**Rakin Ali & Adeel Hussain**
**Cinte 3 - ADK**

**1. Formulera Rekursionen(PartDist i programmet) så kompakt som möjligt med matematiskt notation.**

Låt n och m vara godtyckliga tal sådana att

$n \wedge m > 0$

Då gäller:

$PartDist(0, n) = n$

$PartDist(m, 0) = m$

$PartDist(n, m) = PartDist(n-1, m-1) + PartDist(n-1, m) + PartDist(n, m-1) + 1$

**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?**

| **Ord1 / Ord2** | **“”** | **L** | **A** | **B** | **D** |
|---|---|---|---|---|---|
| **“”** | 0 | 1 | 2 | 3 | 4 |
| **B** | 1 | 1 | 2 | 2 | 3 |
| **L** | 2 | 1 | 2 | 2 | 3 |
| **A** | 3 | 2 | 1 | 2 | 3 |
| **D** | 4 | 3 | 2 | 2 | 2 |

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

Levenshteinavstånd mellan strängarna w1 och w2 som är alltså antal operationer(replace, insertion and deletion) som behöver göras på w2 för att få w1.

**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 operationer per sträng (Insert, Delete & Replace). Alltså vid “worst case scenario” blir det $3^n$ grenar där n är längsta ordets längd. Det rekursiva funktionen går igenom alla möjliga kombinationer tills den hittar det rätta ordet.

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

Börja på matrisen där editering operationen skrivs för ord1 och ord2,

Gå till närmaste minsta editering operation som antingen är till

a. vänster ( M[ i ][ j-1 ] ),
b. snett upp till vänster ( M[ i-1 ][ j-1 ] )
c. eller rakt upp ( M[ i-1 ][ j ] )

2. För varje element där raden och kolumnen har samma bokstav så ska man röra sig diagonalt till vänster.
3. Rörelse sidleds vänster = “Delete operation”
4. Rörelse lodrätt uppåt = “Insert operation”
5. Rörelse diagonalt = “Replace operation”

**Grön = Insert Operation** **Blå = Hoppa diagonalt** **Röd = Delete operation**

| **Ord1 / Ord2** | **“”** | **L** | **A** | **B** | **D** |
|---|---|---|---|---|---|
| **“”** | **0** | 1 | 2 | 3 | 4 |
| **B** | 1 | 1 | 2 | 2 | 3 |
| **L** | 2 | 1 | 2 | 2 | 3 |
| **A** | 3 | 2 | 1 | 2 | 3 |
| **D** | 4 | 3 | 2 | 2 | **2** |

**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?**

M = En tom matris av storleken (m,n)

```
for i = 0 tills i <= m
        M[ i ][ 0 ] = i

for j = 0 tills j <= n
        M[ 0 ][ j ] = j
for i = 1 tills <= m
```

```
for j = 1 tills <= n
        if ( w1.charAt[ i-1 ] == w2.charAt[ j-1 ]) //Om bokstäverna är lika
                M[ i ] [ j ] = M[ i-1 ][ j-1 ]          // Sätt till diagonalens värdesl
        else
                M[ i ] [ j ] = min ( M[ i-1 ][ j-1 ], M[ i-1][ j ], M[ i ][ j-1] ) + 1
```

Den lämpliga beräkningsordningen är först att fylla i den första raden och den första kolumnen. Därefter fyller man i rad för rad i ordning av de element som är kvar. Viktigaste är att börja fylla i första raden och kolumnen först då det sedan inte spelar roll ifall man fyller i kolumnvis eller radvis efteråt. Tidskomplexiteten blir densamma.

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

Vi har två loopar med tidskomplexitet m respektive n, detta ger tidskomplexiteten O(m*n)

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

| **Ord1 / Ord2** | **“”** | **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 |

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

Den sista bokstaven ändras om man jämför labd och labs, detta leder till att sista elementet i matrisen ändras från 2 till 3 operationer.

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

Alla element efter M[p+1][ 0 ] till M[ y ][ x ] i matrisen kan bli olika ifall man jämför orden Y och Z i en matris innehållande X.