Problem set 1

Federico Chinello

January 10, 2024

I assume array and matrix indices start from 1. When I do slicing, I assume both the extremes are selected (i.e., `A[i:j]` comprises both `A[i]` and `A[j]`).

Problem 4

Given a sequence of n integers $a_1, a_2, ..., a_n$, find its longest subsequence that is strictly increasing. Running time: $O(n^2)$.

The input is a sequence `S` of integers of length `n`.

```
\\ dp[i][j] is the length of the increasing
\\ subsequence of S[i:j] having S[i] as first element
dp[][] := n x n matrix

for i = 1,...,n:
    last = S[i]
    for j = i+1,...,n:
        if S[j] > last:
            \\ if S[j] is larger than the last
            \\ item selected as part of the
            \\ increasing subsequence starting with S[i]
            dp[i][j] = dp[i][j-1] + 1
            last = S[j] \\ update last
        else:
            dp[i][j] = dp[i][j-1]

// In the last column of dp, we find
// the lengths of the increasing
// subsequences starting with each item of S
// The general LIS is the longest among them
max = dp[1][n]
max_i = 1
for i = 2,...,n:
    if dp[i][n] > max:
        max = dp[i][n]
        max_i = i

solution[] := list
i = max_i
solution.append(S[i])
for j = i+1,...,n:
    if dp[i][j] != dp[i][j-1]:
        solution.append(S[j])
```