שעור 2 תורת הגרפים – המרחקים הקצרים ביותר, אלגוריתם Floyd-Warshall

אלגוריתם פלויד-וורשאל הוא אלגוריתם במדעי המחשב המשמש למציאת המסלולים הקצרים ביותר בין כל שני זוגות הקדקודים, **בגרף משוקלל ומכוון**. האלגוריתם מבוסס על פרדיגמת **התכנון הדינמי .**

האלגוריתם פורסם על ידי רוברט פלויד בשנת 1962 בגרסתו המוכרת לנו כיום, אך גרסאות דומות של האלגוריתם פורסמו ב 1959-על ידי ברנארד רוי, ועל ידי סטיבן וורשאל ב-1962.

תכנות דינמי הוא חלופה לפתרון בעיות בעזרת חיפוש שלם או אלגוריתמים חמדניים. משתמשים בתכנות דינאמי כאשר ניתן לפתור את הביעה בעזרת תת-בעיה קטנה יותר. השיטה נראית כך:

1. לחלק את המשימה לתת-משימות בגודל קטן יותר.
2. מציאת הפתרון האופטימלי של תת-משימות באופן רקורסיבי.
3. שימוש של הפתרון של התת-משימות לבניית הפתרון של הבעיה המקורית.

חיפוש שלם למציאת המסלולים הקצרים ביותר בין כל שני זוגות הקדקודים, בגרף ייקח המון זמן וזיכרון נוסף.

מבנה של מסלול קצר ביותר.

האלגוריתם מבוסס על התכונה הבאה של המסלול הקצר ביותר.

טענה.

יהיה $p_{1k} = (v_1, v_2, \ldots, v_k)$ הוא המסלול הקצר ביותר בין קדקוד v_1 ל- v_k והיה $p'_{ij} = (v_i, \ldots, v_j)$ הוא תת-מסלול של מסלול p. אם p הוא המסלול הקצר ביותר בין קדקוד v_1 ל- v_k אז p'_{ij} הוא גם מסלול קצר ביותר בין קדקודים v_i ל v_j .

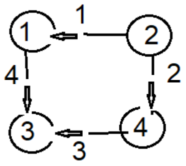
הוכחה בדרך השלילה. עלות של מסלול p_{1k} מורכבת מעלות של המסלולים $(v_1, \ldots, v_{i-1})$, p'_{ij} ו- $(v_{j+1}, \ldots, v_k)$. אם p'_{ij} הוא לא מסלול קצר ביותר בין v_i ל v_j אז קיים מסלול קצר ביותר בין קדקודים v_j ,v_i, כלומר אפשר לקצר את מסלול p_{1k} שזה סתירה להנחה ש-p_{1k} הוא קצר ביותר. ■.

בניית האלגוריתם

נתבונן באוסף $\{v_1, v_2, \ldots, v_n\}$ קדקודי הגרף ומסלול p_{ij} בין קדקודים v_i , v_j העובר דרך בדיוק k קודקודים מורשים $\{v_1, v_2, \ldots, v_k\}$.

כאשר $k = 0$ מתבוננים רק בצלעות שמחברים את הקדקודים (מספר קודקודים מורשים שווה 0). כאשר שני קדקודים לא מחוברים ע"י צלע – המרחק ביניהם (עלות המעבר) נחשב כאינסופי. כאשר $k = 1$ מתבוננים במסלולים העוברים דרך v_1, כאשר $k = 2$ מתבוננים במסלולים העוברים דרך v_1, v_2, וכן הלאה.

לדוגמה בגרף הבא, כאשר $k = 0$ מטריצת שכנות של הגרף מייצגת מרחקים ישרים בין זוגות הקדקודים בלבד. כאשר $k = 1$, ניתן לעבור דרך קדקוד 1 ומקבלים מסלול מקדקוד 2 לקדקוד 3: $dist(2,3) = 5$.



נתבונן במסלול קצר ביותר p_{ij} כאשר הקדקודים המורשים הם $\{v_1, v_2, \ldots, v_{k-1}\}$, נסמן את המרחק ב- d_{ij}. נוסיף קדקוד v_k לרשימת המורשים. כאן יש שתי אפשרויות:

א) קדקוד v_k לא נמצא על המסלול הקצר ביותר בין v_i לבין v_j, כלומר הוספת קדקוד חדש לא שיפרה על המרחק בין v_i ו- v_j, לכן $d_{ij}^{k} = d_{ij}^{k-1}$.

ב) קדקוד v_k נמצא על המסלול הקצר ביותר בין v_i לבין v_j, כלומר המסלול הקצר ביותר בין v_i לבין v_j כרגע עובר דרך קדקוד v_k. מהו המרחק החדש בין v_i ו- v_j?
המסלול החדש נשבר ע"י קדקוד v_k לשני מסלולים: p_{ik} ו- p_{kj}. המסלולים p_{ik} ו- p_{kj} הם תת-מסלולים של המסלול הקצר ביותר, לכן הם גם המסלולים הקצרים ביותר בין v_i, v_k ובין v_k, v_j בהתאם. לכן $d_{ij}^{k} = d_{ik}^{k} + d_{kj}^{k}$. במסלולים האלה קדקוד v_k הוא קדקוד סופי או קדקוד ראשון, כלומר לא שייך לרשימת המורשים שהם קדקודים אמצעים, לכן מותר למחוק אותו מרשימת המורשים, מקבלים: $d_{ij}^{k} = d_{ik}^{k-1} + d_{kj}^{k-1}$.

מסקנה: בשני המקרים המרחק בין v_i ל- v_j מורכב ממרחקים של הקבוצה הקודמת של הקדקודים המורשים. מכאן נובע שבהינתן מטריצת שכנות המתאימה ל- $k = 0$ אנו יכולים לחשב את המרחקים הקצרים ביותר לכל $k = 1,2,\ldots,n$. כאשר $k = n$ מקבלים מסלולים אופטימליים לכל זוגות קדקודי הגרף. כאושר עוברים מ-$k-1$ ל- k d_{ij}^{k} מקבל ערך מינימאלי שמתקבל משני המקרים

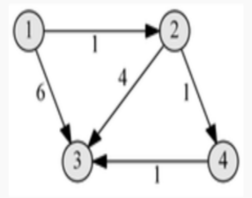
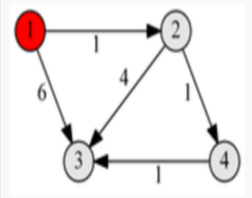
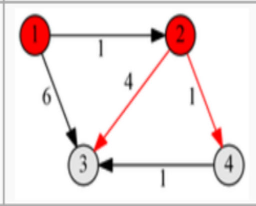
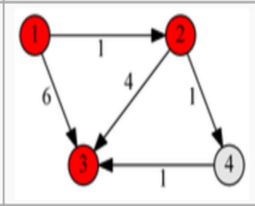
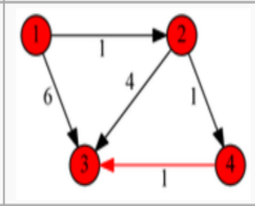
$$d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$$

פסדו-קוד:

```
void FWAlgorithm(w[][])
   // array of minimum distances initialized to ∞ (infinity)
   for each pair of vertices (i,j)
       dist[i][j] = ∞
   end-for
   for each edge vertices(i,j)
       dist[u][v] = w(u,v) // the weight of the edge (u,v)
   end-for
   for each vertex v
       dist[v][v] = 0
   end-for
   for k from 0 to n-1
       for i from 0 to n-1
          for j from 0 to n-1
              if (dist[i][k]!=∞ && dist[k][j]!=∞)
                 dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]
              end if
          end-for
       end-for
   end-for
end-FWAlgorithm
```

סיבוכיות האלגוריתם $O(n^3)$.

דוגמה. בגרף הבא נדגים את מטריצת שכנות לכל $k = 1,2,3,4$:

$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
				
$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 6 & \infty \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{5} & \mathbf{2} \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & 5 & 2 \\ \infty & \times & 4 & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$	$\begin{pmatrix} \times & 1 & \mathbf{3} & 2 \\ \infty & \times & \mathbf{2} & 1 \\ \infty & \infty & \times & \infty \\ \infty & \infty & 1 & \times \end{pmatrix}$

מציאת את המסלולים הקצרים ביותר :

```
for i from 1 to n
   for j from 1 to n
       if (dist[i][j]!=∞) path[i][j] = i→j
       else path[i][j] = ""
    end-for
end-for
for k from 1 to n
    for i from 1 to n
       for j from 1 to n
           if (dist[i][k]!=∞ && dist[k][j]!=∞)
               if (dist[i][j] > dist[i][k]+ dist[k][j])
                   dist[i][j] = dist[i][k]+ dist[k][j]
                   path[i][j] = path[i][k]+ path[k][j]
               end-if
           end-if
       end-for
    end-for
end-for
```

כאשר הגרף מוגדר בעזרת **מטריצה בוליאנית** $B[n][n] = \{b_{ij}\}$, כאשר $b_{ij} = true$ כאשר קדקודים i,j מחוברים ע"י צלע, ו- $b_{ij} = false$ אחרת. במקרה כזה אלגוריתם פלויד-וורשל מאפשרת לבדוק באיזה זוגות של קדקודים מחוברים ע"י מסלול, כלומר האם גרף קשיר ולחשב את רכיבי הקשירות שלו, אבל אין אפשרות לבנות מסלול קצר ביותר בין שני קדקודי הגרף.

פסדו-קוד:

```
void FWBoolean(boolean [][] B)
   for (int k = 0; k<n; k++)
      for (int i = 0; i<n; i++)
         for (int j = 0; j<n; j++){
             B[i][j] = B[i][j] || (B[i][k] && B[k][j]);
         end-for
      end-for
   end-for
end-FWBoolean
```

משקלים שליליים בגרף משוקלל מכוון ולא מכוון

מעגל שלילי - מעגל שסכום משקלי צלעותיו שלילי (מה שגורם לכך שאין תשובה מוגדרת לשאלת המסלולים הקצרים) .

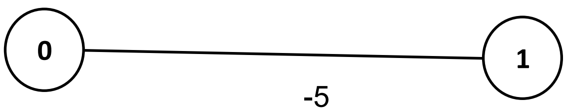
יהיה $G(V,E)$ – **גרף משוכלל ולא מכוון**, $(a,b)\in E$ – צלע, $weight(a,b)=w<0$, מרחק בין קדקודים a,b – שלילי. במקרה של משקלים שליליים אלגוריתם של Floyd-Warshall לא נותן תוצאה נכונה.

הוכחה:

```
m - matrix, m[i,i] ← 0
for k=1 to n
      for i to n
            for j to n
                  m[i,j] = min(m[i][j], m[i,k]+m[k,j])

k=a, i=b, j=b → m[b,b]=min(m[b,b], m[b,a]+m[a,b]) = 2w < 0 → error!
k=b, i=b, j  → m[b,j]=min(m[b,j], m[b,b]+m[b,j]) → error
etc.
```

מסקנה: לאחר הפעלת אלגוריתם של פלויד-וורשל באלכסון מופיעים מספרים שליליים אם ורק אם בגרף יש מעגל שלילי.

Example 1:



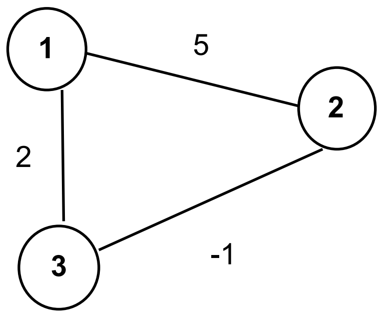

```
Input: 0,-5
      -5, 0
Output:
k=0, i=0, j=0, mat[0,0]=0
k=0, i=0, j=1, mat[0,1]=-5
k=0, i=1, j=0, mat[1,0]=-5
k=0, i=1, j=1, mat[1,1]=-10
k=1, i=0, j=0, mat[0,0]=-10
k=1, i=0, j=1, mat[0,1]=-15
k=1, i=1, j=0, mat[1,0]=-15
k=1, i=1, j=1, mat[1,1]=-20
```

output incorrect:

-10	-15
-15	-20

Example 2:

```
Input:  0, 5, 2
        5, 0,-1
        2,-1, 0
Output:
k=0, i=0, j=0, mat[0,0]=0
k=0, i=0, j=1, mat[0,1]=5
k=0, i=0, j=2, mat[0,2]=2

k=0, i=1, j=0, mat[1,0]=5
k=0, i=1, j=1, mat[1,1]=0
k=0, i=1, j=2, mat[1,2]=-1

k=0, i=2, j=0, mat[2,0]=2
k=0, i=2, j=1, mat[2,1]=-1
k=0, i=2, j=2, mat[2,2]=0

k=1, i=0, j=0, mat[0,0]=0
k=1, i=0, j=1, mat[0,1]=5
k=1, i=0, j=2, mat[0,2]=2

k=1, i=1, j=0, mat[1,0]=5
k=1, i=1, j=1, mat[1,1]=0
k=1, i=1, j=2, mat[1,2]=-1

k=1, i=2, j=0, mat[2,0]=2
k=1, i=2, j=1, mat[2,1]=-1
k=1, i=2, j=2, mat[2,2]=-2

k=2, i=0, j=0, mat[0,0]=0
k=2, i=0, j=1, mat[0,1]=1
k=2, i=0, j=2, mat[0,2]=0

k=2, i=1, j=0, mat[1,0]=1
k=2, i=1, j=1, mat[1,1]=-2
k=2, i=1, j=2, mat[1,2]=-3

k=2, i=2, j=0, mat[2,0]=0
k=2, i=2, j=1, mat[2,1]=-3
k=2, i=2, j=2, mat[2,2]=-4
```



Correct

v	1	2	3
1	0	1	2
2	1	0	-1
3	2	-1	0

incorrect:

	1	2	3
1	0	1	0
2	1	-2	-3
3	0	-3	-4

בעצם כדי לעבוד עם משקלים שליליים חייבים לא להרשות לעבור פעמיים על אותה צלע.

קשירות הגרף.

כאשר לאחר הפעלת אלגוריתם של פלויד-וורשל בגרף לא מכוון ומשוקלל לא נשארו צלעות שמשקלן אינסופי – הגרף קשיר. במקרה שיש ∞ אחד לפחות – הגרף אינו קשירץ

במקרה של מטריצה בוליאנית אם כל איברי המטריצה הם $true$ – הגרף קשיר, כאשר יש לפחות $false$ אחד – הגרף אינו קשיר.

מציאת רכיבי קשירות בגרף המוצג ע"י מטריצה בוליאנית. פסדו-קוד:

```
int[] connectComponentsOfGraphBool(boolean [][] mat)
// connectComp[j] - the component number of vertex j
// num – number of components
      flag = true
      for(int i=0; i<mat.length; i++){
// vertex i doesn't belong to any component
// we must check if vertex i has connection
// with vertex from exist component
            if (connectComp[i]==0)
                  flag = true
                  for (int k=0; flag && k<mat.length; k++)
                        if (connectComp[k]!=0 && mat[i][k])
                              connectComp[i] = connectComp[k]
                              flag = false
                        end-if
                  end-for
                  if (flag)
                        numComponentes++
                        connectComp[i]=numComponentes;
                        for(int j=i+1; j<mat.length; j++)
                  // vertex j is not defined yet, the path exists
                              if (connectComp[j]==0 && mat[i][j])
                                    connectComp[j] = numComponentes
                              end-if
                        end-for
                  end-if
            end-if
      end-for
      return connectComp;
end-connectComponentsOfGraphBool
```