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 3

Given a sequence of length n, find its longest subsequence (of not necessarily consecutive elements) that is a palindrome. Running time: $O(n^2)$.

The input is a string `S`.

```
// dp[i][j] is the length of the longest
// palindromic subsequence of S[i:j]
dp[][] := n x n matrix

// Base case
// A single letter is a palindrome
for i = 1,...,n:
    dp[i][i] = 1

// Recursive case
for i = n-1,...,0:
    for j = i+1,...,n:
        if (S[i] = S[j]) & (j = i+1):
            // Two consecutive equal characters
            // are palindromic (e.g. AA)
            dp[i][j] = 2
        elif S[i] = S[j]:
            // If the extremes of S[i:j] are equal
            dp[i][j] = dp[i + 1][j - 1] + 2
        else:
            // If the extremes of S[i:j] are not equal
            // the recurrence is:
            dp[i][j] = max{dp[i+1][j], dp[i][j-1]}
```

```
// Lq is the length of the max subsequence
// of S that is a palindrome
L = dp[1][n]
// We initialize an array to store the solution
subseq[] := array of length L

i = 1
j = n
k = 1
while i <= j:
    if S[i] = S[j]:
        subseq[k] = subseq[L-k+1] = S[i]
        i++
        j--
        k++
    elif dp[i][j] = dp[i+1][j]:
        i++
    elif dp[i][j] = dp[i][j-1]:
        j--
```

The solution is stored in the array `subseq`.