Uppgifter till föreläsning 1 om dynamisk programmering, DD2350

Talföljder Betrakta följande rekursion för en talföljd.

$$S_n = \begin{cases} 0 & \text{om } n = 0, \\ 1 & \text{om } n = 1, \\ 2 & \text{om } n = 2, \\ S_{n-1} + S_{n-2} + S_{n-3} & \text{om } n > 2 \end{cases}$$

Räkna ut de 7 första talen i följden för hand. Skriv pseudokod för en dynamisk programmeringsalgoritm som beräknar S_n. Hur mycket av historiken behöver man spara under beräkningens gång?

Generaliserade talföljder - tvådimensionell rekursion Givet följande rekursion, konstruera en algoritm som beräknar $M[i, j]$ med dynamisk programmering. Argumentera för att algoritmens beräkningsordning fungerar. Analysera algoritmens tidskomplexitet.

$$M[i, j] = \begin{cases} 0 & \text{om } i = 0 \text{ eller } j = 0, \\ M[i-1, j] + M[i-1, j-1] + 1 & \text{annars.} \end{cases}$$

Matriskedjemultiplikation Matriskedjemultiplikation är problemet att hitta bästa sättet att multiplicera ihop en kedja av matriser av varierande storlek. Genom att multiplicera ihop matriserna med varandra i en listig ordning kan antalet operationer som krävs minimeras. Att multiplicera en $a \times b$-matris och en $b \times c$-matris med vanlig matrismultiplikation tar abc talmultiplikationer.

Anta att man vill beräkna matrisprodukten $A_1 A_2 \cdots A_n$ där matris A_i har dimension $r_{i-1} \times r_i$. Låt $M[i, j]$ stå för det minimala antalet multiplikationer som behövs för att räkna ut matrisprodukten $A_i A_{i+1} \cdots A_j$. Rekursionsekvationen för $M[i, j]$ är:

$$M[i, j] = \begin{cases} 0 & \text{om } i = j \\ \min_{i \le k < j}(M[i, k] + M[k + 1, j] + r_{i-1} r_k r_j) & \text{om } i < j \end{cases}$$

Konstruera en beräkningsordning och en algoritm som beräknar M. Analysera algoritmens tidskomplexitet.
