midterm (last lecture) to fully prepare the exam.

Dynamic programming

1. The Coin Row Problem: Suppose you have a row of coins with values that are positive integers $c_1, \cdots, c_n$. These values might not be distinct. Your task is to pick up coins have as much total value as possible, subject to the constraint that you don't ever pick up two coins that lie beside each other. How would you solve this using dynamic programming?
   Solve the problem for coins with values c_1 to c_6 as follows: $(5, 1, 2, 10, 6, 2)$.

2. The Coin Change Problem: Suppose we have m types of coins with values $c_1 < c_2 < \cdots c_m$ (e.g. in the case of pennies, nickels, dimes, ... we would have $c_1 = 1$, $c_2 = 5$, $c_3 = 10$, $\cdots$). Let $f(n)$ be the minimum number of coins whose values add up to exactly n. Write a recurrence for $f(n)$ in terms of the values of the coins. You may use as many of each type of coin as you wish.
   As an example, suppose the coin values c_1, c_2, and c_3 are 1, 3, 4. Solve the problem for $n = 6$ using dynamic programming.

3. What is the optimal substructure of the Neddleman-Wunch algorithm (i.e. optimal pairwise sequence alignment)?

Divide-and-Conquer

4. In Karatsuba multiplication, when you do the multiplication $(x_1+x_0)\cdot(y_1+y_0)$, the two values you are multiplying might be $n/2 + 1$ digits each, rather than $n/2$ digits, since the addition might have led to a carry e.g. $53 + 52 = 105$. Does this create a problem for the argument that the recurrence is $t(n) = 3t(n/2) + c_n$?

5. Apply the master method to determine the asymptotic behavior of the function $T(n)$.

   1. $T(n) = 2 \cdot T(n/4) + n^{0.51}$

2. $T(n) = 0.5 \cdot T(n/2) + 1/n$
3. $T(n) = 64 \cdot T(n/8) - n^2 \cdot logn$
4. $T(n) = \sqrt{2} \cdot T(n/2) + \log n$
5. $T(n) = 6 \cdot T(n/3) + n^2 \cdot \log n$
6. $T(n) = 3 \cdot T(n/3) + n/2$

6. Write a recurrence that describes its worst-case running time of the quicksort algorithm.

Amortized analysis

7. Suppose we perform a sequence of stack operations on a stack whose size never exceeds k. After every k operations, we make a copy of the entire stack for backup purposes. Show that the cost of n stack operations, including copying the stack, is $O(n)$ by assigning suitable amortized costs to the various stack operations.

8. Suppose we perform a sequence of n operations on a data structure in which the i^{th} operation costs i if i is an exact power of 2, and 1 otherwise. Use aggregate analysis or accounting method to determine the amortized cost per operation.