אלגוריתם דייקסטרה (Dijkstra’s algorithm)

אלגוריתם דייקסטרה פותר את בעיית מציאת המסלול הקצר ביותר ממקור יחיד עבור גרף (מכוון או לא מכוון) ממושקל G=(V,E) במקרה שבו כל משקולות הצלעות הם אי-שליליים. המשקולות בגרף מסמלות מרחק.
משמעותו של המסלול הקצר ביותר בין שתי נקודות היא המסלול בעל סכום המשקולות הנמוך ביותר בין שני הקדקודים.
אלגוריתם של דייקסטרה מוצא מסלולים קצרים ביותר מקדקוד מקור s לכל קדקודי הגרף.
בתמצית ניתן לסכם את פעולת האלגוריתם כך:

<u>פעולת האלגוריתם</u>
הגרף נתון כמטריצת שכנות (או רשימת שכנות). עבור כל קדקוד, מסומן האם ביקרו בו או לא ומה מרחקו מקדקוד המקור. בהתחלה כל הקדקודים מסומנים כאילו לא ביקרו בהם, ומרחקם מוגדר כאינסוף.

לולאת האלגוריתם:

- כל עוד נותרו קדקודים שלא ביקרנו בהם:
  - מסמנים את X (הקדקוד הנוכחי. באיטרציה הראשונה זהו s) כקדקוד שביקרו בו.
  - עבור כל קדקוד Y שהוא שכן של X וגם לא ביקרנו בו:
    - Y מעודכן, כך שמרחקו יהיה שווה לערך המינימלי בין שני ערכים: מרחקו הנוכחי, ומשקל הצלע המחברת בין X לבין Y בתוספת המרחק בין s ל-X.
  - בוחרים קדקוד X חדש בתור הקדקוד שמרחקו בשלב הזה מצומת המקור s הוא הקצר ביותר מבין כל הקדקודים בגרף שטרם ביקרנו בהם.

האלגוריתם מסתיים כאשר קדקוד X החדש הוא היעד או (למציאת כל המסלולים המהירים ביותר) כאשר ביקרנו בכל הקדקודים.

<u>Dijkstra's algorithm Pseudo-Code</u>

```
class Edge
      int vert
      double weight
      public Edge(int v, double w)
            vert = v
            weight = w
      end-constructor
end-class-Edge

// Vertex – contains list of adjacency edges with weights
class Vertex implements Comparable<Vertex>
      int name, pred, dist
      Edge[] edges
      boolean visited

      public Vertex(name)
            this.name = name
            dist = ∞
            pred = nil
            visited = false
      end-constructor
```

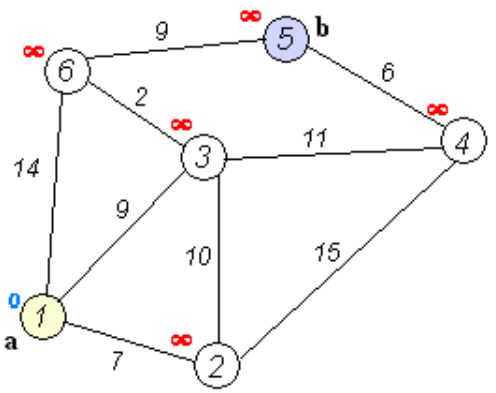
```
      @Override
      int compareTo(Vertex v)
            ans = 0
            if (this.dist - v.dist > 0) ans = 1
            else if (this.dist - v.dist < 0) ans = -1
            return ans
      end-compareTo
end-class-Vertex
```

```
Q – Priority Queue (MinHeap), contains vertices with tags
G – array of vertices
S – the source vertex

Dijkstra(Vertex[] G, s)//G=Vertex[], Vertex s
      s.dist = 0
     //Inserts the specified element into this priority queue
      for i=0 to G.length-1 //O(|V|)
            Q.add(G[i])
      end-for

      //O(|V|) + O(|E|*log|V|) = O(|E|*log|V|)
      while (Q not empty)
      // Retrieves and removes the head of the queue
            u = Q.extractMin()
            for i=0 to u.edges.length-1
                  v = u.edges[i].vert
                  if (!v.visited)
       // Relaxation
                        t = u.dist + weight(v, u)
                        if (v.dist > t)
                              v.dist  = t
                              v.pred= u
      // Update the lable of v
                              Q.decreaseKey(v, t) // O(log|V|)
                        end-if
                  end-if
            end-for
            u.visited= true
      end-while
end-Dijkstra
```

<u>סיבוכיות האלגוריתם</u> תלויה במבנה הנתונים השומר את הקדקודים שטרם ביקרנו בהם:

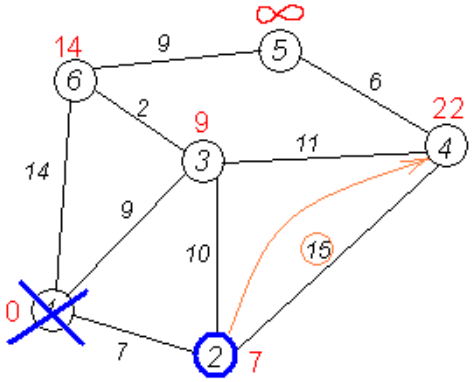
- אם מדובר ברשימה או במערך, הסיבוכיות היא ב-$O(|V|^2 + |E|) = O(|V|^2)$.
- אם משתמשים בערימה בינומית סיבוכיות הריצה משתפרת $O(|E|*\log_2|V|)$

דוגמה: נתבונן בגרף שבאיור. נניח רוצים למצוא את המרחק הקצר ביותר מקדקוד -1 לכל קדקודי הגרף.



ליד כל קדקוד מסומן בתווית אדומה - אורכו של המסלול הקצר ביותר בין קדקוד זה לקדקוד 1. בגלל שעוד לא גילינו את קדקודי הגרף האורך ההתחלתי בין קדקוד 1 לכל קדקוד הגרף הוא אינסופי.

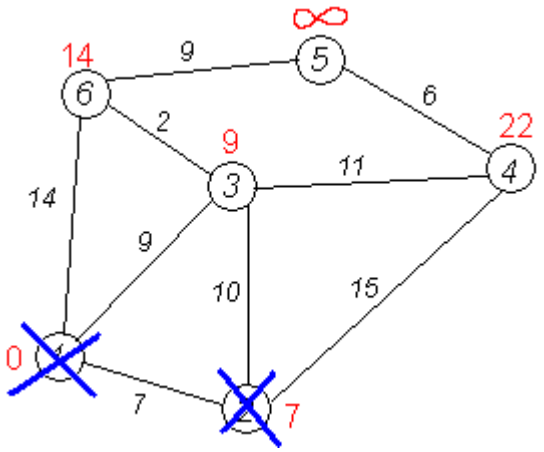
הצעד הראשון. המשקל המינימאלי הוא של קדקוד 1. השכנים של קדקוד 1 הם קדקודים 2, 3 ו -6. השכן הראשון של קדקוד 1 הוא קדקוד 2 כוון שאורכו עד קדקוד 1 מינימאלי ושווה 7. אורכו המסלול הקצר ביותר שווה לסכום של תווית שלו ואורך הצלע שבין 1 ל-2 שווה 0 + 7 = 7. 7 קטן אינסוף לכן התווית החדשה של קדקוד 2 היא 7. אנו בצעים את אותה פעולה עם שני השכנים האחרים של קדקוד 1 – עם קדקוד 2 וקדקוד 6.
גילינו את כל השכנים של קדקוד 1 וחישבנו את המרחק המינימאלי בין קדקוד 1 לבין קדקודים 3, 2, ו-6.
סיימנו טיפול בקדקוד 1 ונסמן אותו כ-visited=true.

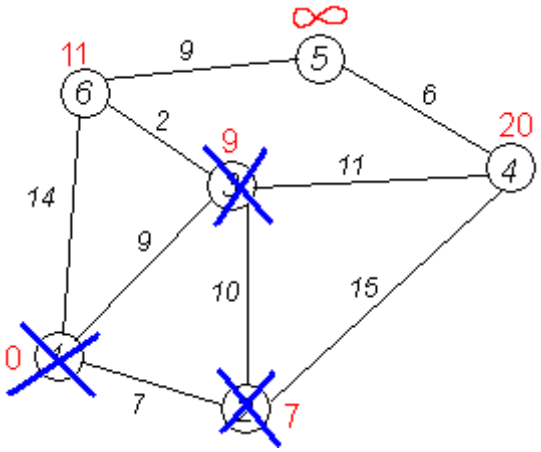


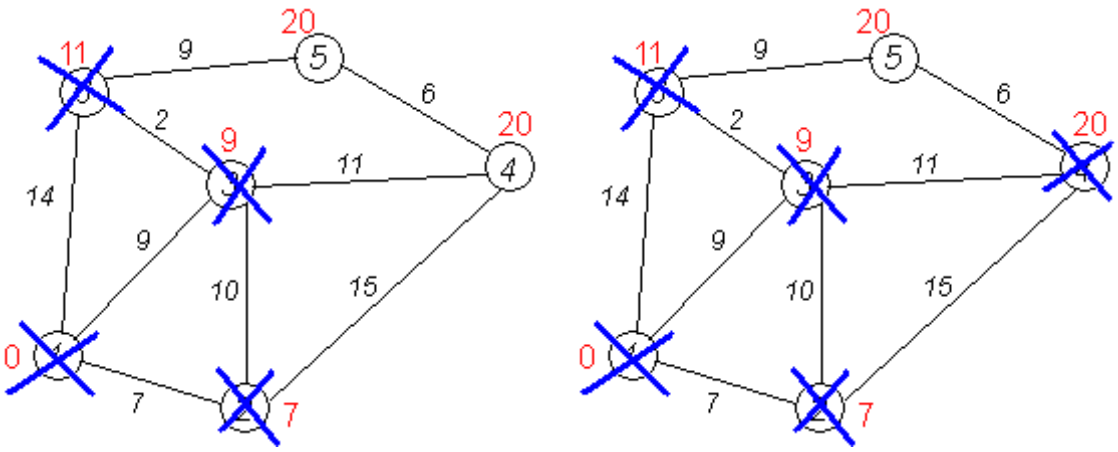
הצעד השני. שוב, אנו מוצאים את הקדקוד "הקרוב ביותר" לקדקוד 1 – זהו קדקוד 2. השכנים של 2 הם 3 ו-4.
ננסה להפחית את התווים שלהם. לשם כך ננסה להשיגם דרך קדקוד 2. המרחק בין 1 ל-3 דרך 2 שווה
10+7= 17 ו- 17>9 לכן התווית של 3 לא משתנה.
השכן הבא של 2 – קדקוד 4. המרחק מ-1עד 4 דרך 2 שווה
7+15=22 ו- 22<∞ לכן קדקוד 2 מקבל תווית חדשה – 22.
סיימנו טיפול בקדקוד 2 ונמחק אותו מהגרף.

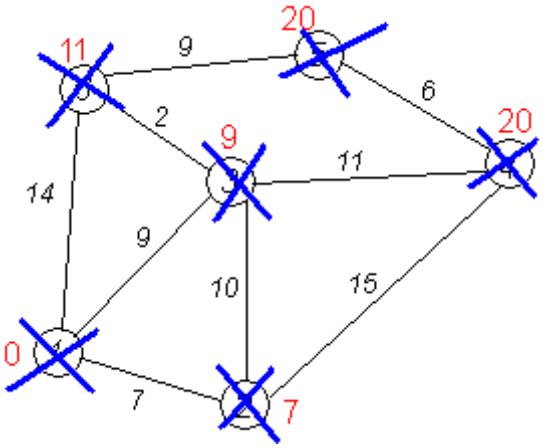
הצעד השלישי. בוחרים בקדקוד 3 ומטפלים כל השכנים שלו. בסיום הטיפול מקבלים:

צעדים הבאים:









האלגוריתם מסתיים כאשר כל הקדקודים נמחקו. התוצאה של האלגוריתם ניתן לראות באיור האחרון: המסלול הקצר ביותר מ -1 ל -2 שווה 7, ל-3 שווה 9, ל- 4,5 שווה 20 ל-6 שווה 11.

נכונות אלגוריתם של דייקסטרה

משפט.

יהי G גרף מכוון ממושקל משקלי הצלעות מספרים לא שליליים, s– קדקוד מקור. אז לאחר הרצת האלגוריתם מתקיים $dist[v] = \delta(s,v)$ לכל $v \in V(G)$, כאשר $\delta(s,v)$ הוא המרחק הקצר ביותר בין קדקודים s, v.

הוכחה:

נוכיח כי לכל קדקוד v כאשר מוצאים אותו מתור עדיפויות Q מתקיים שוויון $dist[v] = \delta(s,v)$.

הוכחה באינדוקציה.

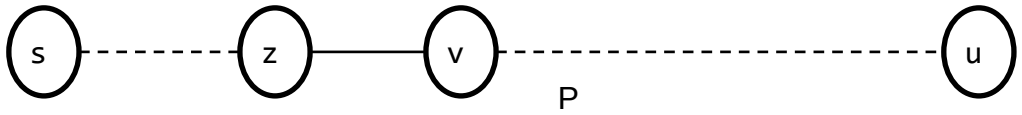
א) בסיס אינדוקציה: בשלב ראשון $dist[s]=\delta(s,s)=0$.

ב) נניח שהטענה נכונה עבור n שלבים ראשונים של האלגוריתם.

ג) נוכיח את הטענה עבור שלב n+1. נניח שבשלב n+1 בחרנו בקדקוד u ונוכיח כי $dist[u]=\delta(s,u)$.

נשים לב כי לכל $x \in V$ מתקיים $dist[x] \geq \delta(s,x)$. מוציאים u מתור.

יהיה P המסלול הקצר ביותר מ-s ל-u. נסמן ב-v את הקדקוד הראשון הלא מטופל במסלול P ויהיה z קדקוד קודם ל-z ב-P, לכן קדקוד Z מטופל ועבורו מתקיים $dist[z] = \delta(s,z)$ לפי הנחת האינדוקציה.



כוון ש-P הוא מסלול קצר ביותר מ-s ל- u התת-מסלול שלו מ- s ל-v הוא גם קצר ביותר, לכן

$\delta(s,v) = \delta(s,z) + weight(v,z) = dist[z] + weight(z,v)$ או

$$\delta(s,v) = dist[z] + weight(z,v)$$

אבל במסלול שהתקבל לפי דייקסטרה קדקוד v קיבל תווית מינימאלית, לכן

$dist[v] \le dist[z] + weight(z,v) = \delta(s,v)$.

כוון ש- $\delta(s,v)$ – הוא המרחק הקצר ביותר מ- s ל-v, אז $dist[v] = \delta(s,v)$.

אבל קדקוד v לא מטופל ואנו בחרנו בקדקוד u, לכן

$$dist[u] \le \mathbf{dist[v]} = \delta(s,v) \le \delta(s,u)$$

ושוב $\delta(s,u)$ הוא המרחק הקצר ביותר מ- s ל-u ולכן

$dist[u]= \delta(s,u)$.

האלגוריתם מסתיים כאשר כל הקדקודים מטופלים, לכן לכל קדקוד מתקיים $dist[u]= \delta(s,u)$.
מש"ל.

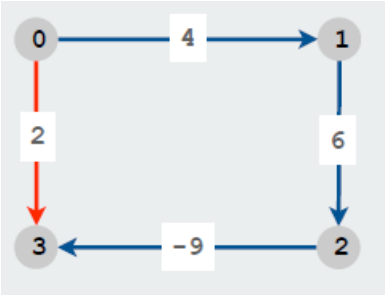
סיבוכיות האלגוריתם: הלולאה העיקרית מתבצעת בסיבוכיות $O(|E|)$ פעמים, הוצאת איבר מינימאלי מתור עדיפויות מתבצעת בסיבוכיות $O(\log_2|V|)$, לכן סיבוכיות של אלגוריתם דייקסטרה $O(|E|\log_2|V|)$.

כמו ב- BFS אלגוריתם של דייקסטרה מאפשר לחשב את המסלול הקצר ביותר בין שני קדקודי הגרף:

```
getPath(G, u, v){
      Dijkstra(G, u)
      t = v
      String path = t
      while(t != u)
            t =  t.pred
            path = t + path;
      end-while
      return path;
end-getPath
```

הערה חשובה: אלגוריתם של דייקסטרה לא עובד עם משקלים שליליים.

דוגמה: ניקח s=0, האלגוריתם של דיקסטרה יבחר בקדקוד 3 מיד אחרי s. זה לא נכון בגלל שהמסלול הקצר ביותר מקדקוד 0 לקדקוד 3 הוא 0→1→2→3 שמשקלו שווה 9-6+4=1.



הוספת 9 לכל הצלעות גם לא ייתן פתרון נכון, בגלל שמוסיפים לכל הצלעות: