Lecture 12 of Geiger & Itai's slide brochure
www.cs.technion.ac.il/~dang/courseDS

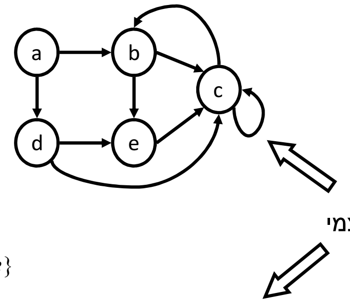
גרפים

חומר קריאה לשיעור זה

Chapter 23.1 – Representations of Graphs
Chapter 23.4 – Topological Sort
Chapter 24 – Minimum Spanning Trees

גרפים מכוונים (Directed Graphs)

גרף מכוון הוא זוג (V, E) המורכב מקבוצת צמתים V וקבוצת קשתות $E \subseteq V \: X \: V$.



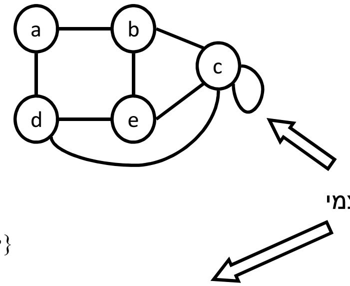
$V = \{a, b, c, d, e\}$

$E = \{(a, b), (a, d), (b, c), (b, e), (c, b), (c, c), (d, c), (d, e), (e, c)\ \}$

נסמן $|V| = n$ וכן $m = |E|$.
בדוגמא: $n = 5$, $m = 9$.
מספר הקשתות m בגרף מכוון קטן\שווה מ-n^2.

גרפים לא-מכוונים (Undirected Graphs)

גרף לא-מכוון הוא זוג (V, E) המורכב מקבוצת צמתים V וקבוצת קשתות E. קשת ב-E היא קבוצה בת שני איברים מתוך V. קשת מסומנת ע"י (i, j) (במקום הסימון המדויק יותר $\{i, j\}$).



$V = \{a, b, c, d, e\}$

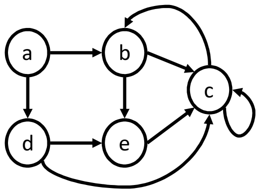
$E = \{(a, b), (a, d), (b, c), (b, e), (c, c), (d, c), (d, e), (e, c)\ \}$

נסמן $|V| = n$ וכן $m = |E|$.
בדוגמא: $n = 5$, $m = 8$.
מספר הקשתות m בגרף לא מכוון קטן\שווה מ-n^2.

ייצוג גרף מכוון במטריצת סמיכויות (Adjacency Matrix)

נגדיר מטריצה בולאנית A בגודל $n \times n$ כך שיתקיים: $A[i, j] = 1 \;\Leftrightarrow\; (i, j) \in E$

דוגמא:



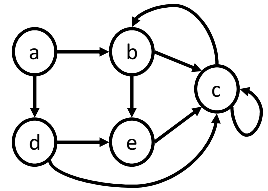
	a	b	c	d	e
a	0	1	0	1	0
b	0	0	1	0	1
c	0	1	1	0	0
d	0	0	1	0	1
e	0	0	1	0	0

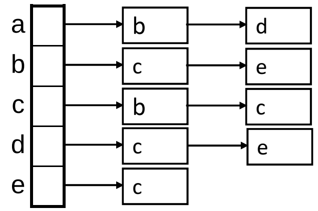
הערה: גרף לא-מכוון אפשר לייצג באמצעות מטריצה סימטרית.

ייצוג גרף מכוון ברשימות סמיכויות (Adjacency lists)

לכל צומת נשמור את רשימת הצמתים אליהם הוא מצביע.

דוגמא:

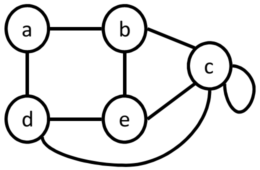


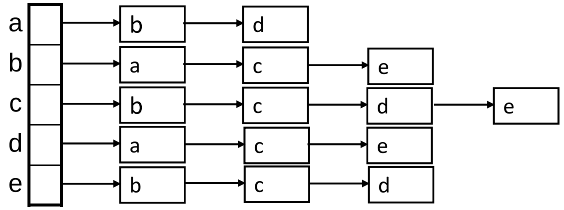


ייצוג גרף לא-מכוון ברשימות סמיכויות

לכל צומת נשמור את רשימת הצמתים אליהם הוא מחובר בקשת.

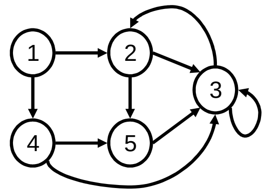
דוגמא:





כל קשת מופיעה פעמיים (מלבד חוגים עצמיים).

תרגיל



נתונה רשימה לא ממוינת של קשתות.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$.

בדוגמא:

$E = \{(5,3), (2,3), (4,5), (4,3), (3,2), (1,2), (3,3), (1,4), (2,5)\ \}$

נתונות m קשתות כאשר כל קשת היא זוג מספרים בטווח $1 \ldots n$.

פתרון: נבצע Radix Sort כאשר הבסיס הוא n.

מיון לפי "ספרה" ימנית

$(3,3)$

$(4,3)$

$(1,2)$ $(2,3)$ $(2,5)$

$(3,2)$ $(5,3)$ $(1,4)$ $(4,5)$

מיון לפי "ספרה" שמאלית

$(1,4)$ $(2,5)$ $(3,3)$ $(4,5)$

$(1,2)$ $(2,3)$ $(3,2)$ $(4,3)$ $(5,3)$

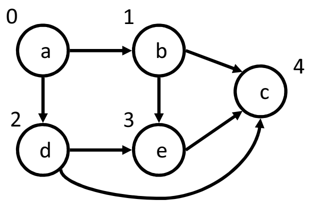
מיון טופולוגי

קלט: גרף מכוון.

פלט: מספור של צמתי הגרף – לצומת i יינתן המספר $N[i]$ – כך שיתקיים:

$N[i] < N[j] \quad \Rightarrow \quad (i,j) \in E$, או הודעה שלא קיים מספור כזה.

דוגמא:



מתי ניתן למצוא מספור כזה ?

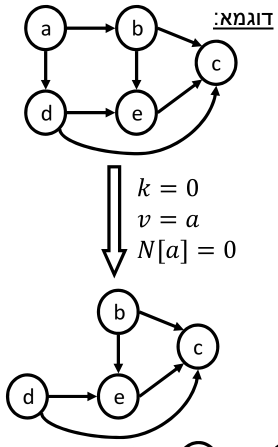
הגדרה: מקור הוא צומת שלא נכנסות אליו קשתות.

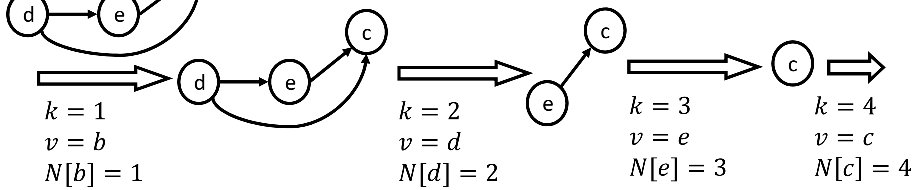
אבחנה: לכל גרף מכוון ללא מעגלים מכוונים יש לפחות מקור אחד.

אלגוריתם למיון טופולוגי

1. <u>אתחול:</u> $k = 0$.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v.
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף (וכן את הקשתות היוצאות ממנו).
3. אם $k = n$ אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

<u>דוגמא:</u>





מבני נתונים למיון טופולוגי

1. <u>אתחול:</u> $k = 0$.
2. כל עוד קיימים מקורות בצע:
   - מצא מקור v.
   - תן ל-v מספר k. קדם את k באחד.
   - סלק את v מהגרף (וכן את הקשתות היוצאות ממנו).
3. אם $k = n$ אז המספור הושלם, אחרת בגרף יש מעגל מכוון.

<u>הפעולות הנדרשות</u>

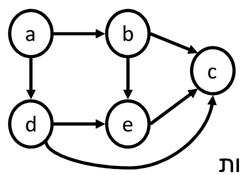
מצא מקור ←

סלק מקור ←

<u>בשקפים הבאים:</u>
- מימוש 1 בייצוג מטריצת סמיכויות
- מימוש 2 בייצוג רשימת סמיכויות

מימוש בייצוג מטריצת סמיכויות

	a	b	c	d	e
a	0	1	0	1	0
b	0	0	1	0	1
c	0	0	0	0	0
d	0	0	1	0	1
e	0	0	1	0	0



<u>מימוש הפעולות הנדרשות:</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת i מקור, נבדוק שעמודה i כולה אפסים. <u>זמן:</u> $O(n)$.
כדי למצוא מקור נבדוק את כל הצמתים. <u>סה"כ:</u> $O(n^2)$.

<u>סלק מקור i</u>: אפס שורה i. (העמודה כבר מאופסת כיון שמסלקים מקור). <u>זמן:</u> $O(n)$.

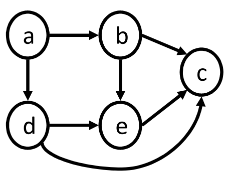
<u>זמן כללי:</u> האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן עבור מימוש זה היא $O(n^3)$.

צוואר הבקבוק במימוש זה היא פעולת "מצא מקור" המתבצעת n פעמים.

מימוש בייצוג מטריצת סמיכויות (שיפור)

בכל השיפורים נבצע עבוד מקדים (Preprocessing). בחישוב זמן הריצה נוסיף את זמן העיבוד המקדים לזמן ריצת האלגוריתם.

	a	b	c	d	e
a	0	1	0	1	0
b	0	0	1	0	1
c	0	0	0	0	0
d	0	0	1	0	1
e	0	0	1	0	0



<u>עיבוד מקדים:</u> בנה מערך in-degree[i] ובו נשמור את מספר הקשתות הנכנסות לצומת i. <u>זמן הבניה:</u> $O(n^2)$.

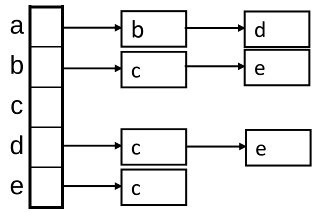
<u>מימוש הפעולות הנדרשות:</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת i מקור, נבדוק in-degree[i]=0. <u>זמן:</u> $O(1)$.
כדי למצוא מקור נבדוק את כל הצמתים. <u>סה"כ:</u> $O(n)$.

<u>סלק מקור i</u>: אפס שורה i.
לכל $A[i, j] = 1$ הקטן באחד את in-degree[j]. זמן $O(n)$.

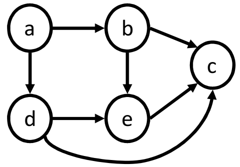
<u>זמן כללי:</u> במימוש זה נבצע עיבוד מקדים שדורש $O(n^2)$ זמן ונבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן של מימוש זה היא $O(n^2)$.

מימוש בייצוג רשימת סמיכויות



<u>מימוש הפעולות הנדרשות:</u>

<u>מצא מקור</u>: כדי לבדוק האם צומת i מקור, נעבור על כל הרשימות. אם הצומת לא נמצא אז הוא מקור. <u>זמן</u>: $O(m)$.
כדי למצוא מקור נבדוק את כל הצמתים.
<u>סה"כ</u>: $O(n \cdot m)$.

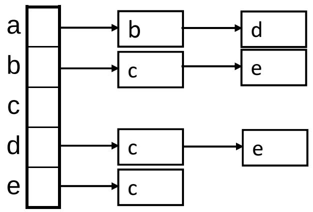


<u>סלק מקור i</u>: סמן שצומת i סולק. זמן $O(1)$.

<u>זמן כללי</u>: האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור". לפיכך סיבוכיות הזמן של מימוש זה היא $O(n^2 \cdot m)$.

<u>שיפור ראשון</u>:
<u>מצא מקור</u>: כדי למצוא מקור נעבור על כל הרשימות ונשמור מערך בולאני ובו 1 לכל צומת בו נתקלנו ו- 0 אחרת. <u>זמן</u>: $O(n+m)$.
לפיכך זמן כללי של מימוש זה הוא $O\big(n(n+m)\big) = O(n^2 + n \cdot m)$.

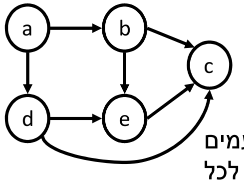
מימוש בייצוג רשימת סמיכויות (שיפור)



<u>שיפור שני</u>: נבנה ונשמור in-degree[i].
בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע.
<u>זמן</u>: $O(n+m)$.

<u>מצא מקור</u>: עבור על המערך in-degree. <u>זמן</u> $O(n)$.

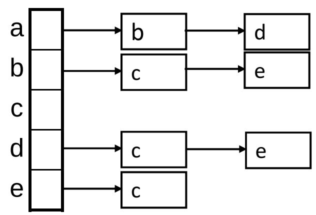
<u>סלק מקור i</u>: עבור על הרשימה ה- i ועדכן את המערך in-degree. זמן $O(out\_degree(i))$.



<u>זמן כללי</u>: פרט לעיבוד המקדים, האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור" – פעם אחת לכל צומת i. לפיכך סיבוכיות הזמן של מימוש זה היא:

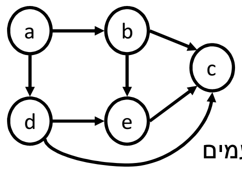
$$\sum_{i=1}^{n} O\big(n + out\_degree(i)\big) = O(n^2 + \sum_{i=1}^{n} out\_degree(i)) = O(n^2 + m)$$

מימוש בייצוג רשימת סמיכויות (שיפור)



<u>שיפור שלישי</u>: נשמור צמתים עבורם מתקיים in-degree[i]= 0 במחסנית, תור, או רשימה.
בנית in-degree[i] ע"י מעבר על כל הרשימות וספירת הפעמים שכל צומת מופיע. כמו קודם.
<u>זמן</u>: $O(n+m)$.

<u>מצא מקור</u>: איבר ראשון במחסנית. <u>זמן</u>: $O(1)$.



<u>סלק מקור i</u>: עבור על הרשימה ה- i ועדכן את המערך in-degree. הכנס למחסנית איברים עבורם דרגת הכניסה מתאפסת.
<u>זמן</u>: $O(out\_degree(i))$.

<u>זמן כללי</u>: פרט לעיבוד המקדים, האלגוריתם מבצע n פעמים את זוג הפקודות "מצא מקור" ו"סלק מקור" – פעם אחת לכל צומת i. לפיכך סיבוכיות הזמן של מימוש זה היא:

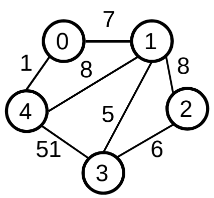
$$\sum_{i=1}^{n} O\big(1 + out\_degree(i)\big) = O(n + \sum_{i=1}^{n} out\_degree(i)) = O(n + m)$$

גרפים ממושקלים

גרף יקרא <u>ממושקל</u> אם לכל קשת e בקבוצה E קיים משקל חיובי $w(e)$.

נתמקד בגרפים ממושקלים לא-מכוונים. גרפים אלה ניתן לייצג בעזרת מטריצת סמיכויות סימטרית או רשימות סמיכויות.

<u>דוגמא:</u>



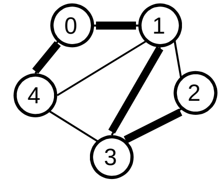
<u>ייצוג באמצעות מטריצת שכנויות:</u>
מטריצה A ממומשת ע"י ייצוג של מטריצה סימטרית. האלכסון מציין חוגים עצמיים. מספר האיברים הנדרש הוא $n(n+1)/2$.

	0	1	2	3	4
0	0	7	0	0	1
1		0	8	5	8
2			0	6	0
3				0	51
4					0

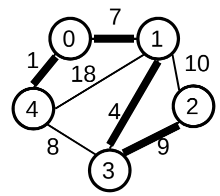
$$A[i,j] = \begin{cases} 0 & (i,j) \notin E \\ w(i,j) & else \end{cases}$$

עץ פורש מינימום

הגדרה: עץ פורש של גרף לא-מכוון G הוא עץ שצמתיו הם הצמתים של G וקשתותיו הן קשתות של G.

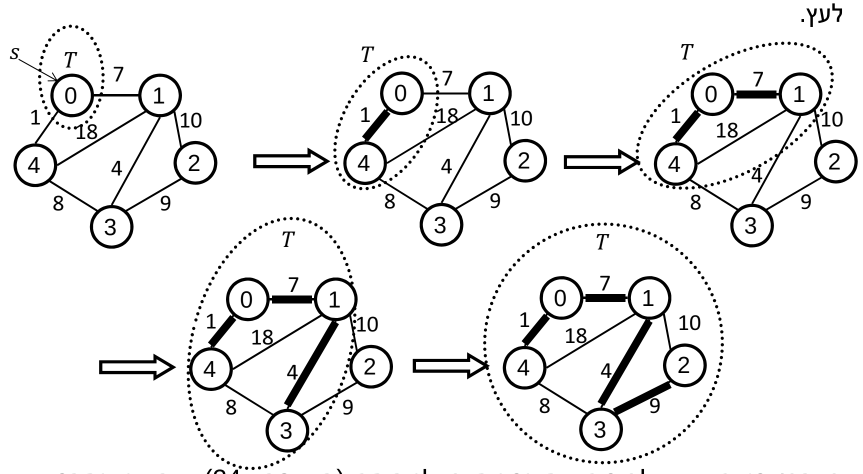


הגדרה: עץ פורש מינימום של גרף ממושקל לא-מכוון G הוא עץ פורש של G שסכום משקלי קשתותיו מינימלי.



מציאת עץ פורש מינימום

רעיון (Prim): הכנס צומת כלשהו s לעץ T. בכל שלב מצא קשת בעלת משקל מינימלי המחברת בין צומת מהעץ T הנוכחי לצומת בשאר הגרף והוסף קשת זו לעץ.



הערה: כמובן שיש להוכיח את נכונות האלגוריתם (ראו פרק 24), אך אנו נתרכז בפרטי המימוש בלבד.

אלגוריתם למציאת עץ פורש מינימום (Prim)

רעיון: כל צומת שאינו ב-T ישמור את הקשת הקלה ביותר המחברת אותו ל-T.

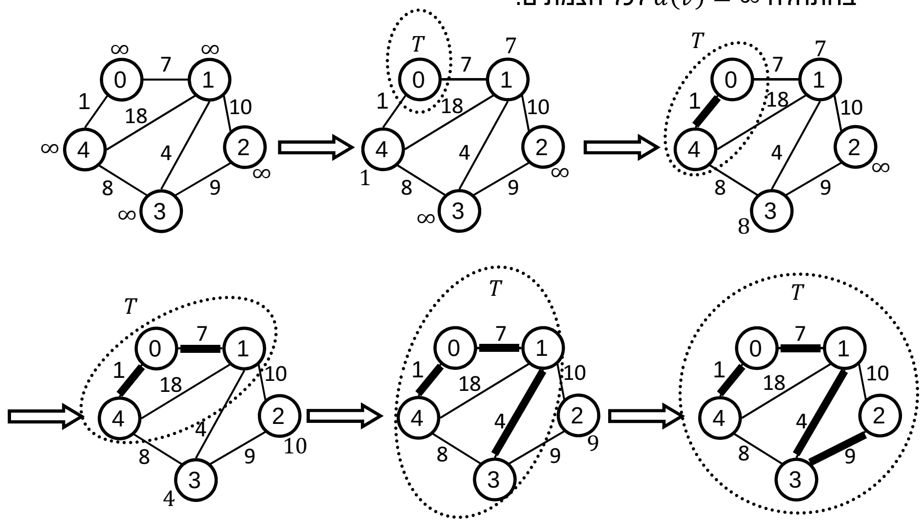
כל צומת $v \notin T$ ישמור:
- קשת (u,v) קלה ביותר המחברת בין v וצומת ב-T (Null אם אין קשת כזו)
- $d(v)$ – משקל הקשת (u,v) הנ"ל (∞ אם אין קשת כזו)

הפעולות הנדרשות	
	אתחול: לכל צומת v יש מפתח $d(v) = \infty$.
	כל עוד העץ לא מכיל את כל הצמתים:
מצא צומת עם $d(i)$ מינימלי והוסף לעץ	1. הוסף צומת $i \notin T$ עם $d(i)$ מינימלי לעץ.
	2. אם i שמר קשת (i,j), הוסף אותה לעץ T.
	3. לכל קשת (i,k):
	אם $k \notin T$ וגם $d(k) > w(i,k)$ אז
$d(k)$ הקטנת	$d(k) \leftarrow w(i,k)$. k יחזיק את הקשת (i,k).

הערה: הצומת הראשון שמתווסף לעץ הוא הצומת s מהשקף הקודם.

דוגמת הרצה

לכל צומת v שאינו ב-T, הערך $d(v)$ מסומן ליד הצומת.
בהתחלה $d(v) = \infty$ לכל הצמתים.



מימוש עם רשימה ומטריצת סמיכויות

נשמור רשימת צמתים שאינם ב-T.

מציאת צומת i עם קשת קלה ביותר שיוצאת מ-T:
- מעבר על הצמתים שאינם ב-T.
- מציאת הצומת עם $d(i)$ מינימלי והוצאתו מהרשימה.

סה"כ זמן: $O(n)$.

מעבר על הקשתות (i,k):
- ע"י מעבר על השורה ה-i במטריצה.
- כל עדכון של $d(k)$ מתבצע בזמן $O(1)$.

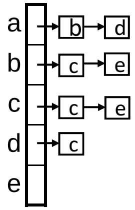
סה"כ זמן: $O(n)$.

	0	1	2	3	4
0	0	7	0	0	1
1	7	0	10	4	18
2	0	10	0	9	0
3	0	4	9	0	8
4	1	18	0	8	0

סה"כ:
כל אחת מ-n האיטרציות לוקחת $O(n)$ זמן, ומכאן שהסיבוכיות היא $O(n^2)$.

מימוש עם רשימה ורשימת סמיכויות

מציאת והוצאת צומת i עם קשת קלה ביותר שיוצאת מ-T: מעבר על הצמתים שאינם ב-T ומציאת הצומת עם $d(v)$ מינימלי והוצאתו מהרשימה. זמן: $O(n)$.



מעבר על הקשתות (i,k):
ע"י מעבר על שכני i. כל עדכון של $d(k)$ מתבצע בזמן $O(1)$.
זמן: $O(degree(i))$.

סה"כ:
לכל צומת i נבצע איטרציה אחת, שתיקח $O(n+degree(i))$.
מכאן שזמן הריצה הכולל הוא

$$\sum_{i=1}^{n} O\big(n+degree(i)\big) = O(n^2+\sum_{i=1}^{n} degree(i)) = O(n^2+m)$$

אבחנה:
צוואר הבקבוק הוא הוצאת הצומת בקצה של קשת קלה ביותר שיוצאת מ-T.

- נתאר מימוש עם מבני נתונים בעלי זמן ריצה טוב יותר לפעולה זו.
- נמשיך להשתמש ברשימת סמיכויות כך שמעבר על קשתות הגרף ידרוש $O(m)$ זמן ולא $O(n^2)$.

מימוש עם ערימה – ניתוח זמנים

נשמור את הצמתים שאינם ב-T בערימה ונוסיף לכל צומת ברשימת הסמיכויות מצביע למיקומו בערימה.
מהן הפעולות המתבצעות על הערימה?
תחילה בונים ערימה בזמן $O(n)$. בנוסף,

בכל שלב:	זמן הפעולה
• מציאת מינימום	$O(1)$
• הוצאת מינימום	$O(\log n)$
• הקטנת מפתח של צומת בערימה, בהינתן מצביע אליו.	$O(\log n)$

סיבוכיות המימוש:
כל צומת נמצא ומוצא מהערימה פעם אחת. פעולות אלו עולות סה"כ $O(n\log n)$.
כל קשת גורמת עדכון פעם אחת לכל היותר. העדכונים עולים סה"כ $O(m\log n)$.
לכן פעולות הערימה מתבצעות בזמן $O(m\log n+n\log n)$. זהו גם הזמן הכולל של מימוש זה.

מתי השיפור עוזר (יחסית לזמן $O(n^2)$)?

מימוש עם ערימת פיבונאצ'י – ניתוח זמנים

קיים מימוש של ערימה הנקרא ערימת פיבונאצ'י, התומך בבניית ערימה בגודל n בזמן $O(n)$ ובשאר פעולות הערימה בסיבוכיות הבאה:

בכל שלב:	זמן הפעולה (משוערך)
• מציאת מינימום	$O(1)$
• הוצאת מינימום	$O(\log n)$
• הקטנת מפתח של צומת בערימה	$O(1)$

סיבוכיות המימוש:
כל צומת נמצא ומוצא מהערימה פעם אחת. פעולות אלו עולות סה"כ $O(n\log n)$.
כל קשת גורמת עדכון פעם אחת לכל היותר. העדכונים עולים סה"כ $O(m)$.
לכן פעולות הערימה מתבצעות בזמן $O(m+n\log n)$. זהו גם הזמן הכולל של מימוש זה.

הערה היסטורית: נציין שבזמן המצאת ערימת פיבונאצ'י מימוש זה שיפר אסימפטוטית את האלגוריתם הטוב ביותר הידוע לחישוב עץ פורש מינימום. זו דוגמה למקרה בו מבנה נתונים עם סיבוכיות משוערכת טובה נותן ביצועים טובים יותר מאשר מבני נתונים עם סיבוכיות טובה יותר במקרה הגרוע.

עוד על ערימות פיבונאצ'י בספר הלימוד.