Minimum Spanning Tree - Boruvka Algorithm

האלגוריתם של Boruvka הוא אלגוריתם **חמדני** המשמש למציאת עץ פורש מינימלי בגרף G(E,V) קשיר, משוקלל לא מכוון. האלגוריתם מתחיל את בניית העץ מבניית יער, שכל עץ בו מורכב מקדקוד אחד, כלומר היער מורכב מ- n=|V| עצים. צעד האלגוריתם מוסיף לעץ את הצלע בעלת המשקל המינימלי (צלע בטוחה) מבין אלה היוצאות מקדקוד.

יותר פורמאלי:

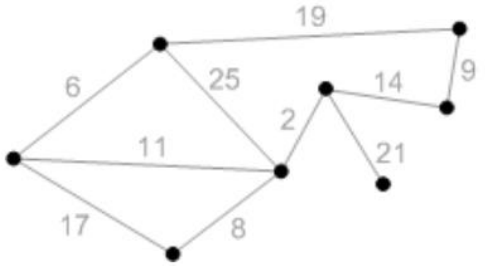
1. בונים יער, המורכב מ-n עצים. כל עץ מורכב הקדקוד אחד ועדיין לא מכיל אף צלע. כל קדקוד כזה מהווה שורש של עץ והוא הנציג של העץ.

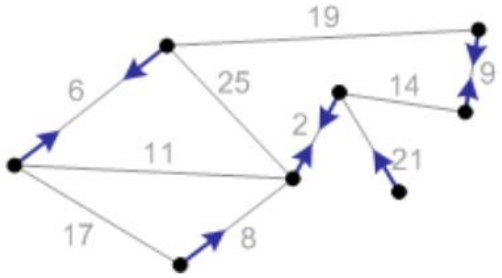
   T={{0}, {1}, . . . , {n}}

2. לכל עץ מוסיפים צלע בטוחה הסמוכה לשורש. יכול להיות שאותה צלע מוסיפים לשורשים שונים. האלגוריתם מונע את המצב וכל צלע נכנסת לעץ פעם אחד בלבד.

3. מחברים עצים (union) ע"י צלע בעלת משקל מינימאלי (צלע בטוחה).

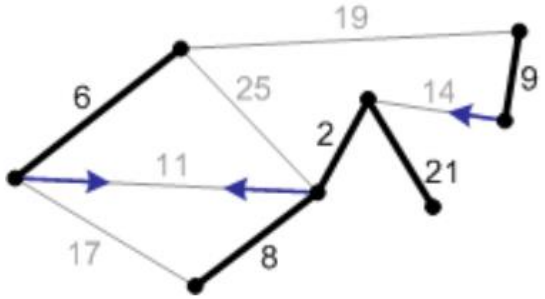
4. כל עוד מספר עצים גדול מ-1 חוזרים לשלב 3.

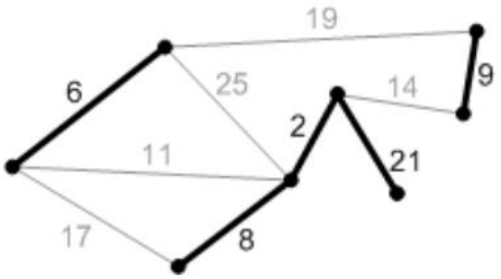




1. בשלב הראשון בונים יער המורכב מעצים, כך שכל מכל עץ מחיל קדקוד אחד בלבד וקבוצה ריקה של צלעות.

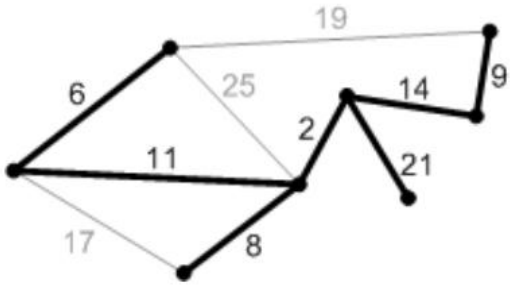
2. לכל עץ (לכל קדקוד) מוסיפים צלע סמוכה עליו בעלת משקל מינימאלי (צלע בטוחה).





3. מוסיפים צלעות בטוחות לעץ פורש מינימאלי.

4. לכל עץ מוצעים צלע בטוחה נוספת.



5 . מוסיפים את הצלעות הבטוחות שמצאנו אותן בשלב 4 לעץ פורש מינימאלי ובז האלגוריתם מסתיים.

Boruvka pseudo code

```
BoruvkaMST(Edge[] graph, int n){//n-number of vertices, m-number of edges

        UnionFind uf = new UnionFind(n)
        mst = new ArrayList<>(n-1)

        while (mst.size() < n-1) {//O(log(n))
        // for-each tree in forest, find closest edge
        // if edge weights are equal, ties are broken in favor of first edge in E
                Edge[] closest = new Edge[n]
                for (Edge e : graph) {//O(|E|)
                        v = e.v
                        u = e.u
                        vRoot = uf.find(v)
                        uRoot = uf.find(u)
                        if (vRoot == uRoot) continue   // same tree
                        if (closest[vRoot] == null || e.weight < closest[vRoot].weight)
                                closest[vRoot] = e
                        if (closest[uRoot] == null || e.weight < closest[uRoot].weight)
                                closest[uRoot] = e
                end-for

                // add newly discovered edges to MST
                for (int i = 0; i < n; i++) {//O(n)
                        Edge e = closest[i]
                        if (e != null) {
                                v = e.v
                                u = e.u
                                // don't add the same edge twice
                                if (!uf.connected(v, u)) {
                                        mst.add(e)
                                        weight = weight + e.weight;
                                        uf.union(v, u)
                                end-if
                        end-if
                end-for
        end-while
end-BoruvkaMST
```

סיבוכיות האלגוריתם: בלולאת while עוברים לכל היותר $\log_2(n)$ פעמים, בגלל שבמקרה הגרוע מספר עצים בכל איטרציה מתחלק ב-2, לולאת for ראשונה עוברת על כל הצלעות, לכן יש $m*\log_2 n$ ואורך של לולאת for השנייה n. מקבלים:

$$O((m+n)*\log_2 n) = O(m*\log_2(n))$$