# Teoriuppgifter till labb 2

## Uppgift 1

***Formulera rekursionen (partDist i programmet) så kompakt som möjligt med matematisk notation.***

*a = w1len, b = w2len*

partDist(0, b) = b,

partDist(a, 0) = a,

partDist(a, b) = partDist(a - 1, b - 1) + partDist(a - 1, b) + partDist(a, b - 1) + 1

## Uppgift 2

***Beräkna partDist("labd", "blad", x, y) för alla x och y mellan 0 och 4 och för in värdena i en matris M. Vad blir M?***

Matris M visas nedan:

| **x-ord / y-ord** | **“”** | **“l”** | **“a”** | **“b”** | **“d”** |
| --- | --- | --- | --- | --- | --- |
| **“”** | 0 | 1 | 2 | 3 | 4 |
| **“b”** | 1 | 1 | 2 | 2 | 3 |
| **“l”** | 2 | 1 | 2 | 3 | 3 |
| **“a”** | 3 | 2 | 1 | 2 | 3 |
| **“d”** | 4 | 3 | 2 | 2 | **2** |

## Uppgift 3

**Vad är det alltså metoden partDist(w1, w2, x, y) beräknar?**

Metoden partDist(w1, w2, x, y) beräknar något som heter *levenshteinavstånd* (*editeringsavstånd* i labblydelsen ). Det är det antalet bokstavsoperationer som krävs för att transformera det ena ordet till det andra. Vi har tre tillåtna bokstavsoperationer från labblydelsen:

1. Ta bort en av bokstäverna i ordet.

2. Lägg till en bokstav någonstans i ordet.

3. Byt ut en bokstav i ordet mot en annan bokstav.

## Uppgift 4

**Visa att tidskomplexiteten för Distance(w1, w2) är exponentiell i ordlängden. Du kan anta att w1 och w2 har samma längd.**

Det finns tre olika operationer att utföra (1. ta bort, 2. lägg till och 3. byt ut). Om vi säger att n är längden av w1 och w2. I värsta fall behöver den utföra operationer tills den kommer till rätt ord. Det betyder att Distance(w1, w2) är exponentiell i ordlängden.

## Uppgift 5

**Visa hur man kan spåra vilka editeringsoperationer som görs i den kortaste editeringsföljden från "labd" till "blad" genom att titta på matrisen M.**

1. Börja på det sista element i Matrisen M som anger editeringsavståndet för orden.
2. Kolla ifall rad och kolumn har samma bokstav: Ja: Hoppa till elementen som är diagonalt till vänster sen hoppa till (3.). Nej: hoppa direkt till (3.)
3. Gå därefter till det minsta editeringsavståndet av de celler som visas nedan:

| b)Byt ut | c)Lägg till |
| --- | --- |
| a)Ta bort | **(Den nuvarande cellen)** |

Ifall två eller fler celler har samma minsta editeringsavstånd följ ordningen a, b och sedan c vid valet. Upprepa sedan från steg (2.)

| **x-ord / y-ord** | **“”** | **“l”** | **“a”** | **“b”** | **“d”** |
| --- | --- | --- | --- | --- | --- |
| **“”** | **0** | 1 | 2 | 3 | 4 |
| **“b”** | **1 (lägg till)** | 1 | 2 | 2 | 3 |
| **“l”** | 2 | **1** | 2 | 3 | 3 |
| **“a”** | 3 | 2 | **1** | **2 (ta bort)** | 3 |
| **“d”** | 4 | 3 | 2 | 2 | **2** |

Vi utför två operationer: lägg till och ta bort. Detta motsvarar editeringsavståndet på sista elementet.

## Uppgift 6

**Visa med pseudokod hur rekursionen kan beräknas med dynamisk programmering, dvs hur en dynprogmatris M kan skapas. Vilken beräkningsordning är lämplig vid beräkning av M?**

Den rekursiva algoritmen beräknar editeringavstånd för samma substrängar flera gånger. Men med dynamisk programmering kan man spara i en matris editeringsavstånd för dessa substrängar så att de finns tillgänliga när man senare konstruerar lösningen till större delproblem (som alltså bygger på dessa redan beräknade värde för editeringsavstånd).

**Pseudokod:**

M = En tom matris (m,n)

M[0,0] //Behövs inte men gör det tydligare.

for a = 1 till m

M[a, 0] = a;

for b = 1 till n

M[0, b] = b

for x = 1 till m

for y = 1 till n

if förstaordet[x] = andraordet[y]

M[x,y] = M[x-1, y-1]

else

M[x,y] = min ( M[x-1,y-1], M[x-1,y], M[x, y-1] ) + 1

Beräkningsordning för denna version av editeringsavstånd är alltså att först fylla i den första och andra kolumnen (basfall). Sedan kan man gå från vänster till höger i varje rad nedåt tills vi har gått igenom alla element.

## Uppgift 7

**Analysera tidskomplexiteten för att bestämma editeringsavståndet mellan ett n-bokstavsord och ett m-bokstavsord med dynamisk programmering.**

Från vår pseudokod har vi två for-loopar med tidskomplexitet n och m. Vi har alltså tidskomplexiteten: O(nm). För m > n så har vi O(m2). Om n >= m så har vi O(n2).

## 

## Uppgift 8

**Beräkna dynprogmatrisen för editeringsavståndet mellan "labs" och "blad".**

| **x-ord / y-ord** | **“”** | **“l”** | **“a”** | **“b”** | **“s”** |
| --- | --- | --- | --- | --- | --- |
| **“”** | 0 | 1 | 2 | 3 | 4 |
| **“b”** | 1 | 1 | 2 | 2 | 3 |
| **“l”** | 2 | 1 | 2 | 3 | 3 |
| **“a”** | 3 | 2 | 1 | 2 | 3 |
| **“d”** | 4 | 3 | 2 | 2 | **3** |

## Uppgift 9

**Vilken del av matriserna för "labd"-"blad" och "labs"-"blad" skiljer?**

Det som skiljer sig mellan de två matriserna är det sista elementet. För matrisen “labd”-”blad” har vi en **2:a** och för “labs”-”blad” en **3:a** i M[4,4].

## Uppgift 10

**Allmänt sett, vilken del av matriserna för Y-X och Z-X skiljer när orden Y och Z har samma första p bokstäver?**

Lika:

Y-X[0…p, 0…p] = Z-X[0…p, 0…p].

Olika:

Y-X[0…, (p+1)...] != Z-X[0…, (p+1)...]

Y-X[(p+1)..., 0…] != Z-X[(p+1)..., 0…]

Vi får allmänt sett de delar av matrisen där minst ena av indexen är större än p.