Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer

26. November 2018

Algorithms & Data Structures — Homework 10 — HS 18

Exercise Class (Room & TA): _____

Submitted by: _____

Peer Feedback by: _____

Points: ____

Hint: This exercise sheet is concerned with *dynamic programming*. A complete description of a dynamic program always includes the following aspects (important also for the exam!):

1. **Definition of the DP table:** What are the dimensions of the dynamic programming table DP[., .]? What is the meaning of each entry (in clearly worded words)?
2. **Calculation of an entry:** Which values of the table are initialized, and how are they initialized? How are entries calculated from other entries? What are the dependencies between entries?
3. **Calculation order:** In what order can you calculate the entries so that these dependencies are fulfilled?
4. **Reading the solution:** How can the solution be read out from the table at the end?

Exercise 10.1 *Number of solutions for Subset Sum.*

Given a sequence of positive integers $A[1], \ldots, A[n]$ and an integer S, your problem is to find the number of subsets of $A[1], \ldots, A[n]$ with sum S. Consider the following dynamic programming algorithm:

1. **Definition of the DP table:** $T[., .]$ is an $(n+1) \times (S+1)$ table, $T[i, s]$ is a number of subsets of $A[1], \ldots, A[i]$ with sum s.
2. **Calculation of an entry:**

$$T[i,s] = \begin{cases} 1, & \text{if } s = 0; \\ 0, & \text{if } s > 0 \text{ and } i = 0; \\ T[i-1,s], & \text{if } i > 0 \text{ and } 0 < s < A[i]; \\ T[i-1,s] + T[i-1, s-A[i]], & \text{if } i > 0 \text{ and } s \geq A[i]. \end{cases}$$

3. **Calculation order:** We can calculate the entries of T row-by-row from top to bottom, and then within each row from left to right.
4. **Reading the solution:** $T[n, S]$ is a solution.

Given are a sequence $A = [6, 1, 2, 7, 3, 8, 5, 8]$ and a desired sum $S = 11$. Use the dynamic programming algorithm as described above to find the number of subsets of A with sum S. Show the DP table.

Exercise 10.2 *Matrix chain multiplication.*

Given is a sequence of matrices: A_1 of size 6×4, A_2 of size 4×5, A_3 of size 5×3, A_4 of size 3×8, A_5 of size 8×2 and A_6 of size 2×7.

Use the dynamic programming algorithm as described in class to find the min cost of multiplication $A_1 \cdot \ldots \cdot A_6$. Show the DP table. For this exercise assume that the cost of multiplying a $k_1 \times k_2$ matrix A and a $k_2 \times k_3$ matrix B is $k_1 \cdot k_2 \cdot k_3$.

Exercise 10.3 *Greedy algorithm for Subset Sum.*

Consider the following variant of the Subset Sum problem:
given a sequence of positive integers $A[1], \ldots, A[n]$, assign each index i, $1 \leq i \leq n$, to one of two parts, S and T, such that $S \cap T = \emptyset$, $S \cup T = \{1, 2 \ldots, n\}$ and $\sum_{i \in S} A[i] = \sum_{i \in T} A[i]$. For simplicity, assume that $A[1], \ldots, A[n]$ is *perfectly balanced*, that is, such a partition in two parts with equal sum is possible. For example, a sequence $2, 1, 1$ is perfectly balanced, but the sequence $2, 1, 2$ is not.

Consider the following simple greedy algorithm: go through the numbers one by one and assign each index to the part with the smallest current sum. That is, if the current index is i and we have a partition of $1, \ldots, i-1$ in two parts S and T such that $\sum_{j \in S} A[j] \leq \sum_{j \in T} A[j]$, we assign i to S.

1. Show that this algorithm doesn't solve the problem: provide a perfectly balanced sequence of integers A such that this greedy algorithm splits indices in two parts S and T such that $S \cap T = \emptyset$, $S \cup T = \{1, 2 \ldots, n\}$ and $\sum_{i \in S} A[i] < \sum_{i \in T} A[i]$.

Let's define a function Greedy-Imbalance(A) that takes as input a perfectly balanced sequence of positive integers $A[1], \ldots, A[n]$ and outputs the imbalance ratio that greedy achieves for that input, that is

$$\text{Greedy-Imbalance}(A) = \frac{\sum_{i \in T} A[i]}{\sum_{i \in T'} A[i]},$$

where T is the larger part computed by the greedy algorithm and T' is an actual solution (that is, $\sum_{i \in T'} A[i] = \frac{1}{2} \sum_{i=1}^{n} A[i]$).

2. For all n provide a perfectly balanced sequence of positive integers $A[1], \ldots, A[n]$ such that Greedy-Imbalance$(A) = \frac{3}{2}$
3. Show that for any perfectly balanced sequence of positive integers A, Greedy-Imbalance$(A) \leq \frac{3}{2}$.

Exercise 10.4 *Game with coins **(1 Point)**.*

Consider the following game: given is a sequence of coins, such that each coin has a positive integer value and the initial number of coins n is divisible by 3. On each turn the player may either take (and keep) the coin from the left or the right end of the sequence, but then two coins disappear from the other end. The disappeared coins are lost to the player. The goal of the player is to maximize the value of the coins taken.

For example, if the sequence of coins is $20, 20, 500, 20, 20, 20, 20, 100, 100, 100, 5, 5, 5, 5, 5, 5, 1, 1, 1, 1, 1$, the maximal value is 803: first we take the right coin, then the left one, then the right twice, and finally we take the left one three times.

1. Describe a dynamic programming algorithm which calculates the maximum value achievable for the player, given a sequence of coins $C[0], \ldots, C[n-1]$. Describe your dynamic program and specify the running time of the algorithm.

2. Assume that you need to report the order of coins that the player should pick to maximize the value of the coins taken. Describe your backtracking procedure and specify the running time of the backtracking. Use letters R and L to output the order, e.g. for the sequence given above the order is R, L, R, R, L, L, L.

Exercise 10.5 *Sub-numbers on a Mechanical Computer* ***(2 Points)****.*

You are given a number of n non-zero digits and an array L of breaking points. Your task is to break the number into several *sub-numbers* i.e. break the sequence of digits into several continuous subsequences. Instead of using a modern computer, you are given a mechanical computer that does not perform any numerical operations, but is capable of splitting numbers into sub-numbers. When you break a number of n digits into two sub-numbers, the mechanical computer uses n units of time to execute the procedure (as copy of the initial number must be performed).

Obviously the order of selecting breaking points to split the number into sub-numbers will affect the overall processing time on the mechanical computer. Consider the number 123456789, and break points $L = \{1, 2, 5\}$. Splitting the number by following the break points from left to right will cost you:

1. 9 units for the first break: $\{1\}$, $\{23456789\}$
2. 8 units for the second break: $\{1\}$, $\{2\}$ $\{3456789\}$, and
3. 7 units for the last break: $\{1\}$, $\{2\}$, $\{345\}$, $\{6789\}$

In total 24 units of time. If you execute the splits following the breaking points from right to left, it will take you:

1. 9 units for the first break: $\{12345\}$, $\{6789\}$
2. 5 units for the second break: $\{12\}$, $\{345\}$, $\{6789\}$, and
3. 2 for the last break: $\{1\}$, $\{2\}$, $\{345\}$, $\{6789\}$

In total 16 units of time.

You are allowed to attach a modern computer to the mechanical computer, such that will compute the order of executing the breaks such that you can minimize the overall time to break the number into sub-numbers. Use dynamic programming, and determine the minimal break cost and the order of the breaks in $O(n^3)$ time complexity and $O(n^2)$ space complexity.

Note that in the mechanical computer the first digit of the input number has index of 1 and last digit has index of n. Any break point l, indicates that break should happen between index l and $l + 1$ of the the input number and $0 < l < n$ for all $l \in L$. Also note that all elements of L are unique.

Submission: On Monday, 3.12.2018, hand in your solution to your TA *before* the exercise class starts.