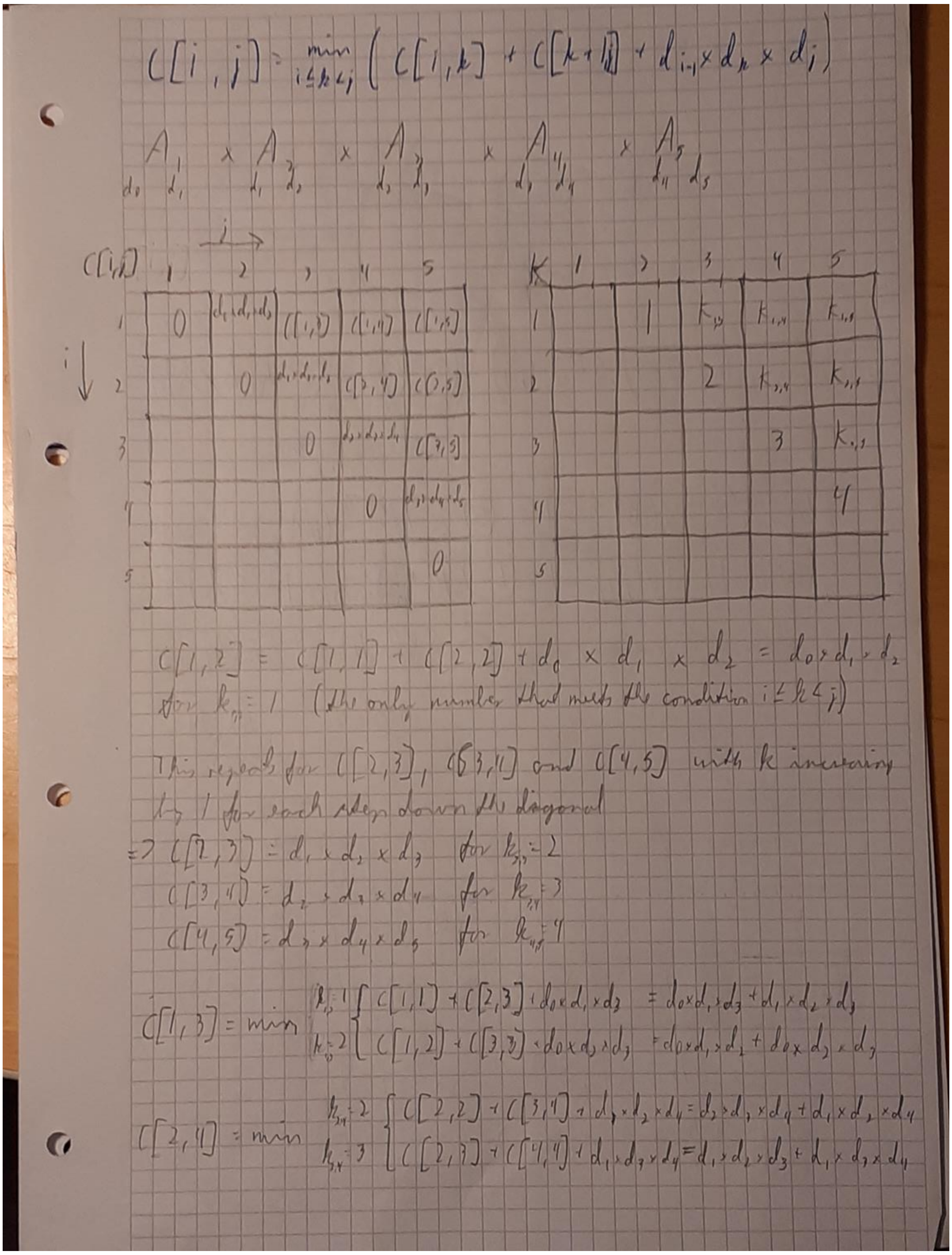
Assignment 2

By: Erlend Kulander Kvitrud

Problem I: Problem Matrix Chain Multiplication

Part 1: I solved this problem using the notation and approach used in the online course Dynamic Programming (DP) by Abdul Bari[1], which differs slightly from the one used by the textbook and lecture notes. Sorry about the poor resolution; I didn't have access to a scanner and my phone camera is not the best.



$$C[i,j] = \min_{i \le k < j} \left(C[1,k] + C[k+1,j] + d_{i-1} \times d_k \times d_j \right)$$

$$\underset{d_0 \; d_1}{A_1} \times \underset{d_1 \; d_2}{A_2} \times \underset{d_2 \; d_3}{A_3} \times \underset{d_3 \; d_4}{A_4} \times \underset{d_4 \; d_5}{A_5}$$

C[i,j]	1	2	3	4	5
1	0	$d_0 \times d_1 \times d_2$	C[1,3]	C[1,4]	C[1,5]
2		0	$d_1 \times d_2 \times d_3$	C[2,4]	C[2,5]
3			0	$d_2 \times d_3 \times d_4$	C[3,5]
4				0	$d_3 \times d_4 \times d_5$
5					0

K	1	2	3	4	5
1		1	$k_{1,3}$	$k_{1,4}$	$k_{1,5}$
2			2	$k_{2,4}$	$k_{2,5}$
3				3	$k_{3,5}$
4					4
5					

$C[1,2] = C[1,1] + C[2,2] + d_0 \times d_1 \times d_2 = d_0 \times d_1 \times d_2$

for $k_{1,2} = 1$ (the only number that meets the condition $i \le k < j$)

This repeats for $C[2,3]$, $C[3,4]$ and $C[4,5]$ with k increasing by 1 for each step down the diagonal

$\Rightarrow C[2,3] = d_1 \times d_2 \times d_3$ for $k_{2,3} = 2$

$C[3,4] = d_2 \times d_3 \times d_4$ for $k_{3,4} = 3$

$C[4,5] = d_3 \times d_4 \times d_5$ for $k_{4,5} = 4$

$$C[1,3] = \min \begin{cases} k_{1,3}=1: & C[1,1] + C[2,3] + d_0 \times d_1 \times d_3 = d_0 \times d_1 \times d_3 + d_1 \times d_2 \times d_3 \\ k_{1,3}=2: & C[1,2] + C[3,3] + d_0 \times d_2 \times d_3 = d_0 \times d_1 \times d_2 + d_0 \times d_2 \times d_3 \end{cases}$$

$$C[2,4] = \min \begin{cases} k_{2,4}=2: & C[2,2] + C[3,4] + d_1 \times d_2 \times d_4 = d_2 \times d_3 \times d_4 + d_1 \times d_2 \times d_4 \\ k_{2,4}=3: & C[2,3] + C[4,4] + d_1 \times d_3 \times d_4 = d_1 \times d_2 \times d_3 + d_1 \times d_3 \times d_4 \end{cases}$$

---

[1] https://www.youtube.com/watch?v=_WncuhSJZyA&list=PLJULIlvhz0rE83NKhnq7acXYIeA0o1dXb&index=13

$$C[3,5] = \min \begin{cases} k_{3,5}=3 & C[3,3] + C[4,5] + d_2 \times d_3 \times d_5 = d_3 \times d_4 \times d_5 + d_2 \times d_3 \times d_5 \\ k_{3,5}=4 & C[3,4] + C[5,5] + d_2 \times d_4 \times d_5 = d_2 \times d_3 \times d_4 + d_2 \times d_4 \times d_5 \end{cases}$$

$$C[1,4] = \min \begin{cases} k_{1,4}=1 & C[1,1] + C[2,4] + d_0 \times d_1 \times d_4 = C[2,4] + d_0 \times d_1 \times d_4 \\ k_{1,4}=2 & C[1,2] + C[3,4] + d_0 \times d_2 \times d_4 = d_0 \times d_1 \times d_2 + d_2 \times d_3 \times d_4 + d_0 \times d_2 \times d_4 \\ k_{1,4}=3 & C[1,3] + C[4,4] + d_0 \times d_3 \times d_4 = C[1,3] + d_0 \times d_3 \times d_4 \end{cases}$$

$$= \min \begin{cases} \min[(d_2 \times d_3 \times d_4 + d_1 \times d_2 \times d_4), (d_1 \times d_2 \times d_3 + d_1 \times d_3 \times d_4)] + d_0 \times d_1 \times d_4 \\ d_0 \times d_1 \times d_2 + d_2 \times d_3 \times d_4 + d_0 \times d_2 \times d_3 \\ \min[(d_0 \times d_1 \times d_3 + d_1 \times d_2 \times d_3), (d_0 \times d_1 \times d_2 + d_0 \times d_2 \times d_3)] + d_0 \times d_3 \times d_4 \end{cases}$$

$$C[2,5] = \min \begin{cases} k_{2,5}=2 & C[2,2] + C[3,5] + d_1 \times d_2 \times d_5 \\ k_{2,5}=3 & C[2,3] + C[4,5] + d_1 \times d_3 \times d_5 \\ k_{2,5}=4 & C[2,4] + C[5,5] + d_1 \times d_4 \times d_5 \end{cases}$$

For the remainder, I will simply write the solution as a recursive function of the costs that have already been calculated in the third or fourth diagonal of the cost matrix

$$C[1,5] = \min \begin{cases} k_{1,5}=1 & C[1,1] + C[2,5] + d_0 \times d_1 \times d_5 \\ k_{1,5}=2 & C[1,2] + C[3,5] + d_0 \times d_2 \times d_5 \\ k_{1,5}=3 & C[1,3] + C[4,5] + d_0 \times d_3 \times d_5 \\ k_{1,5}=4 & C[1,4] + C[5,5] + d_0 \times d_4 \times d_5 \end{cases}$$

$$= \min \begin{cases} C[2,5] + d_0 \times d_1 \times d_5 \\ C[3,5] + d_0 \times d_1 \times d_2 + d_0 \times d_2 \times d_5 \\ C[1,3] + d_3 \times d_4 \times d_5 + d_0 \times d_3 \times d_5 \\ C[1,4] + d_0 \times d_4 \times d_5 \end{cases}$$

Part 2: The source code for my Matrix-chain multiplication Calculator is available at: https://github.com/ErlendKK/DAT600-Assignment2/blob/main/Matrix-chain-Multiplication-Calculator

A preview is available at https://htmlpreview.github.io/?https://raw.githubusercontent.com/ErlendKK/DAT600-Assignment2/main/Matrix-chain-Multiplication-Calculator

Part 3: The parenthesization problem cannot be solved by a greedy algorithm, since there is no guarantee that the globally optimal solution will emerge from greedily selecting locally optimal ones. Consider the very simple example:

Matrix A: 3x1
Matrix B: 1x2
Matrix C: 2x4

=> p = [3, 1, 2, 4]

The first step of the calculation is (for the two possible parenthesizations):

(AB)C: c[1, 2] + 0 = 3*1*2 = 6
A(BC): 0 + c[2, 3] = 1*2*4 = 8

=> (AB)C is the most locally optimal parenthesization, and thus the greedy choice.

Yet at the next step, the total cost is the sum of the cost of the first step and the cost of multiplying the two resulting matrices:

(AB)C: c[1, 3] = c[1, 2] + 3*2*4 = 6 + 24 = 30
A(BC): c[1, 3] = c[2, 3] + 3*1*4 = 8 + 12 = 20

Here we see that the globally optimal parenthesization is A(BC), even though this was locally non-optimal at the first step. A greedy algorithm would have yielded the non-optimal solution (AB)C.

Problem II: Fractional and 0-1 Knapsack

The source code is available at https://github.com/ErlendKK/DAT600-Assignment2/blob/main/knapsack_problems

Problem III: Problem Greedy + Dynamic (coin change)

1 - 3) Assumptions: the array is presorted in ascending order; question 2 is not really a question, but an intro to question 3; the optimal solution to question 3 might be an algorithm that checks whether the coin-set has a greedy solution, then solves it greedily if possible, or with DP if not. Yet since question 5 asks us to compare the runtime of theses algorithms, I assume that this is not what we are supposed to do.

4) The Norwegian coin system seems greedy. It seems like the fact that any larger coin corresponds

to an exact number of smaller coins (i.e. no cases ala selecting '11' as your first coin being globally suboptimal for the target '15'), means that it must be greedy, but I am unsure how to formally prove that this is so. Another indicator is that the two algorithms give the same results for the Norwegain coinset, for any target value I have tried.

5)
I ran each algorithm 1000 times for different target values, averaging their run times for each target, and got the following results. As we would expect, the greedy algorithm is constant with regards to target, while the DP one is linear.