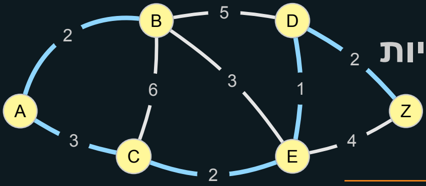


יסודות האלגוריתמים והסיבוכיות

תרגול 07 - עץ פורש מינימלי

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים BFS ו DFS.

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים BFS ו DFS.

אלגוריתמים אלו מוצאים עצים פורשים מינימלים **רק** בגרפים לא ממושקלים.

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים BFS ו DFS.
אלגוריתמים אלו מוצאים עצים פורשים מינימלים **רק** בגרפים לא ממושקלים.

במקרה זה סך משקלם יהיה מספר הקשתות בעץ עם *n* צמתים:

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים BFS ו DFS.

אלגוריתמים אלו מוצאים עצים פורשים מינימלים **רק** בגרפים לא ממושקלים.

במקרה זה סך משקלם יהיה מספר הקשתות בעץ עם n צמתים:

$n-1$ קשתות.

בעיית העץ הפורש המינימלי

הכרנו אלגוריתמים המוצאים עצים פורשים בגרפים קשירים BFS ו DFS.

אלגוריתמים אלו מוצאים עצים פורשים מינימלים **רק** בגרפים לא ממושקלים.

במקרה זה סך משקלם יהיה מספר הקשתות בעץ עם n צמתים:

$n-1$ קשתות.

אך מה אם נרצה למצוא עץ פורש בגרף ממושקל עם סך משקלים מינימלי?

האלגוריתם של פרים

Prim($G = (V, E)$)

$T = \varnothing$
Mark arbitrary vertex as *visited*.
while there exists a non-visited vertex in G **do**
    Find minimum edge $\{v, u\}$ with v visited and u not-visited.
    Mark u as visited and add $\{v, u\}$ to T
return T

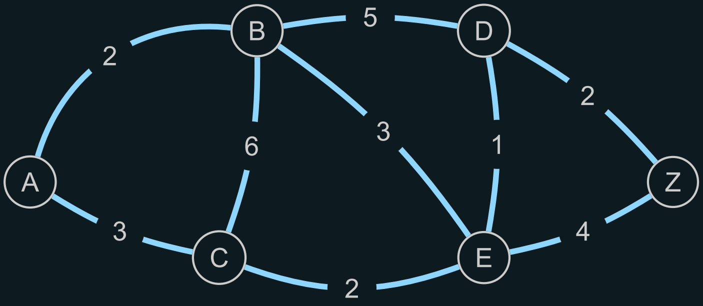
האלגוריתם של פרים

HeapPrim($G = (V, E)$)

```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i, j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i, j} = H.popRoot().
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i, j} .
9:      Mark x as visited and add {i, j} to T.
10:     Add all edges {x, y} ∈ E to H where y is not visited.
11: return T
```

תרגיל 1

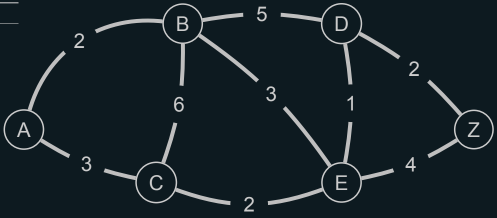
מצאו עץ פורש מינימאלי בגרף הבא:



תרגיל 1

HeapPrim($G = (V, E)$)

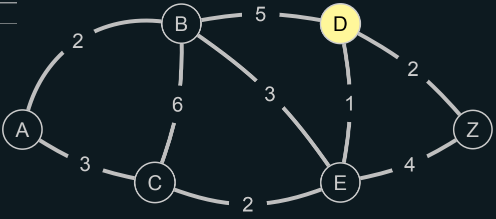
1: $T = \varnothing$
2: Create an empty heap H
3: Mark arbitrary vertex i as *visited*.
4: Add all edges $\{i, j\} \in E$ to H.
5: **while** there exists a non-visited vertex in G **do**
6: $\quad \{i, j\} = H$.popRoot()
7: $\quad$ **if** i and j are both visited **then** Go To 6.
8: $\quad$ Let x be the non-visited vertex of $\{i, j\}$.
9: $\quad$ Mark x as visited.
10: $\quad$ Add all edges $\{x, y\} \in E$ to H for non-visited y.
11: **return** T



תרגיל 1

HeapPrim($G = (V, E)$)

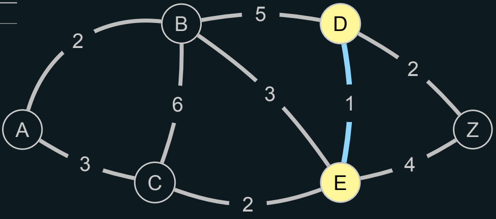
```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i,j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i,j} = H.popRoot()
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i,j} .
9:      Mark x as visited.
10:     Add all edges {x,y} ∈ E to H for non-visited y.
11: return T
```



תרגיל 1

HeapPrim($G = (V, E)$)

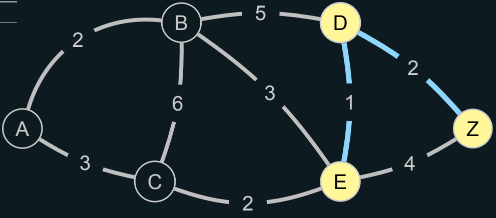
1: $T = \varnothing$
2: Create an empty heap H
3: Mark arbitrary vertex i as *visited*.
4: Add all edges $\{i, j\} \in E$ to H.
5: **while** there exists a non-visited vertex in G **do**
6: $\quad \{i, j\} = H$.popRoot()
7: $\quad$ **if** i and j are both visited **then** Go To 6.
8: $\quad$ Let x be the non-visited vertex of $\{i, j\}$.
9: $\quad$ Mark x as visited.
10: $\quad$ Add all edges $\{x, y\} \in E$ to H for non-visited y.
11: **return** T



תרגיל 1

HeapPrim($G = (V, E)$)

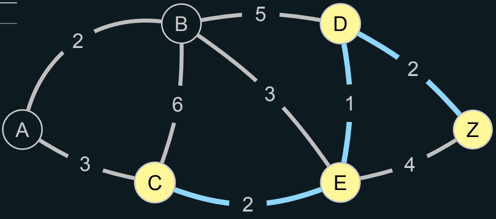
1: $T = \varnothing$
2: Create an empty heap H
3: Mark arbitrary vertex i as *visited*.
4: Add all edges $\{i, j\} \in E$ to H.
5: **while** there exists a non-visited vertex in G **do**
6: $\quad \{i, j\} = H$.popRoot()
7: $\quad$ **if** i and j are both visited **then** Go To 6.
8: $\quad$ Let x be the non-visited vertex of $\{i, j\}$.
9: $\quad$ Mark x as visited.
10: $\quad$ Add all edges $\{x, y\} \in E$ to H for non-visited y.
11: **return** T



תרגיל 1

HeapPrim($G = (V, E)$)

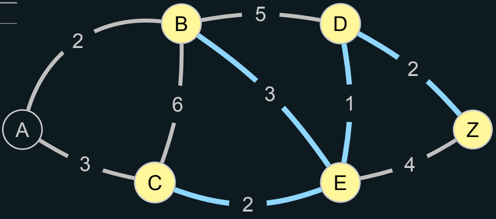
```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i, j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i, j} = H.popRoot()
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i, j} .
9:      Mark x as visited.
10:     Add all edges {x, y} ∈ E to H for non-visited y.
11: return T
```



תרגיל 1

HeapPrim$(G = (V, E))$

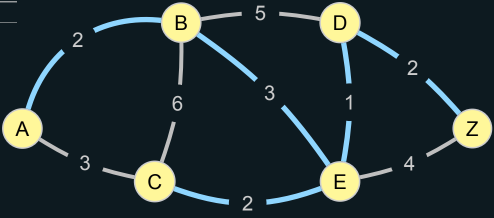
```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i,j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i,j} = H.popRoot()
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i,j} .
9:      Mark x as visited.
10:     Add all edges {x,y} ∈ E to H for non-visited y.
11: return T
```



תרגיל 1

HeapPrim($G = (V, E)$)

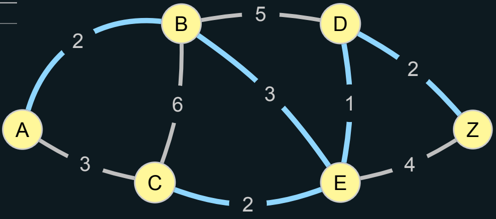
```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i,j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i,j} = H.popRoot()
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i,j} .
9:      Mark x as visited.
10:     Add all edges {x,y} ∈ E to H for non-visited y.
11: return T
```



תרגיל 1

HeapPrim($G = (V, E)$)

```
1:  T = ∅
2:  Create an empty heap H
3:  Mark arbitrary vertex i as visited.
4:  Add all edges {i,j} ∈ E to H.
5:  while there exists a non-visited vertex in G do
6:      {i,j} = H.popRoot()
7:      if i and j are both visited then Go To 6.
8:      Let x be the non-visited vertex of {i,j} .
9:      Mark x as visited.
10:     Add all edges {x,y} ∈ E to H for non-visited y.
11: return T
```



שימו לב שקשת AC יכלה גם כן להיבחר.

הקשת BE תיבחר במידה ותור הקדימויות מומש ע״י ערימה בינארית.

תרגיל 2

נתון גרף לא מכוון קשיר וממושקל בעל n צמתים. תארו אלגוריתם בזמן $\mathcal{O}(n)$ המוצא קשת ספציפית שהורדת משקלה ב-1 יוריד את משקל העץ הפורש המינימאלי של הגרף.

תרגיל 2

נתון גרף לא מכוון קשיר וממושקל בעל n צמתים. תארו אלגוריתם בזמן $\mathcal{O}(n)$ המוצא קשת ספציפית שהורדת משקלה ב-1 יוריד את משקל העץ הפורש המינימאלי של הגרף.

נבחר צומת v כלשהו, נעבור על כל הקשתות היוצאות ממנו נמצא את הקשת המינימלית $\{v, u\}$ ונבחר אותה.

תרגיל 2

נתון גרף לא מכוון קשיר וממושקל בעל n צמתים. תארו אלגוריתם בזמן $\mathcal{O}(n)$ המוצא קשת ספציפית שהורדת משקלה ב-1 יוריד את משקל העץ הפורש המינימאלי של הגרף.

נבחר צומת v כלשהו, נעבור על כל הקשתות היוצאות ממנו נמצא את הקשת המינימלית $\{v, u\}$ ונבחר אותה.

בחירתנו נכונה משום שניתן לפרק את העץ הפורש המינימלי לתתי עצים פורשים שמחוברים בינהם בקשת אחת בלבד - הקשת המינימלי שמחברת בינהם.

תרגיל 2

נתון גרף לא מכוון קשיר וממושקל בעל n צמתים. תארו אלגוריתם בזמן $\mathcal{O}(n)$ המוצא קשת ספציפית שהורדת משקלה ב-1 יוריד את משקל העץ הפורש המינימאלי של הגרף.

נבחר צומת v כלשהו, נעבור על כל הקשתות היוצאות ממנו נמצא את הקשת המינימלית $\{v, u\}$ ונבחר אותה.

בחירתנו נכונה משום שניתן לפרק את העץ הפורש המינימלי לתתי עצים פורשים שמחוברים בינהם בקשת אחת בלבד - הקשת המינימלי שמחברת בינהם.

בנוסף ע״פ האלגוריתם של פרים הקשת המינימלית היוצאת מכל צומת תהיה בעץ הפורש המינימלי.

הסיבוכיות של האלגוריתם היא $\mathcal{O}(n)$ משום שבמקרה הגרוע נעבור על $n-1$ שכנים של v.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.
2. אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.
2. אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.
3. אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.
2. אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.
3. אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1.
4. אם נוצר רכיב קשירות אחד נחזיר "כן" אם ורק אם קיימת בG קשת במשקל 1.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש. $\mathcal{O}(n+m)$
2. אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.
3. אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1.
4. אם נוצר רכיב קשירות אחד נחזיר "כן" אם ורק אם קיימת בG קשת במשקל 1.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. $\mathcal{O}(n+m)$ נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.
2. $\mathcal{O}(1)$ אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.
3. אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1.
4. אם נוצר רכיב קשירות אחד נחזיר "כן" אם ורק אם קיימת בG קשת במשקל 1.

תרגיל 3

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש. $\mathcal{O}(n+m)$
2. אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד. $\mathcal{O}(1)$
3. אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1. $\mathcal{O}(1)$
4. אם נוצר רכיב קשירות אחד נחזיר "כן" אם ורק אם קיימת בG קשת במשקל 1.

תרגיל 3

אוניברסיטת בן-גוריון בנגב
جامعة بن غوريون في النقب
Ben-Gurion University of the Negev

נתון גרף G לא מכוון קשיר וממושקל, שכל המשקלים בו הם 0 או 1. תארו אלגוריתם הבודק אם לגרף זה יש עץ פורש שסכום המשקלים שלו הוא 1.

למעשה אנחנו מחפשים עץ פורש שבו כל הקשתות חוץ מאחת במשקל 0.

1. $\mathcal{O}(n+m)$ נמחק מ G את כל הקשתות במשקל 1 ונמצא את רכיבי הקשירות של הגרף החדש.
2. $\mathcal{O}(1)$ אם מצאנו יותר משני רכיבי קשירות נחזיר "לא" משום שנצטרך לפחות שתי קשתות במשקל 1 כדי לחבר את כל הרכיבי לעץ אחד.
3. $\mathcal{O}(1)$ אם נוצרו שני רכיבי קשירות נחזיר "כן" משום שנתון לנו שG קשיר ולכן בהכרח קיימת בינהם קשת במשקל 1.
4. $\mathcal{O}(m)$ אם נוצר רכיב קשירות אחד נחזיר "כן" אם ורק אם קיימת בG קשת במשקל 1.

תרגיל 4

הוכיחו או הפריכו את הטענה:

בהינתן עץ פורש מינימלי T בגרף $G = (V, E)$ ושני צמתים $v, u \in V$, המסלול הקצר ביותר בין v לu כלול בעץ T.

תרגיל 4

הוכיחו או הפריכו את הטענה:

בהינתן עץ פורש מינימלי T בגרף $G=(V,E)$ ושני צמתים $v,u\in V$, המסלול הקצר ביותר בין v ל-u כלול בעץ T.

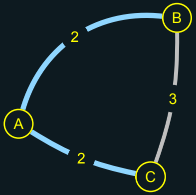
הטענה לא נכונה.

תרגיל 4

הוכיחו או הפריכו את הטענה:
בהינתן עץ פורש מינימלי T בגרף $G = (V, E)$ ושני צמתים $v, u \in V$, המסלול הקצר ביותר בין v לu כלול בעץ T.

הטענה לא נכונה.

ניקח כדוגמת נגד את הגרף הבא



קשת BC אינה מופיעה בעץ הפורש המינימלי והיא המסלול הקצר ביותר בין B ל C.