Minimum Spanning Tree - Kruskal Algorithm

עץ פורש מינימלי (Minimum spanning tree - MST) הוא עץ המורכב מתת-קבוצה של צלעות בגרף קשיר לא מכוון ומקיים את התכונות הבאות:

א) פורש את הגרף - כלומר, כולל את כל קדקודי הגרף.

ב) מינימלי (או: **מזערי**) בסכום משקלי הצלעות שלו, שהוא הקטן ביותר האפשרי מכל העצים הפורשים של הגרף.

האלגוריתם הראשון למציאת עץ פורש מזערי בגרף לא מכוון הומצא בידי המדען הצ'כי **Otakar Boruvka** ב1926.

מטרתו הייתה מציאת כיסוי חשמלי יעיל של חבל מורביה. כיום משתמשים בשני אלגוריתמים ידועים לגרף לא מכוון:

אלגוריתם של קרוסקל (Kruskal) ואלגוריתם של פרים (Prim).

האלגוריתם של קרוסקל הוא אלגוריתם **חמדני** לפתרון בעיית מציאת עץ פורש מינימלי בגרף משוקלל לא מכוון.

שלבי האלגוריתם:

1. בונים T – עץ פורש מינימאלי ריק.

2. ממיינים את צלעות הגרף בסדר עולה (מקטן לגדול).

3. מגדירים מספר קבוצות זרות שכל קדקוד הגרף שייך לקבוצה שלו, כלומר מספר קבוצות שווה למספר קדקודי הגרף |V| .

   במילים אחרות מגדירים יער המורכב מ-|V| עצים שכל עץ מכיל רק את המשורש שלו שהוא קדקוד הגרף.

4. שולפים צלע e=(u, v) בעלת משקל מינימאלי.

5. אם הצלע מחברת בין שני עצים, כלומר קדקודים u ו-v שייכים לקבוצות שונות מוסיפים אותה ל-T ומאחדים

   את העצים. (צלע כזו נקראת "צלע בטוחה" – safe edge).

   במקרה שהצלע מחברת שני קדקודים השייכים לאותו עץ (הצלע כזו סוגרת מעגל) מדלגים עליה.

6. בודקים האם מספר צלעות ב-T שווה ל- |V|-1. אם כן האלגוריתם מסתיים, אם לא חוזרים לשלב 4.

Pseudo – code of Kruskal Algorithm

```
kruskal(G)// T – Result Minimum Spanning Tree
  T←Φ  // build empty tree
  // build groups: the group contains only one vertex
  for each vertex v∈V[G]//O(n)
     makeSet(v)
  end-for
  //sort the edges of E into non-decreasing order by weight
  sort(E) // O(mlog2n)
  for each edge (u,v)∈E //O(m)
     if (set(u) != set(v))
        T←T∪{(u,v)}
        union(u,v) //O(α(n)) ≅ O(1)
     end-if
   end-for
 end kruskal
```

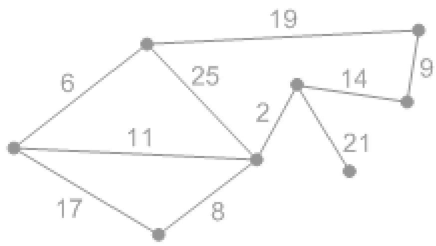
סיבוכיות של אלגוריתם קרסקל: נסמן ב-$|E|$, $m = |V|$, $n =$ סיבוכיות של מיון מערך הצלעות היא $O(mlog_2m)$, במקרה הגרוע, מספר צלעות מקסימאלי הוא בגרף שלם, כאשר כל קדקוד כשור לכל קדקוד אחר בגרף ושווה ל- $m = n(n-1)/2$. לכן סיבוכיות של מיון צלעות במקרה הגרוע שווה

$$O(mlog_2m) = O(mlog_2m) = (mlog_2n^2) = O(2mlog_2n) = O(mlog_2n)$$

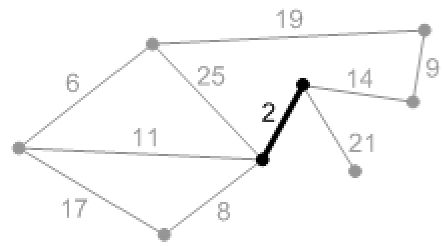
אז סיבוכיות של קרסקל היא $\boldsymbol{O(m \cdot log_2n) + O(m \cdot \alpha(n)) = O(m \cdot log_2n)}$

כאשר אלגוריתם של קרוסקל מקבל מערך הצלעות ממוין, הסיבוכיות של האלגוריתם הופכת ל-$\boldsymbol{O(m \cdot \alpha(n))}$

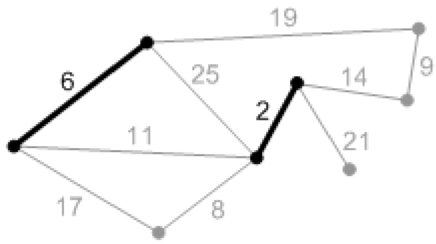
<u>דוגמא:</u>



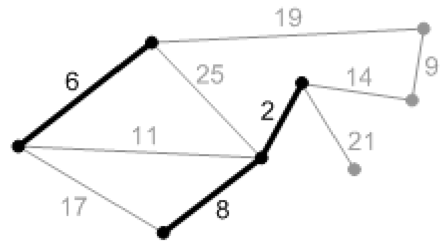
a)



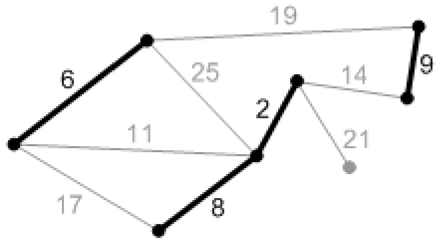
b)



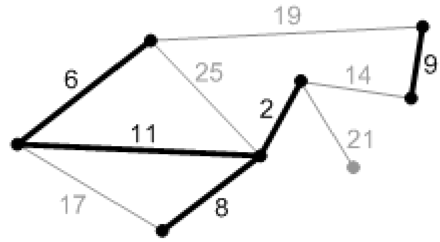
c)



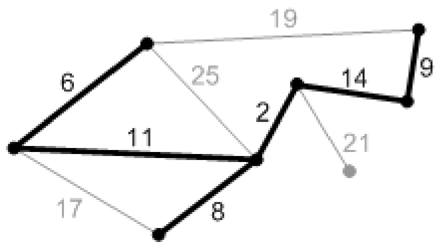
d)



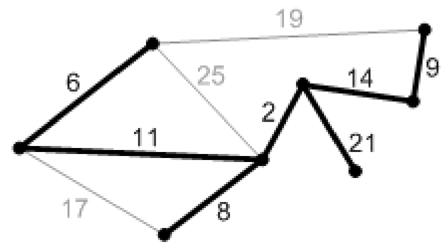
e)



f)



g)



h)

min sum = 71

<u>נכונות של אלגוריתם קרסקל</u>

נוכיח את נכונות של אלגוריתם קרסקל בדרך השלילה. יהיה T עץ שנבנה בעזרת אלגוריתם קרסקל, S עץ פורש מינימאלי ומתקיים אי-שוויון: weight(S) < weight(T), כאשר weight(S) - משקל כולל של S.

מכוון שבגרף יכול להיות מספר עצים פורשים מינימאליים, ניקח כ-S עץ פורש מינימאלי כך שמספר k צלעות משותפות עם T הוא מקסימאלי בין כל עצים פורשים מינימאליים. יהיה

$$T = \{e_1, e_2, \ldots, e_{i-1}, e_i, \ldots, e_{n-1}\}$$
$$S = \{e_1, e_2, \ldots, e_{i-1}, f_i, \ldots e_p, \ldots, f_{n-1}\}$$

סדרת צלעות שמתקבלת אחרי הרצת אלגוריתם של קרסקל, (ברור שהסדרה ממוינת בסדר לא יורד).

יהי $e_i = (a,b)$ הצלע הראשונה בסדרה זו שלא שייכת ל-S, $(S \not\ni e_i)$ (צלעות $e_1, e_2, \ldots, e_{i-1}$ שייכות ל-S).

נוסיף את צלע e_i לעץ S, נקבל גרף חדש $S_1 = S \cup e_i$ בעל n צלעות, אזי ב-S_1 יש מעגל C. מכוון ש-T הוא עץ, במעגל C קיימת צלע e_p שלא שייכת ל- T ($e_p \notin T$). נשווה את משקלי הצלעות e_p ו- e_i : $e_p \in S$ לכן היא לא סוגרת מעגלים עם צלעות $e_1, e_2, \ldots, e_{i-1}$, לכן היא יכולה להיות מועמד לאלגוריתם של קרסקל בשלב i , אבל בשלב i בחרנו ב-e_i ולא ב-e_p , לכן

$$\text{weight}(e_i) \le \text{weight}(e_p).$$

נסיר צלע e_p מגרף S_1, נקבל עץ

$$S_2 = S_1 - \{e_p\} = S \cup \{e_i\} - \{e_p\}.$$

עץ S_2 התקבל מ- S ויש בו $\boldsymbol{i}$ צלעות משותפות עם T. נשים לב כי

$$\text{weight}(S_2) = \text{weight}(S) + \text{weight}(e_i) - \text{weight}(e_p) \le \text{weight}(S)$$

אבל S הוא עץ פורש מינימאלי, המשקל שלו קטן ביותר בגרף G,לכן

$$\text{weight}(S_2) = \text{weight}(S)$$

ו- S_2 הוא גם עץ פורש מינימאלי ויש בו $\boldsymbol{i}$ צלעות משותפות עם T . הגענו לסתירה עם בחירת S, כעץ פורש מינימאלי שמספר צלעותיו משותפות עם T מקסימאלי ושווה ל- $i-1$. מש"ל.

איחוד קבוצות זרות Disjoint-Set Data Structure

במדעי המחשב, **איחוד קבוצות זרות** (**Disjoint-Set Data Structure**), הוא מבנה נתונים אשר מבצע מעקב אחרי קבוצה של אובייקטים המחולקים למספר של תתי-קבוצות זרות ולא חופפות.

מבנה נתונים של קבוצות זרות מאוד נוח לאחסון של רכיבי קשירות בגרף. לדוגמה, האלגוריתם של קרוסקל דורש מבנה נתונים כזה ליישום יעיל.

הפעולות המוגדרות על מבנה זה הן:

1. **יצירה** (*MakeSet*), פעולה היוצרת קבוצה חדשה המכילה אובייקט אחד בלבד (סינגלטון).
2. **חיפוש** *(Find)*: קביעה איזו קבוצה מכילה אובייקט ספציפי. פעולה זו יכולה גם לעזור בקביעה האם שני אובייקטים שייכים לאותה הקבוצה.
3. **איחוד** *(Union)*: איחוד שתי קבוצות לקבוצה אחת.

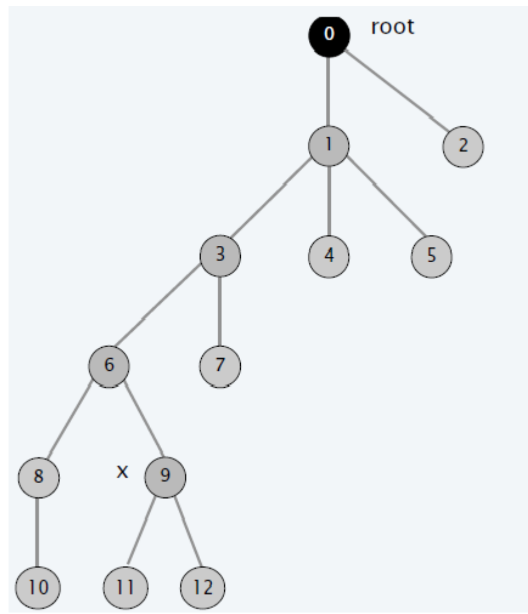
תצוגה: כל קבוצה מיוצגת על ידי עץ, שורש העץ נחשב כנציג של הקבוצה. כל קדקוד מחזיק את המצביע לצומת האב **שלו** (parent node).

Find הולך לפי האבות עד שמגיע לשורש העץ.

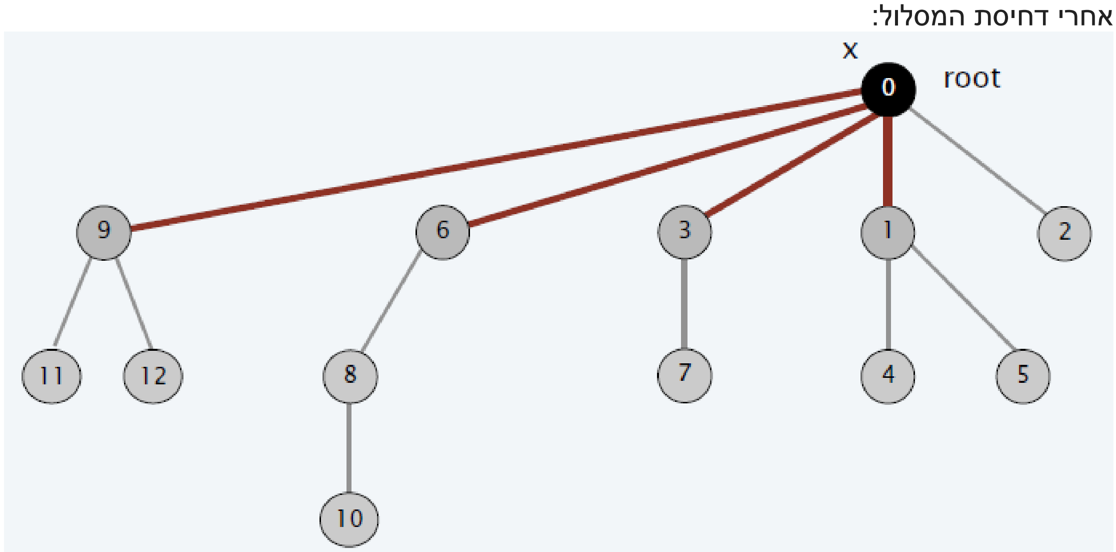
Union משלב שני עצים לתוך אחד על ידי הצמדת שורש אחד לשורש של עץ אחר.

העץ שנוצר לאחר איחוד של שני עצים יכול להיות מאוד לא מאוזן, וחיפוש יהיה מאוד לא יעיל.
אחת מדרכים ליצירת עץ מאוזן יותר נקראת **איגוד לפי דרגה** (**union by rank**). השיטה מחזיקה דרגה לכל קדקוד (הערך ההתחלתי הוא 0) ומצרפת את העץ הקטן לשורש של העץ הגדול.
פונקציית Find(x) מקיימת גם **path compression**, כלומר לאחר שנמצא שורש העץ (root) שמכיל את קדקוד x היא משנה קדקודי האבות של קדקודים הנמצאים במסלול מ-x לשורש ישירות ל-root.

לפני דחיסת המסלול:

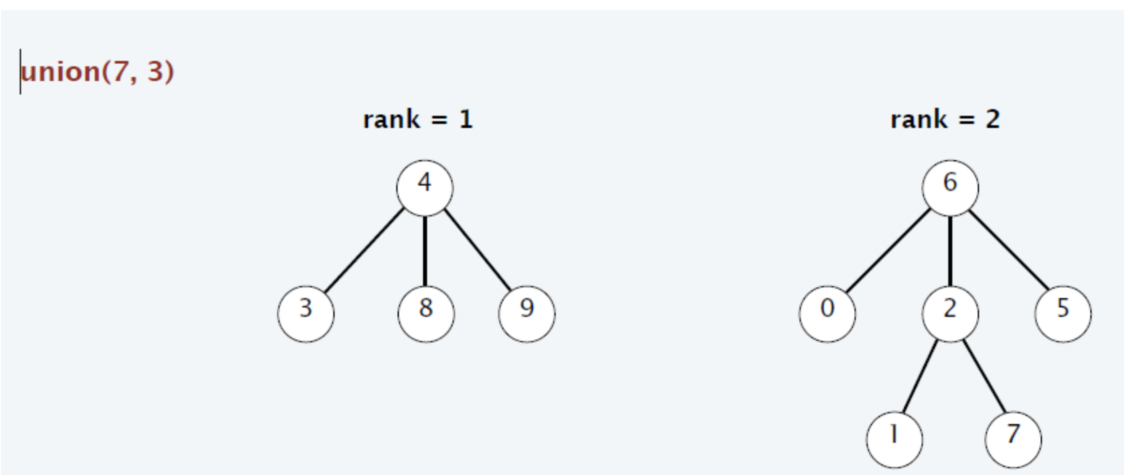


אחרי דחיסת המסלול:

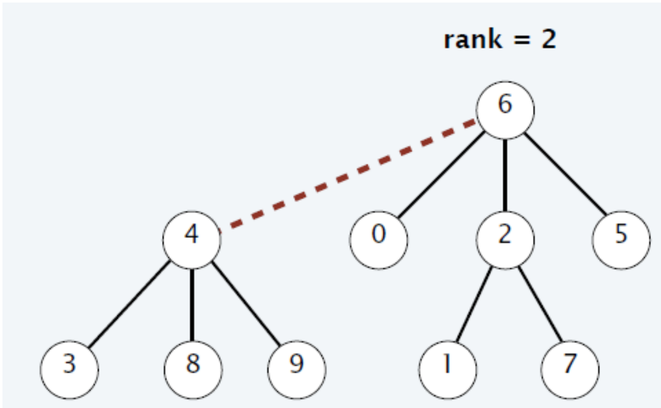


פסאודו-קוד לפונקציות *MakeSet*, *Find* ו-*Union* :

```
MakeSet(x)//O(1)
     x.parent = x
     x.rank   = 0


 Union(x, y)// O(α(n))
     xRoot = Find(x)
     yRoot = Find(y)
     if xRoot == yRoot
         return

      // x and y are not already in same set. Merge them.
     if xRoot.rank < yRoot.rank
         xRoot.parent = yRoot
     else if xRoot.rank > yRoot.rank
         yRoot.parent = xRoot
     else
         yRoot.parent = xRoot
         xRoot.rank = xRoot.rank + 1

 Find(x)// O(α(n))
 // Ackerman של היא פונקציה הפוכה לפונקציה   α(n)
 //O(1)  שזה כמעט
     if x.parent != x
        x.parent = Find(x.parent)
     return x.parent
```



rank = height.



rank=height

union by rank – תכונות:

1. כאשר **x** הוא לא שורש, אזי rank(x) < rank(parent(x)).
הוכחה: קדקוד בדרגה k נוצר בעזרת מיזוג של שני שורשים בדרגה k-1 בלבד.

2. כאשר **x** הוא לא שורש, אזי rank(x) לעולם לא ישתנה שוב.
הוכחה: דרגת הקדקוד משתנה לשורשים בלבד, קדקוד שהוא לא שורש לעולם לא יהפוך לשורש.

3. לכל שורש בדרגה k יש לפחות $n \geq 2^k$ קדקודים בעץ שלו.
הוכחה באינדוקציה:
בסיס האינדוקציה: נכון עבור k=0.
הנחת אינדוקציה: הטענה נכונה עבור k-1.
שלב אינדוקציה: קדקוד בדרגה k נוצר בעזרת מיזוג של שני שורשים בדרגה k-1 בלבד. לפי הנחת אינדוקציה כל תת-עץ מורכב מלפחות 2^{k-1} צמתים, לכן לאחר מיזוג לעץ יש לפחות $n \geq 2^k$ קדקודים.

4. הדרגה הגבוהה ביותר של קדקוד היא קטנה או שווה $\log_2 n$.
הוכחה: $2^k \leq n$ לכן $k \leq \log_2 n$.