University of Zurich[UZH]

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatik II
Assignment 6

May 7, 2018

Dynamic Programming [45 points]

The Knapsack Problem

Task 1. [30 points] Assume a knapsack with `capacity B` and a `set S of n items` are available. Each item i has a given `mass` m_i ($m_i \geq 0$) and a `profit` p_i ($p_i \geq 0$). Choosing A items ($A \leq n$) to be put inside the knapsack, their total profit P and total mass M can be computed. Since the knapsack has a limited capacity, the items chosen should fit within it, that is $M \leq B$. The goal is to compute the `maximum profit P` gained through a proper choice of items to be included.

Example: Given a knapsack of capacity 5, and items with mass = $\{(0,2),(1,4),(2,3),(3,2)\}$ and profit = $\{(0,45),(1,40),(2,25),(3,15)\}$, where in every pair `(a,b)` `a` is the `item number` and `b` is `mass` and `profit` respectively. The best combination would be item 0 (with mass 2 and profit 45) and item 2 (with mass 3 and profit 25) for a total profit of 70. No other combination with mass 5 or less has a greater profit.

Solve the knapsack problem using (a) recursion, (b) memoization and (c) dynamic programming. Create a C program including the following:

- A global matrix *m[50][501]* used in order to store the intermediate results when memoization or dynamic programming is used.
- *int knapsackRecursive(int capacity, int mass[], int profit[], int n)* which computes the maximum total profit P recursively.
- *int knapsackMemoized(int capacity, int mass[], int profit[], int n)* which computes the maximum total profit P using memoization.
- *int knapsackDynamic(int capacity, int mass[], int profit[], int n)* which computes the maximum total profit P using dynamic programming.

Print the results and report the `execution time` of the following problem sets to test the functions created:

Cap.	mass	profit
10	6, 3, 2, 4	50, 60, 40, 20
165	23, 31, 29, 44, 53, 38, 63, 85, 89, 82	92, 57, 49, 68, 60, 43, 67, 84, 87, 72
500	43, 12, ... 42, 91 (`50 items`) randomly generated with range [1 - 100]	2, 86, ... 65, 39 (`50 items`) randomly generated with range [1 - 100]

Task 2. **[15 points]** You are a commodity trader and want to perform a historical analysis to determine how close your actual performance was to the maximum possible profit. In commodity trading, a buyer buys a certain commodity and sells on a future date, the trader is allowed to make at most `k` transactions (one transaction consists of one buy and one sell order). Given the price of a certain commodity (e.g. crude oil) for `n` days in the form of an array `A[0..n-1]` containing integer values and the number `k` of your transactions calculate the maximum possible profit you could have made during this period executing at most `k` transactions.

Example: Given an array of daily prices `A={12, 14, 17, 10, 14, 13, 12, 15}` of a commodity, and the number of maximum transactions `k=3`, the maximum profit is `12`. The maximum profit `12` can be earned by executing the following transactions:

- buy on day `0`, sell on day `2`, `profit=5`.
- buy on day `3`, sell on day `4`, `profit=4`.
- buy on day `6`, sell on day `7`, `profit=3`.

Total profit is `5+4+3=12` with `3` transactions executed.
Note: You can only execute one transaction at a time, no overlapping transactions are allowed.
In any case the maximum profit returned should **not** be `negative`.
Solve the maximum profit problem using dynamic programming. Create a C program that includes the following function:

- *int maxProfit(int price[], int n, int k)*, that calculates and returns the maximum total profit using dynamic programming.

Graphs [15 points]

Task 3. [**15 points**] You are given a `binary` 2D matrix `M`, each element of the matrix represents an island, islands with value `0` are uninhabited and islands with value `1` are inhabited. Each island represented by a cell in the matrix can have up to 8 neighbors (north, east, west, south and 4x diagonally). A group of neighboring inhabited islands form an ecosystem. The goal is to find the number of distinct ecosystems in the archipelago of islands represented by the 2D matrix `M`.

Example:
Given the following matrix of `islands` in an archipelago, `M`:

1	1	0	0	1
0	1	0	0	0
1	0	0	1	1
0	0	0	0	0
1	0	1	0	1

the number of distinct `ecosystems` is `6`. Assuming that the top left cell is (0,0) and the bottom right is (4,4), the distinct `ecosystems` in matrix `M` are:

- `A={(0,0),(0,1),(1,1),(2,0)}`
- `B={(0,4)}`
- `C={(2,3),(2,4)}`
- `D={(4,0)}`
- `E={(4,2)}`
- `F={(4,4)}`

Solve the `ecosystem` discovery problem using `graphs`. Create a C program that includes the following function:

- *int countEcosystems()*, that calculates and returns the number of distinct ecosystems.

Submission

For this exercise, you need to submit a zipped folder *a<exercise number>_<family name>_<matriculation number>.zip* where `family name` and `matriculation number` correspond to your personal data. This folder should include the C-files you created for each of the tasks. Each C-file should be named as *task<task number>.c* and it should also include you personal data in the form of a comment on the top.

Deadline: **Sunday, May 20$^{\text{th}}$ at 23:59**.