אלגוריתמים חמדנים

פרק 4 ב-Kleinberg/Tardos

בעיות ניהול משאבים
קידוד Huffman
בעיית עץ פורש מינימום

הרעיון: בונים את הפלט בהדרגה. בכל צעד בוחרים החלטה אופטימלית לאותו הצעד ולא משנים אותה.

בדרך כלל זהו רעיון גרוע!

בעיית שיבוץ משימות

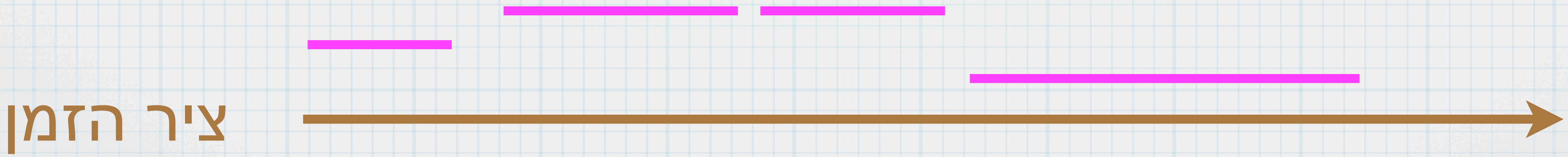
<u>הקלט</u>: קבוצת משימות J. כל משימה $j \in J$ מאופיינת על ידי זמן התחלה $s(j)$ וזמן סיום $t(j)$.

<u>הפלט</u>: תת-קבוצה $J' \subseteq J$ של משימות לא חופפות בזמן, שהיא מקסימלית בגודלה.

בעיית שיבוץ משימות

<u>הקלט</u>: קבוצת משימות J. כל משימה $j \in J$ מאופיינת על ידי זמן התחלה $s(j)$ וזמן סיום $t(j)$.

<u>הפלט</u>: תת-קבוצה $J' \subseteq J$ של משימות לא חופפות בזמן, שהיא מקסימלית בגודלה.



אלגוריתם החמדן

1. בחר משימה.
2. זרוק את כל המשימות החופפות לה.
3. אם עדיין יש משימות שלא נבחרו או נזרקו:
   חזור לצעד מספר 1.

מהו כלל בחירה טוב?

לפי סדר זמן ההתחלה:

לפי משך המשימה:

לפי מספר המשימות החופפות:

אלגוריתם החמדן

1. בחר משימה.
2. זרוק את כל המשימות החופפות לה.
3. אם עדיין יש משימות שלא נבחרו או נזרקו:
חזור לצעד מספר 1.

מהו כלל בחירה טוב?

לפי סדר זמן ההתחלה:

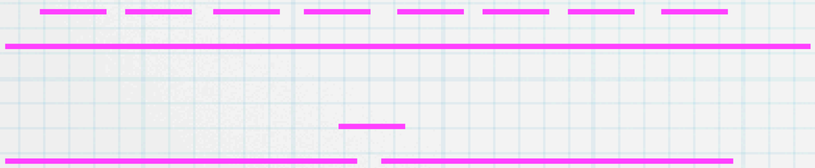
לפי משך המשימה:

לפי מספר המשימות החופפות:

אלגוריתם החמדן

1. בחר משימה.
2. זרוק את כל המשימות החופפות לה.
3. אם עדיין יש משימות שלא נבחרו או נזרקו:
חזור לצעד מספר 1.

מהו כלל בחירה טוב?



לפי סדר זמן ההתחלה:

לפי משך המשימה:

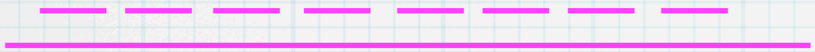
לפי מספר המשימות החופפות:

אלגוריתם החמדן

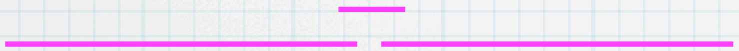
1. בחר משימה.
2. זרוק את כל המשימות החופפות לה.
3. אם עדיין יש משימות שלא נבחרו או נזרקו:
חזור לצעד מספר 1.

מהו כלל בחירה טוב?

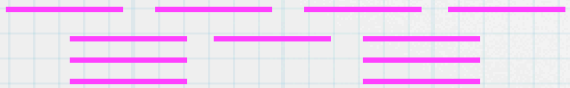
לפי סדר זמן ההתחלה:



לפי משך המשימה:



לפי מספר המשימות החופפות:



אלגוריתם החמדן לשיבוץ משימות

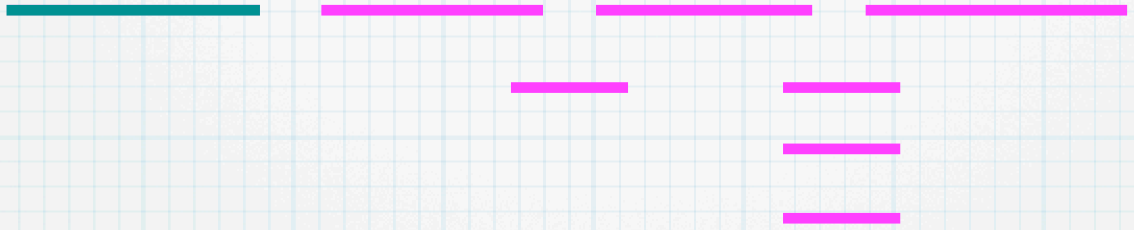
נבחר לפי סדר זמן הסיום!

```
sort(J,t)
while J ≠ ∅ do
  next_job ← first entry in J
  write(next_job)
  for every j ∈ J s.t. s(j) < t(next_job) do
    J ← J ∖ {j}
  end for
end while
```

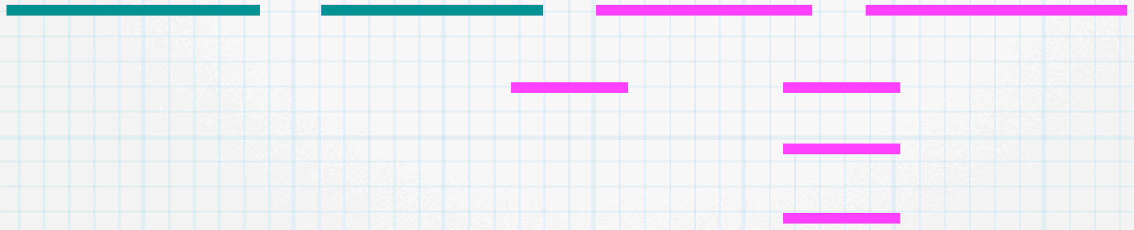
הדגמת הרצה

הדגמת הרצה

הדגמת הרצה



הדגמת הרצה



הדגמת הרצה

הדגמת הרצה

הדגמת הרצה

_____ _____ _____ _____

הדגמת הרצה

הוכחת נכונות

נסמן ב-Alg את פלט האלגוריתם וב-Opt קבוצת משימות מקסימלית כלשהי.

משפט: $|Alg| = |Opt|$

<u>הוכחה</u>

נרשום את המשימות ב-Alg וב-Opt לפי סדר עולה של ערכי t:

$$Alg = \{i_1,i_2,i_3,...,i_k\}$$
$$Opt = \{j_1,j_2,j_3,...,j_m\}$$

נוכיח באינדוקציה שלכל $q=1,2,...,k$ מתקיים

$$t(i_q) \leq t(j_q)$$

המשך ההוכחה

בסיס האינדוקציה: $q = 1$, לפי כלל הבחירה.
נניח נכונות עבור $q-1$, כלומר: $t(i_{q-1}) \leq t(j_{q-1})$

שימו לב כי $s(j_q) \geq t(j_{q-1})$
ולכן, כאשר האלגוריתם הוסיף את i_q, j_q
עדיין לא נזרקה, ולכן, לפי כלל הבחירה
$t(i_q) \leq t(j_q)$
כנדרש.

עכשיו לא ייתכן כי $m > k$, אחרת
$s(j_{k+1}) \geq t(j_k) \geq t(i_k)$
וזו סתירה לתנאי הסיום של האלגוריתם. $\therefore$

סיבוכיות הזמן

החלק ה"יקר" הוא המיון שלוקח $O(n \log n)$.

שאר האלגוריתם ניתן למימוש בזמן $O(n)$.
(זאת בהנחה שמיינו פעמיים, גם לפי ערכי s.)

בעיית מיזעור האחור

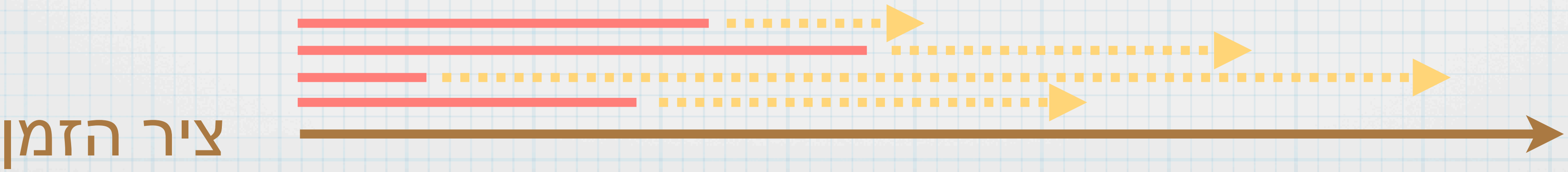
הקלט: משימה $j \in J$ מאופיינת על ידי משך ביצוע $p(j)$ ותאריך יעד $d(j)$.

הפלט: מועד התחלה $s(j)$ לכל משימה, כך שהמשימות תתבצענה ללא חפיפה, והאחור המירבי ביחס לתאריך היעד הוא מינימלי.

האחור המירבי:

$$L = \max_j L(j)$$

$$L(j) = \max\{0, s(j)+p(j)-d(j)\}$$



אלגוריתם החמדן למיזעור האחור

אפשר להציג פתרון לבעייה כפרמוטציה על המשימות,
שתשובצנה בסדר הפרמוטציה בזו אחר זו ללא רווחים.

נשתמש בשיבוץ לפי סדר תאריכי היעד:

```
sort(J,d)
s(0) ← 0
for j ← 1 to |J| do
  s(j) = s(j-1) + p(j-1)
end for
```

הוכחת נכונות

משפט: האלגוריתם מוציא שיבוץ אופטימלי.

<u>הוכחה</u>

נתבונן בשיבוץ אופטימלי כלשהו, פרמוטציה

$j_1, j_2, j_3, \ldots, j_n$

אם שיבוץ זה שונה מפלט האלגוריתם, אזי יש אינדקס i עבורו

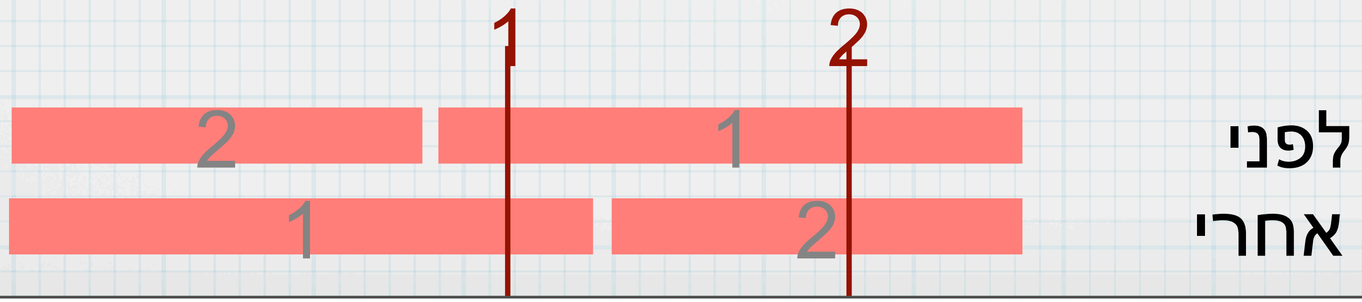
$$d(j_i) \geq d(j_{i+1})$$

המשך ההוכחה

אם נחליף את הסדר בין זוג המשימות הללו, לא נגדיל את ערך הפתרון. כדי לראות זאת,
נסמן ב-’ את המשתנים בפתרון השני.

$$L'(j_i) = \max\{0, s(j_{i+1})+p(j_{i+1})-d(j_i)\} \le \max\{0, s(j_{i+1})+p(j_{i+1})-d(j_{i+1})\} = L(j_{i+1})$$

$$L'(j_{i+1}) = \max\{0, s(j_i)+p(j_{i+1})-d(j_{i+1})\} \le \max\{0, s(j_{i+1})+p(j_{i+1})-d(j_{i+1})\} = L(j_{i+1})$$



המשך ההוכחה

נמשיך להחליף זוגות סמוכים כל עוד הסדר שונה מסדר האלגוריתם. המחיר לא עלה בכל החלפה, כך שמחיר הפתרון של האלגוריתם אינו גדול יותר ממחיר פתרון אופטימלי. ∴

סיבוכיות הזמן

כמו בבעיית השיבוץ הקודמת, הגורם הדומיננטי הוא זמן המיון, שלוקח $O(n \log n)$.

בעיית הדיפדוף

- זיכרון וירטואלי של מחשב מחולק ל"דפים".
- למחשב יש זיכרון מטמון שמכיל k דפים.
- תוכנית מחשב גדולה ניגשת לסדרה ארוכה של דפים לפי סדר נתון, למשל:

a b a b c d a e d b c f d e a b c d e f

- אם דף אינו במטמון, יש להחליף דף אחר במטמון בדף המבוקש.
- כמובן ייתכן שהדף שזה עתה החלפנו יידרש בעתיד, ונצטרך להביאו מחדש.
- רוצים למזער את מספר ההחלפות.

הצגה פורמלית של בעיית הדיפדוף

הקלט: סדרה סופית P של מספרים חיוביים (מספרי דפים מבוקשים).

הפלט: סדרה S של מספרים באורך זהה לאורך הקלט.

- בסדרה זו, ערך חיובי מציין מספר דף שיש לזרוק מהמטמון, וכל ערך אחר מציין שאין צורך לזרוק דף מהמטמון בצעד זה.
- הסדרה צריכה להכיל מספר מינימלי של ערכים חיוביים מבין כל הסדרות החוקיות.
- סדרה היא חוקית אם לכל צעד i, המטמון מכיל לכל היותר k דפים.

אלגוריתם Belady לבעיית הדיפדוף

בכל צעד שדורש החלפה, נזרוק מהמטמון את הדף שיידרש בעתיד הרחוק ביותר (מבין כל הדפים במטמון).

מימוש האלגוריתם

לכל איבר i ב-P נחשב ערך next(i) שהוא המקום הבא בסדרה שבה מופיע דף זה:

```
for p ← 1 to #pages do
  last[p] ← 0
end for
for i ← 1 to |P| do
  next(i) ← ∞
  if last[P(i)] > 0 then
    next(last[P(i)]) ← i
  end if
  last[P(i)] ← i
end for
```

המקום האחרון בו נראה דף p

המשך המימוש

```
cache ← ∅    המטמון
for i ← 1 to |P| do
  S(i) ← 0
  if P(i) ∉ cache and |cache| = k then
    S(i) ← argmax{next(p): p ∈ cache}
    cache ← cache ∖ {S(i)}
  end if
  cache ← cache ∪ {P(i)}
end for
```

זו פרוצדורה שדורשת מימוש

תרגיל בית

1. הוכיחו את נכונות האלגוריתם של Belady.
2. נתחו את סיבוכיות הזמן והמקום.

קידוד Huffman

- נתון קובץ ובו n תווים מא"ב בגודל k.
- קידוד unicode של הקובץ דורש 16n ביטים.
- קידוד באורך קבוע ידרוש n log k ביטים.
- כדי לחסוך במקום, נשתמש בקידוד באורך משתנה - תווים שכיחים יותר יקודדו בפחות ביטים מתווים נדירים.
- לפיענוח קל נדרוש: לכל תו, קידודו איננו רישא של הקידוד של תו אחר.

דוגמה

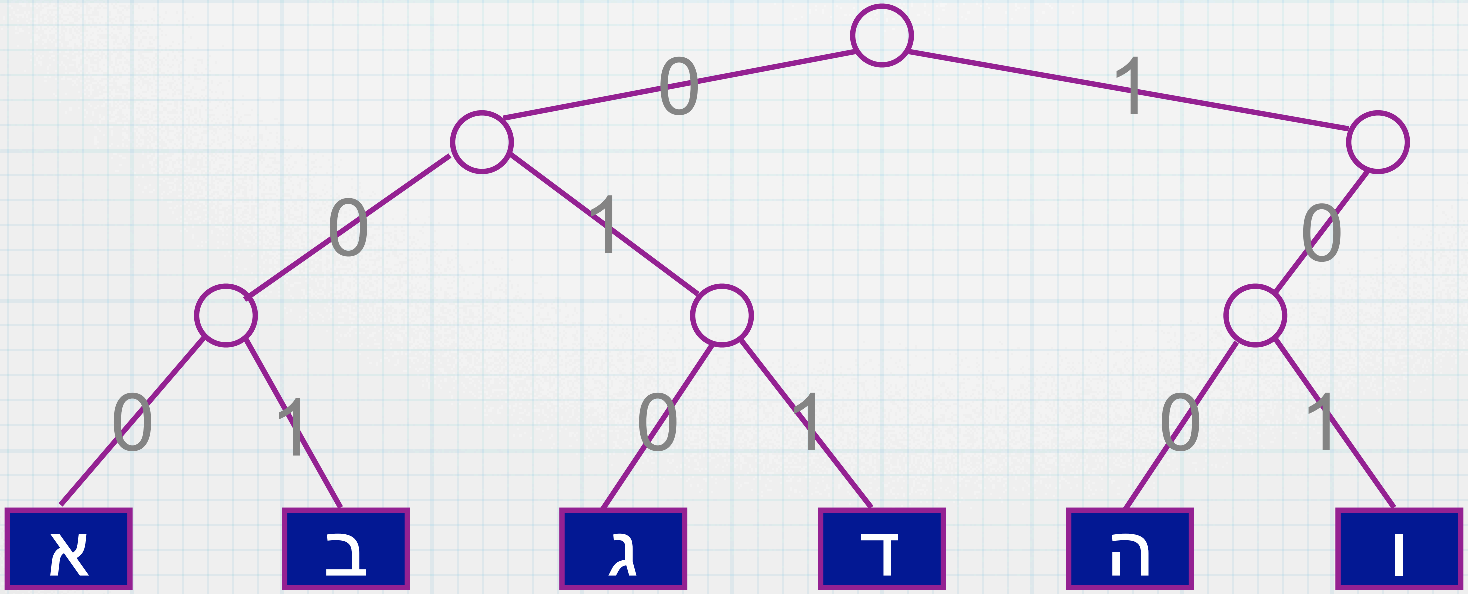
בקובץ 100,000 תווים מהא"ב {א,ב,ג,ד,ה,ו}.

	א	ב	ג	ד	ה	ו	סה"כ
שכיחות	45%	13%	12%	16%	9%	5%	100%
אורך קבוע	000	001	010	011	100	101	300,000
אורך משתנה	0	101	100	111	1101	1100	224,000

ייצוג הקידוד

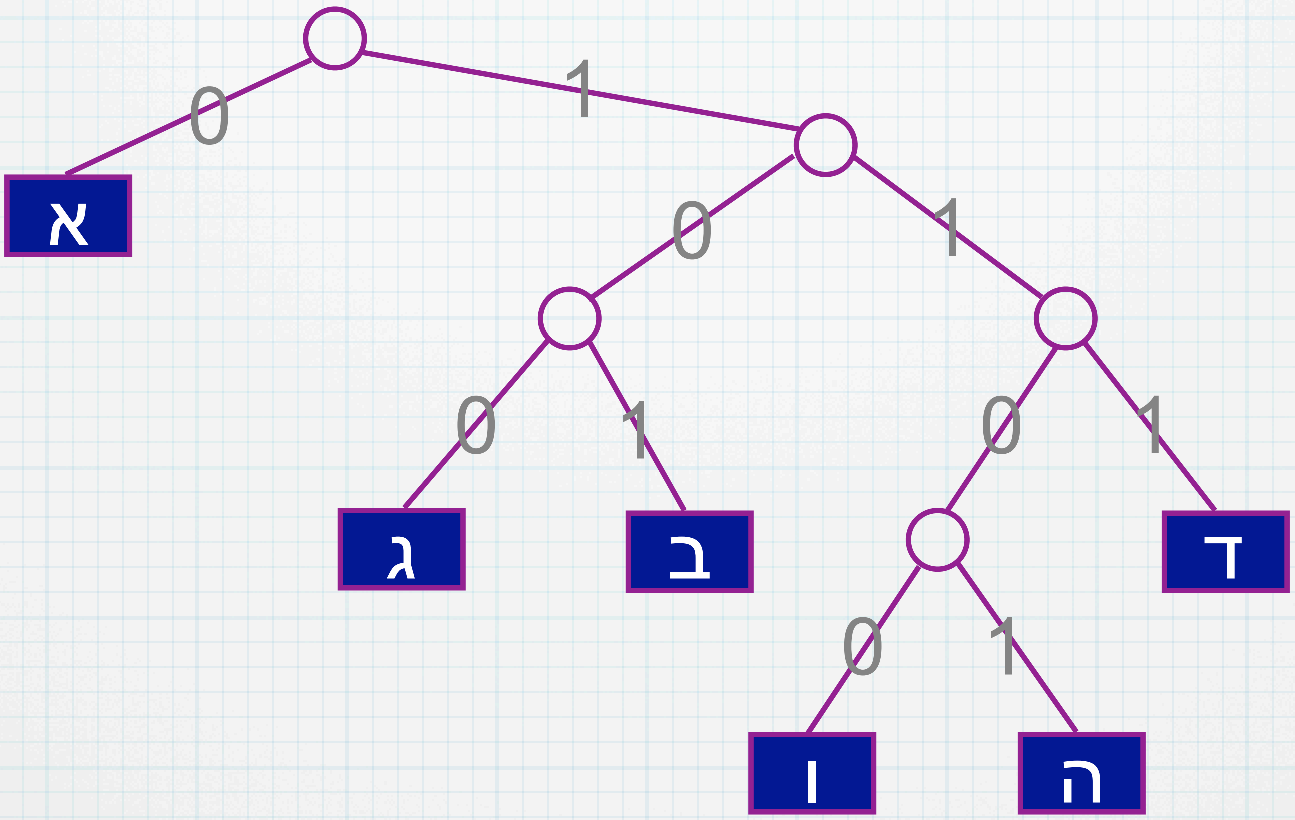
נייצג את הקידוד על ידי עץ בינארי:

- העלים יסומנו על ידי התווים.
- קשת לבן שמאלי תסומן 0.
- קשת לבן ימני תסומן 1.



העץ המתאים לקוד קבוע (3 ביטים)

ייצוג הקידוד (המשך)



העץ המתאים לקוד אופטימלי

ייצוג הקידוד (המשך)

טענה: העץ המתאים לקוד אופטימלי הוא תמיד עץ בינארי מלא (כלומר, לכל צומת שאינו עלה יש שני ילדים).

נסמן ב-C את הא"ב.
לכן בעץ אופטימלי יש $|C|$ עלים ו- $|C|-1$ צמתים פנימיים.

עבור $c \in C$ נסמן ב-$f(c)$ את מספר הפעמים ש-c מופיע בקובץ.

עבור עץ T נסמן ב-$d_T(c)$ את עומק העלה המסומן ב-c ב-T.

המחיר של T הוא:

$$B(T) = \sum_{c \in C} f(c) \cdot d_T(c)$$

בניית קידוד Huffman

בונים את העץ ”מלמטה למעלה“.

מתחילים מ-|C| עלים ומבצעים |C|-1 פעולות מיזוג ליצירת העץ הסופי.

לכל צומת x נחזיק מצביעים left[x] ו-right[x] לילד שמאלי וימני, בהתאמה.

בניית קידוד Huffman

צריך לממש את H במבנה נתונים שמאפשר שליפה מהירה של המינימום, מחיקה והוספה של איברים.

```
Huffman(C,f)
  H ← C
  for i ← 1 to |C|-1 do
    z ← צומת חדש בעץ
    left[z] ← x ← argmin{f(c): c ∈ H}
    right[z] ← y ← argmin{f(c): c ∈ H∖{x}}
    f(z) ← f(x) + f(y)
    H ← H ∖ {x,y} ∪ {z}
  end for
return האיבר הבודד ב-H
```

דוגמת הרצה

5	9	12	13	16	45

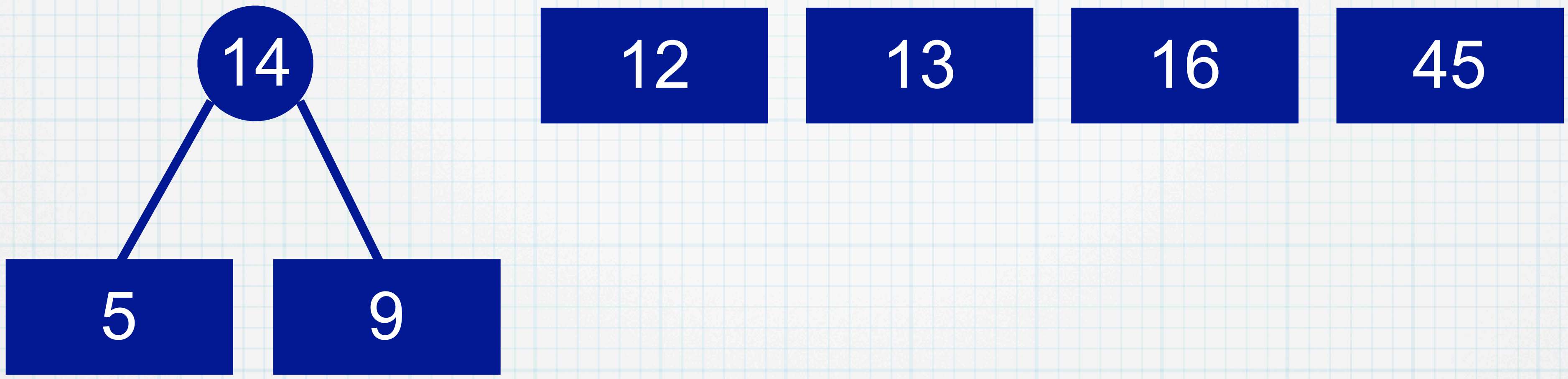
דוגמת הרצה

5	9	12	13	16	45

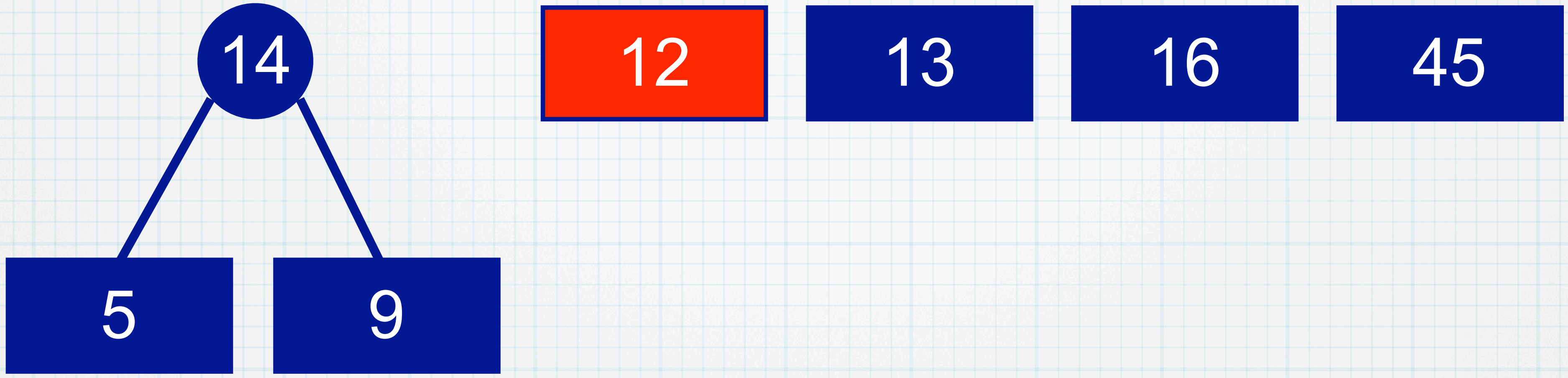
דוגמת הרצה

5	9	12	13	16	45

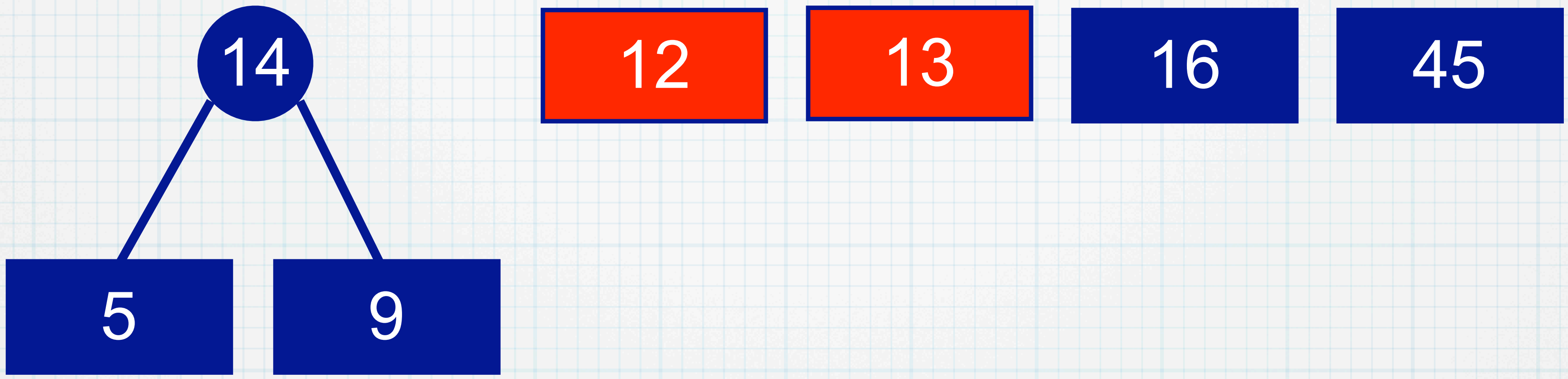
דוגמת הרצה



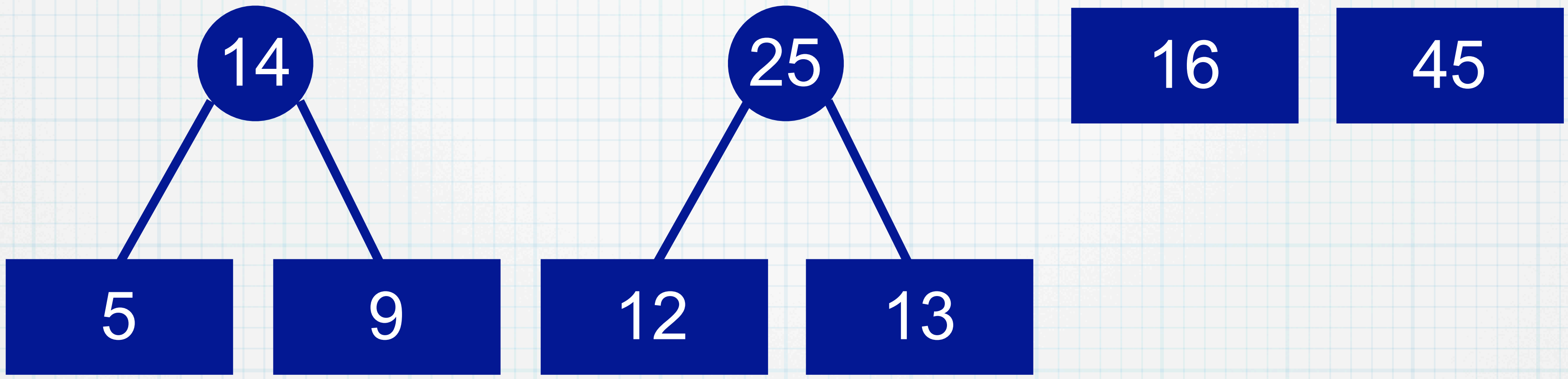
דוגמת הרצה



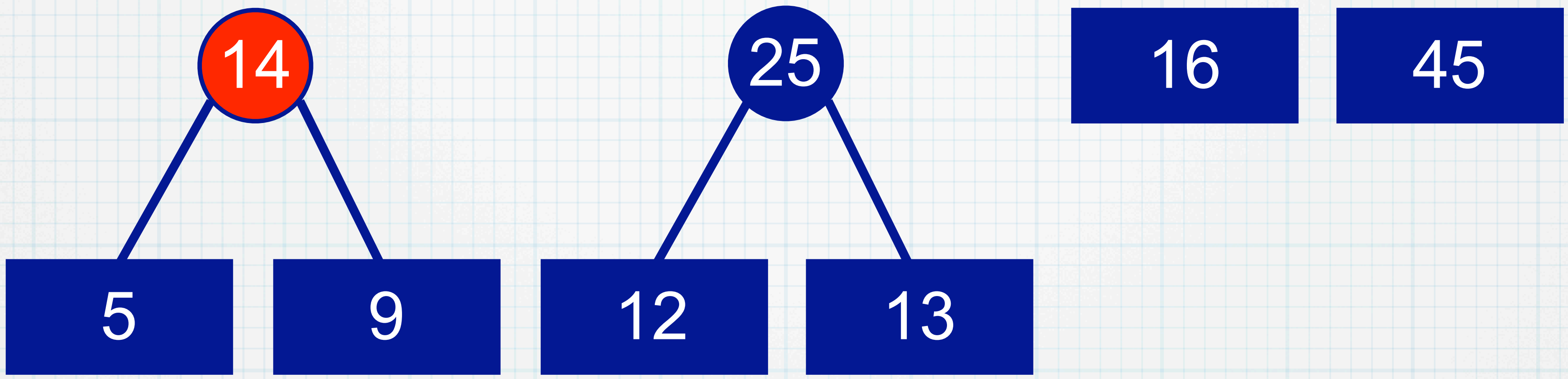
דוגמת הרצה



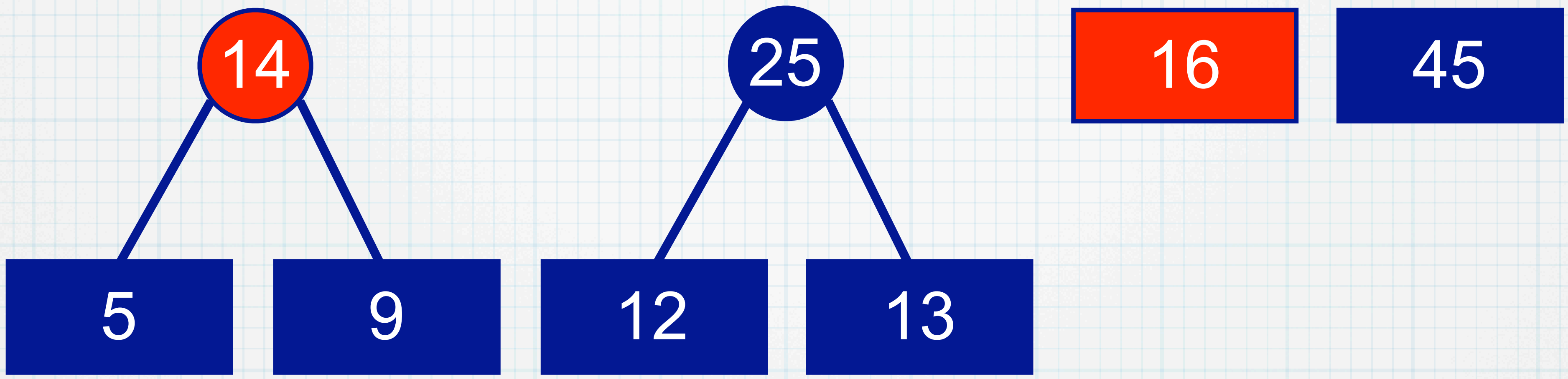
דוגמת הרצה



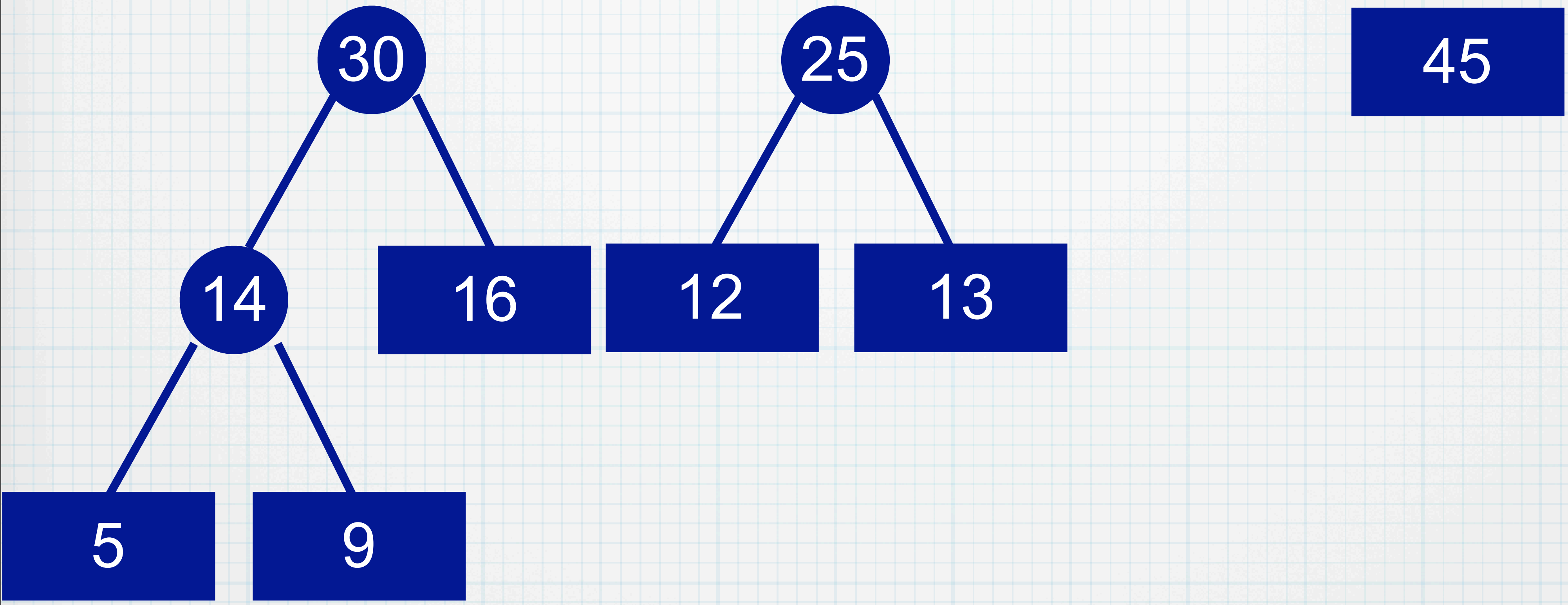
דוגמת הרצה



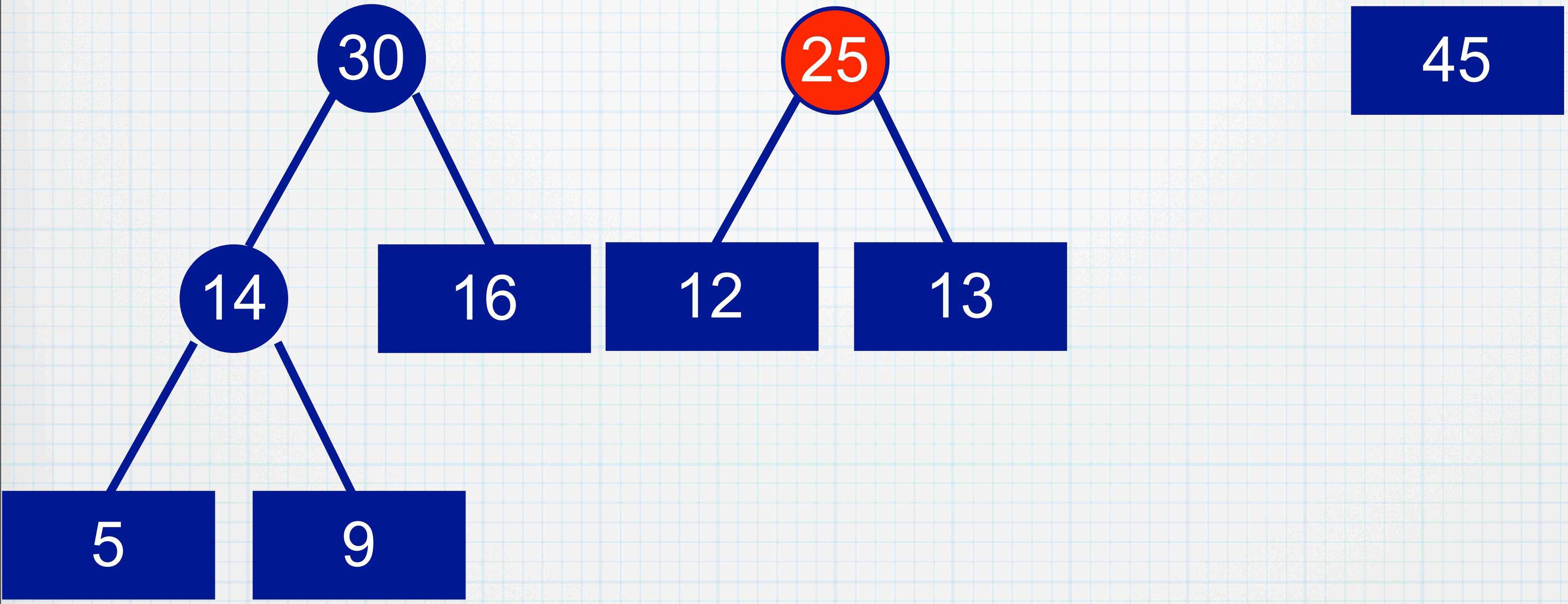
דוגמת הרצה



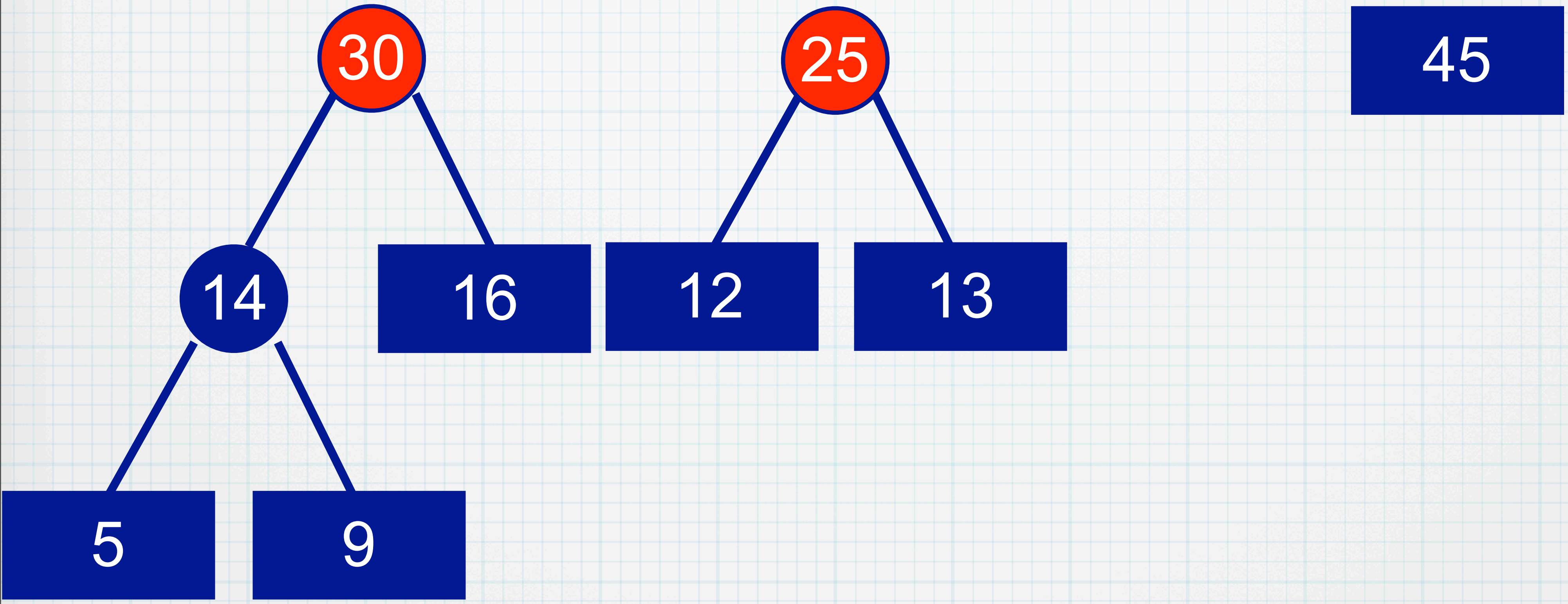
דוגמת הרצה



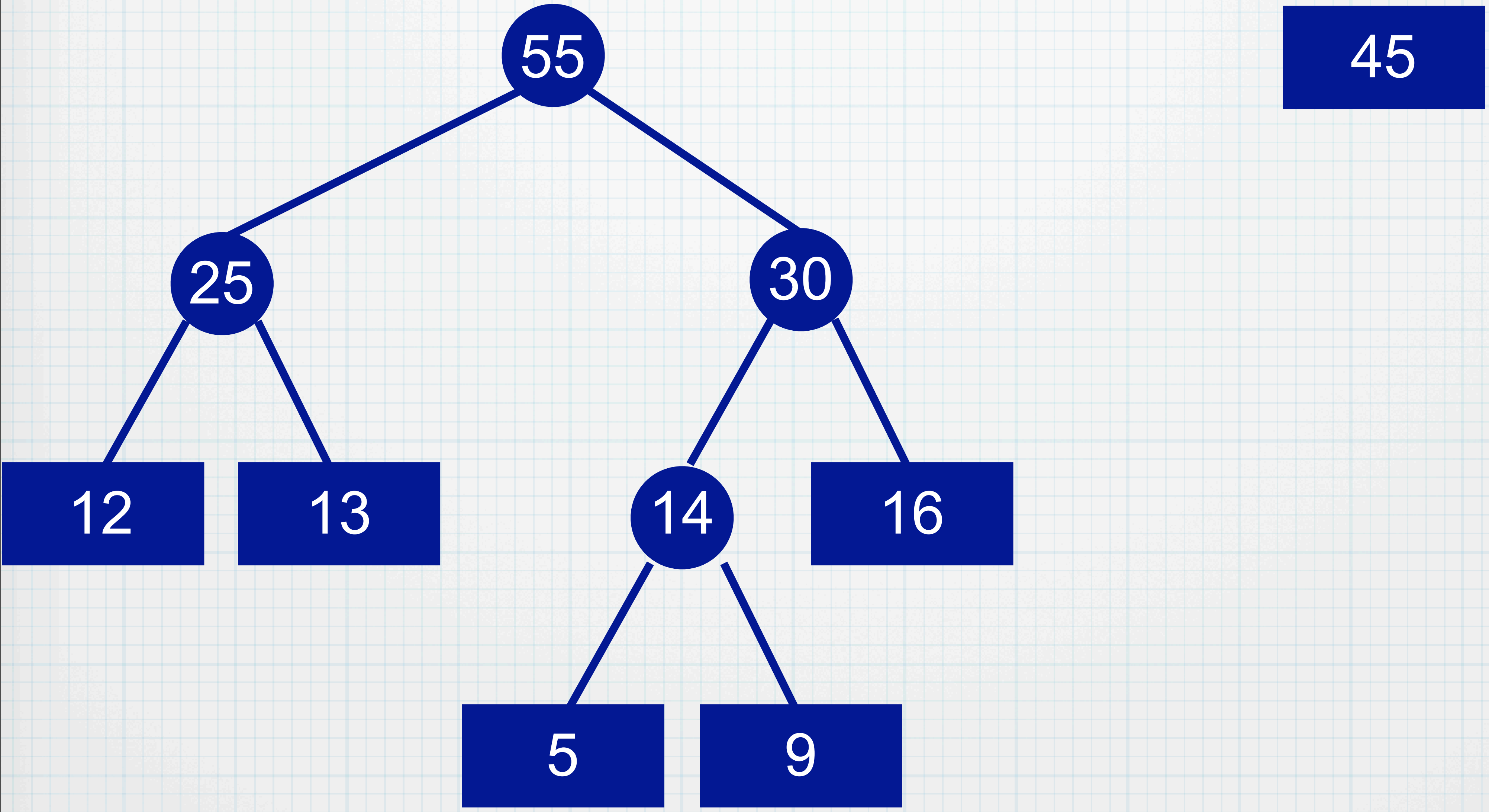
דוגמת הרצה



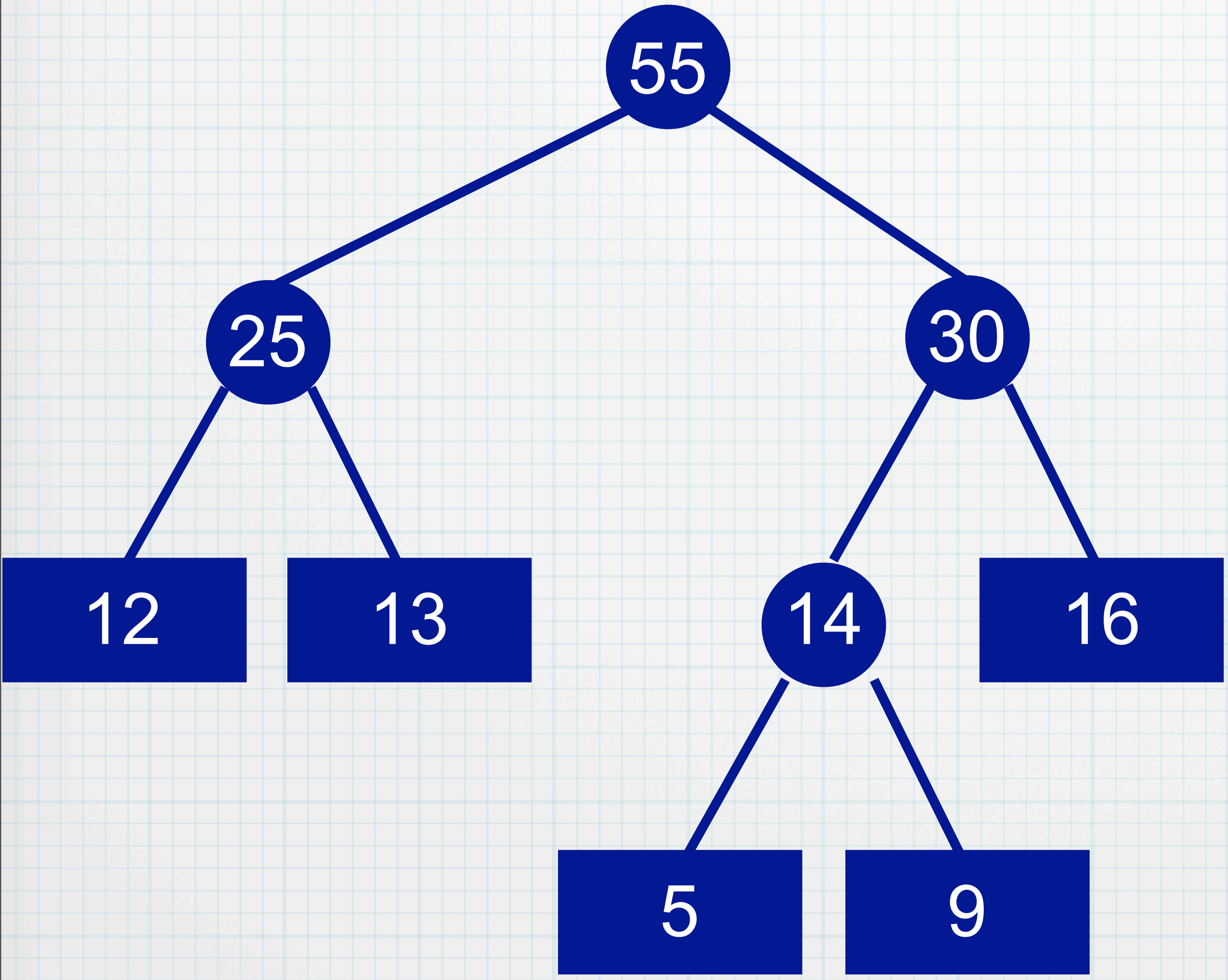
דוגמת הרצה



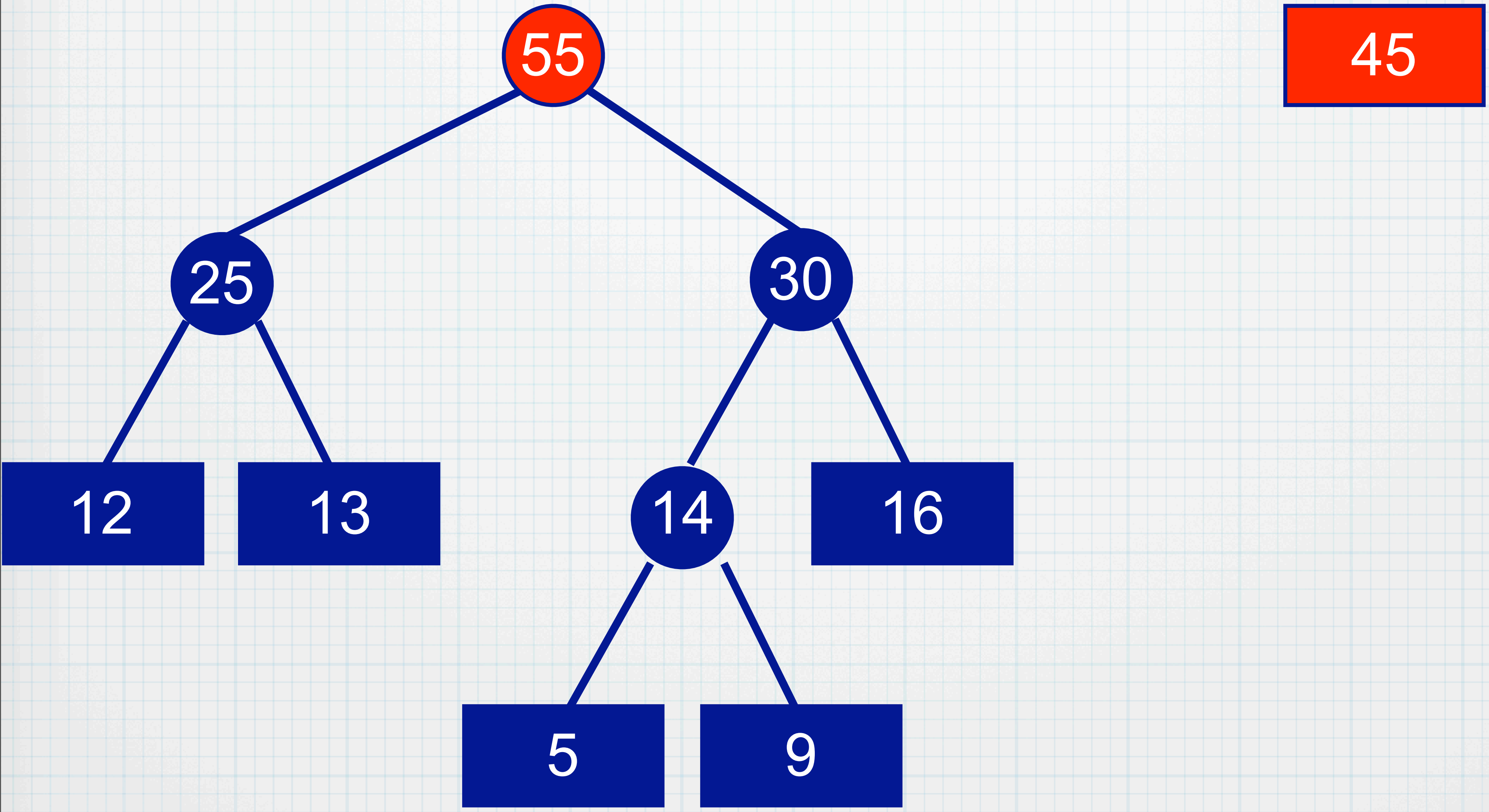
דוגמת הרצה



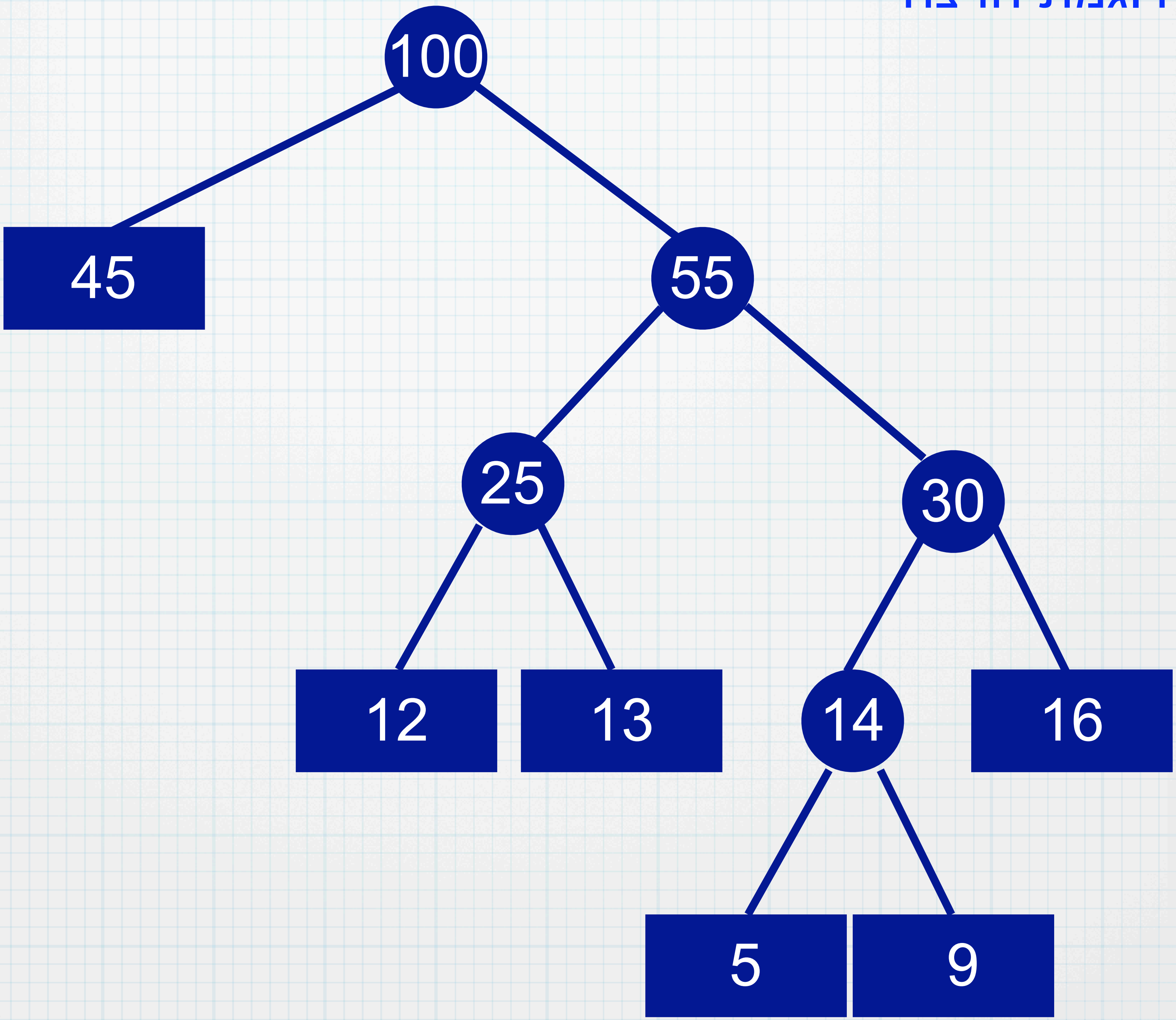
דוגמת הרצה



דוגמת הרצה



דוגמת הרצה



סיבוכיות

מימוש פשוט

בלולאה מתבצעות $|C|-1$ איטרציות.

באיטרציה ה-i מוצאים פעמיים מינימום. מספר הערכים ≥ $|C| + 1 - i$

זמן הריצה: $O(|C|^2)$

מימוש מתוחכם

נממש את H על ידי ערמה (heap).

הבנייה: $O(|C|)$

מציאת מינימום/הוספה או הורדה של איבר: $O(\log |C|)$

זמן הריצה: $O(|C| \log |C|)$

הוכחת נכונות

נראה נכונות באינדוקציה על גודל הא"ב $|C|$

בסיס: א"ב בגודל 2 - האלג' מקודד כל תו על ידי ביט אחד.

צעד האינדוקציה: נניח נכונות עבור א"ב בגודל $|C|-1$.
אחרי איטרציה אחת, נותרים עם $H = C \setminus \{x,y\} \cup \{z\}$,
כאשר x,y הם התווים עם מספר המופעים המזערי בקובץ, z הוא
תו חדש שלא מופיע בא"ב, ו-$f(z) = f(x) + f(y)$.

לפי הנחת האינדוקציה, האלג' בונה עץ אופטימלי T' עבור H.
הפלט T הוא העץ T' שבו מוסיפים את x,y כילדים של z.

הוכחת נכונות (המשך)

למה: קיים עץ אופטימלי T^* עבורו x,y הם עלים אחים בעומק מירבי ב-T^*.

הוכחה

יהי T עץ קידוד אופטימלי כלשהו עם עלים אחים בעומק מירבי u,v. נניח ש-$f(u) \le f(v)$ וכן ש-$f(x) \le f(y)$. נחליף בין u ל-x ובין v ל-y כדי לקבל את T^*.

$$B(T) - B(T^*) = \sum_{c \in C} f(c)\, d_T(c) - \sum_{c \in C} f(c)\, d_{T^*}(c)$$

$$= f(x)\, d_T(x) + f(u)\, d_T(u) - f(x)\, d_{T^*}(x) - f(u)\, d_{T^*}(u) + f(y)\, d_T(y) + f(v)\, d_T(v) - f(y)\, d_{T^*}(y) - f(v)\, d_{T^*}(v)$$

$$= f(x)\, d_T(x) + f(u)\, d_T(u) - f(x)\, d_T(u) - f(u)\, d_T(x) + f(y)\, d_T(y) + f(v)\, d_T(v) - f(y)\, d_T(v) - f(v)\, d_T(y)$$

$$= (f(u)-f(x))\cdot(d_T(u)-d_T(x)) + (f(v)-f(y))\cdot(d_T(v)-d_T(y)) \ge 0$$

הוכחת נכונות (המשך)

יהי ”T העץ *T שבו הסרנו את העלים האחים x,y. נסמן את ההורה שלהם ב-z. זהו עץ קידוד עבור $H = C \setminus \{x,y\} \cup \{z\}$

לפי הנחת האינדוקציה:

$$B(T^*) = B(T'') + f(x) + f(y) \geq B(T') + f(x) + f(y) = B(T)$$

כי

$$f(x)d_{T^*}(x)+f(y)d_{T^*}(y) = (f(x)+f(y))\cdot(d_{T''}(z)+1)$$
$$= f(z)d_{T''}(z)+f(x)+f(y)$$

$$f(x)d_T(x) + f(y)d_T(y) = (f(x)+f(y))\cdot(d_{T'}(z)+1)$$
$$= f(z)d_{T'}(z) + f(x) + f(y)$$

$\therefore$

עץ פורש מינימום (עפ"מ)

<u>הקלט</u>: גרף לא מכוון וקשיר $G=(V,E)$, בייצוג של רשימת סמיכויות. משקלות חיוביים על הקשתות
$w:E \rightarrow \mathbb{N}$

<u>הפלט</u>: עץ T שהוא תת-גרף של G, מכיל את כל V, והמשקל הכולל של קשתותיו מינימלי.

אלגוריתם Kruskal לעפ"מ

נעבור על הקשתות בסדר לא יורד של משקל.
נוסיף לעץ כל קשת שלא סוגרת מעגל בתורה.

האלגוריתם של Kruskal לעפ"מ

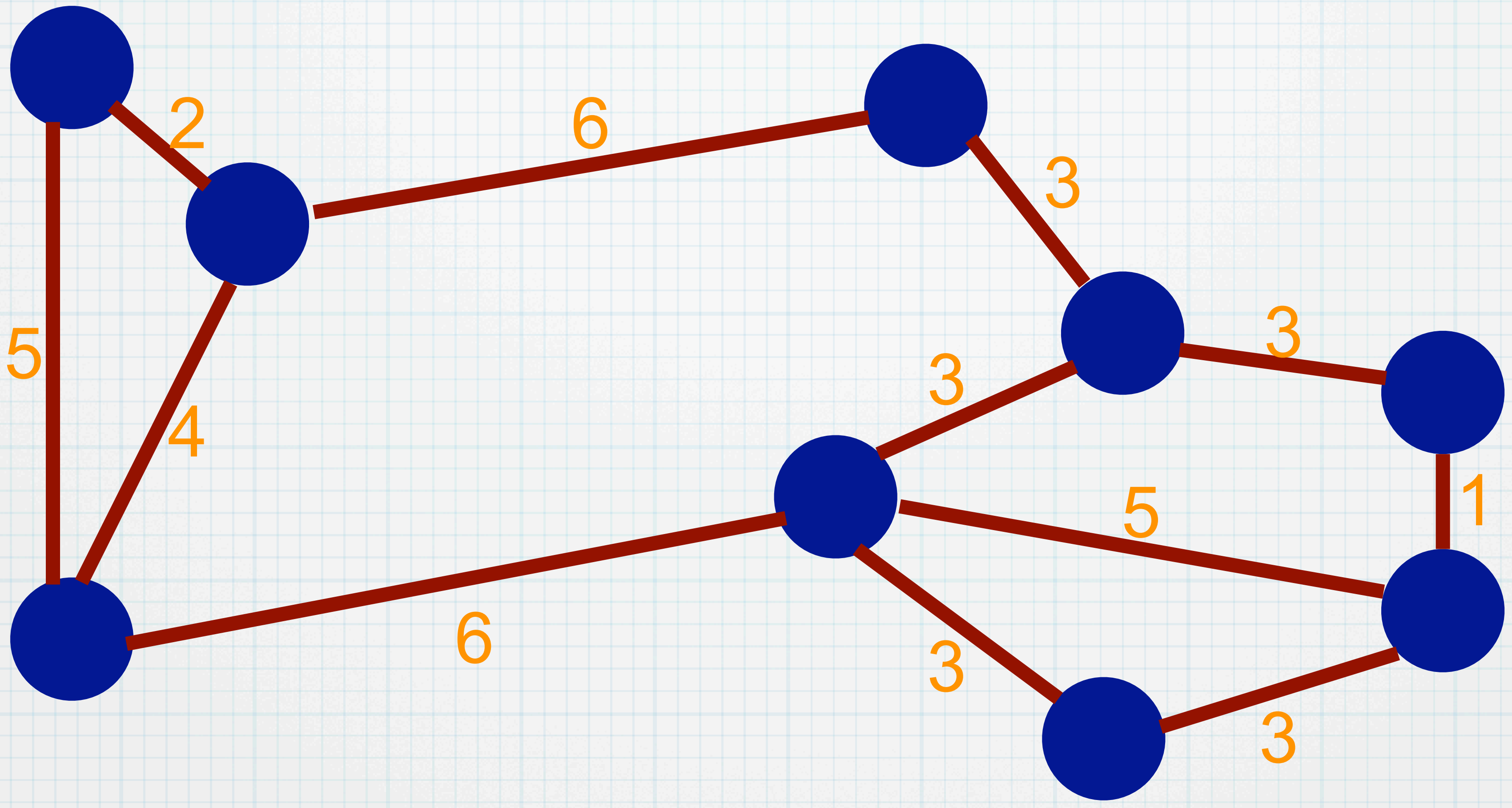
```
Kruskal_MST(G,w)
  T ← (V,∅)
  CC ← {{x}: x ∈ V}
  sort(E,w)
  for each {x,y} ∈ E do
    if C(x) ≠ C(y) then
      E(T) ← E(T) ∪ {{x,y}}
      CC ← CC ∖ {C(x),C(y)} ∪ {C(x) ∪ C(y)}
    end if
  end for
return T
```

המשתנה T יחזיק את העץ ההולך ונבנה

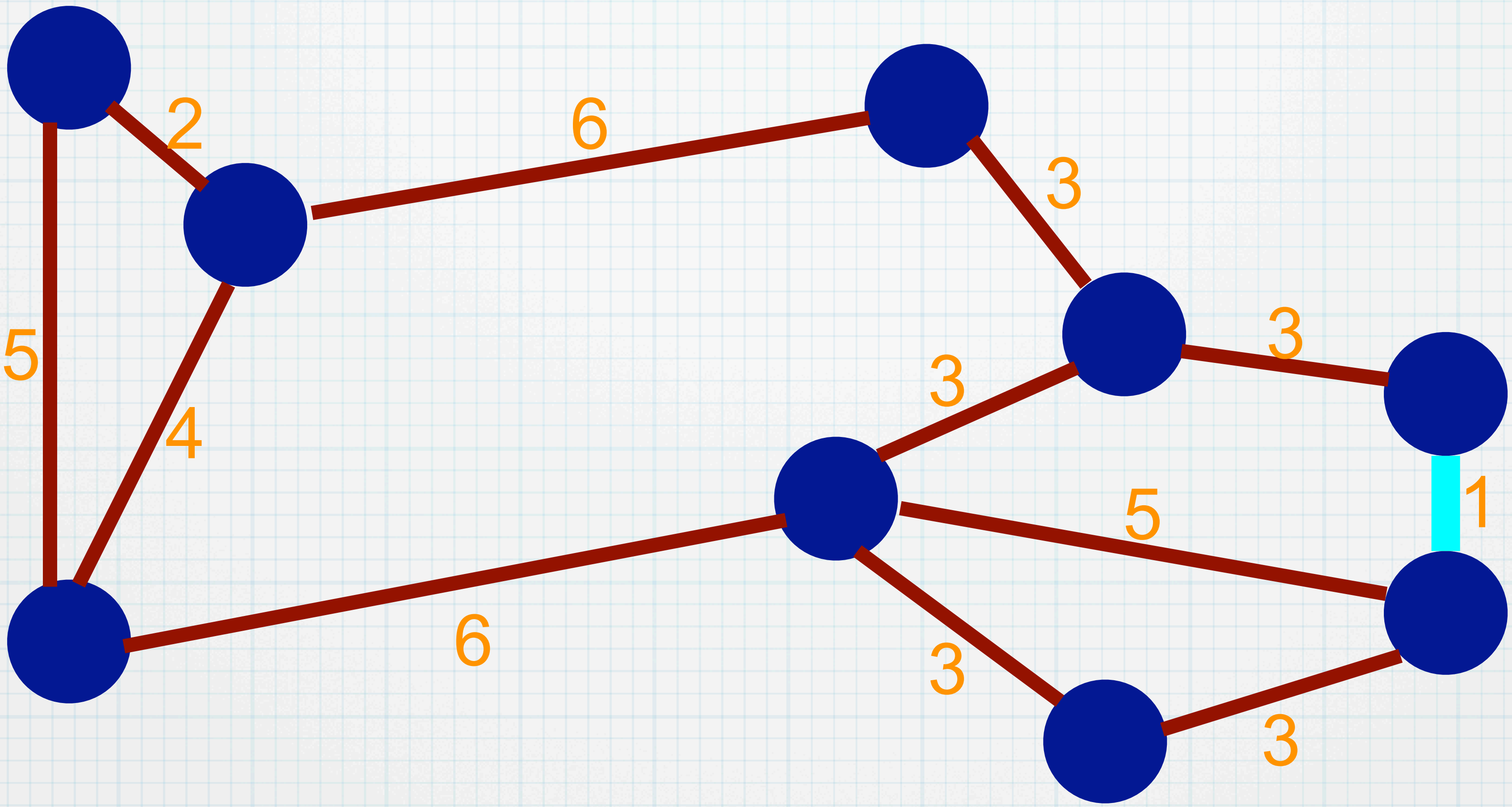
המשתנה CC יחזיק את הרכיבים הקשירים של T

הערך C(x) הוא הרכיב ב-CC שבו נמצא x.

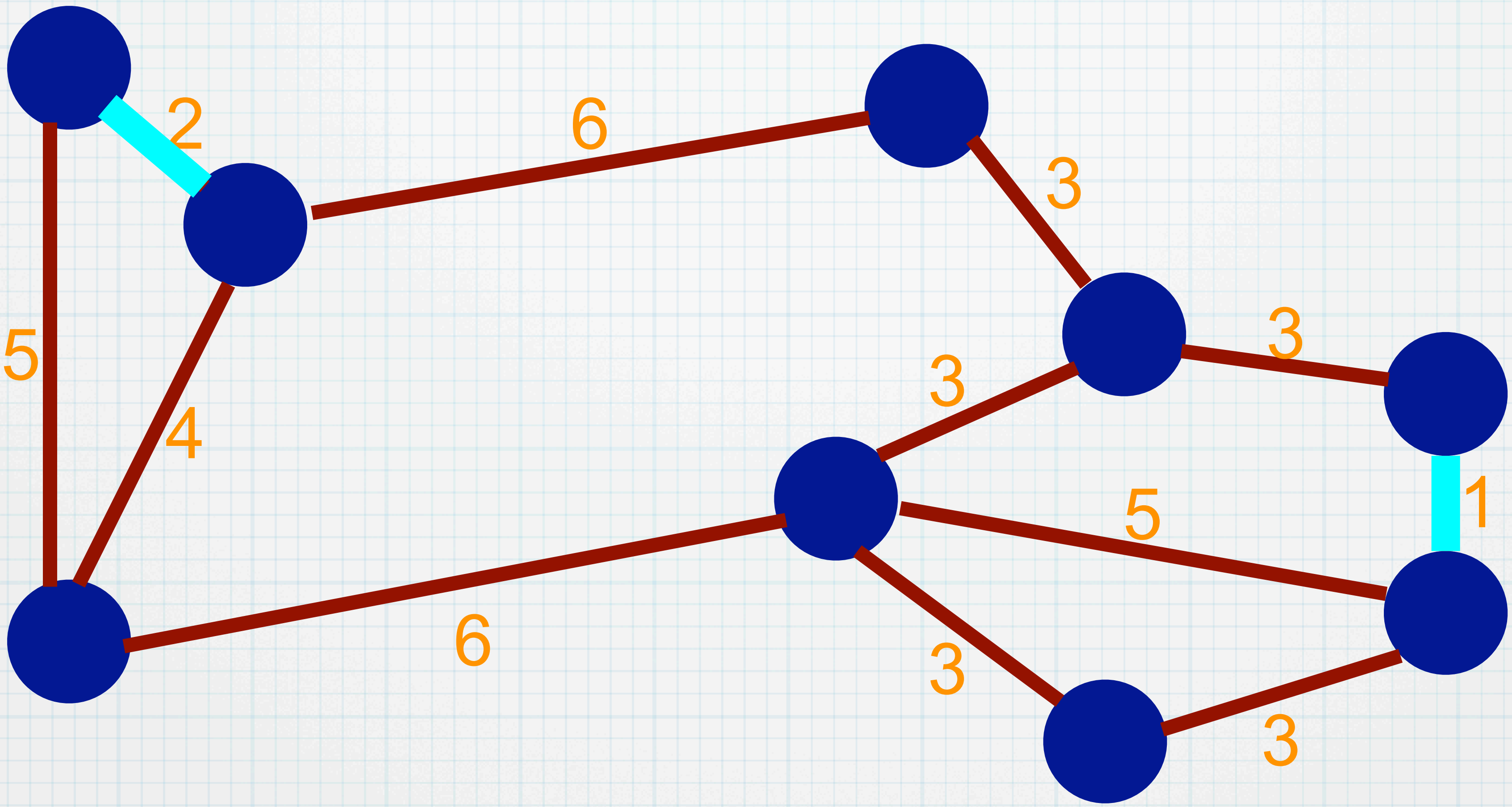
דוגמת הרצה



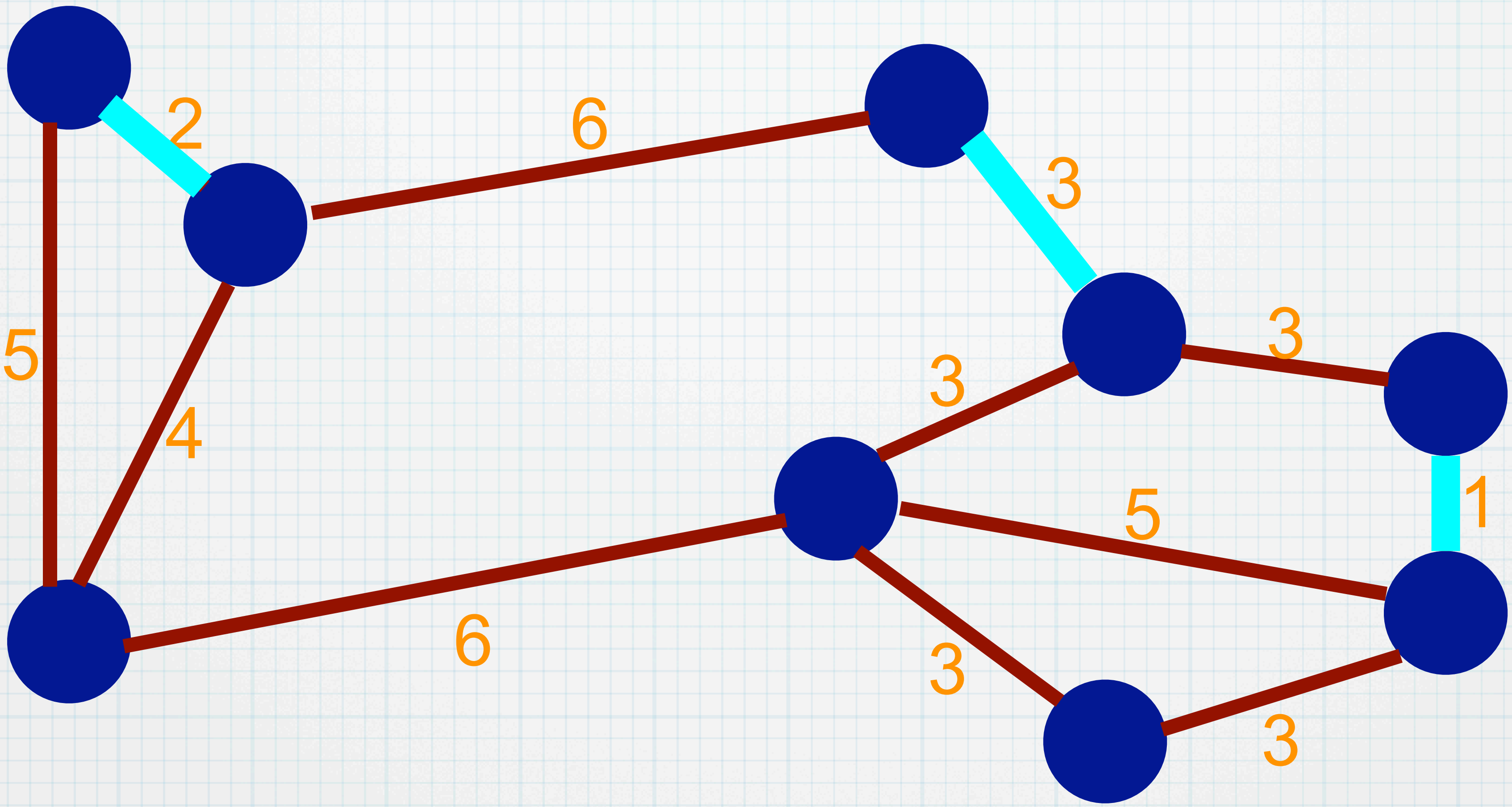
דוגמת הרצה



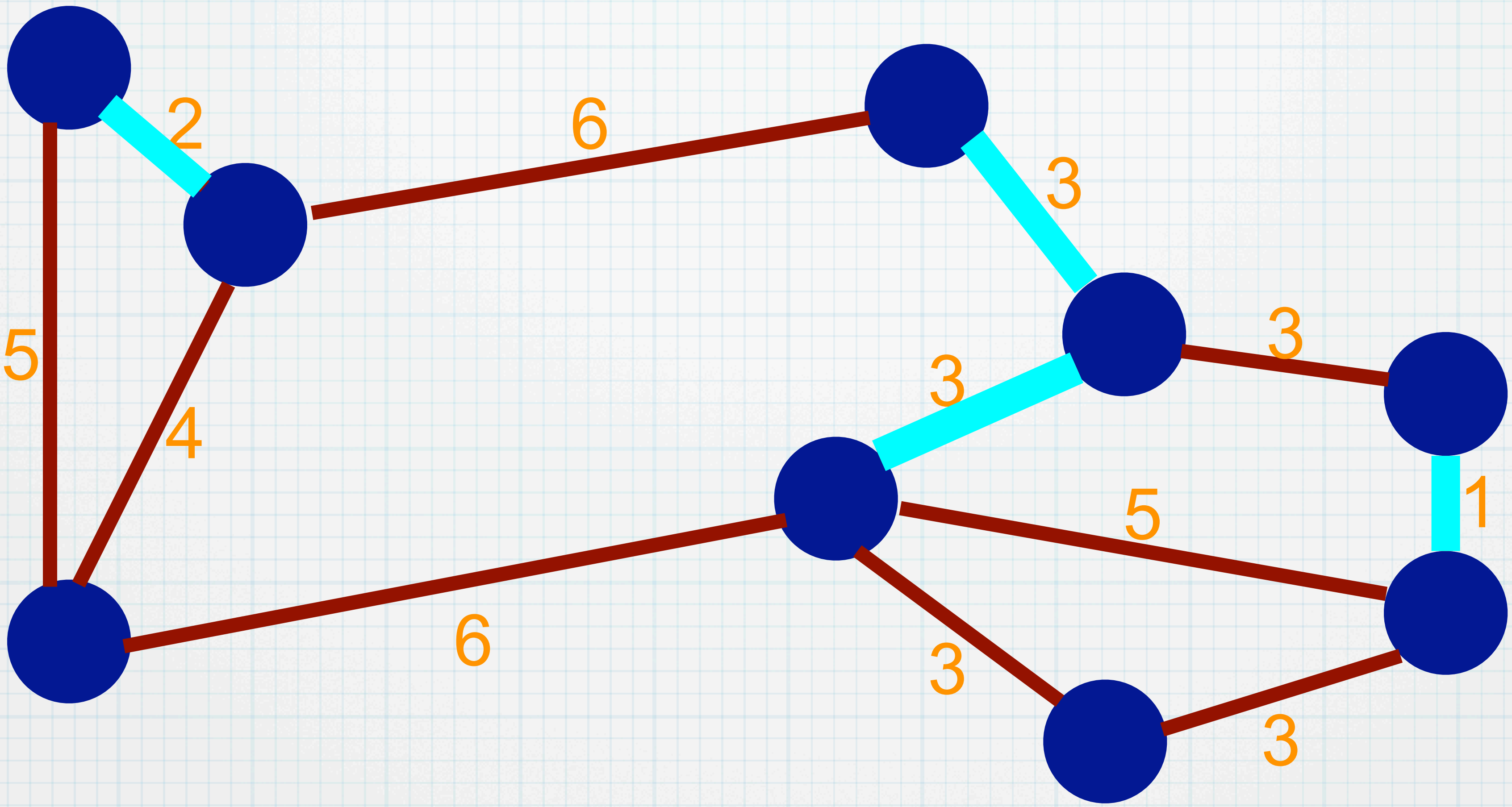
דוגמת הרצה



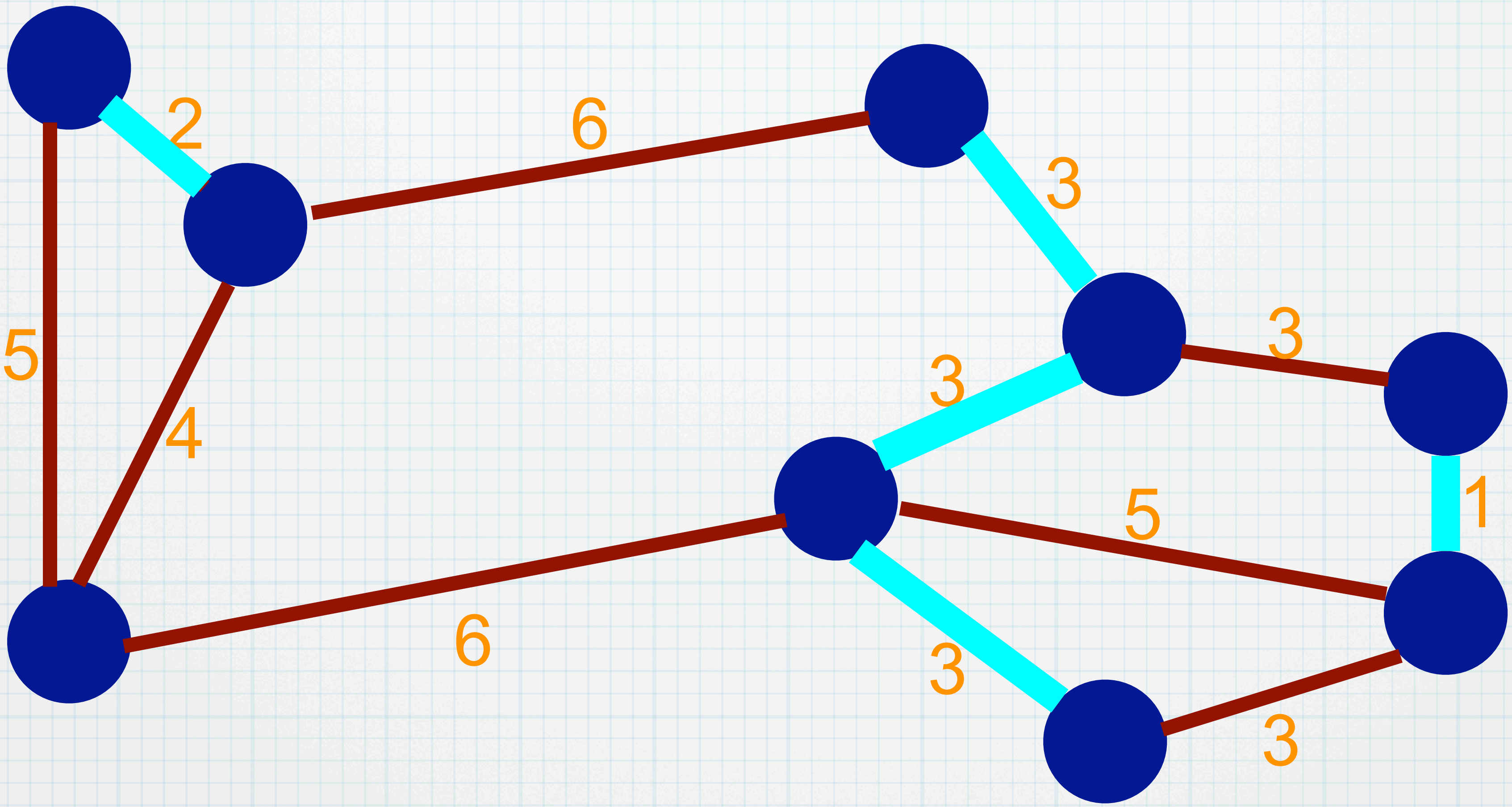
דוגמת הרצה



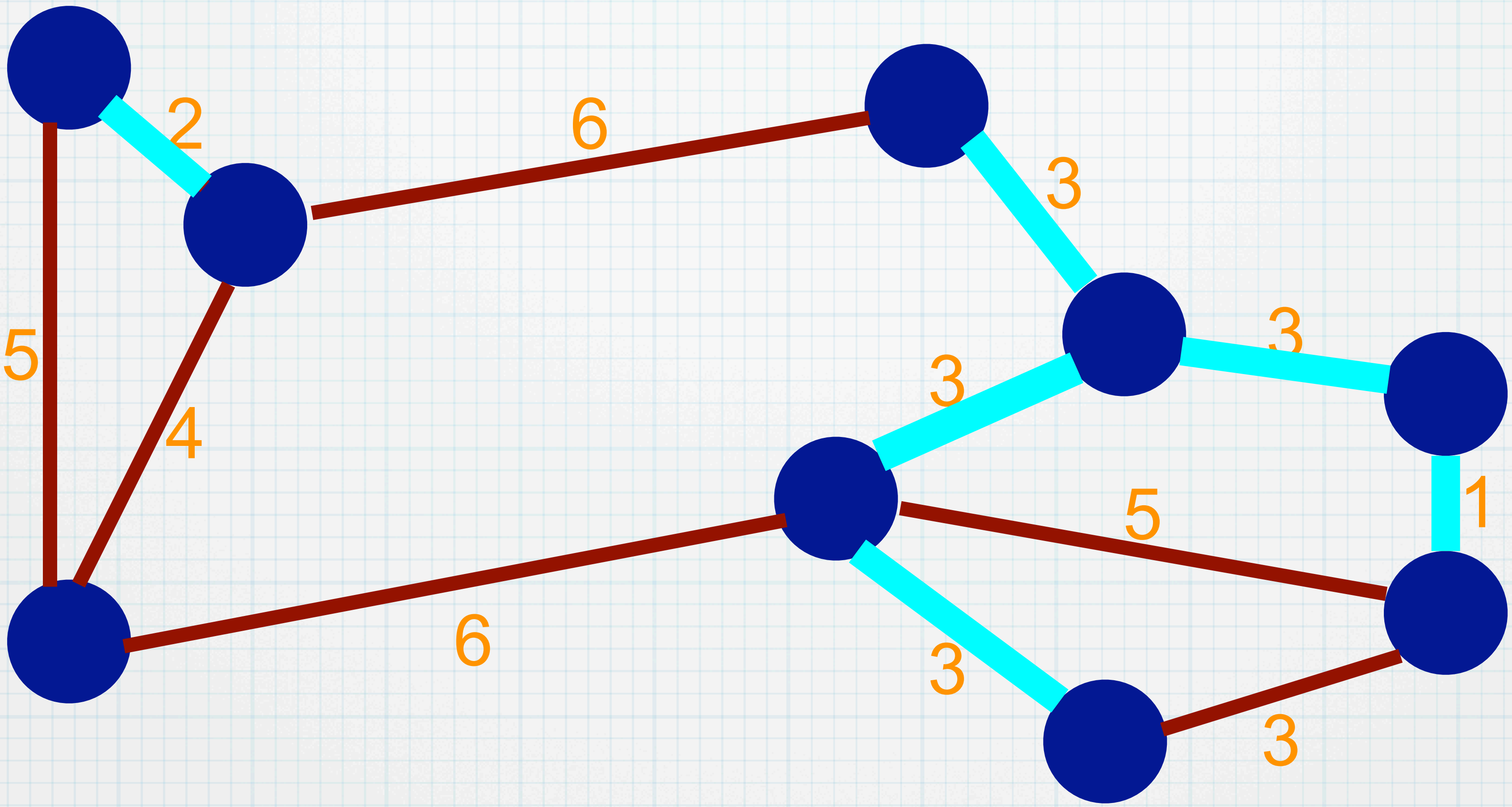
דוגמת הרצה



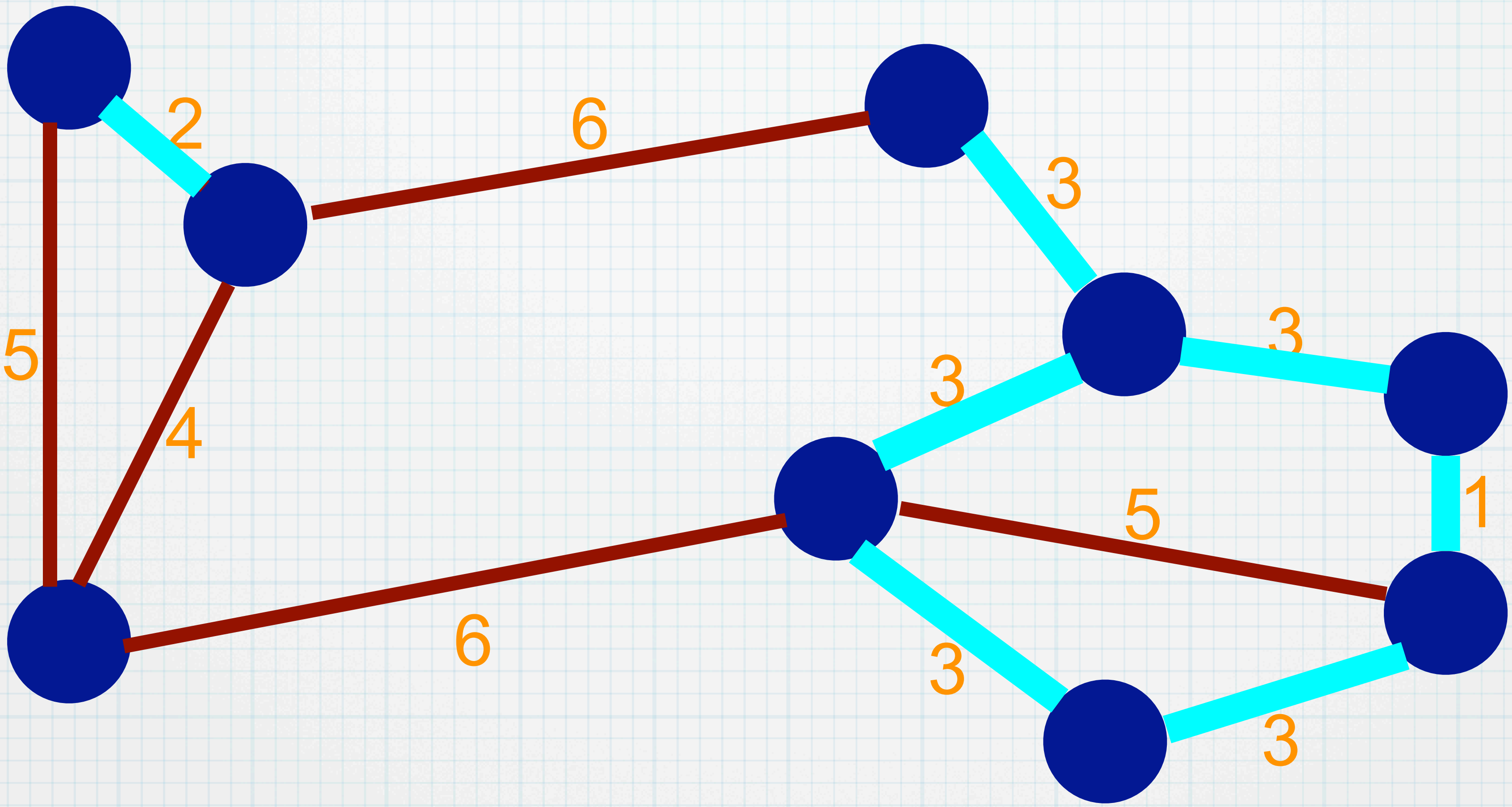
דוגמת הרצה



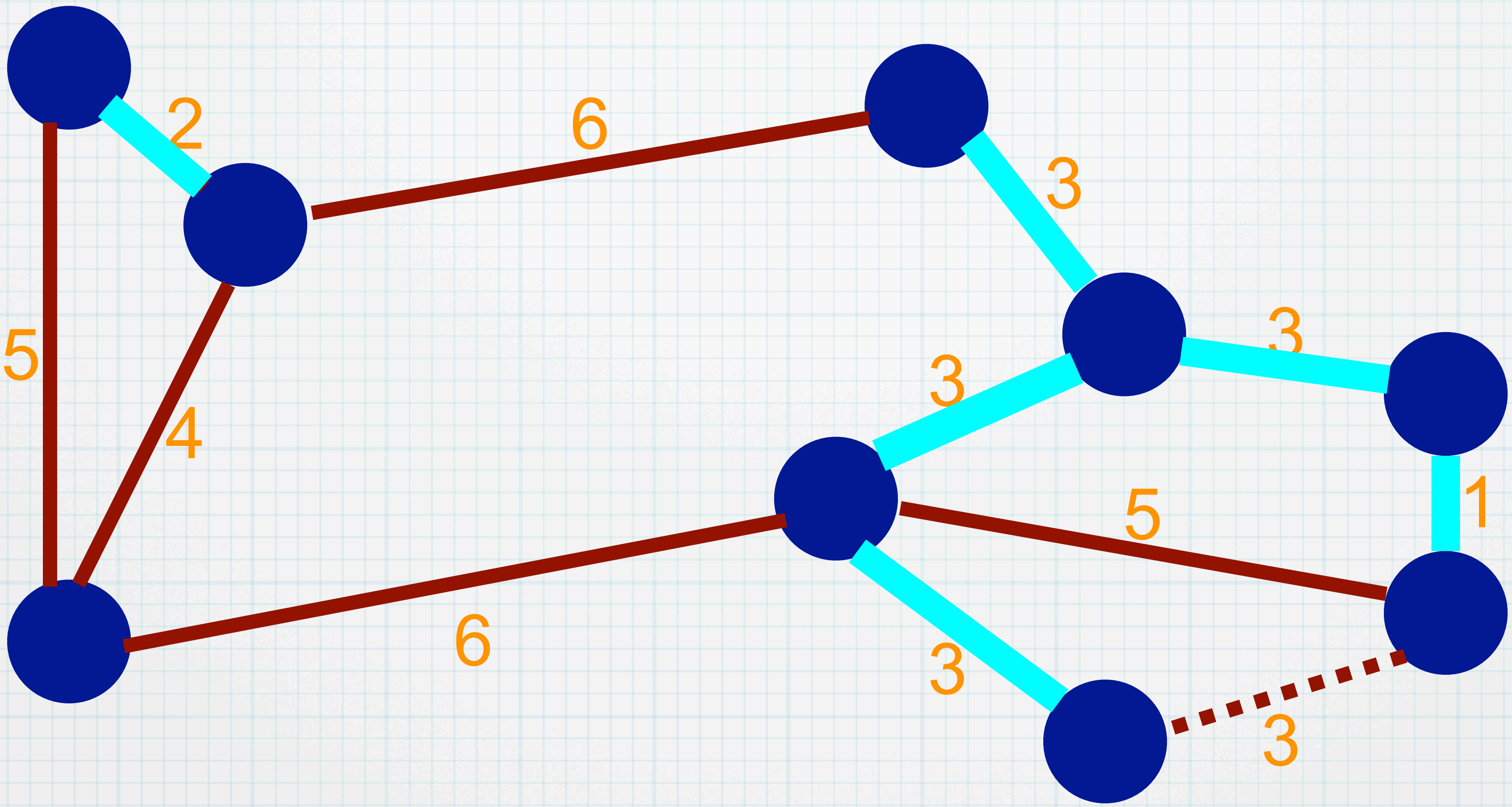
דוגמת הרצה



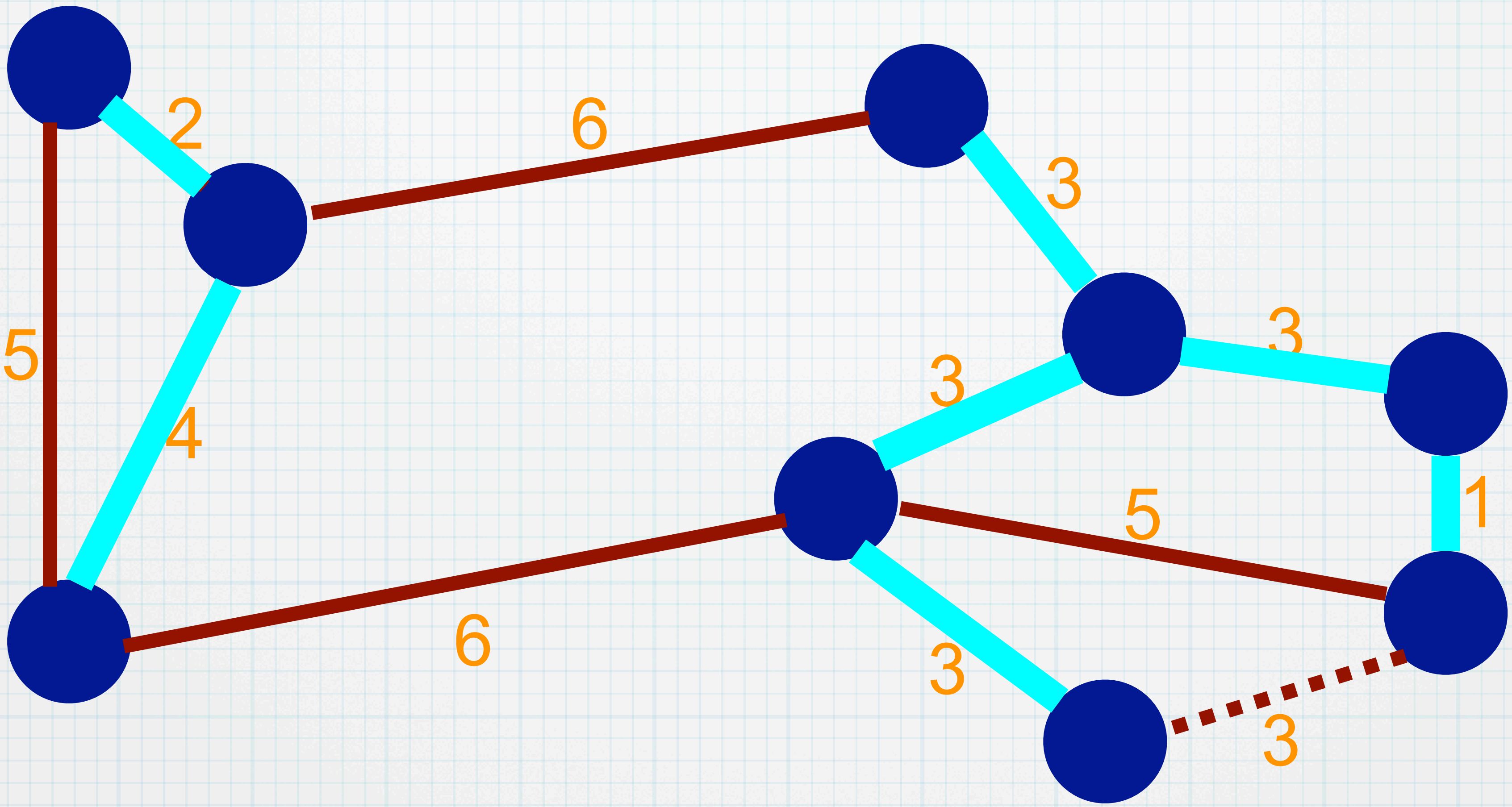
דוגמת הרצה



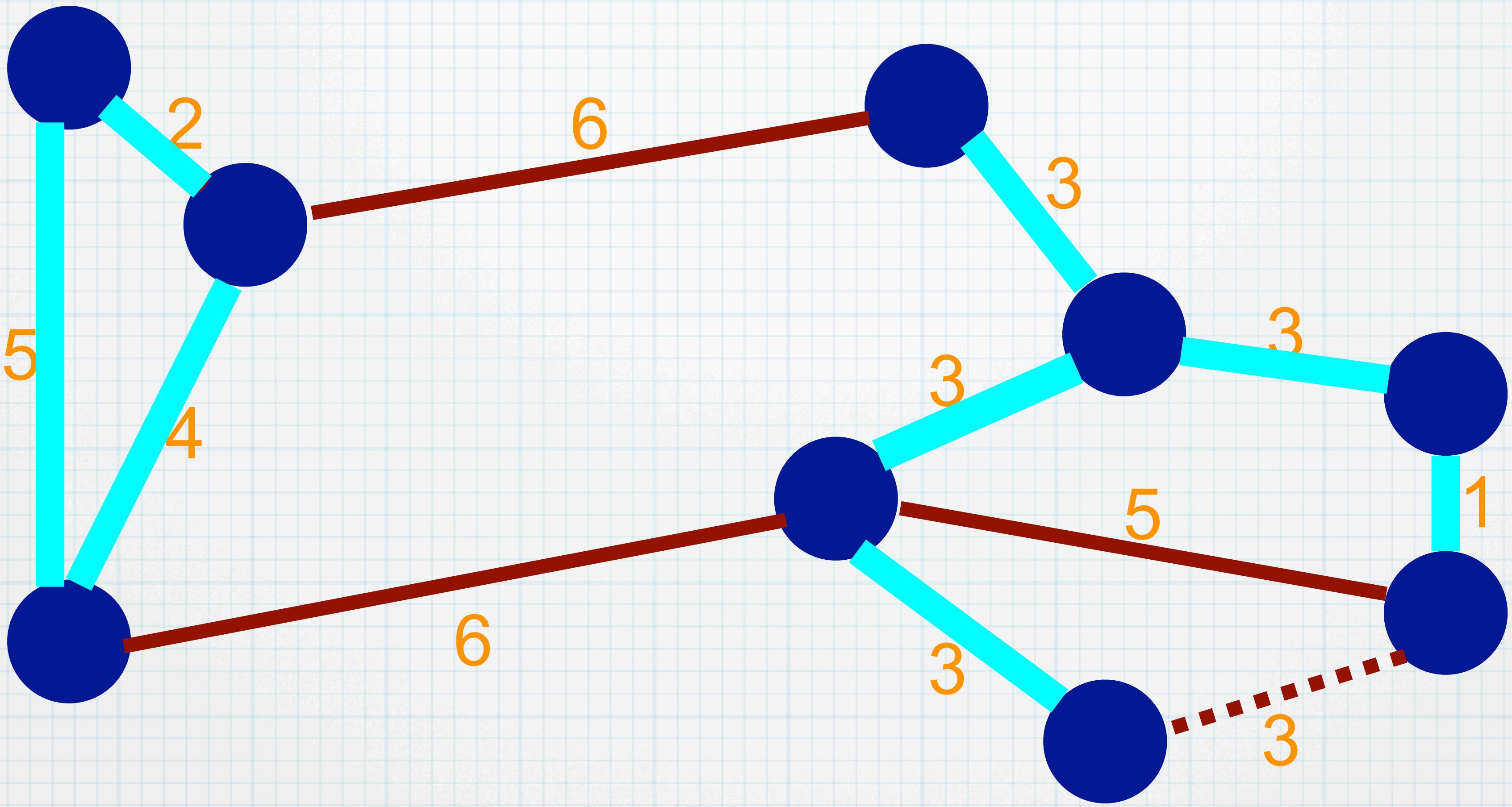
דוגמת הרצה



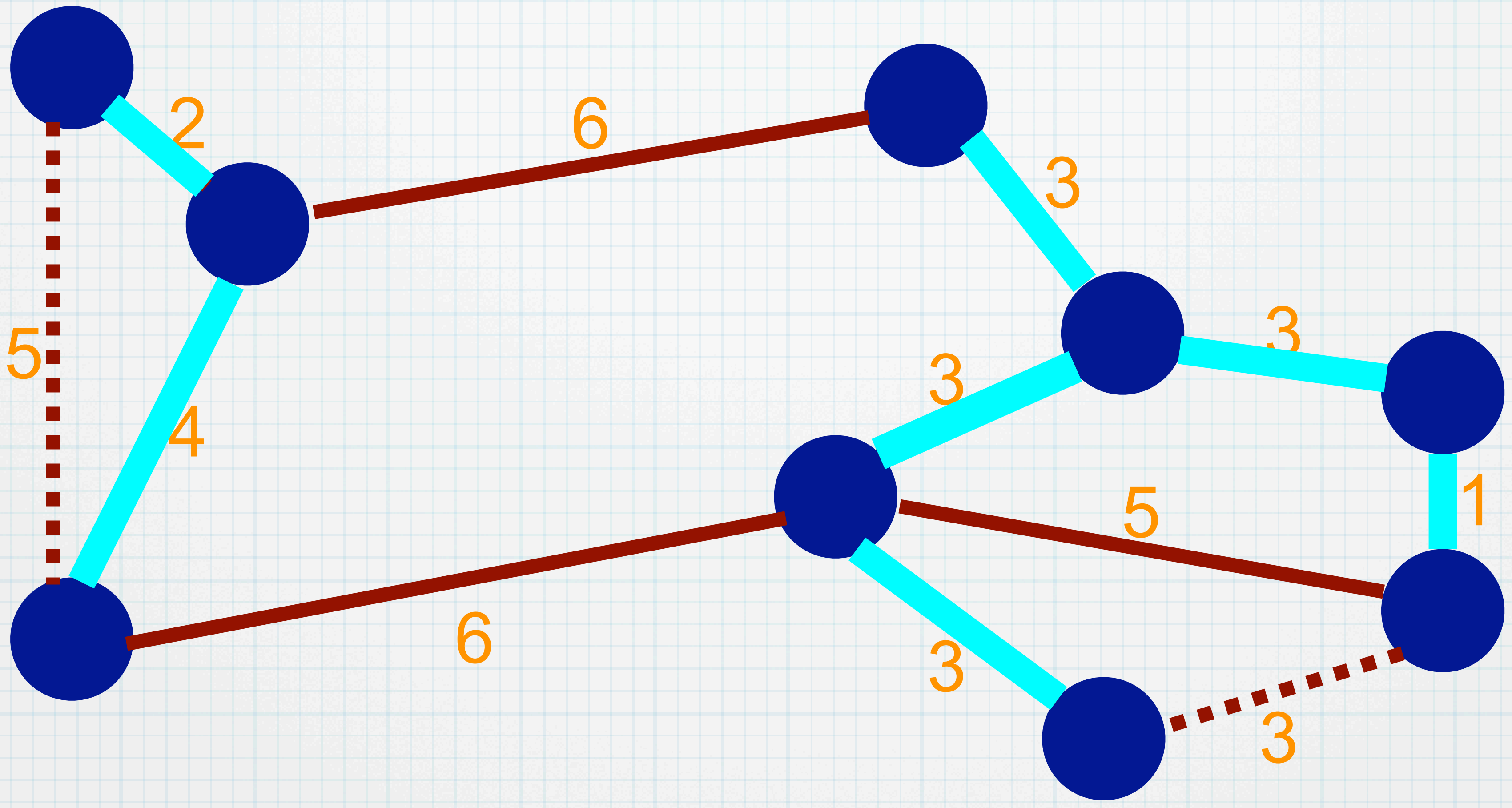
דוגמת הרצה



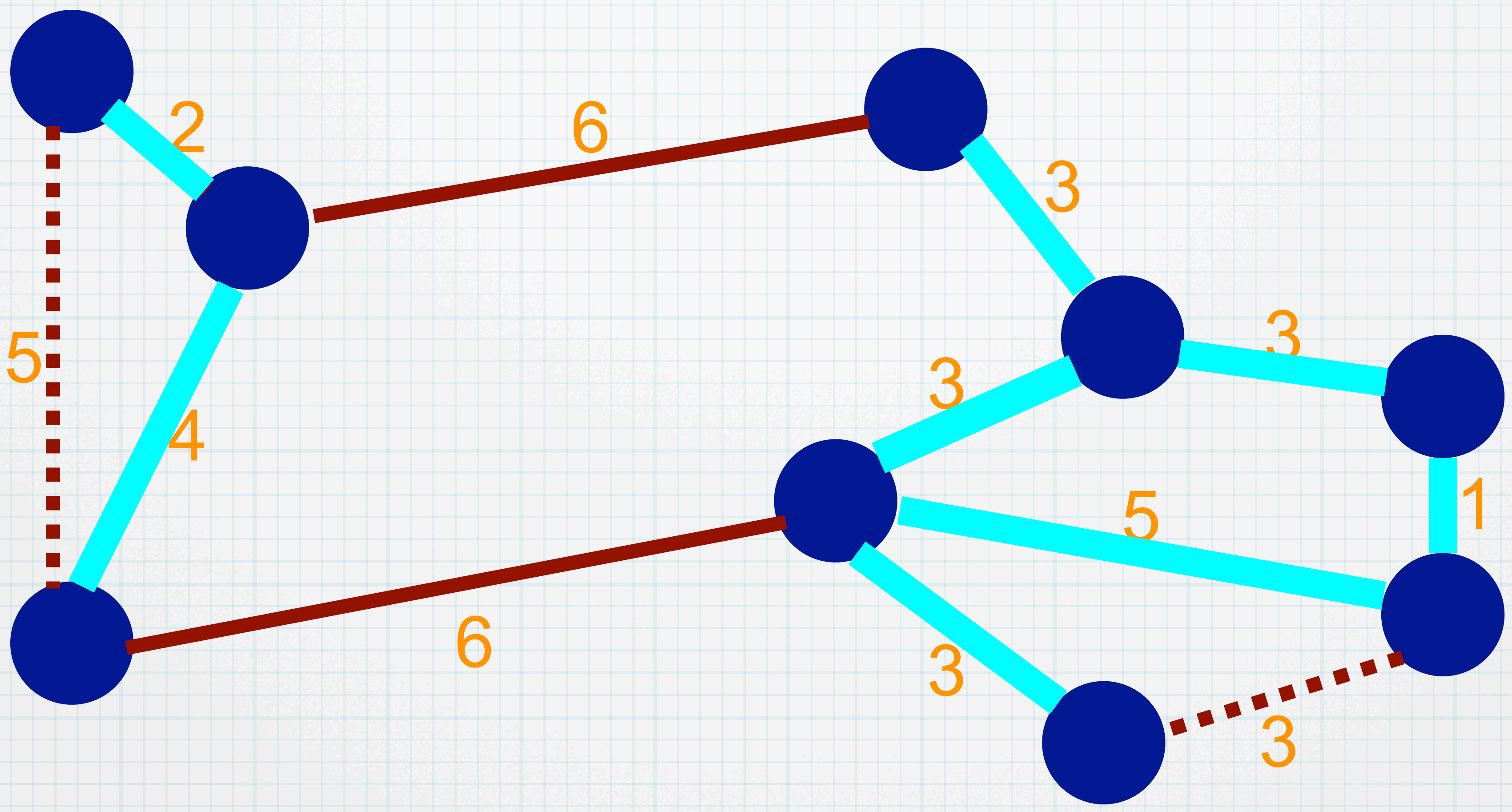
דוגמת הרצה



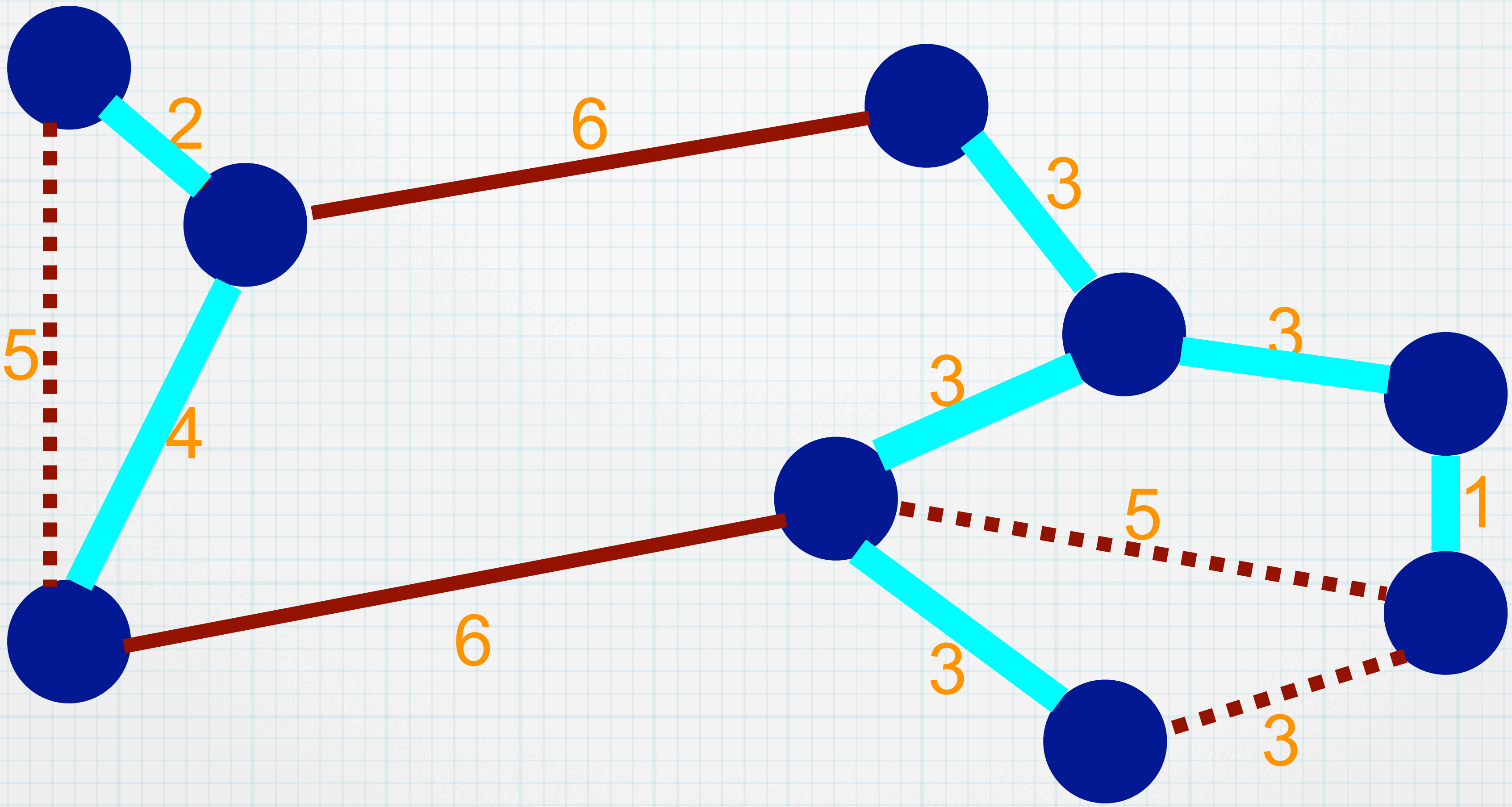
דוגמת הרצה



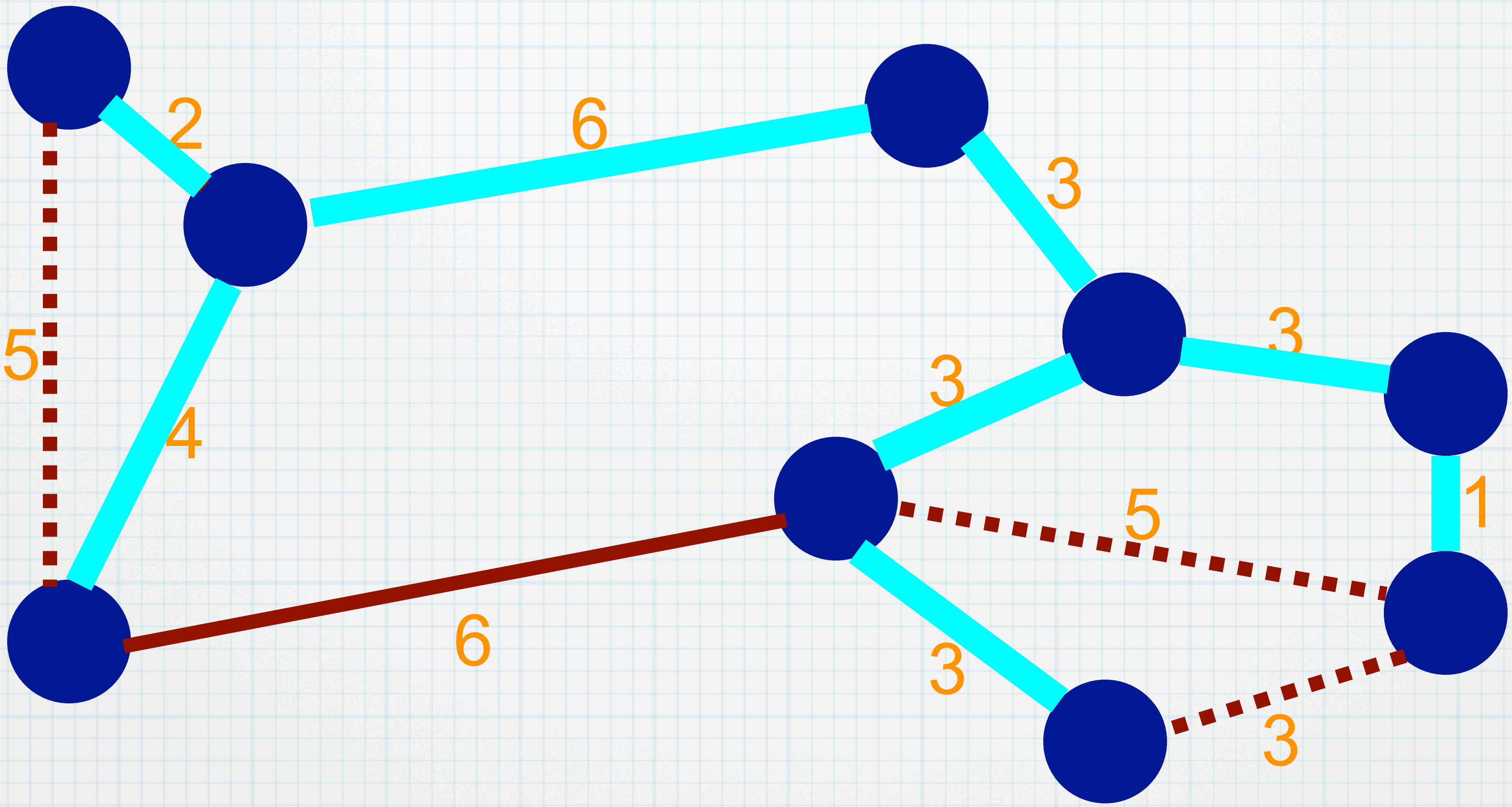
דוגמת הרצה



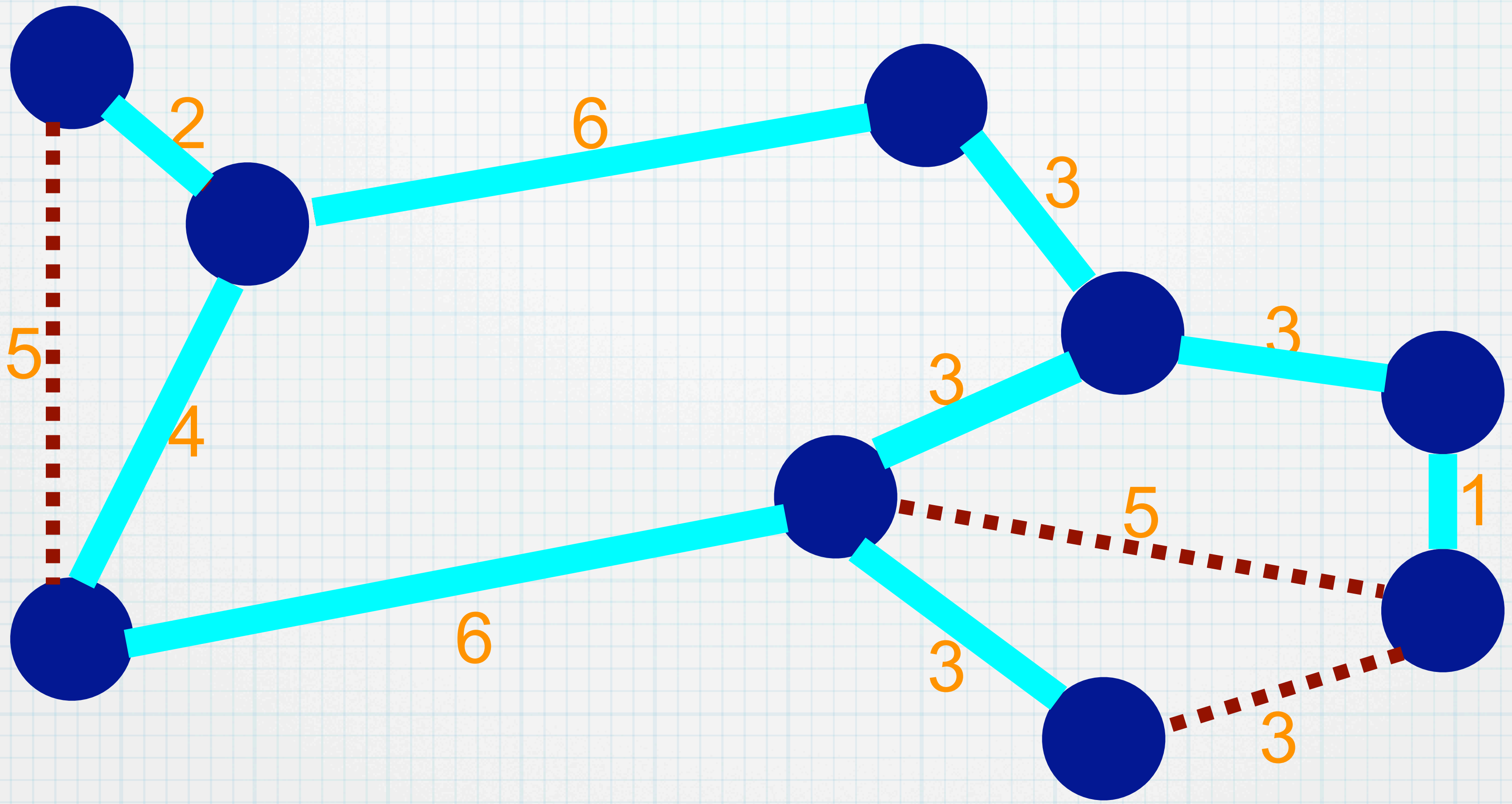
דוגמת הרצה



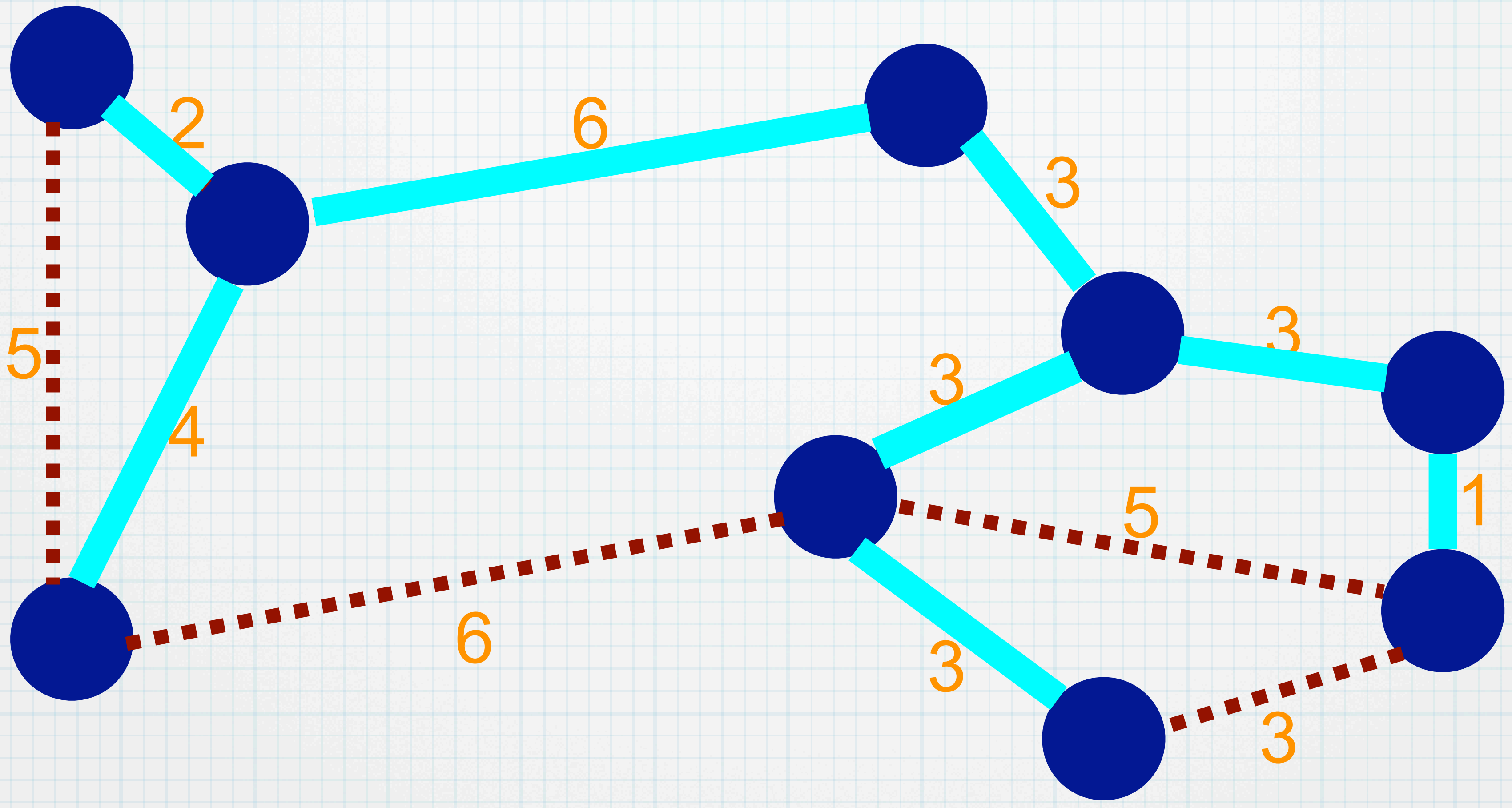
דוגמת הרצה



דוגמת הרצה



דוגמת הרצה

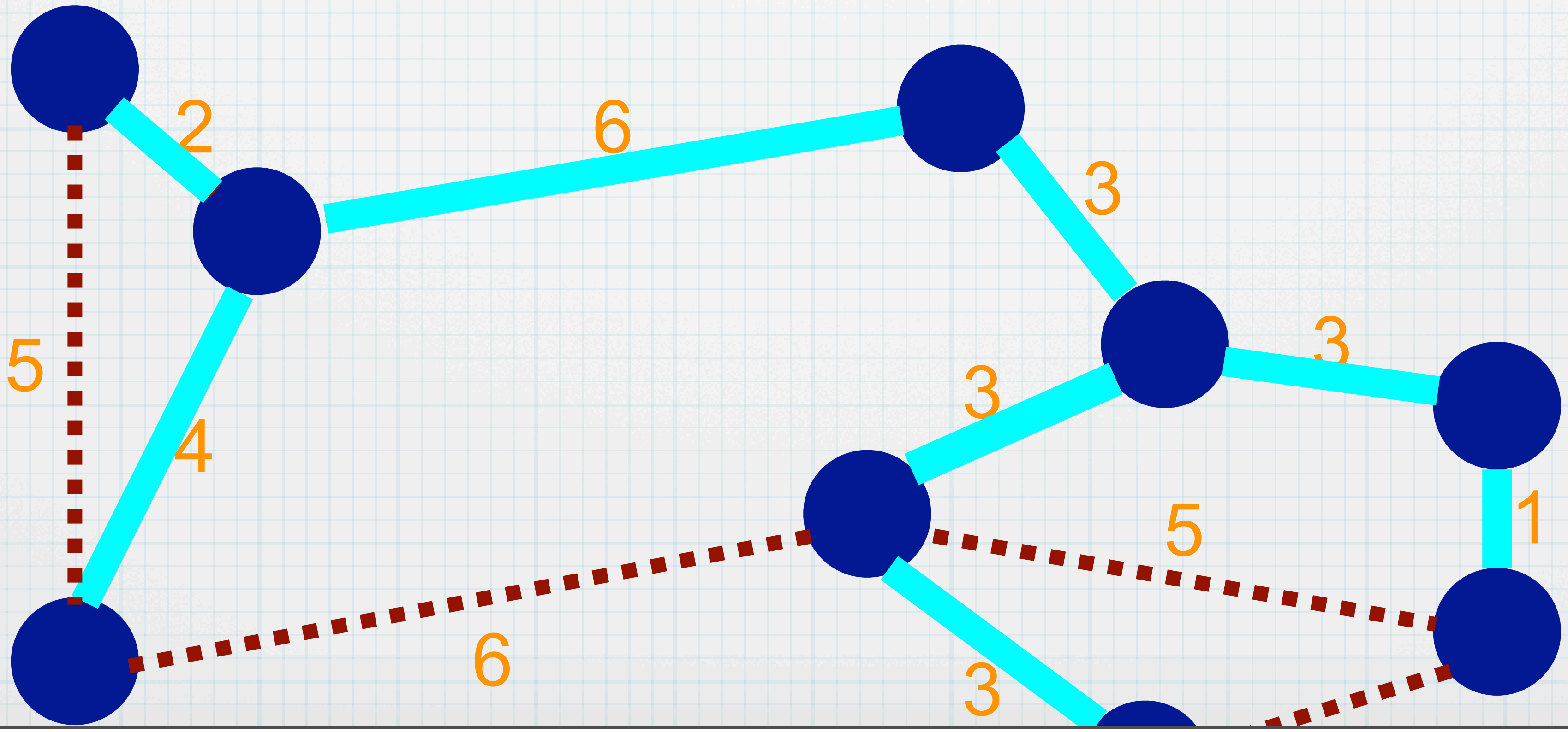


הוכחת נכונות

<u>הגדרה</u>: חתך ב-G הוא קבוצה $\emptyset \neq S \subsetneq V$
שפת החתך היא קבוצת הקשתות עם קצה אחד ב-S.

בעץ כל קשת e מגדירה חתך S(e) שעבורו {e} שפת החתך.

<u>משפט</u>: T הוא עפ"מ אםם לכל $e \in E(T)$ ולכל $f \in E(G)$ בשפת החתך S(e) מתקיים $w(e) \leq w(f)$.

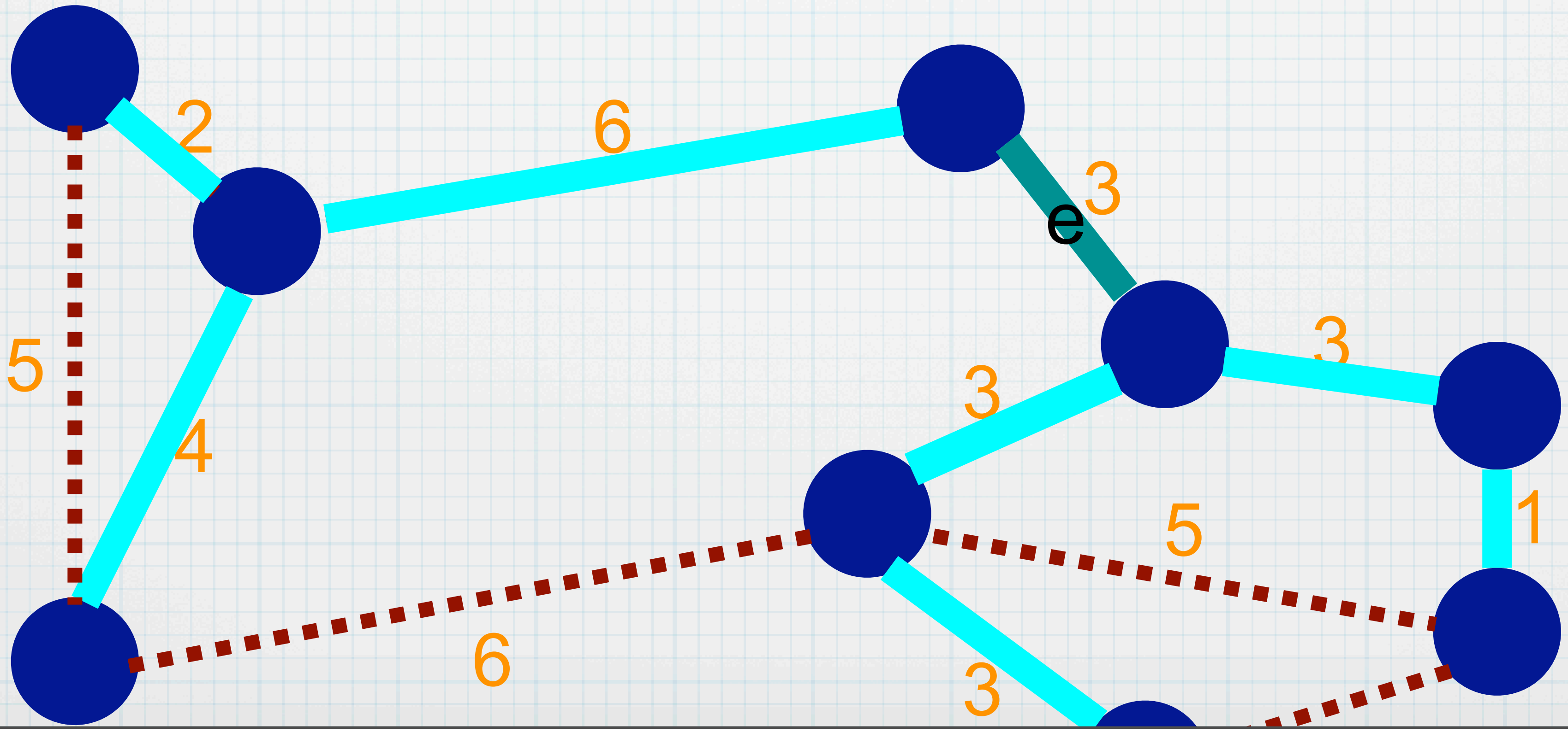


הוכחת נכונות

<u>הגדרה</u>: חתך ב-G הוא קבוצה $\emptyset \neq S \subsetneq V$
שפת החתך היא קבוצת הקשתות עם קצה אחד ב-S.

בעץ כל קשת e מגדירה חתך S(e) שעבורו {e} שפת החתך.

<u>משפט</u>: T הוא עפ"מ אםם לכל $e \in E(T)$ ולכל $f \in E(G)$ בשפת החתך S(e) מתקיים $w(e) \leq w(f)$.

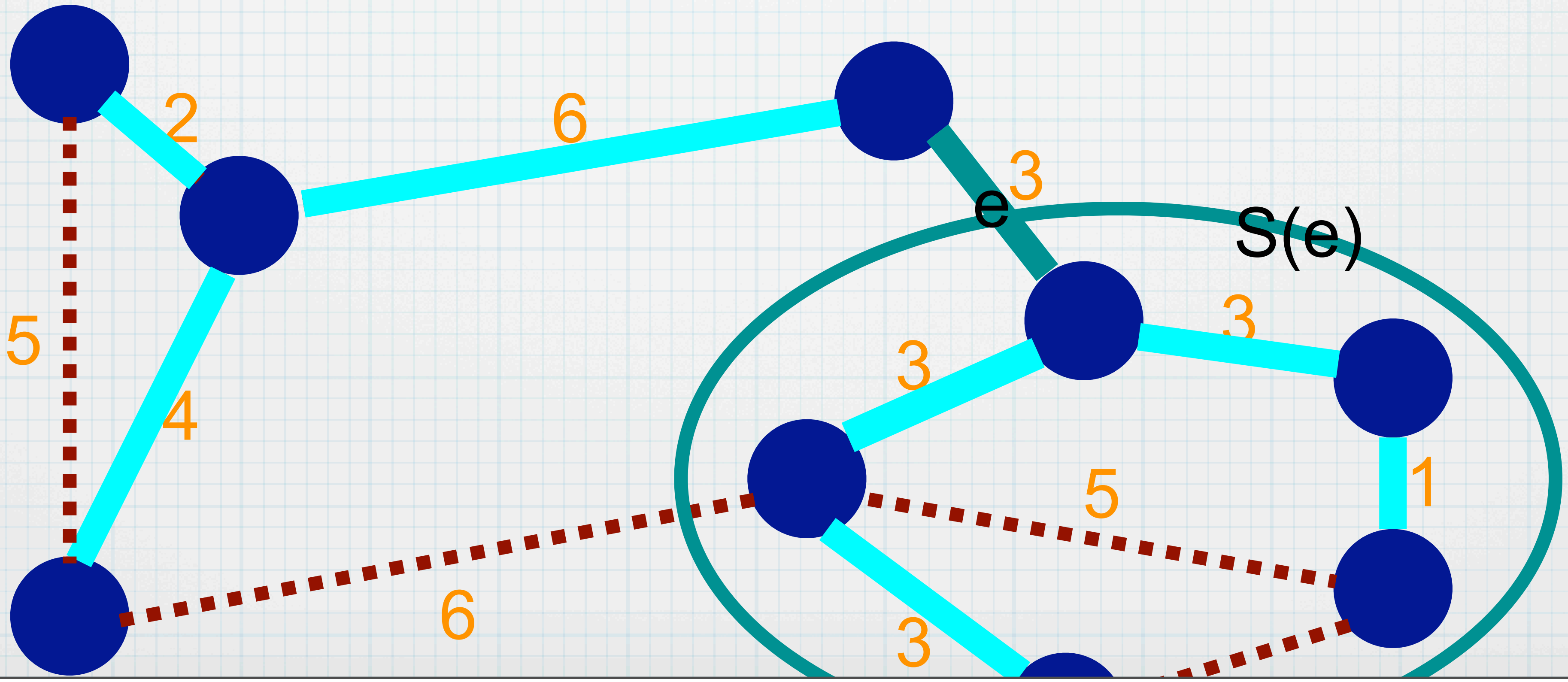


הוכחת נכונות

הגדרה: חתך ב-G הוא קבוצה $\emptyset \neq S \subsetneq V$
שפת החתך היא קבוצת הקשתות עם קצה אחד ב-S.

בעץ כל קשת e מגדירה חתך S(e) שעבורו {e} שפת החתך.

משפט: T הוא עפ"מ אםם לכל $e \in E(T)$ ולכל $f \in E(G)$ בשפת החתך S(e) מתקיים $w(e) \leq w(f)$.

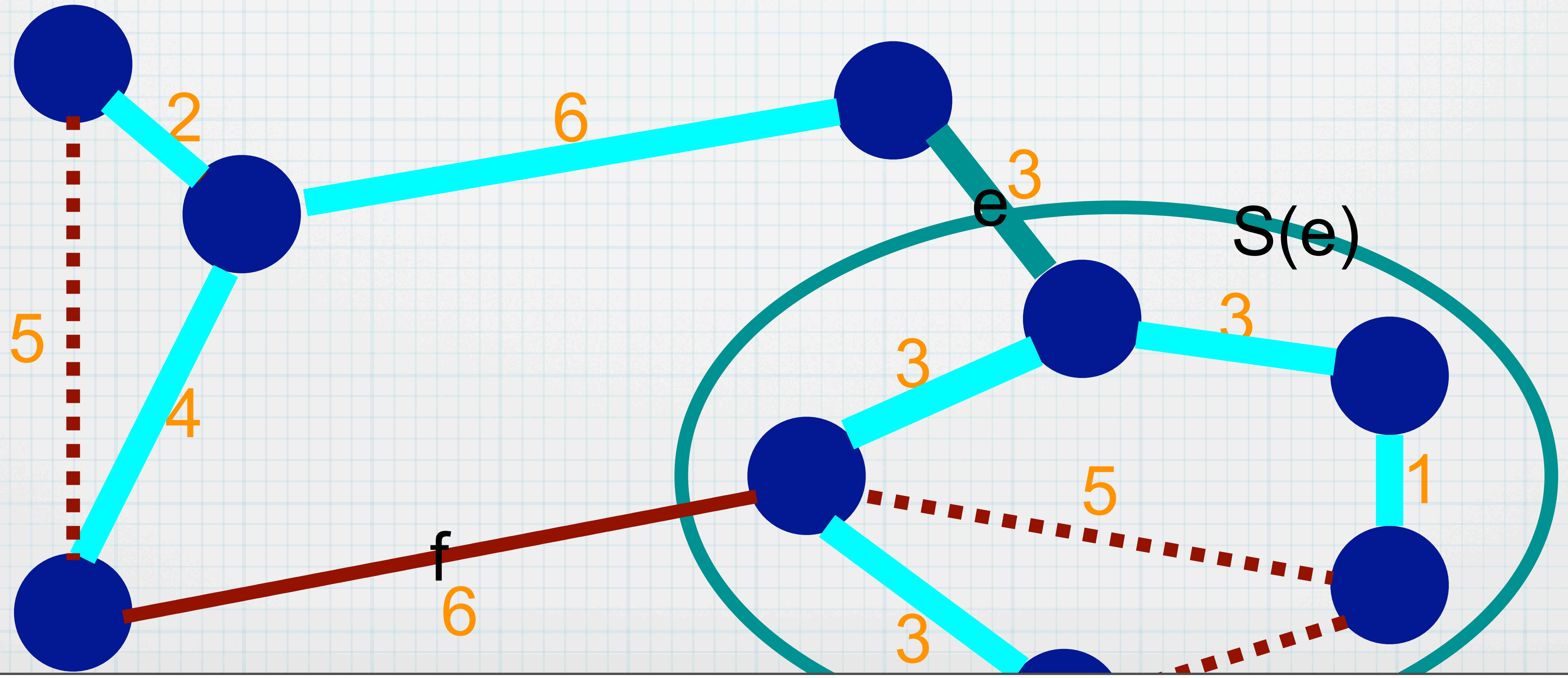


הוכחת נכונות

<u>הגדרה</u>: חתך ב-G הוא קבוצה $\emptyset \neq S \subsetneq V$
שפת החתך היא קבוצת הקשתות עם קצה אחד ב-S.

בעץ כל קשת e מגדירה חתך S(e) שעבורו {e} שפת החתך.

<u>משפט</u>: T הוא עפ"מ אםם לכל $e \in E(T)$ ולכל $f \in E(G)$ בשפת החתך S(e) מתקיים $w(e) \leq w(f)$.



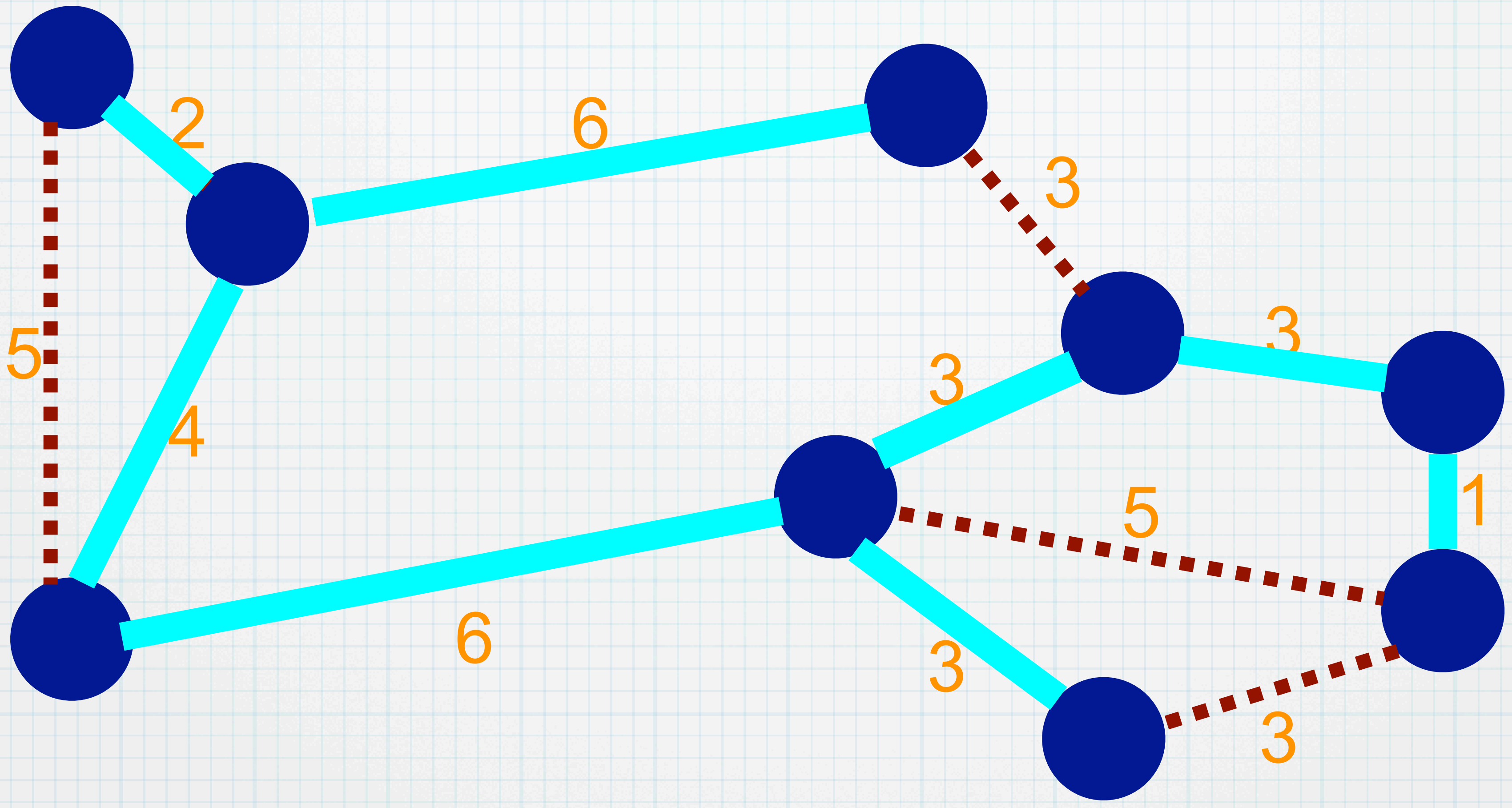
הוכחת המשפט

הרעיון המרכזי: לכל עץ פורש T ולכל $f \in E(G) \setminus E(T)$ יש $e \in E(T)$ כך ש- $(V, E(T) \setminus \{e\} \cup \{f\})$ הוא עץ פורש.

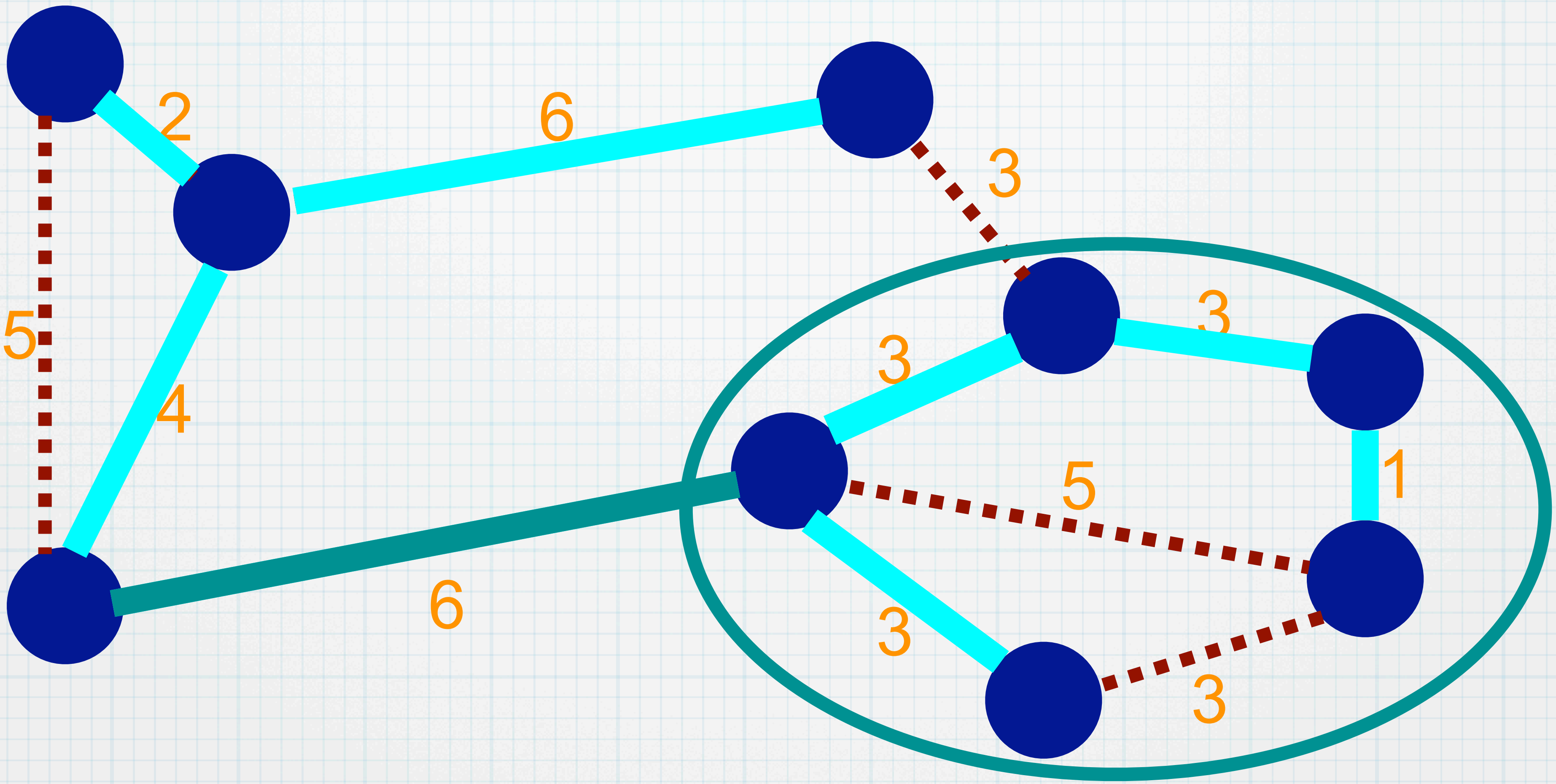
בכיוון: עפ"מ ⇐ e מינימלית ב-$S(e)$.

נתון עפ"מ T של G. תהי $e \in E(T)$. נניח שיש $f \in E(G)$ בשפת החתך $S(e)$ שעבורה $w(f) < w(e)$. אזי העץ הבא: $(V, E(T) \setminus \{e\} \cup \{f\})$ פורש את G ומשקלו נמוך יותר מזה של T. לכן זו סתירה להנחה ש-T עפ"מ של G.

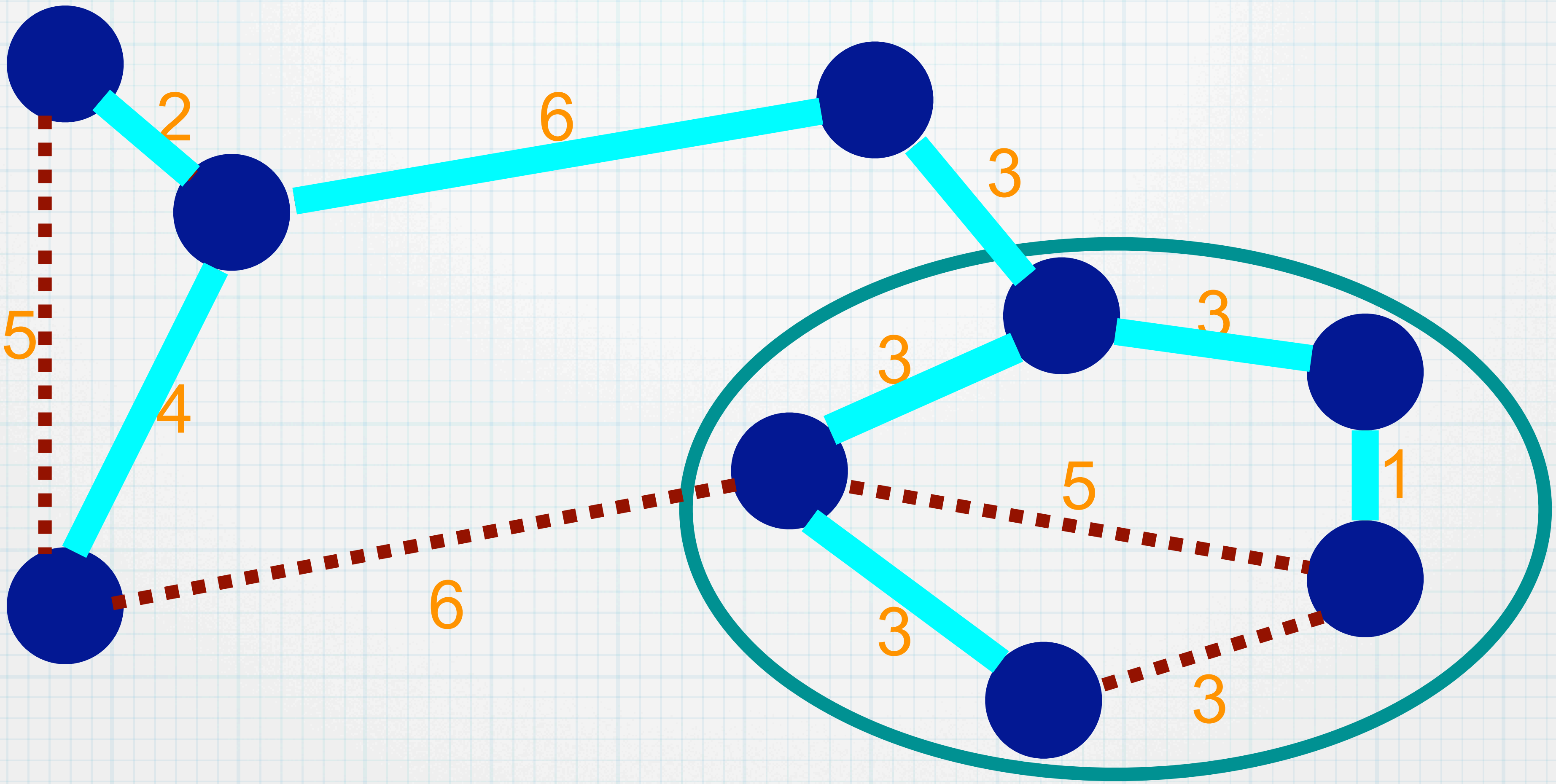
הדגמת ההוכחה



הדגמת ההוכחה



הדגמת ההוכחה

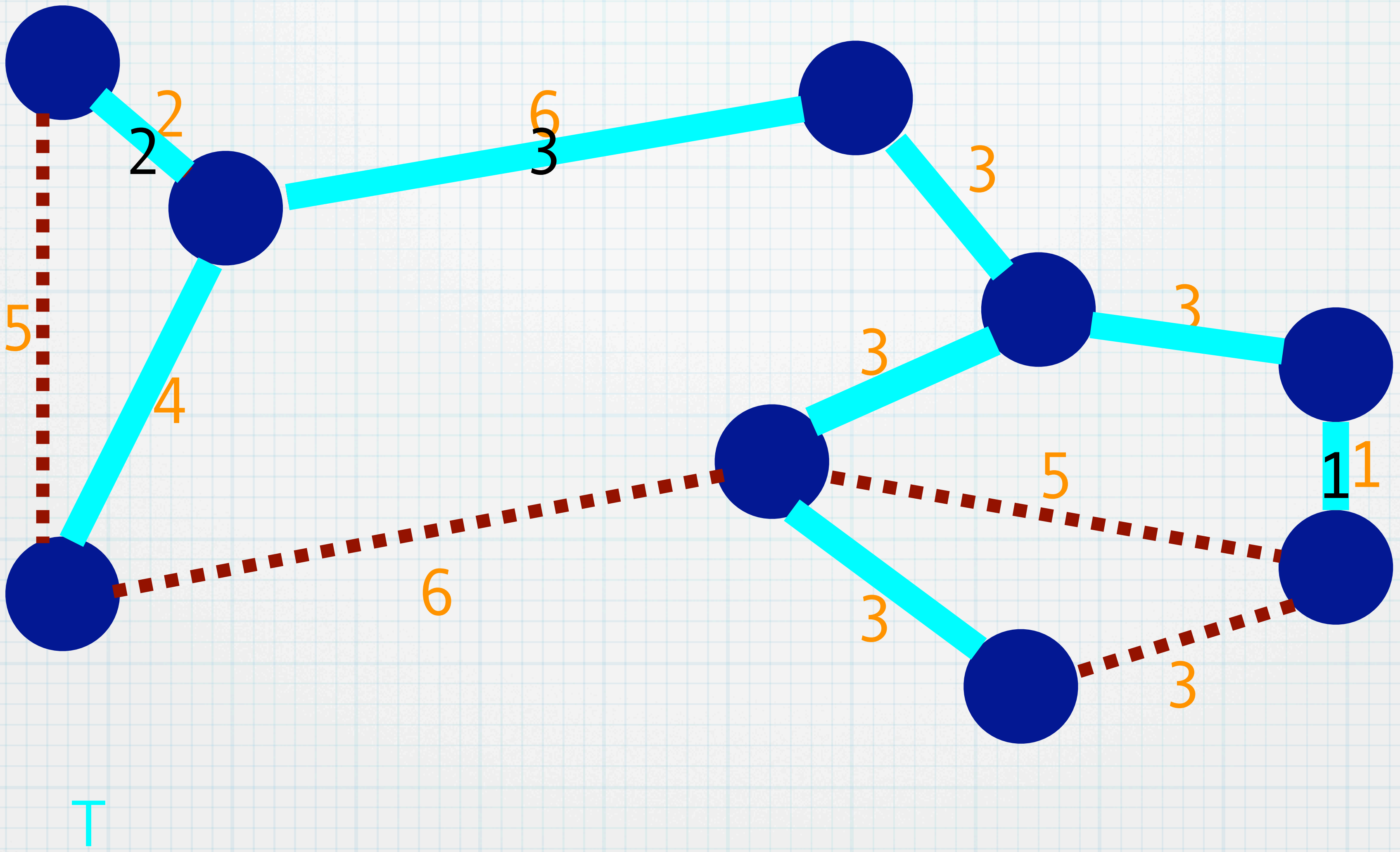


הוכחת המשפט (המשך)

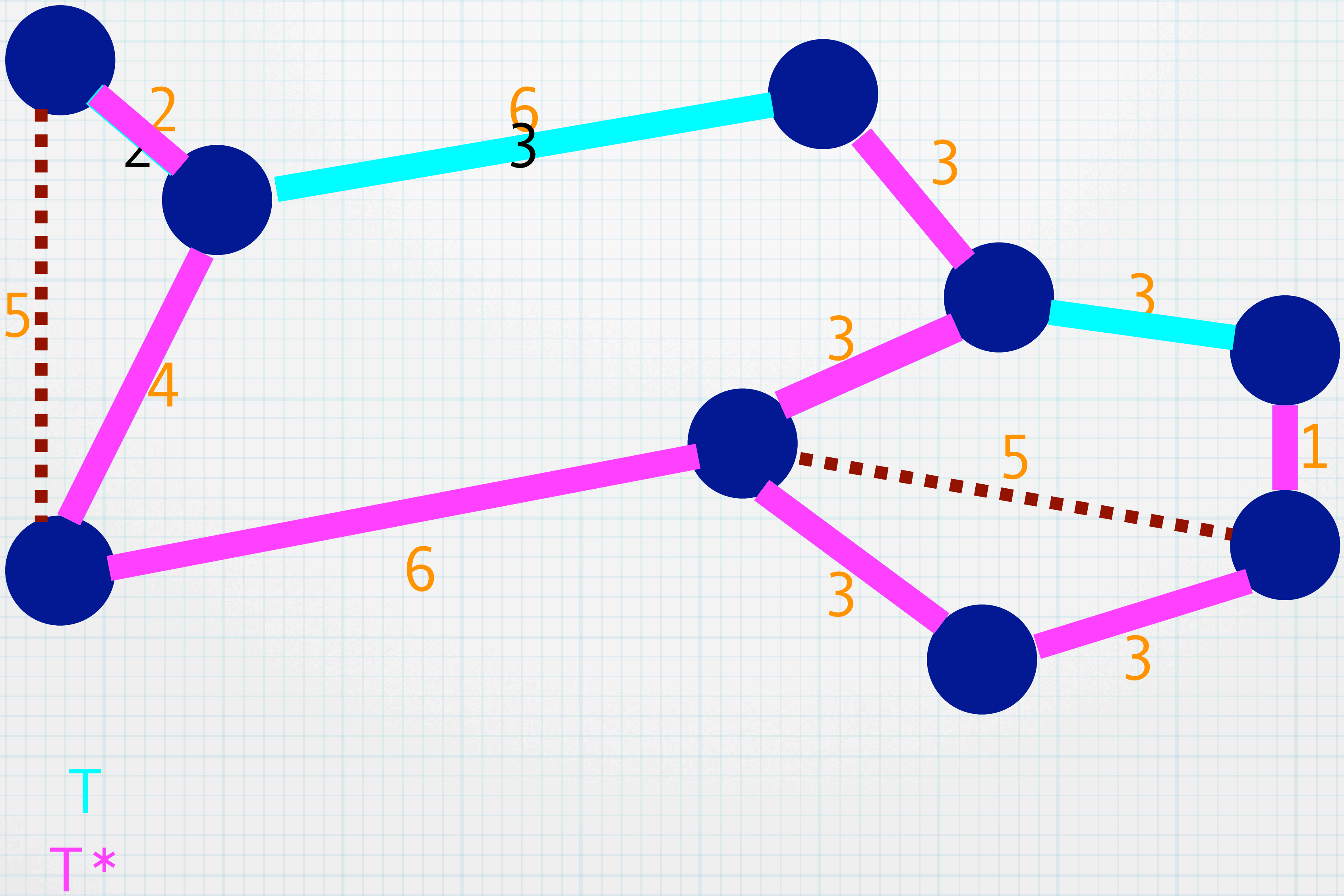
נוכיח את הכיוון ההפוך באינדוקציה. נמספר את קשתות T, ונראה שלכל $k \in \{0,1,2,...,|V|-1\}$, יש עפ"מ שמכיל את k הקשתות הראשונות של T.

בסיס האינדוקציה: $k = 0$, הטענה טריוויאלית.
נניח נכונות ל- k-1. יהי T^* עפ"מ שמכיל את k-1 הקשתות הראשונות של T אבל לא את הקשת הבאה e.
אם נוסיף את e ל-T^*, נסגור מעגל C. מעגל זה חייב להכיל קשת f משפת החתך S(e) (לפי T).
לפי הגדרת S(e), לא ייתכן כי $f \in E(T)$. לכן, אם נוסיף ל-T^* את e ונסיר את f, נקבל עץ כנדרש שמשקלו אינו גדול יותר ממשקל T^*. $\therefore$

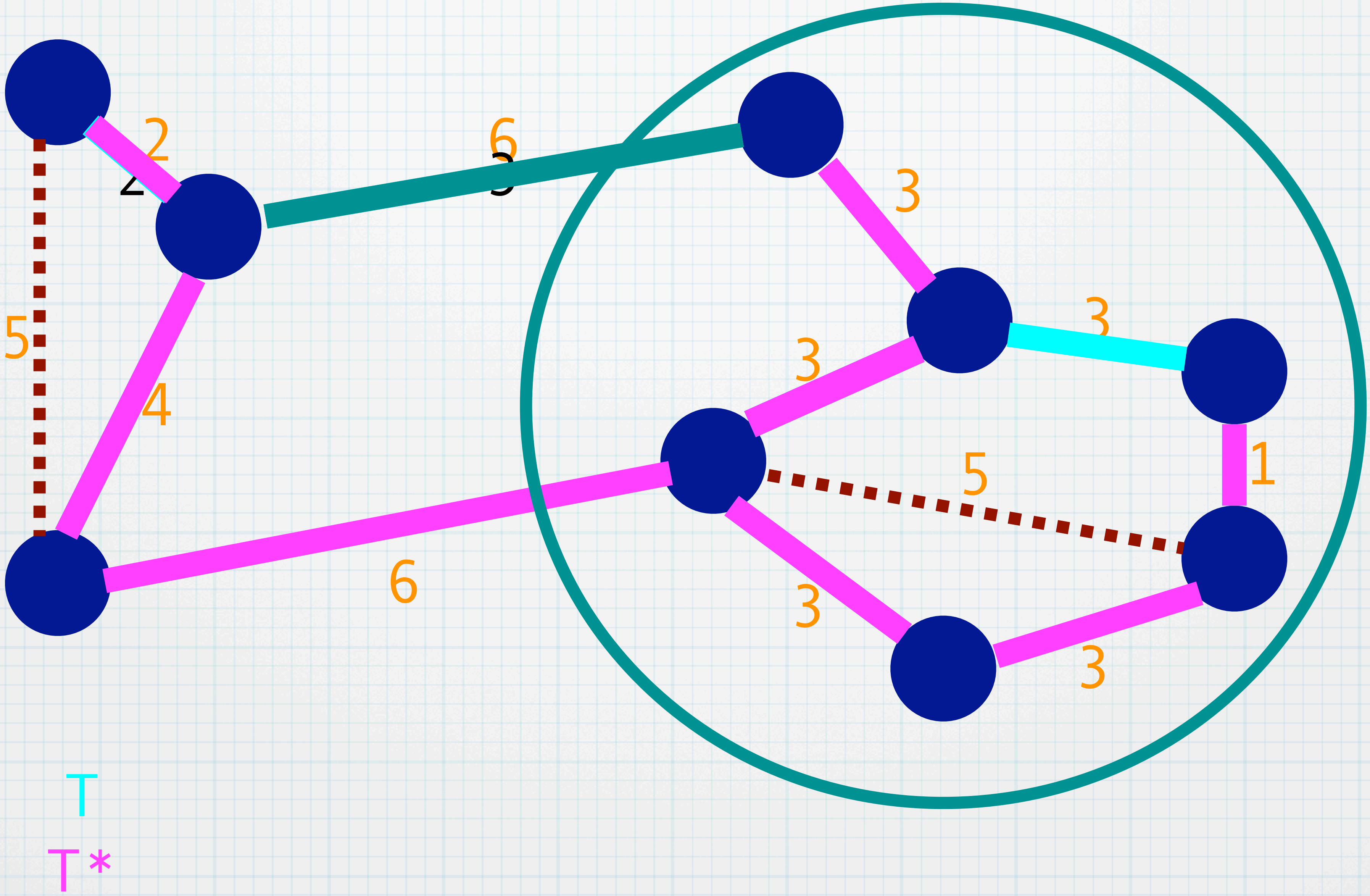
הדגמת ההוכחה



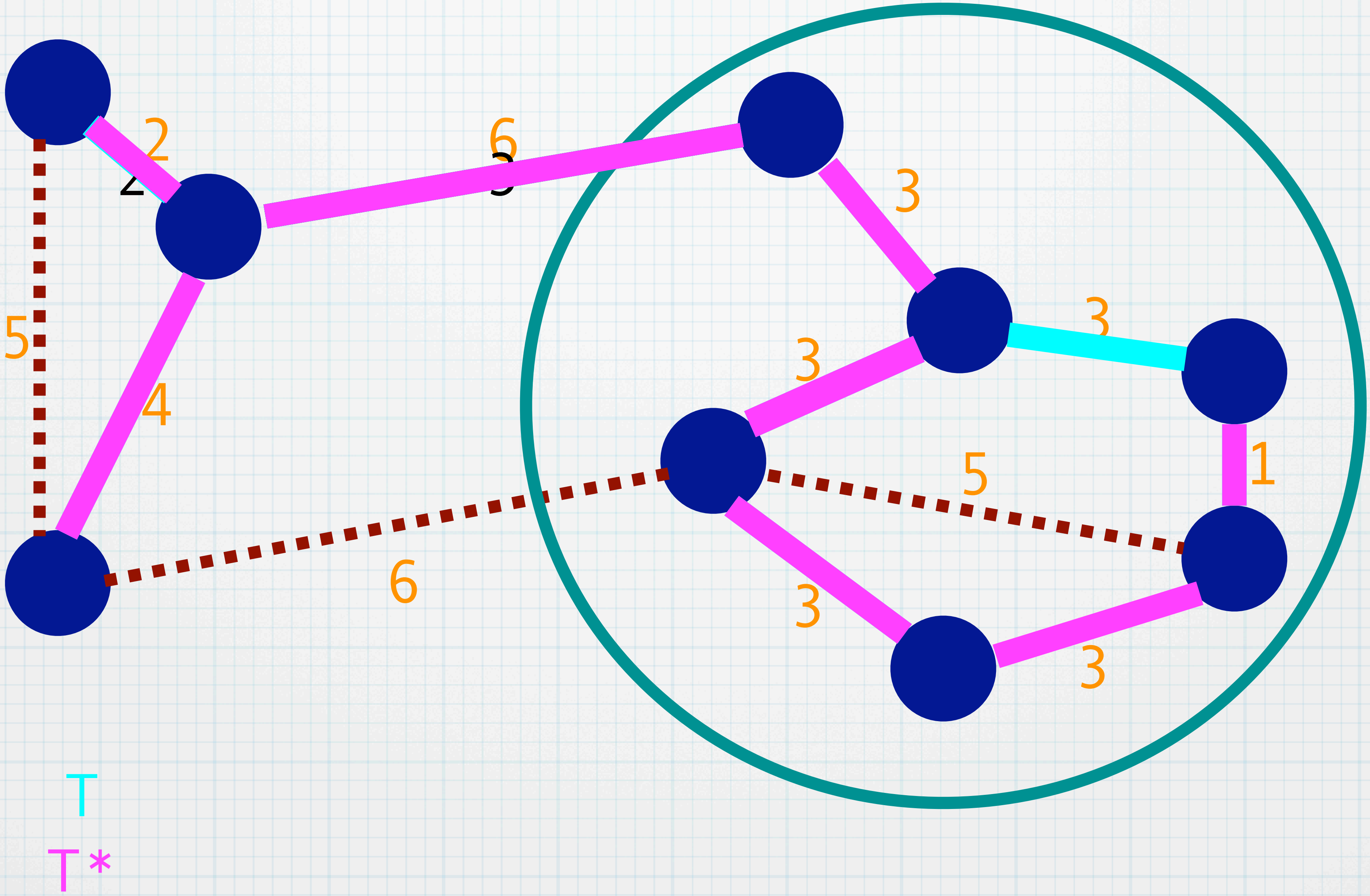
הדגמת ההוכחה



הדגמת ההוכחה



הדגמת ההוכחה



סיבוכיות האלגוריתם של Kruskal

מיון הקשתות: $O(|E| \log |E|)$.

מימוש CC על ידי מבנה נתונים ל-union/find.
מספר האיברים: $|V|$.
אנחנו מבצעים $2|E|$ פעולות find, ופחות מ-$|V|$ פעולות union.
נסתפק במימוש פשוט של union/find. הזמן הכולל לביצוע הפעולות הוא: $O(|E| \log |V|)$.

סיבוכיות הזמן: $O(|E| \log |E|)$ (G קשיר, אז $|E| \geq |V|-1$).

סכמה כללית לאלגוריתמים חמדנים לחישוב עפ"מ

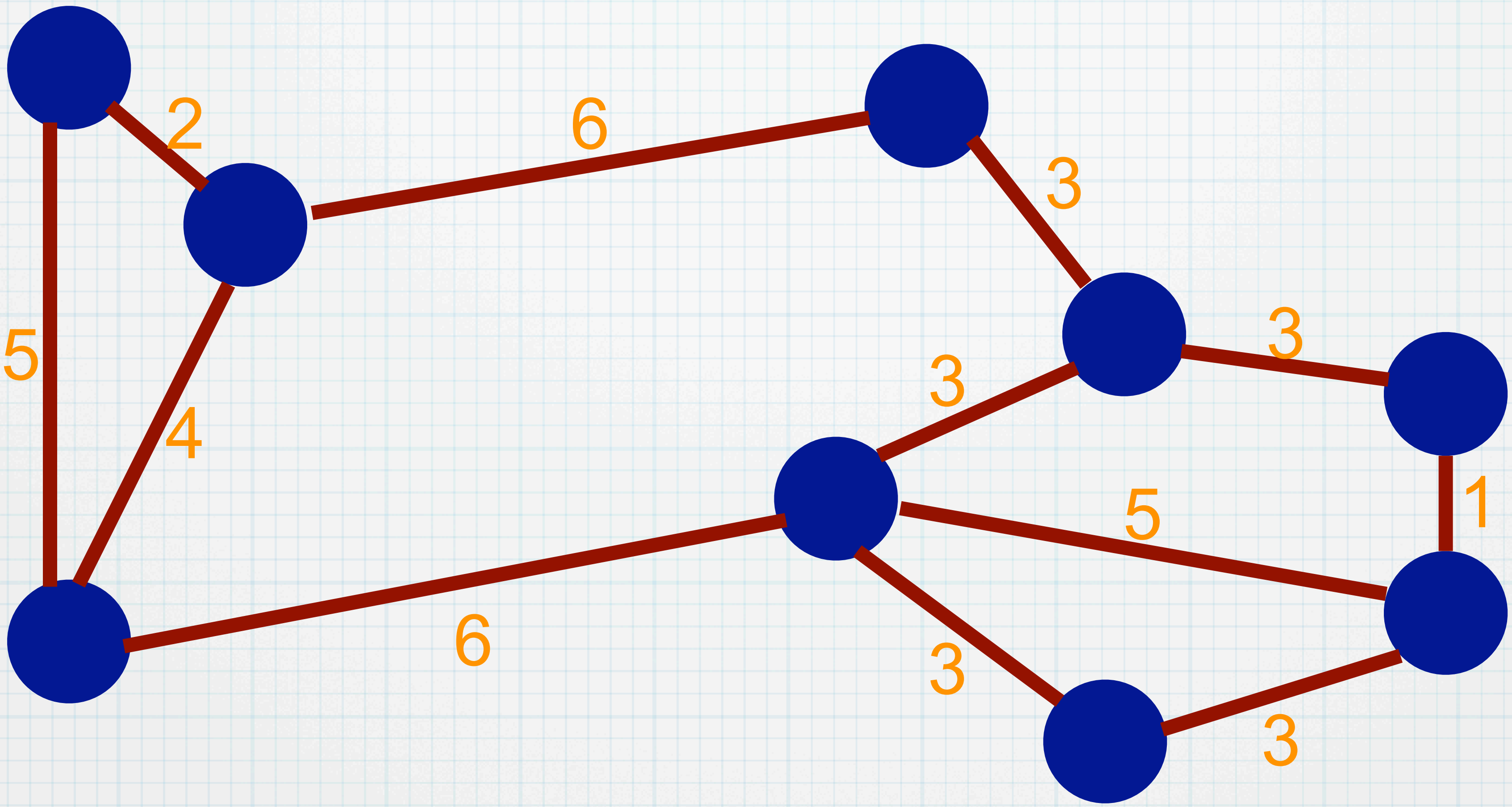
1. (איתחול) כל הקשתות לא צבועות.
2. אם בכל שפת חתך יש קשת כחולה ובכל מעגל יש קשת אדומה, עוצרים.
3. אחרת, מפעילים את אחד הכללים הבאים:

הכלל הכחול: בשפת חתך שבה אין קשת כחולה, צובעים בכחול קשת קלה ביותר שאינה צבועה.

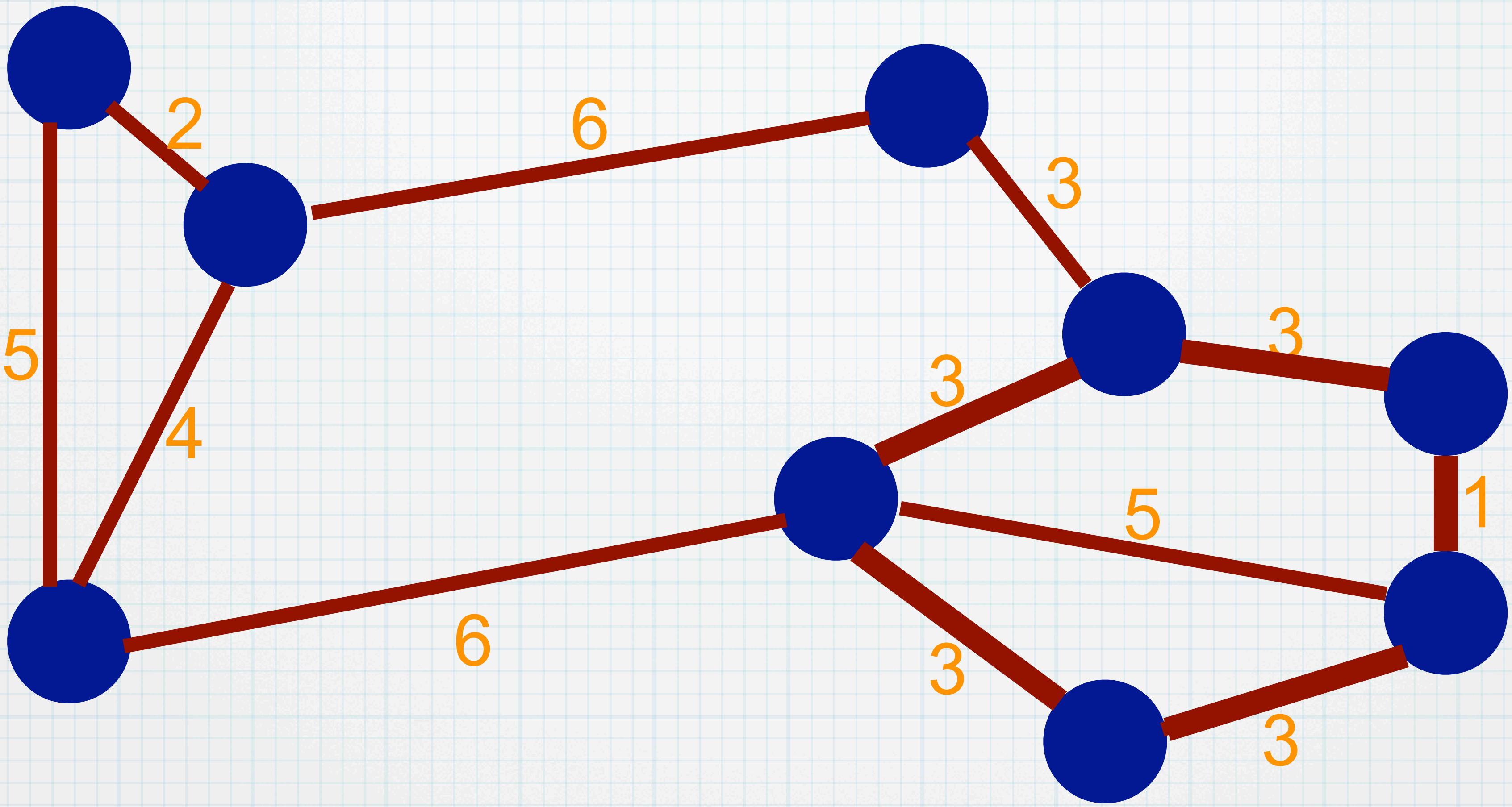
הכלל האדום: במעגל שאין בו קשת אדומה, צובעים באדום קשת כבדה ביותר שאינה צבועה.

4. חוזרים ל-2.

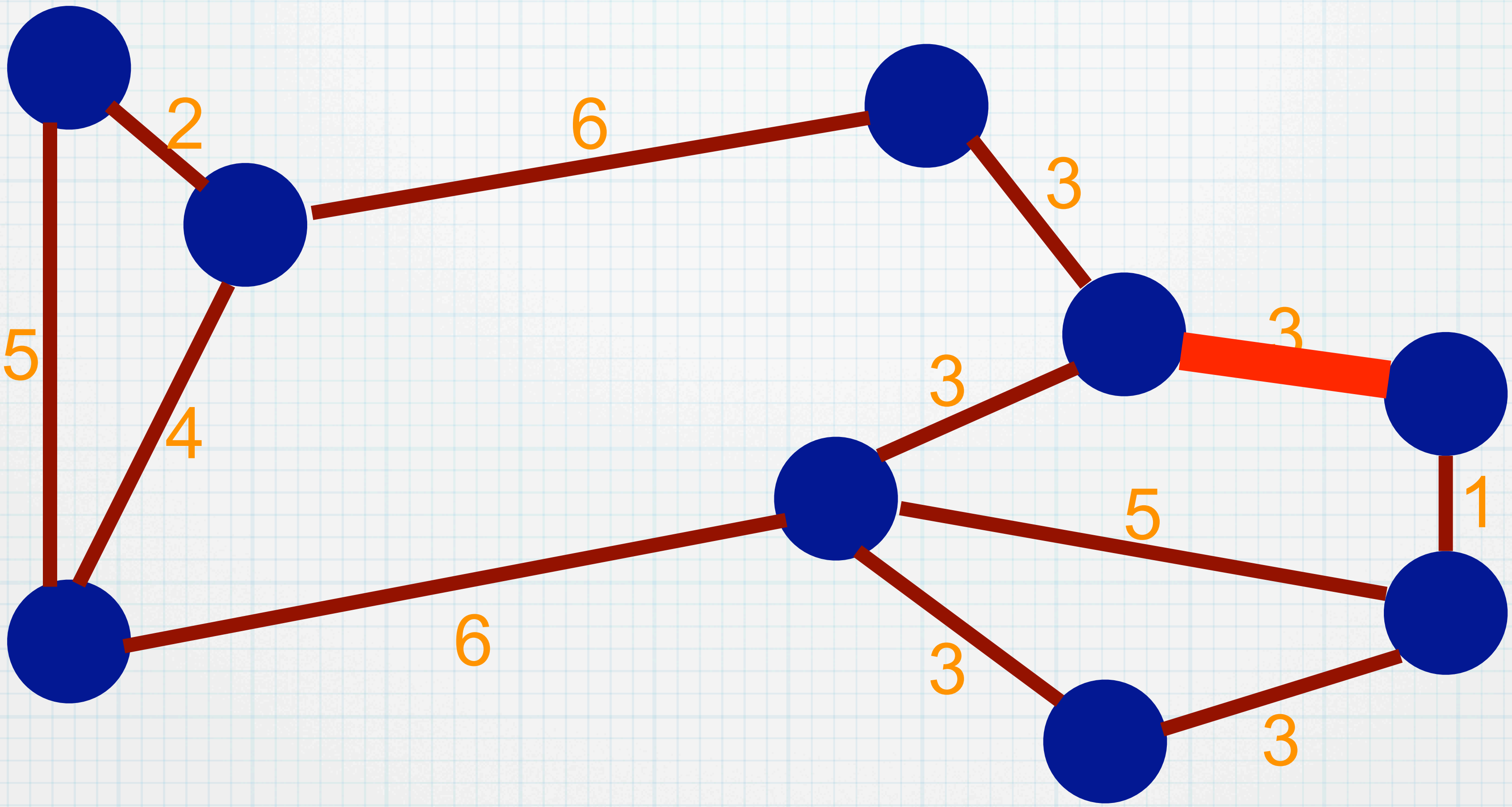
דוגמת הרצה



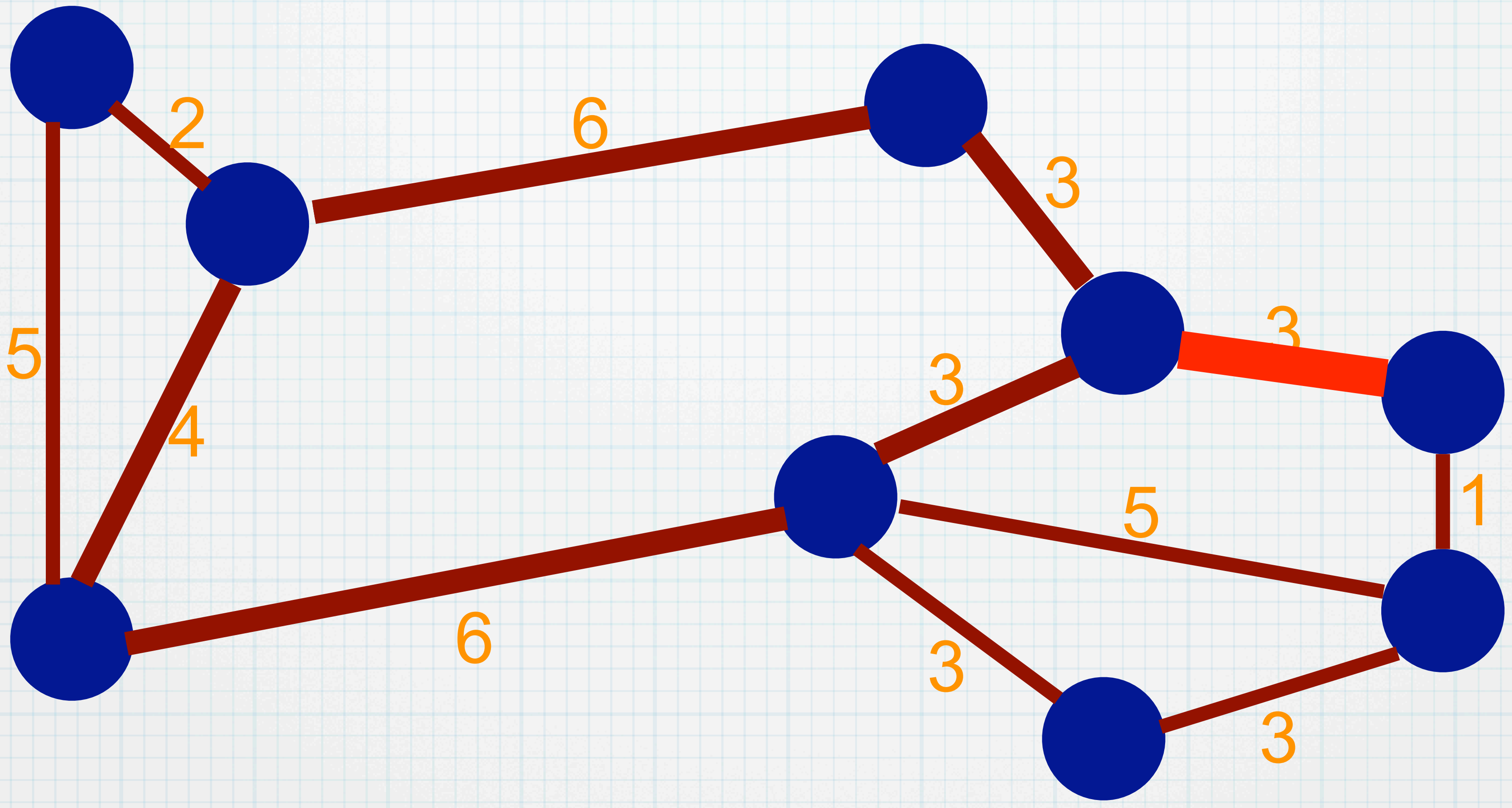
דוגמת הרצה



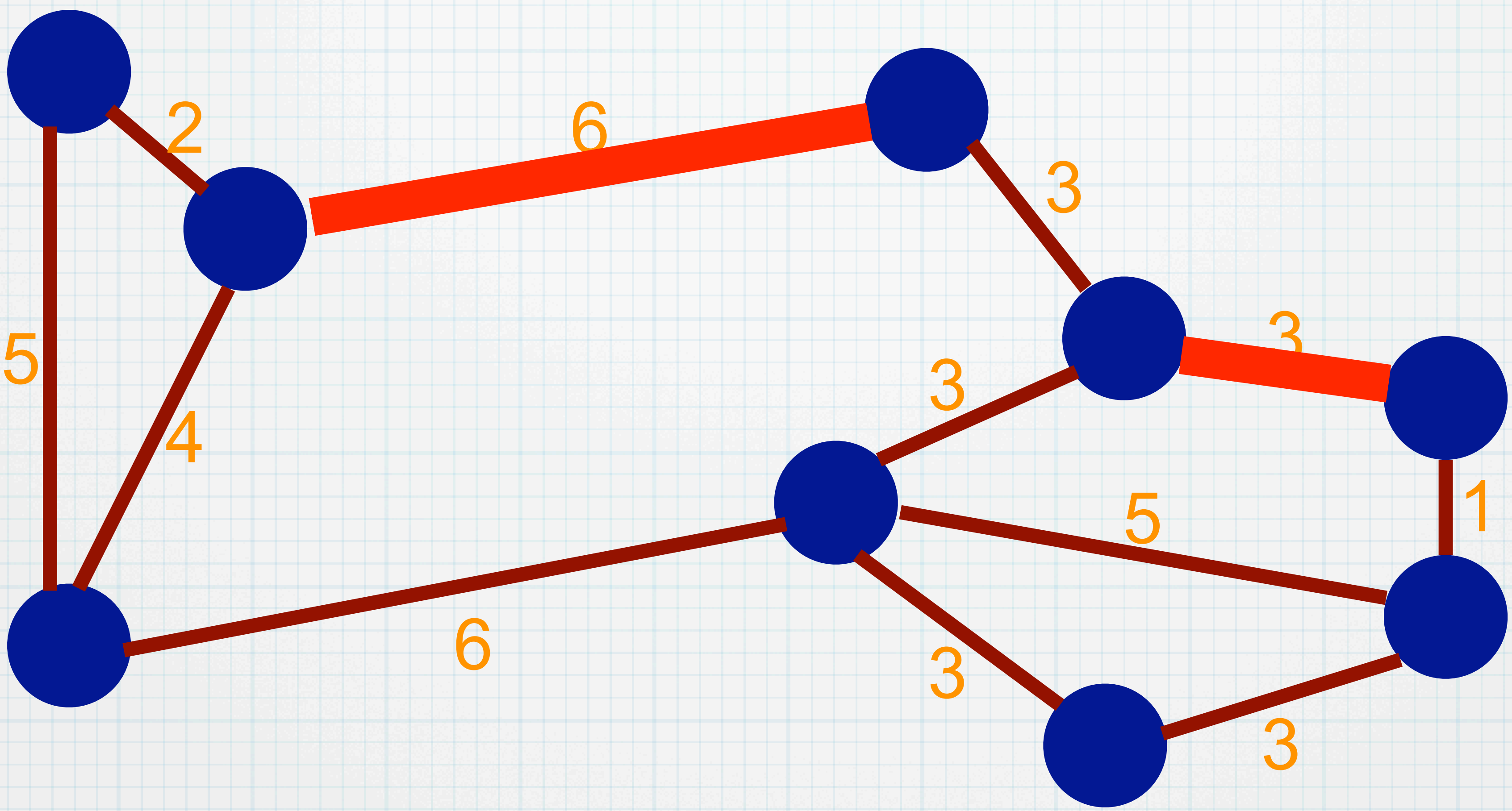
דוגמת הרצה



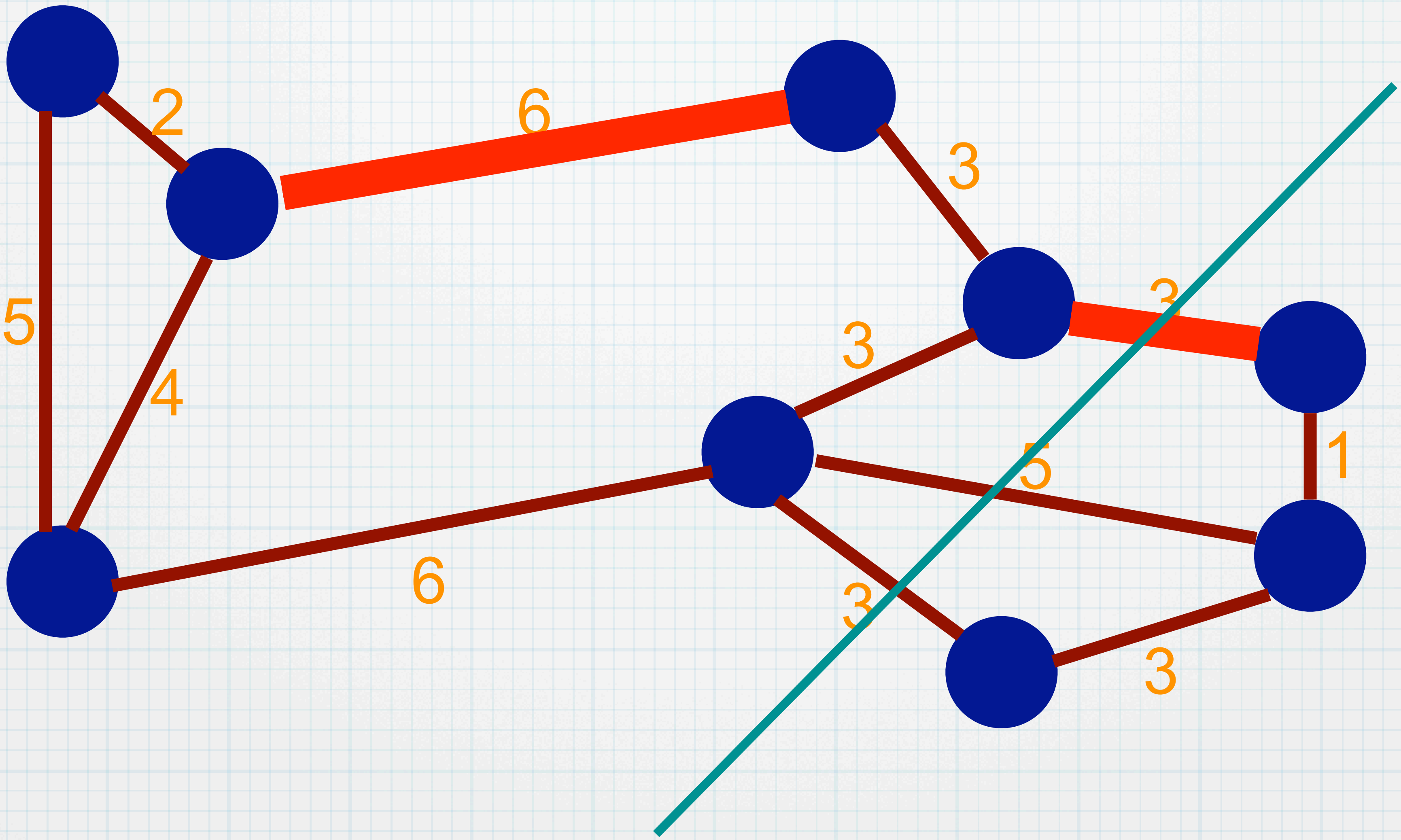
דוגמת הרצה



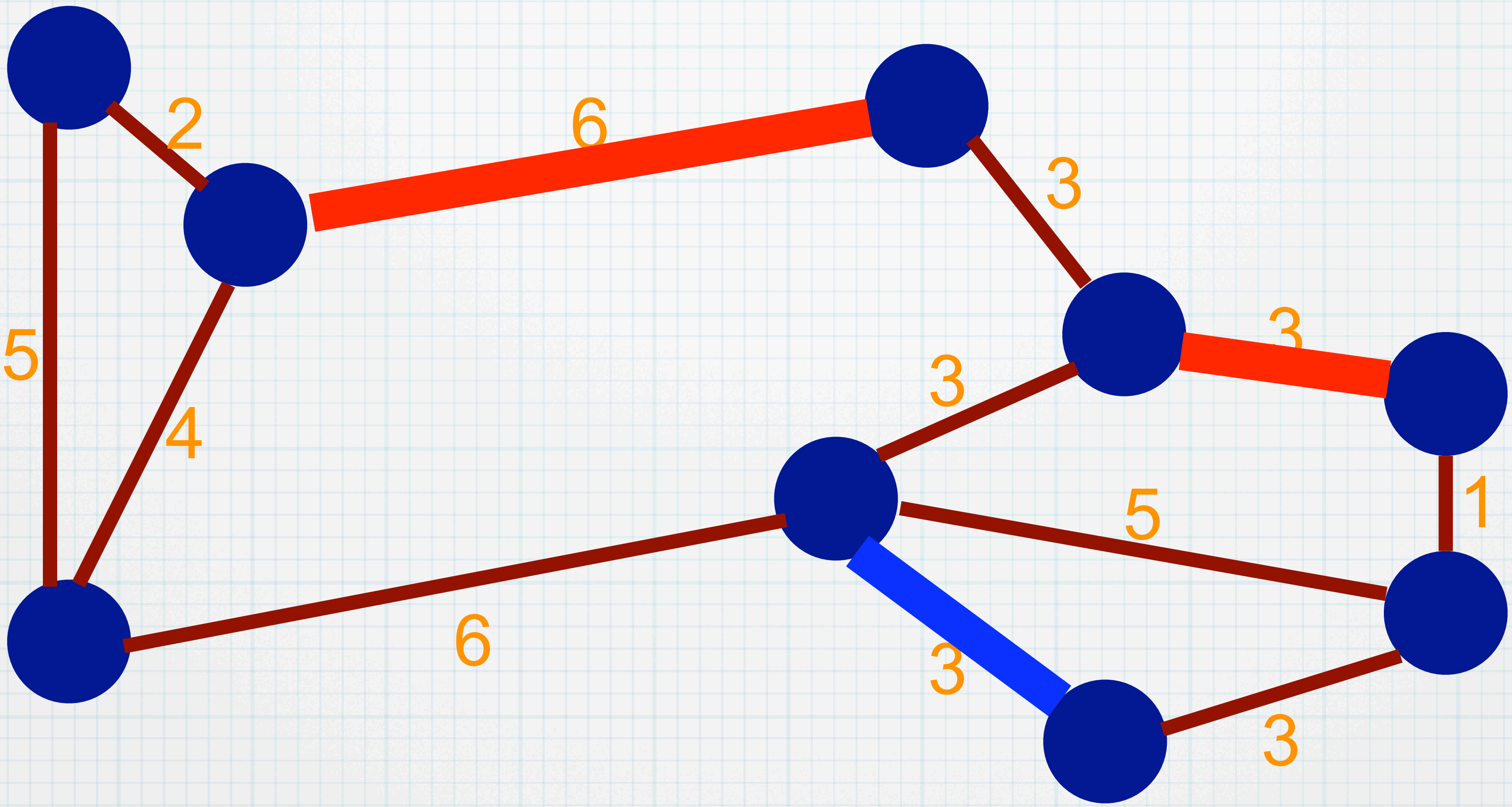
דוגמת הרצה



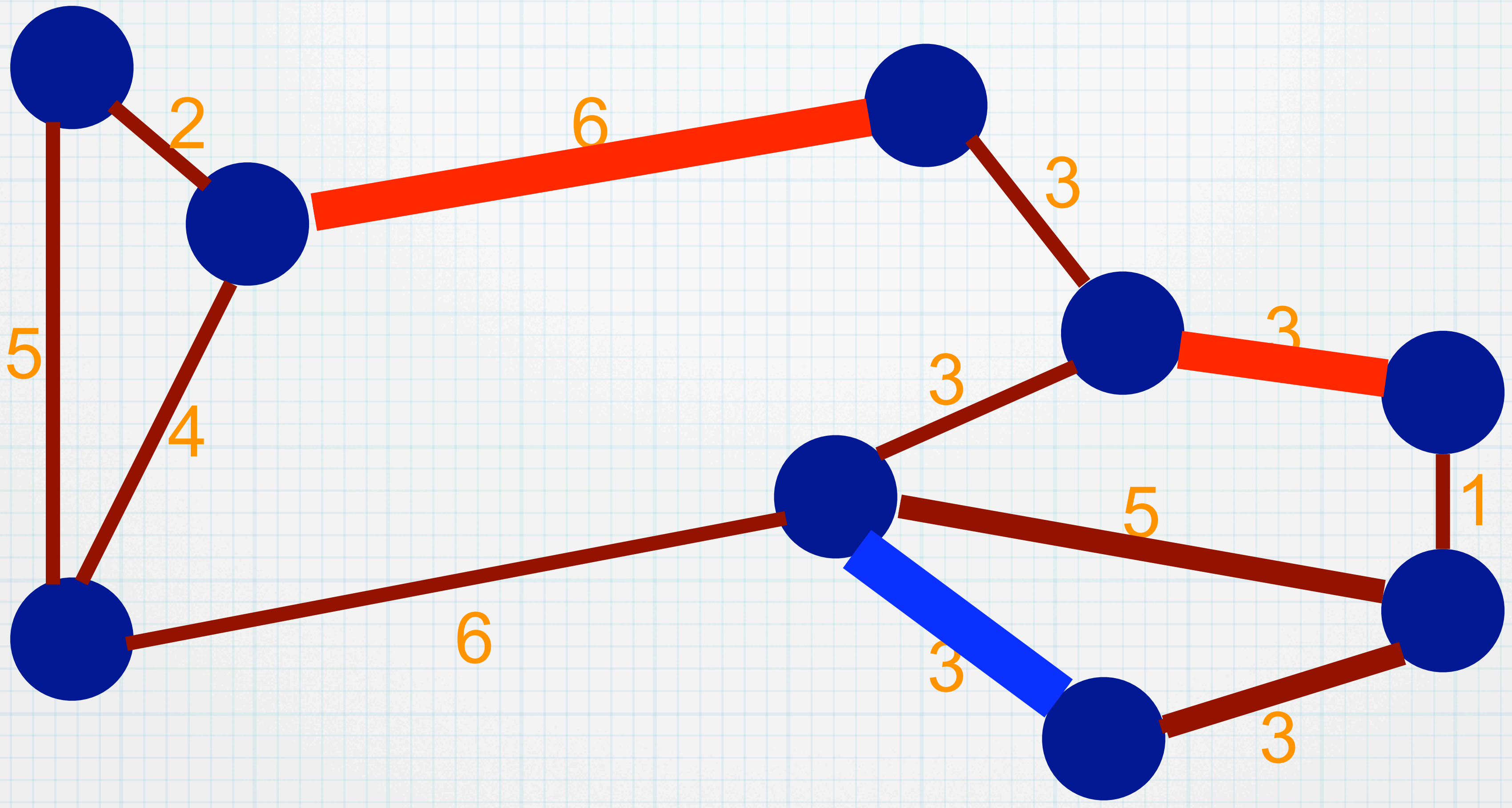
דוגמת הרצה



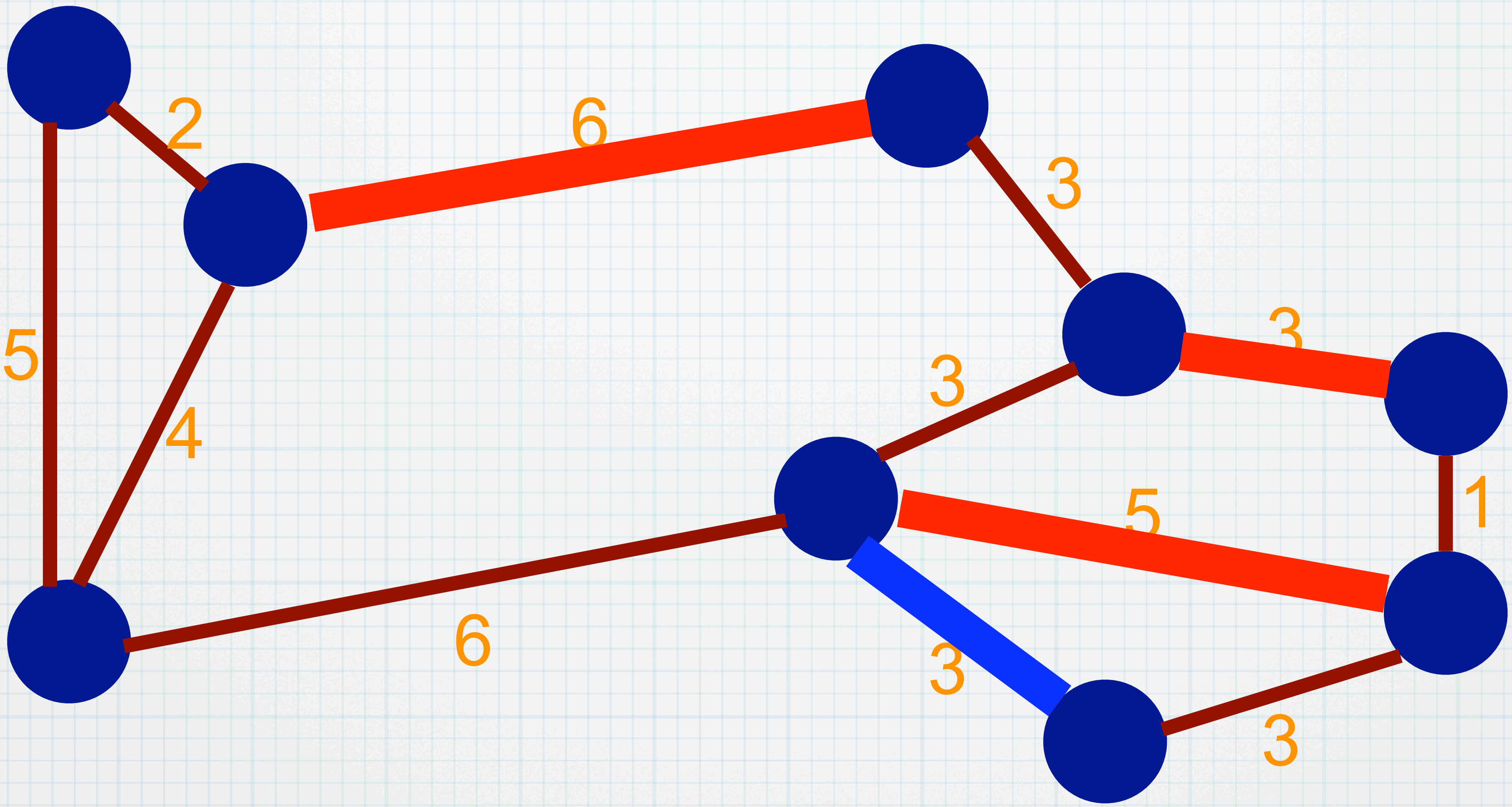
דוגמת הרצה



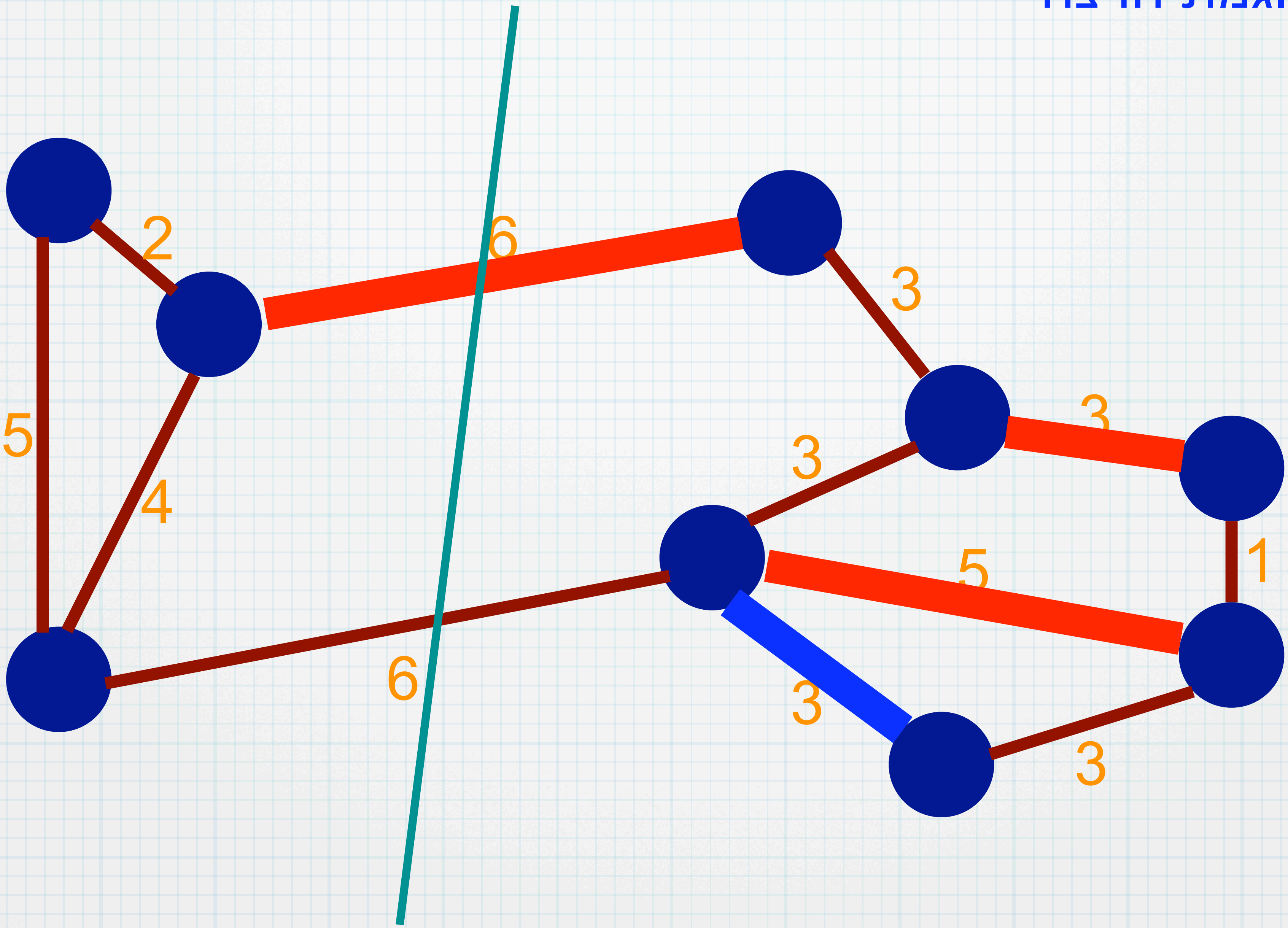
דוגמת הרצה



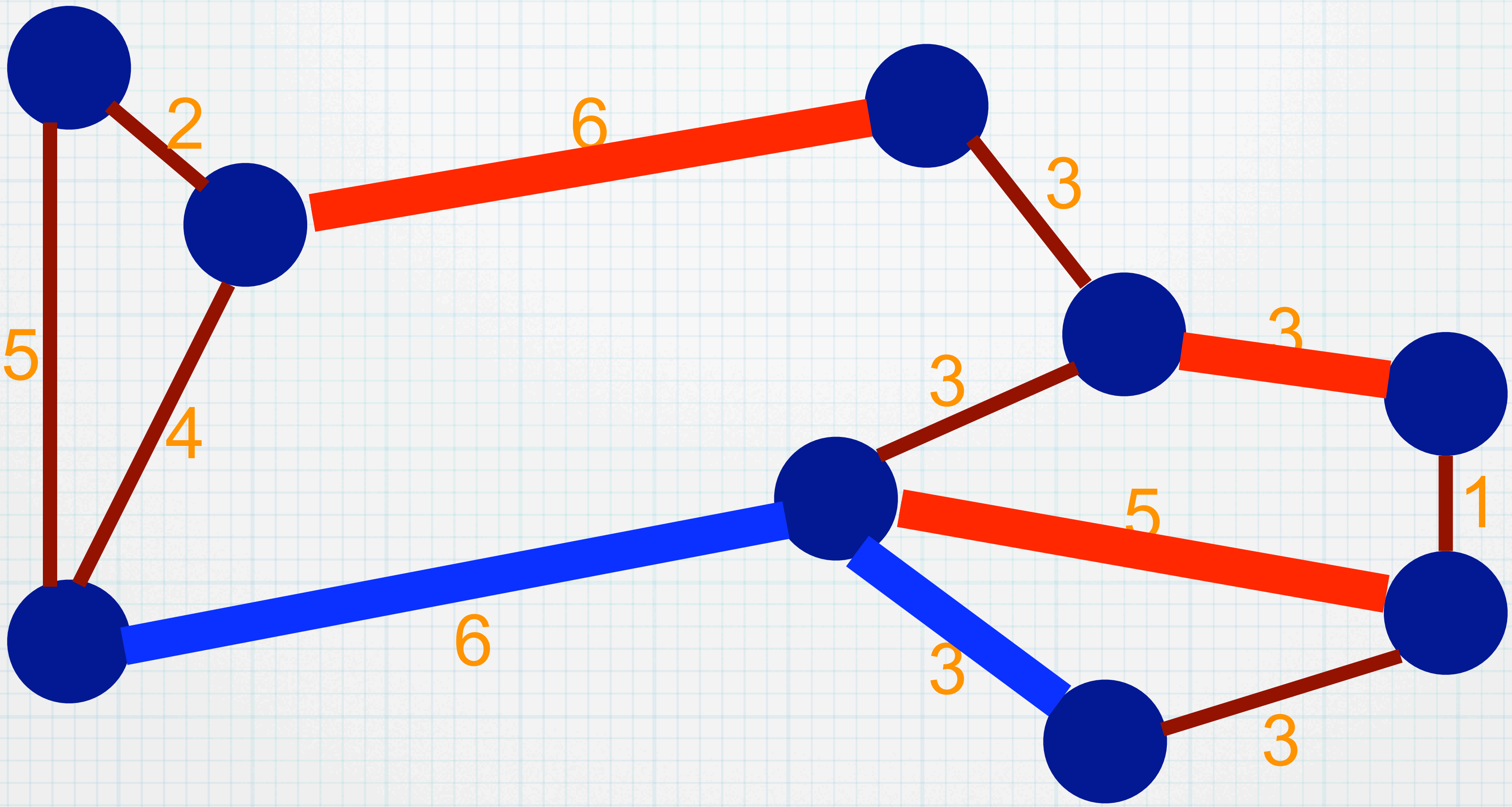
דוגמת הרצה



דוגמת הרצה



דוגמת הרצה



הוכחות

משפט: בסיום התהליך כל הקשתות צבועות ותת-הגרף הנפרש על ידי הקשתות הכחולות הוא עץ פורש מינימום.

למה: אחרי צביעת k קשתות, יש עפ"מ שמכיל את כל הקשתות הכחולות ואף קשת אדומה.

הוכחות

<u>הוכחת הלמה</u>: אינדוקציה על k. הבסיס הוא $k=0$. אין קשתות צבועות, ולכן הטענה בוודאי נכונה.

צעד האינדוקציה: נניח שצבענו את הקשתות $e_1, e_2, \ldots, e_{k-1}$ והטענה התקיימה, כלומר יש עפ"מ T שמכיל את כל הקשתות הכחולות עד עתה ואף קשת אדומה.
עכשיו צובעים את הקשת $e_k = (u,v)$.

מקרה א: הקשת נצבעת בכחול. כלומר, הכלל הכחול הופעל על חתך ש- e_k בשפתו. ב-T יש מסלול מ-u ל-v. מסלול זה חייב לחצות את החתך לפחות פעם אחת, והקשת החוצה e לא צבועה באדום (כי אין קשת אדומה ב-T) או בכחול (כי שפת החתך לא מכילה קשתות כחולות). אבל אז, לפי הכלל הכחול, משקל e_k אינו גדול ממשקל e ולכן אפשר להחליף ביניהן ולקבל עפ"מ 'T שמכיל גם את e_k.

הוכחות

מקרה ב: הקשת נצבעת באדום. כלומר, הכלל האדום הופעל על מעגל C שמכיל את הקשת e_k. נניח ש-T מכיל את הקשת הזו. נתבונן בחתך $S(e_k)$, המוגדר על פי T. המעגל C חוצה את החתך לפחות עוד פעם אחת בקשת e . קשת זו אינה צבועה כחול (אחרת T מכיל מעגל). היא גם אינה צבועה אדום (כי C אינו מכיל קשתות אדומות). לכן משקל e לכל היותר כמשקל e_k, לכן אפשר להחליף ביניהן ולקבל עפ"מ T' שאינו מכיל את e_k.

∴

הוכחות

הוכחת המשפט: על פי הלמה, בכל שלב הקשתות הכחולות פורשות יער (שיכול לכלול צמתים מבודדים). נניח שנותרה קשת לא צבועה e. אם e מחברת שני עמים כחולים, אז חתך המפריד ביניהם אינו כולל קשתות כחולות, ולכן אפשר להפעיל עליו את ה כלל הכחול. אם e סוגרת מעגל ששאר קשתותיו כחולות, אזי היא חייבת להיות קשת כבדה ביותר במעגל זה, ולכן אפשר לצבוע אותה באדום לפי הכלל האדום. ∴

מטרואידים ואלגוריתם החמדן

הגדרה: זוג סדור (S,I) הוא **מטרואיד** אם S קבוצה סופית ו-I קבוצה לא ריקה של תת-קבוצות של S המקיימת:

1. אם $I \ni A$ אזי לכל $A \supset B$, גם $I \ni B$. (ירושה)
2. אם $I \ni A,B$ ו-$|A| > |B|$, אזי יש $B \setminus A \ni x$ כך שמתקיים $I \ni B \cup \{x\}$. (החלפה)

איברי I מכונים **קבוצות בלתי-תלויות**. כל איבר מקסימלי ביחס להכלה נקרא **בסיס**.

דוגמאות

הקבוצה S היא קבוצת השורות של מטריצה. קבוצת שורות A נמצאת ב-I אםם איברי A בת"ל.

הקבוצה S היא קבוצת הקשתות של גרף. קבוצת קשתות A נמצאת ב-I אםם תת-הגרף המושרה ע"י A אינו מכיל מעגל פשוט.

אלגוריתם החמדן

נתונה פונקציית משקל $w:S \to \mathbb{R}$.

אלגוריתם החמדן מנסה למצוא קבוצה ב-I בעלת משקל כולל מירבי.

Greedy(S,w)
 $A \leftarrow \emptyset$
 while $\exists x \in S,\ A \cup \{x\} \in I$ do
  $x \leftarrow \text{argmax}\{w(x): x \in S \wedge A \cup \{x\} \in I\}$
  $A \leftarrow A \cup \{x\}$
 end while
return A

משפט Rado-Gale-Edmonds

תהי I קבוצה לא ריקה של תת-קבוצות של S שמקיימת את תכונת הירושה: $A \subset B \in I \Rightarrow A \in I$.

אלגוריתם החמדן מוצא, לכל פונקציית משקל חיובית w, קבוצה $A \in I$ בעלת משקל כולל מירבי אםם (S,I) מטרואיד.

הוכחת המשפט

בכיוון: מטרואיד ⇐ אלג׳ החמדן אופטימלי.

נניח בשלילה שהטענה לא נכונה. יהיו $a_1,a_2,a_3,\ldots,a_m$ האיברים בפלט האלג׳ בסדר לא עולה של משקל, ו- $b_1,b_2,b_3,\ldots,b_n$ האיברים בפתרון אופטימלי כלשהו באותו סדר.

יהי k האינדקס הראשון עבורו $w(b_k) > w(a_k)$.

$\{a_1,a_2,\ldots,a_{k-1}\}$, $\{b_1,b_2,\ldots,b_k\} \in I$

לכן קיים $1 \le i \le k$ עבורו $b_i \notin \{a_1,a_2,\ldots,a_{k-1}\}$ ו- $\{a_1,a_2,\ldots,a_{k-1},b_i\} \in I$.

זו סתירה לבחירה של האלג׳ החמדן ב- a_k.

הוכחת המשפט (המשך)

בכיוון: אלג' חמדן אופטימלי $\Leftarrow$ מטרואיד.

נניח בשלילה שהטענה לא נכונה. אזי יש קבוצות
$\{a_1,a_2,a_3,\ldots,a_m\}$, $\{b_1,b_2,b_3,\ldots,b_n\} \in I$
עבור $m < n$, אבל לכל $i \in \{1,2,\ldots,n\}$,
$\{a_1,a_2,a_3,\ldots,a_m,b_i\} \notin I$.
אלגוריתם החמדן ייכשל עם פונקציית המשקל

$w(a) = m + 2,\ \forall a \in \{a_1,a_2,a_3,\ldots,a_m\};$

$w(b) = m + 1,$

$\quad \forall b \in \{b_1,b_2,b_3,\ldots,b_n\} \setminus \{a_1,a_2,a_3,\ldots,a_m\};$

$w(c) = 0$, otherwise.

$\therefore$

תרגיל בית

1. נתונים n מערכים ממויינים של מספרים טבעיים באורכים שונים. מיזוג של שני מערכים באורך a ו-b יוצר מערך ממויין אחד באורך a+b בזמן O(a+b). תנו אלגוריתם לחישוב סדר אופטימלי של מיזוגים של זוגות מערכים (כולל תוצאות ביניים) ליצירת מערך ממויין אחד מכל n מערכי הקלט.

2. הוכיחו את נכונות האלגוריתם של Kruskal על סמך משפט Rado-Gale-Edmonds.