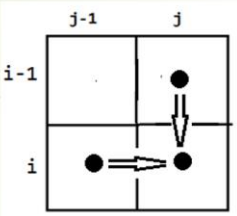
גישה 3 תכנות דינאמי



העלות הטובה ביותר בקדקוד (i,j) שווה לעלות המינימאלי בין העלויות הגעה מנקודה (i-1,j) או מנקודה (i,j-1)

$$mat(i,j).price = \min(mat(i-1,j).price + \ mat(i-1,j).y,$$
$$mat(i,j-1).price + mat(i,j-1).x)$$

1) מספר מסלולים בעלי עלות מינימאלית.

משווים עלויות מלמטה ומשמאל:

- כאשר העלויות שונות מעתיקים מספר המסלולים מכוון של העלות הנמכה יותר,
- כאשר העלויות שוות – סוכמים את מספר המסלולים משני הכוונים.

הסיבוכיות: O(M*N).

```
a = mat[i-1][j].price+mat[i-1][j].y //from above
b = mat[i][j-1].price+mat[i][j-1].x // from left
if (a < b) //from above
      mat[i][j].price = a
      mat[i][j].numOfPaths = mat[i-1][j].numOfPaths
else if (a > b) // from left
      mat[i][j].price = b
      mat[i][j].numOfPaths = mat[i][j-1].numOfPaths
else{//x==y, the same price
      mat[i][j].numOfPaths = mat[i][j-1].numOfPaths+mat[i-1][j].numOfPaths
end-if
```

2) חישוב מסלול אחד טוב ביותר.

מתחילים לעלות מנקודה (N,M) בכל נקודה בודקים מאיזה כוון הגענו לנקודה זו בעזרת השוואת העלויות.

כאשר העלויות שונות - הולכים לכוון העלות הנמוכה יותר,

כאשר העלויות שוות – קובעים את הכיוון האחיד למצב כזה לכל הנקודות, לדוגמה תמיד יורדים שורה, או תמיד הולכים ימינה.

הסיבוכיות: O(M+N)

```
i = M-1, j = N-1, ans = ""
while (i > 0 && j > 0)
      a = mat[i-1][j].price + mat[i-1][j].y //from above
      b = mat[i][j-1].price + mat[i][j-1].x // from left
      if (a < b) // down
            ans = "1" + ans
            i = i - 1
      else //a ≥ b, to left
            ans = "0" + ans
            j = j - 1
      end-if
end-while
while (j > 0)
      ans = "0" + ans
      j = j - 1
end-while
while (i > 0)
      ans = "1" + ans
      i = i - 1
end-while
return ans
```

3) חישוב את כל המסלולים הטובים ביותר. פונקציה רקורסיבית.

בגלל שמספר מסלולים יכול להיות גדול מדי $\binom{M+N}{M}$, מגדירים מספר שלם חיובי **teta** שמהווה גבול קביל למספר המסלולים.

```
void allPathsRecurs(teta)
    if (numOfPaths <= teta)
        ArrayList<String> paths = new  ArrayList<String>(numOfPaths)
        buildPaths(new String(), mat.length-1, mat[0].length-1, paths)
        System.out.println(paths);
    end-if
end-allPathsRecurs
```

הסיבוכיות: O((M+N)*nPaths)

3) חישוב את כל המסלולים הטובים ביותר. פונקציה רקורסיבית.

```
public void buildPaths(String path, int i, int j, ArrayList<String> paths)
      if (i>0 && j>0)
            a = mat[i-1][j].price+mat[i-1][j].y
            b = mat[i][j-1].price+mat[i][j-1].x
            if (a < b)
                  buildPaths("1"+path, i-1, j, paths)
            else if(a > b)
                  buildPaths("0"+path, i, j-1, paths)
            else //a==b
                  buildPaths("1"+path, i-1, j, paths)
                  buildPaths("0" + new String(path), i, j-1, paths)
            end-if
      else if (i==0 && j==0)
            paths.add(path)
      else if (i==0)
            String t = new String()
            for  k=0 to j-1
                  t = t +  "0"
            paths.add(t + path)
      else if (j==0)
            String t = new String()
            for  k=0 to i-1
                  t = t +  "1"
            paths.add(t + path)
      end-if
end-buildPaths
```