Fondamenti di Programmazione 2

2 Dicembre 2022

Exercise 1

Write a program that takes in input an integer n, an integer $0 < k < n$, a sequence of n integers $x_1, ..., x_n$ and computes a subsequence of k elements with the maximum sum.

The program should output the optimal subsequence's elements, their sum, and their index in the original sequence.

Example Consider the sequence $9, 13, -50, 3, 2, 9$ e $k = 3$. In this case, the optimal subsequences is composed by the elements $9, 13, 9$ that appear at indexes $0, 1, 5$ and sum up to 31.

Exercise 2

Write a program that reads an array A of n integers, and computes the optimal indexes $0 \leq i \leq j < n$ such that the quantity $S[i : j] = \sum_{k=i}^{k=j} A[k]$ is maximized.

Example Consider the array $9, 13, -5, 3, 2, 9$. In this case, the optimal choice would be $i = 0, j = 1$, as $S[0 : 1] = A[0] + A[1] = 9 + 13 = 22$.

Consider the array $9, 60, -50, 60, -5, -10$. In this case, the optimal choice would be $i = 0, j = 3$, as $S[0 : 3] = A[0] + A[1] + A[2] + A[3] = 9 + 60 - 50 + 60 = 79$.

Exercise 3

We say a graph is k-colorable if it is possible to assign a color, among $c_1, ..., c_k$, to each one of its nodes so that there does not exist an edge of the graph whose endpoints are colored with the same color. That is, if (u, v) is an edge of the graph, it is not possible to color u, v with the same color.

Write a program that takes in input an undirected graph and returns a D-coloring for the graph, where D is the maximum degree of the graph's nodes.

This coloring always exists for all the graphs, and can be computed with a greedy strategy - do not confuse this with the problem of k-coloring a graph - where k does not depend on the input graph - that is a much harder problem from the computational standpoint.

Exercise 4

Let M be a matrix of $n \times m$ positive integers. Each path that joins the $(0, 0)$ to $(n-1, m-1)$ has an associated cost, that is computed as the sum of the integers on the cells that belong to the path. We want to reach the bottom right cell, $(n-1, m-1)$ starting from the top left cell, $(0, 0)$, using the path with minimum cost.

Write a program that takes as input a matrix M and produces the optimal path.

Example Consideriamo the 4×3 matrix:

```
10 30 90
50 30 90
10 90 90
10 10 10
```

One of the possible paths to reach $(3, 2)$ from $(0, 0)$ is →→↓↓↓, and its cost is $10 + 30 + 90 + 90 + 90 + 10 = 320$.

The optimal choice would be ↓↓↓→→, with a cost of $10+50+10+10+10+10 = 80$.

Exercise 5

Consider an array of n integers A. From a generic position i, we can jump to any of the positions $i + 1, ..., i + A[i]$.

Write a function that computes which positions we need to jump on, starting from 0, to reach the position $n - 1$ with the least amount of jumps.

Example Consider the array $3, 1, 1, 2, 0, 0$. Starting from 0, we can jumps in the positions $1, 2, 3$, as $A[0] = 3$:

To reach the last position, we can do the jumps `0 -> 3` e `3 -> 5`, and we reach our goal in 2 jumps. Alternatively, in a non-optimal way, we can do the jumps `0 -> 1 -> 2 -> 3 -> 5`.

```
Optimal solution:
[3]  1   1   2   0   0
 3   1   1  [2]  0   0
 3   1   1   2   0  [0]

One non-optimal solution:
```

```
[3]  1   1   2   0   0
 3  [1]  1   2   0   0
 3   1  [1]  2   0   0
 3   1   1  [2]  0   0
 3   1   1   2   0  [0]
```

Exercise 6

Write a function that returns the longest *palindromic* subsequence of an input string. We say a string is palindromic when it is the same as its inverse, such as in the case of `OTTO`.

Example Consider the word `SCACCIAPENSIERI`. `CCC` is a palindromic subsequence, `CAPRI` is a subsequence, but is not palindromic. `SACCAS`, `IENEI` are examples of palindromic subsequences, in this case they are also (from) the longest ones in the input string.