CS590 homework 4 – Dynamic Programming, Greedy Algorithms

Report by: Sanjeet Vinod Jain

Q4. Find the maximum alignment for X = dcdcbacbbb and Y = acdccabdbb by  
using the Smith-Waterman algorithm(see slides). Execute the pseudocode algorithm and fill the necessary tables H and P in a bottom-up fashion. Reconstruct the strings X‘ and Y‘ using the tables H and P.

**H Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | D | C | D | C | B | A | C | B | B | B |
|  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| A | 0 | -1 | -1 | 2 | 1 | 0 | -1 | -1 | 2 | 1 | 0 |
| C | 0 | -1 | 1 | 1 | 4 | 3 | 2 | 1 | 1 | 1 | 0 |
| D | 0 | -1 | 0 | 3 | 3 | 3 | 2 | 1 | 3 | 2 | 1 |
| C | 0 | -1 | 1 | 2 | 5 | 5 | 4 | 3 | 2 | 2 | 1 |
| C | 0 | -1 | 0 | 1 | 4 | 4 | 4 | 6 | 5 | 4 | 4 |
| A | 0 | 2 | 1 | 0 | 3 | 3 | 6 | 5 | 5 | 4 | 3 |
| B | 0 | 1 | 4 | 3 | 2 | 5 | 5 | 5 | 4 | 4 | 3 |
| D | 0 | 0 | 3 | 3 | 2 | 4 | 4 | 7 | 6 | 6 | 6 |
| B | 0 | -1 | 2 | 2 | 2 | 3 | 3 | 6 | 6 | 8 | 8 |
| B | 0 | -1 | 1 | 1 | 1 | 2 | 2 | 5 | 5 | 8 | 10 |

**P Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | D | C | D | C | B | A | C | B | B | B |
|  | - | - | - | - | - | - | - | - | - | - | - |
| A | - | d | d | d | l | L | D | d | d | l | l |
| C | - | d | d | u | d | D | L | l | u | d | d |
| D | - | d | u | d | u | D | D | d | d | l | l |
| C | - | d | d | u | d | D | L | l | u | d | d |
| C | - | d | u | u | u | D | D | D | l | d | d |
| A | - | d | l | u | u | D | D | u | d | d | d |
| B | - | u | d | l | d | D | U | d | d | d | d |
| D | - | u | u | d | d | U | D | d | l | d | d |
| B | - | d | u | d | D | U | D | d | d | d | d |
| B | - | d | u | d | D | U | D | d | d | d | d |

Exercise 15.1-2:

Show, by means of a counter example, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the density of a rod of length i to be pi/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where 1 <= i <= n, having maximum density. It then continues by applying the greedy strategy to the remaining piece of length n – i.

Example :

Let the rod length required be 4

And we have some assumed prices of each section of length as per the table below

Hence we get

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Length (i) | 1 | 2 | 3 | 4 |
| Price (Pi) | 5 | 28 | 60 | 52 |
| Density (Pi/i) | 0.2 | 14 | 20 | 13 |

Now by greedy strategy we will sort all these sections by its density

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Length (i) | 3 | 2 | 4 | 1 |
| Price (Pi) | 60 | 28 | 52 | 5 |
| Density (Pi/i) | 20 | 14 | 13 | 0.2 |

Due to this the greedy algorithm tells us to cut a rod of length 3 at the cost of 45

Total cost now is 60

Length remaining is 4-3 = 1

Now again we cut based on max density hence we cut a rod of length 1

Now the total cost is 60 + 5 = 65

But the optimal way to save cost would be to cut it into 2 rods of length 2 each to minimize the cost where price would be 2\*28 = 54

This shows that the greedy approach is not always the optimal solution

**Exercise 15.1-5: The Fibonacci numbers are defined by recurrence (3.22). Give an O(n) time dynamic-programming algorithm to compute the n-th Fibonacci number. Draw the subproblem graph. How many vertices and edges are in the graph?**

Algorithm :

*FIBONACCI(n)*

*let fib[0. . n] be a new array*

*fib[0] = 1*

*fib[1] = 1*

*for i = 2 to n*

*fib[i] = fib[i - 1] + fib[i - 2]*

*return fib[n]*

here the complexity is only determined by the for loop which runs n-2 times or in general has run time of O(n).

The graph of this problem has n+1 nodes where each edge corresponds to the relationship between the larger subproblems to the smaller ones.

Example for say n=5 node

F[2] will connect to node F[1] and F[0] as F[2]=F[1] + F[0]

Similarly F[3] = F[2] + F[1]

and F[4] = F[3] + F[2]

and F[5] = F[4] + F[3]

Hence the tree structure would look like

Diagram

Description automatically generated

Cross indicating we already have memorized the value for that node and we don’t need to evaluate it

Hence the graph of the following would be

A picture containing scissors

Description automatically generated

Hence there are n+1 vertices in the problem ( 5+1 )

Total edges would be 2n-2 ( 2\*5 -2 = 8 )

**Exercise 15.4-1**

**Determine an LCS of ⟨1,0,0,1,0,1,0,1⟩ and ⟨0,1,0,1,1,0,1,1,0⟩**

**Calendar

Description automatically generated**