גרף מכוון קשיר: גרף מכוון G עבורו לכל $u,v\in V\left(G\right)$ קיים מסלול מ־u ל־v או מסלול מ־v ל־u.

גרף מכוון קשיר חזק: גרף מכוון G עבורו לכל $u,v\in V\left(G\right)$ קיים מסלול מ־u ל־v.

אלגוריתם BFS: יהי G גרף ויהי $s\in V\left(G\right)$ אזי

```
function BFS(G, s):
    (d, π, color) ← dict(V (G))
    for u ∈ V (G) \ {s} do
        color[u] ← White
        d[u] ← ∞
        π[u] ← Null
    end
    color[s] ← Grey
    d[s] ← 0
    π[s] ← Null
    Q ← queue()
    while Q ≠ ∅ do
        u ← Q.head
        for v ∈ Neighbor(u) do
            if color[v] = White then
                color[v] ← Grey
                d[v] ← d[u] + 1
                π[v] ← u
                Q.enqueue(v)
            end
        end
        Q.dequeue()
        color[u] ← Black
    end
    return (d, π, color)
```

טענה: יהי G גרף ויהי $s\in V\left(G\right)$ אזי סיבוכיות זמן הריצה של $\text{BFS}\left(G,s\right)$ הינה $\mathcal{O}\left(|V|+|E|\right)$.

משפט: יהי G גרף ויהי $s\in V$ אזי $\left\{v\in V\mid \text{BFS}\left(G,s\right).\text{color}\left[v\right]=\text{Black}\right\}=\left[s\right]_{\to}$.

סימון: יהי G גרף ויהיו $u,v\in V$ אזי $\delta\left(v,u\right)=\min\left(\left\{\text{len}\left(\sigma\right)\mid \sigma\text{ טיול בין }v,u\right\}\right)$.

טענה: יהי G גרף ויהיו $v,u,w\in V$ באשר $\left(w,u\right)\in E$ אזי $\delta\left(v,u\right)\le\delta\left(v,w\right)+1$.

למה: יהי G גרף ויהיו $s,v\in V$ אזי בכל שלב בהרצת $\text{BFS}\left(G,s\right)$ מתקיים $d\left[v\right]\ge\delta\left(v\right)$.

למה: יהי G גרף יהי שלב בהרצת $\text{BFS}\left(G,s\right)$ בו $Q=\left(v_1\ldots v_n\right)$ אזי מתקיים $d\left[v_i\right]\le d\left[v_{i+1}\right]$ וכן $d\left[v_i\right]\le d\left[v_1\right]+1$.

משפט נכונות מרחקים: יהי G גרף ויהיו $s,v\in V$ אזי $\text{BFS}\left(G,s\right).d\left[v\right]=\delta\left(v,s\right)$.

עץ BFS: יהי G גרף ויהי $s\in V$ נגדיר $V_\pi=\left\{v\in V\mid \text{BFS}\left(G,s\right).\pi\left[v\right]\neq\text{Null}\right\}\cup\left\{s\right\}$ וכן $E_\pi=\left\{\left(\pi\left[v\right],v\right)\mid v\in V_\pi\backslash\left\{s\right\}\right\}$ אזי $G_\pi=\left(V_\pi,E_\pi\right)$.

טענה: יהי G גרף ויהי $s\in V$ אזי

- מתקיים $\deg^-_{G_\pi}\left(s\right)=0$.
- לכל $v\in V\left(G_\pi\right)$ מתקיים $\deg^-_{G_\pi}\left(v\right)=1$.
- לכל $v\in V\left(G_\pi\right)$ קיים מסלול ב־G_π בין s,v.
- G_π הינו עץ.
- יהי $v\in V\left(G_\pi\right)$ ויהי σ מסלול ב־G_π בין s,v אזי σ המסלול הקצר ביותר בין s,v ב־G.

מסלול אוילר: מסלול העובר על כל הקשתות בגרף.

מעגל אוילר: מסלול אוילר שהינו מעגל.

טענה: יהי G גרף קשיר ולא מכוון אזי (יש מעגל אוילר ב־G)$\Longleftrightarrow$(לכל $v\in V$ מתקיים $\deg\left(v\right)\in\mathbb{N}_{\text{even}}$).

אלגוריתם למציאת מעגל אוילר: יהי G גרף קשיר ולא מכוון עבורו לכל $u\in V$ מתקיים $\deg\left(u\right)\in\mathbb{N}_{\text{even}}$ אזי

```
function EulerCircle(G, v):
    σ ← List(E(G))
    u ← Neighbor(v)
    while u ≠ v do
        σ.append({v, u})
        G = G\{{v, u}}
        u ← Neighbor(u)
    end
    if length(σ) = |E(G)| then
        return σ
    end
    else
        w ← {x ∈ V(G) | (∃y ∈ V(G).(x, y) ∈ σ) ∧ (deg(x) > 0)}
        σ[w] = EulerCircle(G, w)
    end
    return σ
```

טענה: יהי G גרף קשיר ולא מכוון עבורו לכל $u \in V$ מתקיים $\deg(u) \in \mathbb{N}_{\text{even}}$ ויהי $v \in V(G)$ אזי סיבוכיות זמן הריצה של $\text{EulerCircle}(G, v)$ הינה $\mathcal{O}(|E|)$.

טענה: באלגוריתם EulerCircle כל עוד לולאת ה־while פעילה מתקיים $|\text{Neighbor}(u)| \neq \varnothing$.

משפט: יהי G גרף קשיר ולא מכוון עבורו לכל $v \in V$ מתקיים $\deg(v) \in \mathbb{N}_{\text{even}}$ אזי $\text{EulerCircle}(G)$ הינו מעגל אוילר.

טענה: יהי G גרף קשיר ולא מכוון אזי (יש מסלול אוילר שאינו מעגל ב־G)$\Longleftrightarrow$($|\{v \in V(G) \mid \deg(v) \in \mathbb{N}_{\text{odd}}\}| = 2$).

אלגוריתם למציאת מסלול אוילר: יהי G גרף קשיר ולא מכוון עבורו $|\{v \in V(G) \mid \deg(v) \in \mathbb{N}_{\text{odd}}\}| = 2$ אזי

```
function EulerPath(G):
    {v, u} ← {v ∈ V(G) | deg(v) ∈ N_odd}
    G = G + {{v, u}}
    σ = EulerCircle(G, v)
    return σ\{v, u}
```

טענה: יהי G גרף לא מכוון אזי (G דו־צדדי)$\Longleftrightarrow$(לא קיים ב־G מעגל באורך אי־זוגי).

אלגוריתם זיהוי גרפים דו־צדדיים: יהי G גרף לא מכוון ופשוט אזי

```
function IsBipartite(G):
    (d, π, color) ← BFS(G)
    for (v, u) ∈ V do
        if d(v) = d(u) then
            return false
        end
    end
    return true
```

טענה: יהי G גרף לא מכוון ופשוט אזי (G דו צדדי)$\Longleftrightarrow$($\text{IsBipartite}(G) = \text{true}$).

מסלול קצר ביותר בין קודקודים: יהי G גרף ויהיו $s, t \in V$ אזי מסלול σ מ־s ל־t עבורו $|\sigma| = \min\{|\tau| \mid t\text{־}s \text{ מסלול מ־} \tau\}$.

גרף המסלולים הקצרים ביותר מקודקוד (גרף מק"ב): יהי G גרף ויהי $s \in V$ נגדיר $E' = \{e \in E \mid s\text{־חלק ממסלול קצר ביותר היוצא מ} \ e\}$ אזי (V, E').

אלגוריתם למציאת גרף המסלולים הקצרים ביותר מקודקוד: יהי G גרף ויהי $s \in V$ אזי

```
function ShortestPathGraph(G, s):
    (d, π, color) ← BFS(G)
    E' ← E(G_π)
    for (u, v) ∈ E(G) do
        if |height_{G_π}(u) − height_{G_π(v)}| = 1 then
            E'.append((u, v))
        end
    end
    return (V(G), E')
```

טענה: תהא $e\in E$ אזי (e מחברת בין רמות עוקבות ביער BFS G_{π})$\Longleftrightarrow$(e קשת במק"ב).

מסקנה: יהי G גרף ויהי $s\in V$ אזי $\text{ShortestPathGraph}\left(G,s\right)$ הינו גרף מק"ב מ־s.

גרף המסלולים הקצרים ביותר בין קודקודים: יהי G גרף ויהיו $s,t\in V$ נגדיר $E'=\{e\in E\mid$ חלק ממסלול קצר ביותר היוצא מ־s ל־t $e\}$ אזי (V,E').

טענה: יהי G גרף מכוון ויהיו $s,t\in V$ אזי קיים אלגוריתם לחישוב גרף המסלולים הקצרים ביותר מ־s ל־t בסיבוכיות $\mathcal{O}\left(|V|+|E|\right)$.

אלגוריתם DFS: יהי G גרף ויהי $s\in V$ אזי

```
function DFS(G, s):
    (k, π) ← dict(V)
    color ← dict(E)
    k[s] ← 1
    π[s] ← Null
    for u ∈ V\s do
        k[u] ← 0
        π[u] ← Null
    end
    for e ∈ E do
        color[e] ← White
    end
    i ← 2
    v ← s
    while (∃u ∈ Adj(v).color[(v, u)] = White) ∨ (π[v] ≠ Null) do
        if {u ∈ Adj(v) | color[(v, u)] = White} ≠ ∅ then
            w ← {u ∈ Adj(v) | color[(v, u)] = White}
            color[(v, w)] ← Black
            if k[w] = 0 then
                k[w] ← i
                π[w] ← v
                v ← w
                i ← i + 1
            end
        else
            v ← π[v]
        end
    end
    return (k, π)
```

טענה: יהי G גרף ויהי $s\in V$ אזי סיבוכיות זמן הריצה של $\text{DFS}\left(G,s\right)$ הינה $\mathcal{O}\left(|V|+|E|\right)$.

זמן גילוי: יהי G גרף ויהי $s\in V\left(G\right)$ אזי k בהרצת $\text{DFS}\left(G,s\right)$.

טענה: יהי G גרף ויהיו $s,v\in V$ באשר $v\in[s]_{\rightarrow}$ אזי בהרצת $\text{DFS}\left(G,s\right)$ מתקיים $k\left[v\right]>0$.

עץ DFS: יהי G גרף ויהי $s\in V$ נגדיר $V_{\pi}=\{v\in V\mid \text{DFS}\left(G,s\right).\pi\left[v\right]\neq\text{Null}\}\cup\{s\}$ וכן $E_{\pi}=\{\left(\pi\left[v\right],v\right)\mid v\in V_{\pi}\backslash\{s\}\}$ אזי $G_{\pi}=\left(V_{\pi},E_{\pi}\right)$.

טענה: עץ DFS הינו עץ.

קשתות ביחס לריצת DFS: יהי G גרף ויהי G_π יער DFS אזי

- קשתות עץ: קשת $e \in E(G)$ עבורה $e \in E(G_\pi)$.
- קשתות קדמיות: קשת $(u,v) \in E(G)$ עבורה $(u,v) \notin E(G_\pi)$ וכן u הינו אב של v.
- קשתות אחוריות: קשת $(u,v) \in E(G)$ עבורה $(u,v) \notin E(G_\pi)$ וכן v הינו אב של u.
- קשתות חוצות: קשת $e \in E(G)$ שאינה קשת עץ או קדמית או אחורית.

טענה: יהי G גרף לא מכוון ותהא (u,v) קשת עץ אזי u צאצא של v בגרף G_π או v צאצא של u בגרף G_π.

מסקנה: יהי G גרף לא מכוון אזי לא קיימות קשתות חוצות.

אלגוריתם DFS בעל זמני נסיגה: יהי G גרף אזי

```
function DFS(G):
    (k, f, π, color, low) ← dict(V)
    for u ∈ V do
        k[u] ← 0
        π[u] ← Null
        color ← White
        low ← ∞
    end
    i ← 0 for s ∈ V do
        if k[s] = 0 then
            DFS-VISIT(s, k, f, π, i)
        end
    end
    return (k, f, π, low)
```

```
function DFS-VISIT(v, k, f, π, color, low, i):
    color[u] ← Gray
    i ← i + 1
    k[v] ← i
    for w ∈ Adj(v) do
        if (color[v] = Gray) ∧ (v ≠ π[u]) then
            low ← min(low[u], k[v])
        else if color[v] = White then
            π[w] ← v
            DFS-VISIT(w, k, f, π, color, low, i)
            low ← min(low[u], low[v])
    end
    color[u] ← Black
    i ← i + 1
    f[v] ← i
```

זמן נסיגה: יהי G גרף ויהי $s \in V(G)$ אזי f בהרצת $\mathrm{DFS}(G)$.

טענה Gray Path Lemma: יהיו $v,u \in V$ אזי (v צאצא של u ביער G_π)$\iff$($k[u] < k[v] < f[u]$).

טענה: יהיו $v,u \in V$ אזי ((u,v) קשת חוצה)$\iff$($f[v] < k[u]$).

משפט הסוגריים: יהי G גרף ויהיו $u,v \in V$ אזי מתקיים בדיוק אחד מהבאים

- מתקיים $[k(u), f(u)] \cap [k(v), f(v)] = \varnothing$ וכן u,v אינם צאצא־אב ביער G_π.
- מתקיים $[k(u), f(u)] \subset [k(v), f(v)]$ וכן u צאצא של v ביער G_π.
- מתקיים $[k(u), f(u)] \supset [k(v), f(v)]$ וכן v צאצא של u ביער G_π.

משפט המסלול הלבן: יהי G גרף ויהיו $u,v \in V$ אזי (v צאצא של u ביער G_π)$\iff$(בזמן $k(u)$ באלגוריתם $\mathrm{DFS}(G)$ יש מסלול לבן מ־u ל־v).

גרף מכוון אציקלי: גרף מכוון G בו לא קיים מעגל.

מיון טופולוגי: יהי G גרף מכוון אזי יחס סדר $\prec$ על V המקיים לכל $u,v\in V$ אם $(u,v)\in E$ אזי $u\prec v$.

משפט: יהי G גרף מכוון אזי (G אציקלי)$\Longleftrightarrow$(קיים מיון טופולוגי על G).

טענה אלגוריתם קנות': יהי G גרף מכוון אזי קיים אלגוריתם לבדיקת קיום מיון טופולוגי בסיבוכיות $\mathcal{O}(|V|+|E|)$.

משפט: יהי G גרף מכוון אזי (G רציקלי)$\Longleftrightarrow$(אין קשתות אחוריות ב־G).

טענה: יהי G גרף מכוון אציקלי אזי f המתקבלת מהרצת $\text{DFS}(G)$ משרה מיון טופולוגי על G.

קודקוד מנתק: יהי G גרף מכוון אזי $v\in V(G)$ עבורו $\left|G/\underset{G-\{v\}}{\longrightarrow}\right| \le \left|G/\underset{G}{\longrightarrow}\right|$.

אב חורג: יהי G גרף מכוון ויהי $v\in V$ אזי $w\in V$ עבורו (w,v) קשת אחורית.

זמן גילוי האב החורג המוקדם ביותר: יהי G גרף אזי low בהרצת $\text{DFS}(G)$.

אלגוריתם למציאת כל הקודקודים המנתקים: יהי G גרף מכוון וקשיר אזי

```
function DetachableVertices(G):
    s ← V
    (k, f, π, low) ← DFS(G, s)
    A ← set(V)
    if |Adj_{G_π}(s)| ≠ 1 then
        A.append(s)
    end
    for u ∈ V\{s} do
        if ∃v ∈ children(u).low[v] ≥ k[u] then
            A.append(u)
        end
    end
    return A
```

טענה: יהי G גרף מכוון וקשיר אזי סיבוכיות $\text{DetachableVertices}(G)$ הינה $\mathcal{O}(|V|+|E|)$.

טענה: יהי G גרף מכוון וקשיר אזי $\text{DetachableVertices}(G)$ הינה קבוצת כל הקודקודים המנתקים.

רכיב קשיר היטב (רק"ה): יהי G גרף מכוון אזי קבוצה $C\subseteq V$ מקסימלית בגודלה עבורה לכל $u,v\in C$ קיים מסלול מ־u ל־v וכן מ־v ל־u.

גרף הופכי/משוחלף: יהי G גרף מכוון נגדיר $E'=\{(v,u)\mid(u,v)\in E\}$ אזי $G^T=(V,E')$.

טענה: יהי G גרף מכוון ותהא $C\subseteq V$ אזי (C רק"ה של G)$\Longleftrightarrow$(C רק"ה של G^T).

אלגוריתם למציאת רכיבים קשירים היטב: יהי G גרף מכוון אזי

```
function SCC(G):
    (k, f, π) ← DFS(G)
    /* In the next line, in the main loop of DFS we go through the vertices in decreasing order of f[u]    */
    (k', f', π') ← DFS(G^T)
    A ← set(set(V))
    for v ∈ V do
        A.append([v]_{→G_π^T})
    end
    return A
```

גרף הרכיבים: יהי G גרף מכוון נגדיר $E^*=\left\{(A,B)\in\text{SCC}(G)^2\mid\exists u\in A.\exists v\in B.(u,v)\in E\right\}$ אזי $G^*=(\text{SCC}(G),E^*)$.

אלגוריתם למציאת גרף הרכיבים: יהי G גרף מכוון אזי

```
function ComponentGraph(G):
    V* ← SCC(G)
    E* ← set((V*)^2)
    for (u,v) ∈ E do
        if [v]_{G_π^T →} ≠ [u]_{G_π^T →} then
            E*.append(([v]_{G_π^T →}, [u]_{G_π^T →}))
        end
    end
    return (V*, E*)
```

למה: יהי G גרף מכוון אזי G^* אציקלי.

הגדרה: יהי G גרף ותהא $U \subseteq V$ אזי

- זמן גילוי: $k\left(U\right)=\min_{u\in U}\left(k\left[u\right]\right)$.
- זמן נסיגה: $f\left(U\right)=\max_{u\in U}\left(f\left[u\right]\right)$.

למה: יהי G גרף מכוון ויהיו $C_1,C_2\subseteq V$ רק"ה באשר $\left(C_1,C_2\right)\in E\left(G^*\right)$ אזי $f\left(C_2\right)<f\left(C_1\right)$.

מסקנה: יהי G גרף מכוון ויהיו $C_1,C_2\subseteq V$ רק"ה באשר $\left(C_1,C_2\right)\in E\left(\left(G^T\right)^*\right)$ אזי $f\left(C_2\right)>f\left(C_1\right)$.

משפט: יהי G גרף מכוון ויהי $C\subseteq V$ אזי (C רק"ה)$\Longleftrightarrow$($C\in \mathsf{SCC}\left(G\right)$).

קבוצת מוצא: יהי G גרף מכוון אזי $S\subseteq V$ המקיימת $\forall v\in V.\exists s\in S.s\to v$.

אלגוריתם למציאת קבוצת מוצא מינימלית: יהי G גרף מכוון אזי

```
function MinimalOriginSet(G):
    A ← set(V(G))
    G* ← ComponentGraph(G)
    for C ∈ V(G*) do
        v ← {u ∈ C | ∄w ∈ V(G)\C.(w,u) ∈ E(G)}
        A.append(v)
    end
    return A
```

טענה: יהי G גרף מכוון אזי $\text{MinimalOriginSet}\left(G\right)$ קבוצת מוצא מינימלית.

טענה: יהי G גרף מכוון וקשיר אזי סיבוכיות $\text{MinimalOriginSet}\left(G\right)$ הינה $\mathcal{O}\left(\left|V\right|+\left|E\right|\right)$.

טענה: יהי G גרף מכוון ותהא $S\subseteq V$ אזי קיים אלגוריתם הבודק האם קיים הילוך σ העובר על S בסיבוכיות $\mathcal{O}\left(\left|V\right|+\left|E\right|\right)$.

גרף ממושקל: יהי G גרף ותהא $w:E\to\mathbb{R}$ אזי (G,w).

עץ פורש: יהי G גרף קשיר לא מכוון אזי תת־גרף $T\leq G$ באשר T עץ וכן $V\left(T\right)=V\left(G\right)$.

משקל עץ פורש: יהי G גרף קשיר לא מכוון ויהי $T\leq G$ עץ פורש אזי $w\left(T\right)=\sum_{e\in E\left(T\right)}w\left(e\right)$.

עץ פורש מינימלי (עפ"מ): יהי G גרף קשיר לא מכוון אזי עץ פורש $T\leq G$ עבורו $w\left(T\right)=\min\left\{ w\left(S\right)\mid G\text{שלעץ פורש }S\right\}$.

חתך: יהי G גרף אזי $A,B\subseteq V\left(G\right)$ עבורם $A\uplus B=V\left(G\right)$.

קשתות החתך: יהי G גרף ויהי $A,B\subseteq V\left(G\right)$ חתך אזי $\left\{ \left(u,v\right)\in E\left(G\right)\mid\left(u\in A\right)\wedge\left(v\in B\right)\right\}$.

טענה: יהי $G\leq T$ עץ פורש ותהא $e\in E\left(G\right)\backslash E\left(T\right)$ אזי $T+\left\{ e\right\}$ בעל מעגל יחיד.

טענה: יהי $G\leq T$ עץ פורש תהא $e_1\in E\left(G\right)\backslash E\left(T\right)$ ותהא $e_2\in E\left(T+\left\{ e_1\right\}\right)$ אשר הינה חלק ממעגל אזי $T+\left\{ e_1\right\}-\left\{ e_2\right\}$ עץ פורש.

טענה: יהי $G\leq T$ עץ פורש ותהא $e\in E\left(T\right)$ אזי $T-\left\{ e\right\}$ הינו יער בעל שני עצים.

מסקנה: יהי $G\leq T$ עץ פורש תהא $e\in E\left(T\right)$ ויהי $v\in V\left(G\right)$ אזי $V\left(G\right)\backslash\left[v\right]_{\xrightarrow[T-\{e\}]{}},\left[v\right]_{\xrightarrow[T-\{e\}]{}}$ חתך של G.

אלגוריתם גנרי למציאת עץ פורש מינימלי: יהי G גרף קשיר לא מכוון וממושקל w אזי

```
function MST(G, w):
    color ← dict(E)
    for e ∈ E do
        color[e] = White
    end
    while ∃e ∈ E.color[e] = White do
        Blueless ← {A ⊆ V | ∀e ∈ (A^2 ∩ E).color[e] ≠ Blue}
        Redless ← {σ circle in G | ∀i ∈ [len(σ)].color[(σ[i], σ[i+1])] ≠ Red}
        if Blueless ≠ ∅ then
            A ← Blueless
            f ← argmin_{e∈A^2∩E}(w(e))
            color[f] = Blue
        end
        if Redless ≠ ∅ then
            σ ← Redless
            f ← argmax_{e∈σ}(w(e))
            color[f] = Red
        end
    end
    return (V, {e ∈ E | color[e] = Blue})
```

טענה: יהי G גרף קשיר לא מכוון וממושקל w ותהא $a \in E$ עבורה $\text{color}\left[a\right] = \text{White}$ באיטרציה של $\text{MST}\left(G\right)$ אזי קיימת $e \in E$ אשר ניתנת לצביעה.

מסקנה: יהי G גרף קשיר לא מכוון וממושקל w אזי $\text{MST}\left(G\right)$ צובעת $|E|$ קשתות.

טענה: יהי G גרף קשיר לא מכוון וממושקל w אזי בכל איטרציה של $\text{MST}\left(G\right)$ קיים $T \leq G$ עפ"מ עבורו

- לכל $e \in E$ המקיימת $\text{color}\left[e\right] = \text{Blue}$ מתקיים $e \in E\left(T\right)$.
- לכל $e \in E$ המקיימת $\text{color}\left[e\right] = \text{Red}$ מתקיים $e \notin E\left(T\right)$.

מסקנה: יהי G גרף קשיר לא מכוון וממושקל w אזי $\text{MST}\left(G\right)$ עפ"מ של G.

אלגוריתם פרים למציאת עץ פורש מינימלי: יהי G גרף קשיר לא מכוון וממושקל w אזי

```
function Prim'sAlgorithm(G):
    color ← dict(E)
    U ← set(V)
    for e ∈ E do
        color[e] = White
    end
    r ← V
    U.append(r)
    while U ≠ V do
        (u, v) ← argmin_{e∈U×(V\U)}(w(e))
        color[(u, v)] = Blue
        U.append(v)
        for w ∈ U do
            if (w, v) ∈ E then
                color[(w, v)] = Red
            end
        end
    end
    return (V, {e ∈ E | color[e] = Blue})
```

טענה: יהי G גרף קשיר לא מכוון וממושקל w אזי כל צביעת קשת באלגוריתם $\text{Prim'sAlgorithm}\left(G\right)$ נעשית כמו באלגוריתם הגנרי.

מסקנה: יהי G גרף קשיר לא מכוון וממושקל w אזי $\text{Prim'sAlgorithm}\left(G\right)$ עפ"מ של G.

משפט: יהי G גרף קשיר לא מכוון וממושקל w אזי ניתן לממש את $\text{Prim'sAlgorithm}\left(G\right)$ עם ערימת מינימום בסיבוכיות $\mathcal{O}\left(|E|\log|V|\right)$.

הערה: יהי G גרף קשיר לא מכוון וממושקל w אזי ניתן לממש את $\text{Prim'sAlgorithm}(G)$ בסיבוכיות $\mathcal{O}(|E| + |V|\log|V|)$.

אלגוריתם קרוסקל למציאת עץ פורש מינימלי: יהי G גרף קשיר לא מכוון וממושקל w אזי

```
function Kruskal'sAlgorithm(G):
    color ← dict(E)
    L ← sort(E)
    for (u,v) ∈ L do
        if ∃σ ∈ {u → v}.∀i ∈ [n].color(σ(i)) = Blue then
            color[e] = Red
        end
        else
            color[e] = Blue
        end
    end
    return (V, {e ∈ E | color[e] = Blue})
```

טענה: יהי G גרף קשיר לא מכוון וממושקל w אזי כל צביעת קשת באלגוריתם $\text{Kruskal'sAlgorithm}(G)$ נעשית כמו באלגוריתם הגנרי.

מסקנה: יהי G גרף קשיר לא מכוון וממושקל w אזי $\text{Kruskal'sAlgorithm}(G)$ עפ"מ של G.

משפט: יהי G גרף קשיר לא מכוון וממושקל w אזי ניתן לממש את $\text{Kruskal'sAlgorithm}(G)$ עם Union-Find בסיבוכיות $\mathcal{O}(|E|\log|V|)$ וכן סיבוכיות amortized $\mathcal{O}(|E| \cdot \alpha(|V|))$.

אלגוריתם Borůvska למציאת עץ פורש מינימלי: יהי G גרף קשיר לא מכוון וממושקל w באשר w חח"ע אזי

```
function Borůvska'sAlgorithm(G):
    Trees ← set(set(G))
    for v ∈ V do
        Trees.append({v})
    end
    while |Trees| ≠ 1 do
        for T ∈ Tree do
            (u,v) ← argmin_{(u,v)∈V(T)×V(G)}(w((u,v)))
            S ← {S ∈ Tree | u ∈ V(S)}
            S ← S + T + {(u,v)}
            Trees.Remove(T)
        end
    end
    A ← Trees
    return A
```

טענה: יהי G גרף קשיר לא מכוון וממושקל w באשר w חח"ע אזי סיבוכיות $\text{Borůvska'sAlgorithm}(G)$ הינה $\mathcal{O}(|E|\log|V|)$.

משפט: יהי G גרף קשיר לא מכוון וממושקל w באשר w חח"ע אזי קיים ויחיד $T \leq G$ עפ"מ.

משפט: יהי G גרף קשיר לא מכוון וממושקל w באשר w חח"ע אזי $\text{Borůvska'sAlgorithm}(G)$ עפ"מ של G.

משפט: יהי G גרף קשיר לא מכוון וממושקל w תהא $A \subseteq E$ יהי C מעגל ותהא $e \in E$ בעלת משקל מקסימלי אזי קיים עפ"מ $T \leq G$ עבורו $A \subseteq E(T)$ וכן $e \notin E(T)$.

טענה: יהיו $T_1, T_2 \leq G$ עפ"מ ויהיו $\alpha_1 \leq \ldots \leq \alpha_n$ ו־$\beta_1 \leq \ldots \leq \beta_m$ משקליי הקשתות כולל כפילויות אזי $n = m$ וכן $\alpha_i = \beta_i$ לכל $i \in [n]$.

אלגוריתם למציאת עץ פורש מינימלי המכיל מספר מקסימלי של קשתות מקבוצה מסוימת: יהי G גרף קשיר לא מכוון וממושקל w ותהא $F \subseteq E$ אזי

```
function PrioritizeMST(G, w, F):
    ε ← min({w(e_1) − w(e_2) | (e_1, e_2 ∈ E) ∧ (w(e_1) ≠ w(e_2))})/2
    for e ∈ E do
        if e ∈ F then

        end
        else
            w'(e) ← w(e)
        end
    end
    return Kruskal'sAlgorithm(G, w')
```

טענה: תהא $F\subseteq E$ ויהי T עפ"מ ביחס ל־w' באלגוריתם PrioritizeMST אזי T עפ"מ ביחס ל־w.

מסקנה: תהא $F\subseteq E$ אזי $\text{PrioritizeMST}\left(G,w\right)$ עפ"מ ב־G ביחס ל־w.

בעיית שיבוץ המשימות: יהיו $s_1\dots s_n,f_1\dots f_n\in\mathbb{R}$ באשר $s_i<f_i$ לכל $i\in[n]$ אזי $\max\left\{|A|\mid\left(A\subseteq\{[s_1,f_i]\}_{i=1}^{n}\right)\wedge\left(\forall I,J\in A.I\cap J=\varnothing\right)\right\}$.

אלגוריתם חמדן לבעיית שיבוץ המשימות: יהיו $s_1\dots s_n,f_1\dots f_n\in\mathbb{R}$ באשר $s_i<f_i$ לכל $i\in[n]$ אזי

```
function ActivitySelectionProblem(s_1, ..., s_n, f_1, ..., f_n):
    F ← list([s_1, f_1], ..., [s_n, f_n])
    /* In the next line, we put all the parameters in F and sort it based on f_i   */
    F ← sort({f_1, ..., f_n})
    X ← list([s_1, f_1], ..., [s_n, f_n])
    X ← ∅
    for k ∈ [1, ..., n] do
        if X = ∅ then
            X.append(L[k])
        else if L[k] ∩ X.last = ∅ then
            X.append(L[k])
    end
    return X
```

טענה: יהיו $s_1\dots s_n,f_1\dots f_n\in\mathbb{R}$ באשר $s_i<f_i$ אזי סיבוכיות ActivitySelectionProblem הינה $\mathcal{O}\left(n\log\left(n\right)\right)$.

משפט: לכל $k\in[n]$ באיטרציה ה־k בלולאה ב־ActivitySelectionProblem קיים פתרון לבעיה X^* עבורו $\left([s_i,f_i]\in X^*\right)\Longleftrightarrow\left([s_i,f_i]\in X\right)$.

מסקנה: יהיו $s_1\dots s_n,f_1\dots f_n\in\mathbb{R}$ באשר $s_i<f_i$ אזי ActivitySelectionProblem פתרון לבעיית שיבוץ המשימות.

הערה: כאשר משקל הגרף הוא ℓ הכוונה היא "אורך הקשת" ובכך אנו "מכלילים" גרף ללא משקל בתור גרף עבורו $\ell=1$.

מעגל שלילי: יהי G מעגל ממושקל ℓ אזי מעגל C עבורו $\ell\left(C\right)<0$.

מסלול קצר ביותר בין קודקודים: יהי G גרף ממושקל ℓ ויהיו $s,t\in V$ אזי מסלול σ מ־s ל־t עבורו $\ell\left(\sigma\right)=\min\left\{\ell\left(\tau\right)\mid\tau\in\{s\to t\}\right\}$.

למה: יהיו $s,t\in V$ עבורם קיים מסלול מ־s ל־t וכן כל מסלול מ־s ל־t לא עובר דרך קשת השייכת למעגל שלילי אזי קיים מסלול פשוט קצר ביותר בין s ל־t.

למה: יהיו $s,t\in V$ עבורם קיים מסלול מ־s ל־t וכן קיים מסלול מ־s ל־t העובר דרך קשת השייכת למעגל שלילי אזי לא קיים מסלול פשוט קצר ביותר בין s ל־t.

סימון: יהיו $s,t\in V$ אזי $\delta\left(s,t\right)=\inf_{\sigma\in\{s\to t\}}\ell\left(\sigma\right)$.

בעיית המסלולים הקצרים מנקודת מוצא (SSSP): יהי $s\in V$ אזי $T\leq G$ עץ פורש בו כל מסלול מ־s ל־v הינו מסלול קצר ביותר ב־G.

למה אי־שיוויון המשולש: יהיו $u,v,w\in V$ אזי $\delta\left(u,v\right)\leq\delta\left(u,w\right)+\delta\left(w,v\right)$.

למה תת־מסלול קצר ביותר: יהי σ מסלול קצר ביותר אזי $\left(\sigma\left[i\right],\dots\sigma\left[i+k\right]\right)$ מסלול קצר ביותר.

אלגוריתם בלמן־פורד למציאת מסלולים קצרים ביותר מנקודת מוצא: יהי G ממושקל ℓ ויהי $s\in V$ אזי

```
function BellmanFord(G, ℓ, s):
    (d, π) ← dict(V)
    d[s] ← 0
    for u ∈ V do
        d[u] ← ∞
        π[u] ← None
    end
    (c, i) ← 1
    while (i ≤ |V|) ∧ (c ≠ 0) do
        c ← 0
        for (u, v) ∈ E do
            c ← Relax(ℓ, d, u, v)
        end
    end
    return c

function Relax(ℓ, d, u, v):
    if d[v] > d[u] + ℓ(u, v) then
        d[v] ← d[u] + ℓ(u, v)
        π[v] ← u
        return 1
    end
    return 0
```

למה: יהיו $s,u,v \in V$ באשר $(u,v) \in E$ וכן בריצת BellmanFord מתקיים $\delta(s,u) \leq d[u]$ אזי $\delta(s,v) \leq d[u] + \ell((u,v))$.

מסקנה: יהיו $s,u,v \in V$ באשר $(u,v) \in E$ וכן בריצת BellmanFord מתקיים $\delta(s,u) \leq d[u]$ וכן $\delta(s,v) \leq d[v]$ אזי לאחר הרצת $\text{Relax}(u,v)$ מתקיים $\delta(s,v) \leq d[v]$.

למה: יהי $s \in V$ עבורו לכל $v \in V$ בריצת BellmanFord מתקיים $\delta(s,v) \leq d[v]$ אזי לאחר כל רצף פעולות Relax נקבל כי לכל $v \in V$ מתקיים $\delta(s,v) \leq d[v]$.

מסקנה: יהיו $s,v \in V$ עבורם בריצת BellmanFord מתקיים $d[v] = \infty$ אזי לאחר כל רצף פעולות Relax נקבל כי $d[v] = \infty$.

מסקנה: יהיו $s,v \in V$ עבורם בריצת BellmanFord מתקיים $d[v] = \delta(s,v)$ אזי לאחר כל רצף פעולות Relax נקבל כי $d[v] = \delta(s,v)$.

למה: יהיו $s,t \in V$ עבורם בריצת BellmanFord מתקיים $d[s] = 0$ ויהי $\sigma \in \{s \to t\}$ מסלול אזי לאחר הפעלת הרצף $\text{Relax}(\sigma[0], \sigma[1]), \ldots, \text{Relax}(\sigma[n-1], \sigma[n])$ נקבל כי $d[t] \leq \ell(\sigma)$.

למה: יהי $s \in V$ עבורו לא קיים מעגל שלילי אשר ניתן להגיע אליו מ־s אזי BellmanFord יוצא מהלולאה הראשית כאשר $i < |V|$ וכן מחזיר 0 וכן לכל $v \in V$ מתקיים $d[v] = \delta(s,v)$.

למה: יהי $s \in V$ עבורו קיים מעגל שלילי אשר ניתן להגיע אליו מ־s אזי BellmanFord יוצא מהלולאה הראשית כאשר $i = |V|$ וכן מחזיר 1.

מסקנה: יהי $s \in V$ אזי

- (BellmanFord החזיר 1)$\iff$(קיים מעגל שלילי אשר ניתן להגיע אליו מ־s).
- (BellmanFord החזיר 0)$\iff$(לא קיים מעגל שלילי אשר ניתן להגיע אליו מ־s וכן לכל $v \in V$ מתקיים $d[v] = \delta(s,v)$).

למה: יהי $s \in V$ ויהי C מעגל בעץ BellmanFord באיזשהו שלב של הרצת BellmanFord אזי C מעגל שלילי.

למה: יהי $s \in V$ עבורו לא קיים מעגל שלילי אשר ניתן להגיע אליו מ־s אזי עץ BellmanFord הינו עץ G_π.

למה: יהי $s \in V$ עבורו קיים מעגל שלילי אשר ניתן להגיע אליו מ־s אזי עץ BellmanFord מכיל מעגל שלילי.

מסקנה: יהי $s \in V$ אזי BellmanFord פתרון לבעיית SSSP.

משפט: יהי $s \in V$ אזי אלגוריתם BellmanFord רץ בסיבוכיות $\mathcal{O}(|E| \cdot |V|)$.

הערה: נניח כי $\ell : E \to \mathbb{Z}$ וכן $\ell(e) \geq -W$ אזי קיים אלגוריתם לבעיית SSSP בסיבוכיות $\mathcal{O}\left(|E| \log^2(|V|) \log(|V| \cdot W) \log\log(|V|)\right)$.

אלגוריתם לבדיקת קיום מעגל במשקל 0 בגרף מכוון חסר מעגלים שליליים: יהי G גרף מכוון חסר מעגלים שליליים אזי

```
function IsZeroCircle(G, ℓ):
    s ∉ V
    V ← V ∪ s
    for v ∈ V\{s} do
        E ← E ∪ {(s, v)}
        ℓ((s, v)) ← 0
    end
    BellmanFord(G, ℓ, s)
    for e ∈ E do
        if δ(s, v) ≠ δ(s, u) + ℓ(u, v) then
            E ← E\{(s, v)}
        end
    end
    if ∃ circle C ∈ G then
        return true
    end
    return false
```

טענה: בריצת IsZeroCircle לאחר מחיקת כל הקשתות נקבל את גרף מק"ב מ־s.

טענה: אם בריצת IsZeroCircle לאחר מחיקת כל הקשתות קיים מעגל C אזי $\ell\left(C\right)=0$.

טענה: יהי C מעגל עבורו $\ell\left(C\right)=0$ אזי בריצת IsZeroCircle לאחר מחיקת כל הקשתות נקבל כי C בגרף.

מסקנה: יהי G גרף מכוון חסר מעגלים שליליים אזי (G בעל מעגל ממשקל 0)$\Longleftrightarrow$(IsZeroCircle מחזיר true).

טענה: יהי G גרף מכוון חסר מעגלים שליליים אזי סיבוכיות IsZeroCircle הינה $\mathcal{O}\left(\left|V\right|\cdot\left|E\right|\right)$.

אלגוריתם למציאת מסלולים קצרים ביותר מנקודת מוצא בגרף מכוון אציקלי: יהי G מכוון אציקלי ויהי $s\in V$ אזי

```
function SSSP-DAG(G, ℓ, s):
    (d, π) ← dict(V)
    d[s] ← 0
    for u ∈ V do
        d[u] ← ∞
        π[u] ← None
    end
    /* Knuth'sAlgorithm is an algorithm to compute a topological sorting.   */
    f ← Knuth'sAlgorithm(G)
    for i ∈ [1, ..., |V|] do
        for v ∈ Adj(f(i)) do
            Relax((f(i), v))
        end
    end
    return (d, π)
```

טענה: יהי G מכוון אציקלי ויהי $s\in V$ אזי $\text{SSSP-DAG}\left(G\right)$ פתרון לבעיית SSSP.

טענה: יהי G מכוון אציקלי ויהי $s\in V$ אזי סיבוכיות $\text{SSSP-DAG}\left(G\right)$ הינה $\mathcal{O}\left(\left|E\right|+\left|V\right|\right)$.

אלגוריתם דייקסטרה למציאת מסלולים קצרים ביותר מנקודת מוצא בגרף ללא משקלים שליליים: יהי G גרף עבורו $\ell\geq0$ ויהי $s\in V$ אזי

```
function Dijkstra(G, ℓ, s):
    Q ← heap((V, int))
    (d, π) ← dict(V)
    d[s] ← 0
    for u ∈ V do
        d[u] ← ∞
        π[u] ← None
    end
    Q.insert((s, d[s]))
    while Q ≠ ∅ do
        u ← Q.min
        for v ∈ Adj(u) do
            if d[v] = ∞ then
                π[v] ← u
                d[v] ← d[u] + ℓ(u, v)
                Q.insert((v, d[v]))
            else if d[u] + ℓ(u, v) < d[v] then
                π[v] ← u
                d[v] ← d[u] + ℓ(u, v)
                Q.decrease-key((v, d[v]))
        end
    end
    return (d, π)
```

למה: יהיו $s,u\in V$ עבורם בריצת Dijkstra הצומת u נמחקה מ־Q אזי $d\left[u\right]=\delta\left(s,u\right)$.

משפט: יהי $s\in V$ אזי Dijkstra פתרון לבעיית SSSP כאשר $\ell\geq 0$.

משפט: יהי $s\in V$ אזי ניתן לממש את Dijkstra עם Fibonacci heaps בסיבוכיות $\mathcal{O}\left(|E|+|V|\cdot\log\left(|V|\right)\right)$.