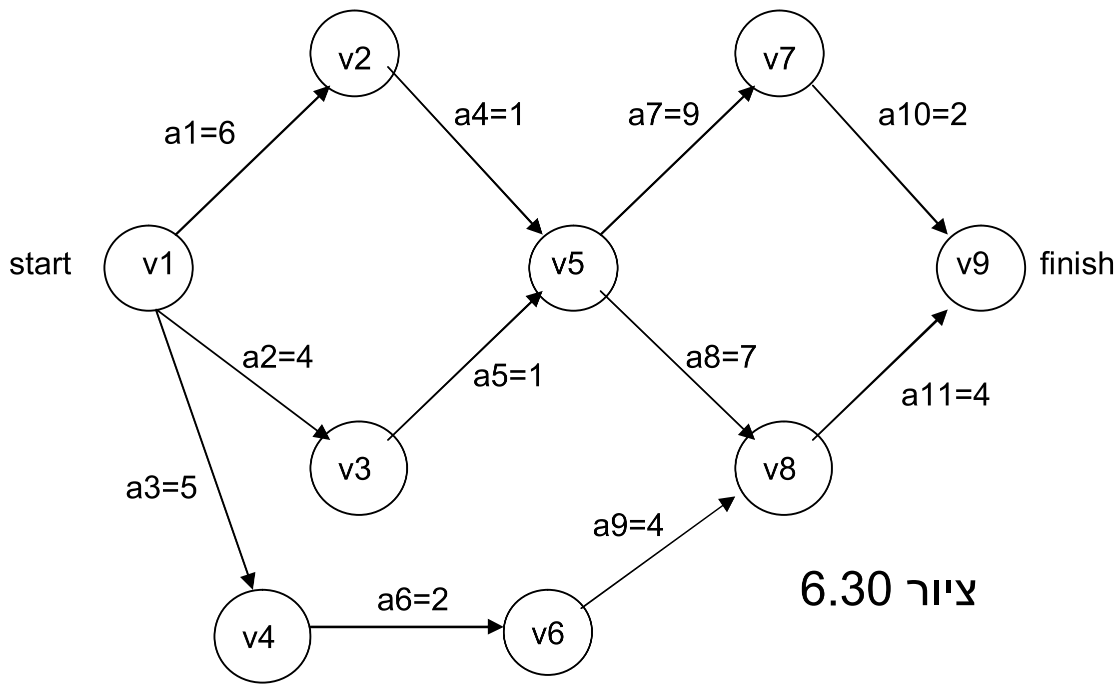
<u>מסלולים קריטיים</u>

- רשת AOE – (ACTIVITY ON EDGES) רשת שבה המשימות לביצוע מיוצגות ע"י חיצים מכוונים. הצמתים ברשת זו מייצגים מאורעות.
- מאורע – מציין סיום פעילות מסוימת (זו שהתבצעה על הקשת המובילה אליו) מאורע קורה רק כאשר כל הפעיליות הנכנסות אליו הושלמו ואילו הפעילויות היוצאות מהצומת לא יתחילו עד אשר המאורע שבצומת קרה.
  קשת=פעילות



ציור 6.30

בציור 6.30 אנו רואים 11 פעילויות A1-A11 אשר משך זמן ביצוע כל אחת מצוין בגרף ותשעה צמתים (מאורעות) V1-V9.
(משך זמן הביצוע שמצוין בגרף הינו הערכה בלבד ובהמשך ההרצאה נראה כיצד נשפרו).

- הוספת אילוצים – במקרה שפעילות מסוימת חייבת להתחיל אחרי שקרה מאורע כלשהו ואנו לא רואים זאת בגרף ניתן להוסיף פעילות דמה שזמנה הוא 0 .

- שימושים של רשת AOE - הערכת פרויקטים אשר כוללת החלטה לגבי עובדות על פרויקט כמו : מהו הזמן המינימלי שבו ניתן לסיים את הפרויקט (בהנחה שאין מעגלים ברשת) וכן אילו פעילויות כדאי לזרז על מנת לקרב את זמן סיומו.

  כיוון שהפעילויות ברשת AOE יכולות להתבצע במקביל הזמן המינימלי להשלמת הפרויקט הוא אורכו של המסלול הארוך ביותר מצומת ההתחלה לצומת הסיום.
  (אורך מסלול זה הוא סכום הזמנים שעל הפעילויות במסלול).

- מסלול קריטי – המסלול בעל הזמן הארוך ביותר מצומת ההתחלה לצומת הסיום

  לדוגמא – בציור 6.30 מסלול הקריטי עובר דרך הצמתים V1,V2,V5,V7,V9 ואורכו 18.
  ברשת יכול להיות יותר ממסלול קריטי אחד.

לדוגמא– V1,V2,V5,V8,V9 שאורכו הוא בהכרח 18.

- הזמן המקודם ביותר שבוא יכול לקרות מאורע V1 הוא אורכו של המסלול הארוך ביותר מצומת המקור לצומת V1 .זמן זה הוא הזמן המקודם ביותר של כל הפעילויות (המיוצגות ע"י חיצים) העוזבות את צומת V1 .

- נסמן(E (i -(EARLIEST) הזמן המוקדם ביותר שפעילות Ai יכולה להתחיל
לדוגמא: E(7)=E(8)=7

- נסמן L(i)- (LATEST) הזמן המאוחר ביותר שפעילות Ai יכולה להתחיל מבלי לפגוע או להגדל את משך הפרויקט כולו.
לדוגמא : E(6)=5 , L(6)=8 ; E(8)=7 , L(8)=7

- הגדרה : פעילות קריטית היא פעילות שבה E(i)=L(i)
- ההפרש E(i) – L(i) הוא מידה לקריטיות של הפעולה והוא מראה כמה זמן ניתן לעקב או להאיט ביצוע פעילות i.
לדוגמא : אם פעילות A6 מואטת ולוקחת יומיים נוספים זה לא ישפיע על זמן הסיום של הפרויקט.

- האצת פעילות קריטית תגרום לזירוז הפרויקט כולו בתנאי שפעילות זו קיימת על כל המסלולים הקריטיים בגרף.

לדוגמא : הפעילות A11 היא קריטית אולם אם נאיץ אותה כך שתיקח שלושה ימים במקום ארבעה משך ביצוע הפרויקט לא יתקצר . לעומת זאת פעילות A1 נמצאת על כל המסלולים הקריטיים וקצורה אכן מקצר את משך הפרויקט כולו.

- המטרה של ניתוח מסלולים קריטיים היא לזהות פעילויות קריטיות וכך שהמשאבים יוכלו להתרכז בפעילויות האלה על מנת להקדים את זמן סיום הפרויקט.

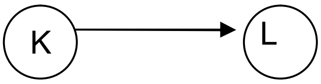
- ניתוח המסלולים הקריטיים יכול להיות מובוצע גם ע"י רשתות AOV כאשר אורך המסלול יהיה סכום זמני הפעילויות הרשומים בצמתים כלומר צומת תיצג את הפעילות.

- זמני הפעילויות הן רק הערכות לכן לאחר ביצוע כל שלב יש לבצע הערכה מחדש של משך הפרויטק כולו כך שניתן יהיה לשפר את הדיוק בזמני הערכה . שינויים אלו בזמני הפעילויות יכולים לגרום לפעילויות שקודם לא היו קריטיות להיות קריטיות ולהפך.

<u>תכנון האלגוריתם להערכת E(i) ו L(i) לכל פעילות ברשת AOE</u>

כאשר E (i) ו L(i) ידועים אז הפעילויות הקריטיות ניתנות לזיהוי בקלות ע"י ההפרש (Li-Ei),
במקום לשמור את E (i) ו L(i) לכל פעילות ברשת AOE נעדיף לשמור את EE(j)

(EARLEST EVENT) ו LE(j) (LAST EVENT)
לכל מאורע (צומת) ברשת , מאשר לשמור זאת עבור הקשתות (פעילויות).
לכן את פעילות Ai מיוצגת ע" הקשת <K,L> נוכל לחשב את E (i) ו L(i) ע"י הנוסחאות :



E(i)=EE(k)
L(i)=LE(L)-Duration(Ai)

הזמנים EE(j) (j) LE מחושבים בשני שלבים , שלב ראשון מציאת EE(j) ע"י הליכה קדימה על הגרף .שלב שני מציאת LE(j) ע"י חזרה בגרף מהסוף להתחלה .

<u>אלגוריתם למציאת סדר טופולגוי</u> : בתוספת המשפט המסומן ב **

```
FOR I=1 TO n DO
  COUNT[I]=INDEGREE OF VERTEX I;
  TOP=0;
  FOR I=1 TO N DO
    IF COUNT[I]=0 THEN
      BEGIN
        COUNT[I]=TOP;
        TOP=I;
      END;
```

```
FOR I=1 TO N DO
  BEGIN
    IF TOP=0 THEN
      WRITE('there is a direct cycle'); STOP!;
      J=TOP;
      TOP=COUT[TOP]
      WRITE(J);
      PRT=LINK(J);
      WHILE PTR <> 0 DO
        BEGIN
          K=VERTEX(PTR);
          COUN[K]=COUNT[K]-1;

****** EE(K)=MAX{EE(K),EE(J)+DUR(PTR)}
עבור מסלולים קריטיים
****** LE(K)=MIN{LE(K),LE(J)-DUR(PTR)}

          IF COUNT[K]=0 THEN
            BEGIN
              COUNT[K]=TOP;
              TOP=K;
            END;
          PTR=LINK[PTR];
        END;
END;
```

סיבוכיות האלגוריתם :

הסיבוכיות של האלגוריתם היא :

$O(\sum_{i=1}^{n} Di+V)$ והשווה ל – $O(E+V)$

- סיבוכיות היא לינארית

- <u>שלב ראשון</u> : מבוסס על הליכה קדימה בגרף מצומת ההתחלה לצומת הסיום מכאן שמתחילים עם
EE (i)=0 ומחשבים את זמני ההתחלה ע"י הנוסח
: 6.2

EE(j)=MAX {EE(i)+ <i,j > משך פעילות }
i € P(j)

- כאשר P(j) היא קבוצת כל צמתי האב של הצומת j
.

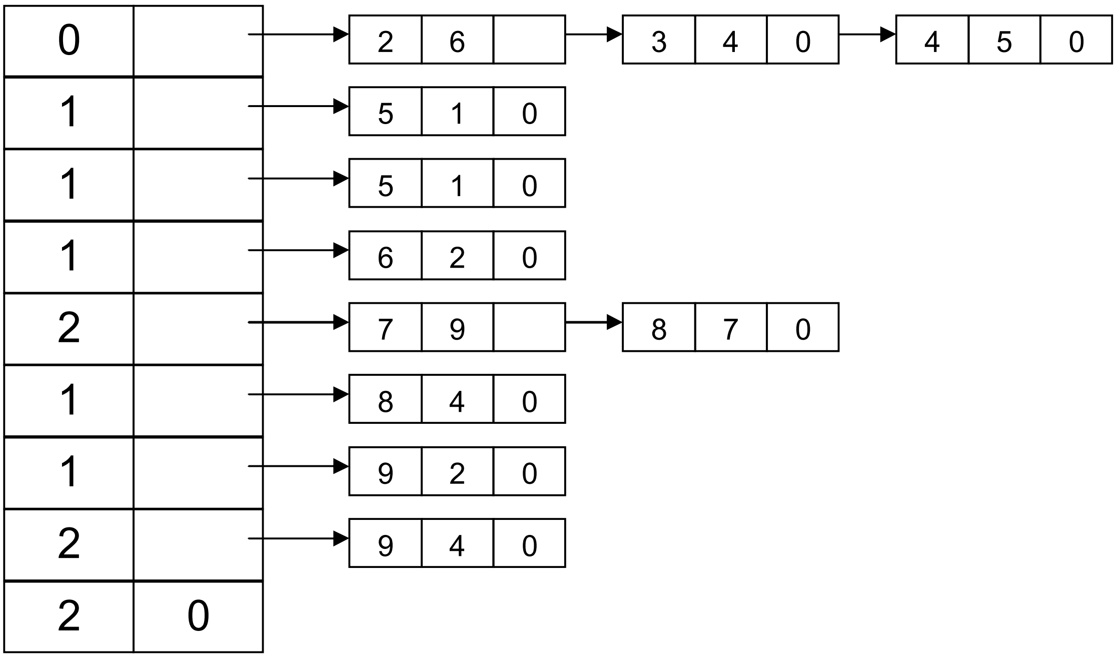
- במידה וחישוב זה היה מתבצע בסדר טופולוגי הרי שזמן ההתחלה המוקדם של כל המקדימים של j היה מחושב לפני החישוב של EE(j) (לכן אפשרית ההליכה קדימה) .

- האלגוריתם לביצוע חישוב של EE(j) יכול להיות מבוצע בקלות בעזרת האלגוריתם של הסדר הטופולוגי עם הכנסת הצעד הבא בין שורות 23 ו 24 באלגוריתם :

```
EE(k)=MAX { EE(k),EE(j)+DUR(ptr)}
```

אנו מניחים שהמערך EE מאותחל לאפס וש-DUR הוא שדה נוסף ברשימת הקשרים אשר מכיל את משך הפעילות.

EE(j) מעודכן בכל פעם ש- EE(i) של אחד מן המקימים שלו ידוע.



	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	stack
initial	0	0	0	0	0	0	0	0	0	1
output v1	0	6	4	5	0	0	0	0	0	4 3 2
output v4	0	6	4	5	0	7	0	0	0	6 3 2
output v6	0	6	4	5	0	7	0	11	0	3 2
output v3	0	6	4	5	5	7	0	11	0	2
output v2	0	6	4	5	7	7	0	11	0	5
output v5	0	6	4	5	7	7	16	14	0	8 7
output v8	0	6	4	5	7	7	16	14	16	7
output v7	0	6	4	5	7	7	16	14	18	9

output
v9

- הכנסת הצעד לתוך האלגוריתם יגרום למציאת ה - EE(j) המינימלי עבור כל צומת j ביחד עם ביצוע הסדר הטופולוגי.

- תרגיל : הפעל את האלגוריתם של המיון הטופולוגי עם הוספה הנ"ל על גבי תרשים 6.30 .

- סיבוכית האלגוריתם נשארת O (E+V) מפני שלא שינינו מאומה באלגוריתם להוציא משפט השמה אחד.

שלב שני : בשלב זה אנו הולכים אחורה מצומת הסיום לצומת ההתחלה לחישוב LE (i) כאשר נשתמש באותו אלגוריתם שהשתמשנו בשלב הראשון.
נתחיל עם LE(n)=EE(n) (כלומר השלב הסופי שהוא שווה 18).
ונשתמש בנוסחה 6.3

LE(j)=MIN{LE (i)-DURATION OF <j,i>}
i € S(j)

- כאשר S(j) היא קבוצת הצמתים המוצבים ע"י צומת j .

- משוואה זו אומרת : אם <j,i> היא פעילות והזמן המאוחר ביותר של מאורע i הוא LE(i) אזי מאורע j חייב להופיע לא יאוחר מאשר LE(i)- <j,i> DURATION OF כלומר חייבים לחשב קודם את הזמן המאוחר של המאורעות העוקבים למאורע j .

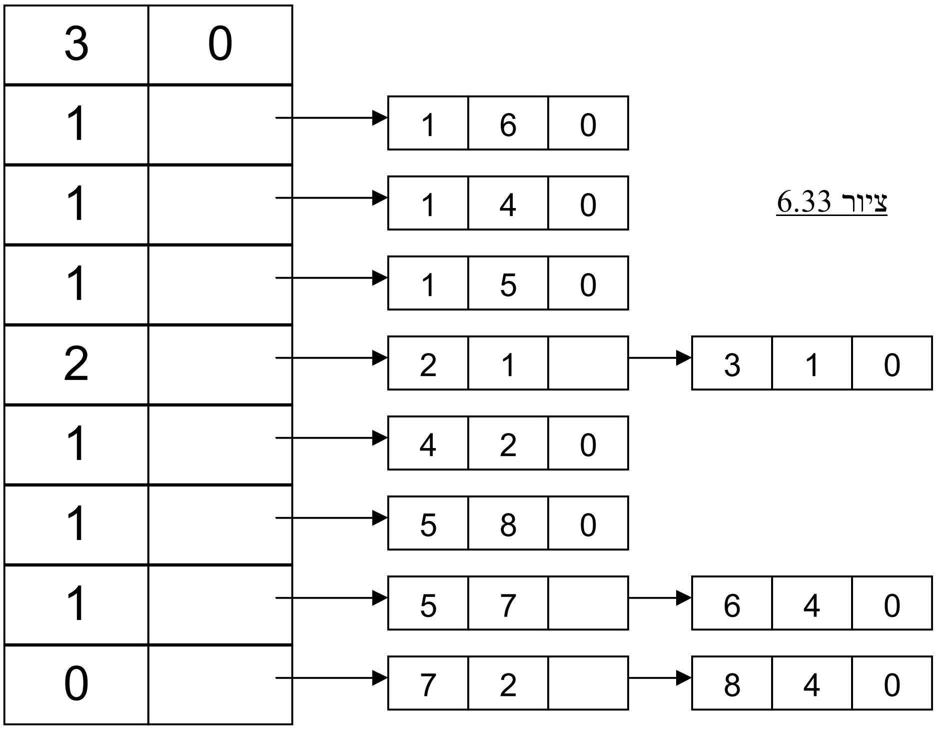
- הערכים ההתחלתיים של LE(i) יכולים להיות שווים ל - EE(n) (מפני שבסופו של דבר הם גדולים או שווים להם).

- מבנה הנתונים המוצע:זהה לחישוב של EE(i) , פשוט נשתמש ברשימת קשרים הפוכה כאשר שדה COUNT יכיל את ה- OUTDEGREE של כל צומת ושדה LINK יצביע לרשימת הצמתים המצביעים על הצומת שבה מדובר.

- האלגוריתם יהיה זהה לאלגוריתם הקודם רק שעכשיו נכניס את הצעד הבא :

  LE(k)=MIN {LE(k),LE(j)-DUR(ptr)}

  באותו מקום (בין השורות 23-24 באלגוריתם של הסדר הטופולוגי)



ציור 6.33

- ציור 6.33 מתאר את התהליך ל אותה רשת מציור 6.30

Activity networks, Topological sort and critical path

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	stack
initial	18	18	18	18	18	18	18	18	18	9
output v9	18	18	18	18	18	18	16	14	18	8 7
output v8	18	18	18	18	7	10	16	14	18	6 7
output v6	18	18	18	18	7	10	16	14	18	4 7
output v4	3	18	18	8	7	10	16	14	18	7
output v7	3	18	18	8	7	10	16	14	18	5
output v5	3	6	6	8	7	10	16	14	18	3 2
output v3	2	6	6	8	7	10	16	14	18	2
output v2	0	6	6	8	7	10	16	14	18	1

(b) Computation of topological-order modified to compute latest event times

k(9)	=EE(9)
k(7)	=min{le(9)-2}=16
k(8)	=min{le(9)-4}=14
k(5)	=min{le(7)-9, le(8) - 7}=7
k(2)	=min{le(5)-1}=6
k(3)	=min{le(5)-1}=6
k(6)	=min{le(8)-4}=10
k(4)	=min{le(6)-2}=8
k(1)	=min{le(2)-6, le(3) -4, le(4) -5}=0

(c) Computations of le Directly from Equesion (6.3) Using a reverse Topological Order

<u>ציור 6.33</u>

ע"י השימוש בערכי EE ו LE של כל צומת נוכל לחשב את E(i) ו L(i) ואת רמת הקריטיות של כל משימה (Li – Ei).

activity	E	L	L-E
a1	0	0	0
a2	0	2	2
a3	0	3	3
a4	6	6	0
a5	4	6	2
a6	5	8	3
a7	7	7	0
a8	7	7	0
a9	7	10	3
a10	16	16	0
a11	14	14	0

<u>ציור 6.34</u>

הפעיליות הקריטיות על פי התרשים 6.34 הם
A11,A10,A8,A7,A4,A1.
ע"י מחיקת כל הפעיליות שאינן קריטיות נקבל את כל המסלולים הקריטיים שהם בהכרח גם מסלולים על הגרף.

- הערה : האלגוריתם של סדר טופולוגי מוצא רק מעגלים מכונים ברשת.

יכולות להיוצר בעיות אחרות כמו גרף שבו יותר מצומת אחת בעלות INDEGREE שהוא אפס ואז האלגוריתם זה אינו מטפל בהם.