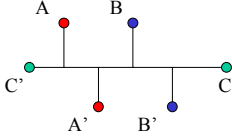


דרישות ממנגנון הנתוב

1. בצועים
2. נכונות: הודעות צריכות להגיע ליעדן
3. אמינות
4. יציבות (התכנסות)
5. הגינות:



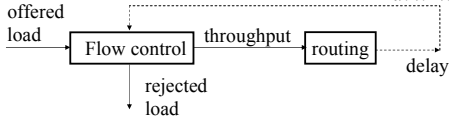
תעבורה מקסימלית
תגרום להרעבת
ההודעות בקשר c-c'

נתוב (Routing)

- הגדרה: קביעת מסלול רצוף בין צמת מקור לצמת יעד דרכו תועבר חבילה (או חבילות) מן המקור ליעד.
 - תהליך הנתוב מסובך:
 - דורש תאום בין מספר רב של צמתים (לא רק שבנים)
 - דורש להתמודד עם מספר רב של מצבים (גם בלתי צפויים)
 - קיימת דרישת בצועים קפדנית שכן הנתוב הוא פעולה בסיסית של הרשת.
 - שני הבטים:
 1. בחירת המסלול
 2. עצם העברת ההודעה ליעדה
- הבט 2 בוד"כ פשוט ביחס ל- 1: נפתר ע"י טבלאות; רשום בגוף ההודעה; וכו'

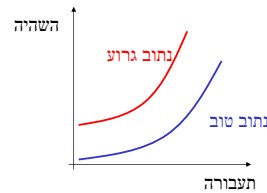
השפעת הנתוב על בצועי הרשת

- הרשת מפעילה ממנגנונים להגבלת התעבורה הנכנסת כדי לוטת את ההשהיה.

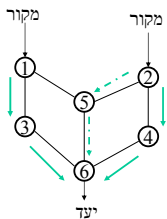


קריטריונים לבצועים

1. כמות התעבורה (throughput)
2. השהיה נמוכה
 1. הבט הרשת
 2. הבט המשתמש
3. מחיר

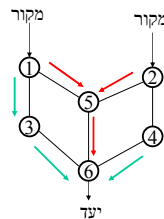


נתוב - דוגמא



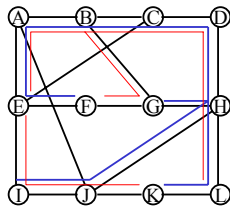
- קבול כל הקווים 10 יחידות.
- א. עמס נמוך:
 - 5 יחידות זרימה מכל מקור.
 - נתוב טוב מול רע.
- ב. עמס גבוה:
 - 5 יח' ממקור 1, 15 ממקור 2.
 - אם מותר מסלול יחיד: 5 יח' תדחנה.
 - אם מותר לפצל זרימה: $2 \times 7.5 = 15$
- ג. עמס מקסימלי:
 - מסלולים נפרדים: גרוע: 10, טוב: 20.
 - פצול זרימה: נתוב טוב: 30.

נתוב - דוגמא



- קבול כל הקווים 10 יחידות.
- א. עמס נמוך:
 - 5 יחידות זרימה מכל מקור.
 - נתוב טוב מול רע.

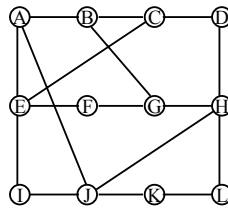
נתוב סטטי – בלתי מפוצל – טבלא לצמת A



יעד	שכנו
A	--
B	B
C	B
D	B
E	E
F	E
G	B
H	B
I	B
J	B
K	B
L	B

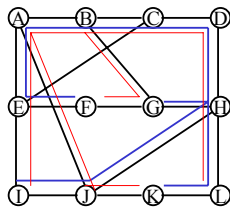
את אוסף המסלולים צריך לבחור כך שלא תוצרנה לולאות, לדוגמא A מנתב ל-F דרך E בעוד ש-E מנתב ל-F דרך A <=> לולאת נתוב.

נתוב סטטי – בלתי מפוצל



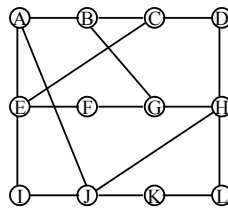
- בכל צמת יש טבלאות נתוב.
- הטבלא מכילה עבור כל יעד את הצמת הבא בנתיב.
- הטבלא שונה בכל צמת
- מלוי הטבלא:
 - בוחרים אוסף מסלולים
 - עבור כל צמת ועבור כל יעד ממלאים את השכנים.
- את אוסף המסלולים צריך לבחור כך שלא תוצרנה לולאות.

נתוב סטטי – מפוצל – טבלא לצמת J



יעד	שכנו			
	שכנו	שכנו		
A	H	0.1	A	0.9
B	H	0.2	A	0.8
C	H	0.5	A	0.5
D	H	0.8	A	0.2
E	H	0.3	A	0.7
F	H	0.2	A	0.8
G	H	0.8	A	0.2
H	H	0.9	A	0.1
I	I	0.9	A	0.1
J	--	--	--	--
K	H	0.1	K	0.9
L	H	0.8	A	0.2

נתוב סטטי - מפוצל



- בטבלת הנתוב, לכל יעד מספר שכנים אפשריים (אחד לכל נתיב).
- השמוש בשכנו לפי התדירות הרשומה בטבלא.
- ההחלטה נעשית באופן סטטיסטי ע"י שמוש במספר אקראי.
- תתכנה לולאות.

אלגוריתם למציאת עץ פורש

- הוכחת נכונות:
- האם קיימת קשת (i,j) בצעד 3?
- מדוע A' יוצר עץ (הוכחה באנדוקציה)?
- מדוע העץ שנוצר הוא פורש?

שמוש עץ פורש ב-LAN.

- נתון: הגרף $G=(N,A)$ (קשיר)
1. בחר צמת אקראי v , $N'=\{v\}$, $A'=\{ \}$
 2. אם $N'=N$ עצור.
 3. בחר קשת (i,j) כך ש $j \notin N'$, $i \in N'$ עדכן: $A'=A' \cup \{(i,j)\}$
 $N'=N' \cup \{j\}$
 4. חזור לצעד 2

עם סיום האלגוריתם A' הוא העץ הפורש

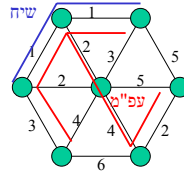
גרפים - הגדרות

- גרף $G=(N,A)$ או $G=(V,E)$
- קשת: a_i או (n_i, n_j)
- מסלול כללי (walk)
- מסלול פשוט (path): מסלול חסר מעגלים
- מעגל (cycle) מסלול המתחיל ומסתיים באותו צמת.
- גרף קשיר: קיים מסלול בין כל זוג צמתים
- תת גרף (subgraph)
- עץ: גרף קשיר חסר מעגלים
- עץ פורש של גרף (spanning tree): עץ המכיל את כל צמתי הגרף.

עפ"מ (המשך)

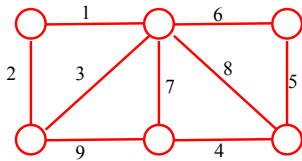
- טענה:** בהנתן שיח F , תהא $\alpha=(i,j)$ הקשת היוצאת מ- F בעלת המשקל המינימלי $(i \in F, j \notin F)$. ניתן להרחבה ע"י הוספת צמת j וקשת α (כלומר, F עם j ו- α הוא שיח גם כן).
- הוכחה:** יהא M עפ"מ שעבורו F הוא תת-עץ. אם $\alpha \in M$ סיימנו.
נניח $\alpha \notin M$ ונראה כי קיים עפ"מ אחר שעבורו הטענה נכונה. מההנחה $M \cup \alpha$ יוצר מעגל.
 $j \notin F$, לכן קיימת קשת $\beta \neq \alpha$ השייכת למעגל היוצאת מ- F . נוריד את β ונוסיף את α . מתקבל גרף M' שהוא קשיר ובעל $N-1$ קשתות (וללא מעגלים) ולכן M' הוא עץ פורש. מאחר ש- $w_\alpha \leq w_\beta$, M' הוא עפ"מ הצכיל את F ואת α ולכן $F \cup \{\alpha\}$ הוא שיח.

עץ פורש מינימלי (עפ"מ) Minimum Spanning Tree (MST)



- אם לקשתות בגרף משקל w_{ij} – לעצים פורשים שונים משקל שונה. רוצים לבחור עץ בעל תכונות אופטימליות כלומר משקל מינימלי.
- הגדרות:
 - שיח (bush): תת גרף קשיר של עפ"מ.
 - קשת יוצאת: קשת בעלת קצה אחד בשיח וקצה שני מחוצה לו.

דוגמא לעפ"מ



אלגוריתמים לבנית עפ"מ

מאחר וכל צמת עצמו הוא שיח ניתן לבנות ע"ס הטענה שהוכחנו אלגוריתמים לעפ"מ.

Prim – Dijkstra

- בחר צמת כלשהו (arbitrarily) לשיח התחלתי.
- הרחב את השיח בעזרת הטענה שהוכחנו.

Kruskal

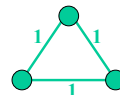
- התחל עם N שיחים כל אחד בן צמת בודד.
- בחר קשת בעלת משקל מינימלי וחבר יחד שני שיחים

איזה אלגוריתם קל יותר לבור?

נתוב לאורך מסלולים קצרים

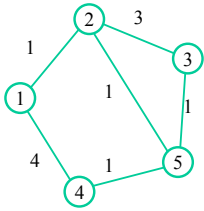
- נתוב לאורך עץ פורש אינו תמיד הגיוני:
 - אין שמוש בכל קשתות הרשת.
 - קיימים מסלולים קצרים יותר.
- נחשב את המסלול הקצר ביותר בין כל מקור לכל היעדים.
 - מסלולים אלה נכנים לטבלאות הנתוב הסטטי.
- הגדרת הבעיה:** נתון גרף עם משקלות על הקשתות. עבור כל זוג מקור-יעד מצא את המסלול הקצר ביותר.
- הערות:**
 - אורך מסלול = סכום אורכי הקשתות.
 - כל המשקלות חיוביים (מעגלים חיוביים).
 - הגרף יכול להיות מכונן.
 - משקלות חיוביים \Leftarrow המסלול הקצר חסר לולאות.

יחידות עפ"מ



- באופן כללי עפ"מ אינו יחיד
- טענה:** אם כל משקלות הגרף שונים אזי קיים לו עפ"מ יחיד.
- הוכחה:** בדומה להוכחת הטענה הקודמת ובהתבסס על חד משמעיות בחירת α .
- כיצד ניתן להבטיח יחידות כאשר המשקלים לא שונים? להשתמש בסדר לקסיקוגרפי (w_{ij}, i)

דוגמא לפעולת אלג' Bellman-Ford



צמת	אתחול	שלב 1	שלב 2	שלב 3
1	0	0	0	0
2	∞	1	1	1
3	∞	∞	4	3
4	∞	4	4	3
5	∞	∞	2	2

$$D_i^{h+1} = \min_j (d_{ij} + D_j^h)$$

אלגוריתם Bellman-Ford

- **הרעיון:** עבור יעד מסוים, לסווג את המסלולים לפי אורכם בקשתות. המסלול האופטימלי באורך h מחושב ע"ס המסלולים האופטימלים באורך $h-1$. (תכנות דינמי)
- **סימונים:**
 - נסמן את צמת היעד ב-1.
 - $d_{ij} > 0$ אורך הקשת (i,j) . $d_{ij} = \infty$ אם הקשת לא קיימת. $d_{ii} = 0$.
 - D_i^h אורך המסלול בין צמת i ליעד המכיל לכל היותר h קשתות.
- **האלגוריתם:**
 - אתחול: $\forall i \neq 1, D_i^0 = \infty, D_1^0 = 0$.
 - עבור כל $h \geq 0$: $D_i^{h+1} = \min_j (d_{ij} + D_j^h)$.
 - סיום: $\forall i, D_i^{h+1} = D_i^h$.

אלג' Bellman-Ford: הוכחת נכונות (המשך)

נניח נכונות הטענה עבור כל $k \leq h$ ונוכיח לגבי $k = h+1$.
מתוך ההגדרה:

$D_i^k \leq D_i^{k-1} \forall k \leq h$ אוסף המסלולים בני k קשתות מכיל את אלה בני $k-1$.
המסלול הקצר בעל $h+1$ קשתות לכל היותר בין i ליעד:

- פחות מ- $h+1$ קשתות \Leftarrow ארכו D_i^h עפ"י הנחת האינדוקציה.
- בן $h+1$ קשתות \Leftarrow מורכב מקשת כלשהי לצמת j ומשם מסלול אל היעד. אורך המסלול לכן:

$$X_i^{h+1} = \min(D_i^h, \min_j (d_{ij} + D_j^h))$$

$$D_i^{h+1} = \min_j (d_{ij} + D_j^h) \leq \min_j (d_{ij} + D_j^{h-1}) = D_i^h$$

$$X_i^{h+1} = \min(D_i^h, D_j^{h+1}) = D_j^{h+1}$$

2.

משקלות חיוביים \Leftarrow אין לולאות \Leftarrow המסלול הארוך ביותר הוא בן $n-1$ קשתות.

אלג' Bellman-Ford: הוכחת נכונות

טענה:

1. המספר D_i^h הוא אורך המסלול הקצר ביותר בן h קשתות לכל היותר בין צמת i ליעד.
2. האלג' מסתיים לאחר $h \leq n$ צעדים ואז D_i^h הוא אורך המסלול הקצר ביותר בין צמת i ליעד.

הוכחה: בעזרת אינדוקציה על h .

$$D_i^1 = \min_j (d_{ij} + D_j^0) = d_{i1} \quad h=0$$

ואכן d_{i1} הוא המסלול הקצר בן קשת אחת.

תכנות דינמי

מתי נשתמש בתכנות דינמי? לפתרון בעיות אופטימיזציה.

1. כאשר הפתרון האופטימלי מכיל בתוכו פתרונות אופטימליים תתי-בעיה.
2. כאשר יש חפיפה בין תתי-בעיה של חלקים שונים בפתרון. זה לא מתקיים ברוקורסיה רגילה.

אלג' B-F: סבוכיות חשוב ומציאת המסלול

- עבור כל h : מציאת המינימום הינה בן n גורמים ומבוצעת לכל צמת כלומר n פעמים. במקרה הגרוע $n = h$ ולכן נקבל $O(n^3)$.
 - חשוב יותר מדויק יתן סבוכיות של $O(n \cdot m)$.
 - מציאת המסלול הקצר מידיעת אורכו:
- לאחר שהאלג' הסתיים נרשום את משוואת Bellman:
- $$D_i = 0, \quad D_j = \min_j (d_{ij} + D_j)$$
- מצמת i נבחר את צמת k המקיים את משוואת Bellman (קיים לפחות צמת אחד כזה) ונוסיף את הקשת (i,k) למסלול. נחזור על הפעולה לצמת k וכולי עד ליעד.
- אסף המסלולים הקצרים לצמת מסויים יוצר עץ.

Dijkstra's Algorithm

1 Initialization:

```

2 N = {A}
3 for all nodes v
4   if v adjacent to A
5     then D(v) = c(A,v)
6   else D(v) = ∞
7

```

8 Loop

```

9 find w not in N such that D(w) is a minimum
10 add w to N
11 update D(v) for all v adjacent to w and not in N:
12   D(v) = min( D(v), D(w) + c(w,v) )
13 /* new cost to v is either old cost to v or known
14 shortest path cost to w plus cost from w to v */
15 until all nodes in N

```

Notation:

$c(i,j)$: link cost from node i to j .

$D(v)$: current value of cost of path from source to dest.

$p(v)$: predecessor node along path from source to v .

N : set of nodes whose least cost path definitively known

האלגוריתם של Dijkstra

שיטה למציאת מסלולים קצרים בגרף בעל משקולות חיוביים.

הרעיון: לגלות את הצמתים לפי סדר מרחקם מהיעד: קודם הצמת הקרוב ביותר, אח"כ השני הקרוב ביותר, וכו'.

הצמת הקרוב ביותר ליעד הוא השכן בקצה הקשת הקצרה ביותר היוצאת מהיעד.

מי יבחר שני:

- שכן של היעד בקצה הקשת היוצאת השנייה הקצרה ביותר, או
 - שתי קפיצות שהראשונה בהן היא הקשת שבחרנו.
- השיטה:** נצייין כל צמת בתווית זמנית המשערת את מרחקו מצמת היעד. כאשר השערוך מאומת, התווית הופכת להיות קבועה.

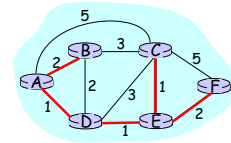
סבוכיות אלגוריתם Dijkstra

האלגוריתם מורכב מ- n איטרציות. בכל איטרציה יש לבחון את כל הצמתים שעדיין לא סומנו שה"כ $O(n^2)$.

ניתן להשתמש במבני נתונים יעילים יותר ולקבל סיבוכיות $O(m+n \log n)$.

Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
→ 0	A	2,A	5,A	1,A	infinity	infinity
→ 1	AD	2,A	4,D		2,D	infinity
→ 2	ADE	2,A	3,E			4,E
→ 3	ADEB		3,E			4,E
→ 4	ADEBC					4,E
5	ADEBCF					



PI אלגוריתם

• סימון:

- כל צמת ממספר את הקשתות מקומית.
- MSG(l) הודעה שהתקבלה על קשת l .

Algorithm PI:

```

m ← 0 // initialization
Upon START or reception of MSG(l)
  if m=0 then
    m ← 1
  send MSG to all neighbors

```

• הערות:

- האלגוריתם מבוזר. כל צמת פועל בנפרד ובלוח זמנים משלו.
- הדגל m דואג לכך שצמת לא יעביר הודעה יותר מפעם אחת.

נתוב בעזרת הצפה

- הרעיון:** צמת המקור רוצה להעביר מידע לכל שאר צמתי הרשת.
 - לא ידוע עץ פורש (אחרת זה פתרון סביר).
- הנחה:** כל צמת מכיר רק את שכניו.
- הכצוה:** כל צמת המקבל את ההודעה מעביר אותה לכל שכניו.
- יתרונות:**
 - אין צורך לדעת/ללמוד את מבנה הרשת.
 - כל צמת קשיר יקבל את ההודעה בזמן המהיר ביותר.

תכונות אלגוריתם PI

1. במשך פעולת האלגוריתם תעבור בדיוק הודעה אחת על כל קשת בכל כוון.
2. כל צמת הקשור למקור יבצע $m \leftarrow -1$ תוך זמן סופי.
3. יהא p_i הצמת הראשון ממנו קבל צמת i את ההודעה. אוסף הקשתות (p_i, i) יוצר יער פורש.
 - אם יש מקור אחד – יוצר עץ פורש.
 - אם לקשתות יש משקלות, העץ הפורש הוא עץ המסלולים הקצרים ביותר מהמקור.
4. צמת המקור אינו יודע מתי האלגוריתם הסתיים (כלומר מתי כל הצמתים קבלו את ההודעה).

הצפה עם אינדיקצית סיום (אלג' PIF)

- **הרעיון:** נשתמש בעץ הפורש שנוצר ע"י ההצפה להעברת מידע הסיום. אינדיקצית הסיום הינה עקיפה ולא מפורשת.
- **השיטה:** צמת שמקבל מידע מעביר אותו לכל שכניו מלבד זה שממנו התקבלה (ההורה בעץ הפורש). צמת יעביר הודעה להורה לאחר שקבל הודעות מכל שכניו.
- **מבנה הנתונים בצמתים:**
 - m – דגל השתתפות בפרוטוקול.
 - $N(i)$ – דגלים המסמנים האם התקבלה הודעה מהצמת בקצה קשת i .
 - p – השכן ממנו קבל הצמת לראשונה את ההודעה (ההורה).
- סימונים:
 - משתנה השייך לצמת i יסומן: $m_i, N_i(i), p_i$.
 - ריצת הפרוטוקול הקשורה לאתחול בצמת s : $m^s, N^s(i), m^s$.

פרוטוקול PIF

algorithm for any node

Init: $\forall i N(i) \leftarrow -0; m \leftarrow -0; p \leftarrow -0$

Upon receipt of $MSG^s(i)$

```

N(i) ← -1
if m = 0 then
    p ← -1
    send  $MSG^s$  to all  $l \in N - \{i\}$ 
    m ← -1
if  $\forall l'$  holds  $N(l') = 1$  then
    send  $MSG^s$  to p
    m ← -0
     $\forall l' N(l') \leftarrow -0$ 
    
```

algorithm for source node s

Init: $\forall i N(i) \leftarrow -0; m \leftarrow -0; p \leftarrow -0$

```

Upon START
if m = 0 then
    send  $MSG^s$  to all  $l \in N$ 
    m ← -1
    
```

Upon receipt of $MSG^s(i)$

```

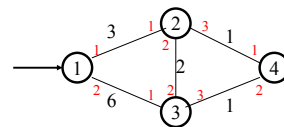
N(i) ← -1

if  $\forall l'$  holds  $N(l') = 1$  then
    m ← -0
     $\forall l' N(l') \leftarrow -0$ 
    
```

תכונות PIF

1. כל הצמתים הקשורים יקבלו את ההודעה.
2. כל הודעה תעבור על כל קשת בכל כוון בדיוק פעם אחת.
3. אוסף הקשתות (i, p_i) יוצר עץ פורש.
4. כל הצמתים יבצעו $m \leftarrow -0$ ויסיימו את הפרוטוקול בזמן סופי.
5. כל צמת יסיים את הפרוטוקול לפני ההורה שלו.
6. צמת המקור יסיים את הפרוטוקול $(m \leftarrow -0)$ אחרון. זוהי אינדיקצית הסיום.

דוגמא לפעולת אלג' PIF



שימוש ב- PIF לבדיקת קשירות הרשת

- כל צמת מתחיל PIF משלו כאשר
 1. מקבל הודעת START, או
 2. מקבל הודעת PIF של צמת אחר (תחילה משתתף ב-PIF שקבל ורק אחר"כ מתחיל את שלו)
- כל צמת שומר משתנים נפרדים לכל פרוטוקולי ה-PIF בהם הוא משתתף.
- תכונות:
 - ניתן לאתחול אסינכרוני
 - כל צמת ילמד על כל השכנים הקשורים אליו.
 - כאשר צמת מסיים את ה-PIF שלו מובטח לו ששמע מכל הצמתים הקשורים אליו. למה?

אלגוריתם Bellman-Ford מבוזר

- הנחות:
 - כל צמת יודע את אורכי הקשתות היוצאות ממנו
 - הרשת נשארת קשירה. נטפל בנפילת קשתות בנפרד.
 - האלגוריתם מבצע את החשוב עבור המרחק לצמת 1.
- תזכורת:

משוואות Bellman Ford:

$$D_i^{h+1} = \min_j (d_{ij} + D_j^h) : i \neq 1$$

עבור $D_i^0 = 0, \forall i \neq 1 \quad D_i^0 = \infty$ תנאי התחלה: $D_1^0 = 0$

לאחר h צעדים המשוואות מתכנסות ל:

$$D_i = \min_{j \in N(i)} (d_{ij} + D_j) : i \neq 1 \quad D_1 = 0$$

עבור $D_1 = 0$ וכמובן:

חשוב מבוזר של מסלולים קצרים ביותר

- מוטיבציה:
 - טופולוגית הרשת ומשקלי הקשתות משתנים לכן יש לחשב את טבלאות הנתוב מחדש מדי פעם.
 - חשוב מבוזר הוא נח מכיון שאין צורך להעביר את המידע הטופולוגי לכל הרשת.
- מטרה: חשוב מחדש של המסלולים הקצרים ביותר לכל יעד.
- הבחנה: צמת לא חייב לדעת את כל המסלול אל היעד. מספיק לדעת את השכן הראשון בדרך ליעד. בד"כ גם לא דרוש אורך המסלול.
- בסיס: צמת יודע רק את הנעשה בסביבה המיידית שלו וצריך ללמד את השאר.

אלגוריתם Bellman-Ford מבוזר אסינכרוני

משוואות Bellman Ford יותר כלליות מכפי שהצגנו.
הרעיון:

- כל צמת מבצע את משוואת B מדי פעם
- מדי פעם מעביר כל צמת את D_i שלו לכל שכניו

הנחות:

- צמת לא יעכב לעד את בצוע העדכון של D_i .
- צמת לא יעכב לעד הודעה לשכניו על השנויים.
- השהיית ההודעות ברשת סופית.

אלגוריתם Bellman-Ford מבוזר סינכרוני

- כל צמת i מחשב בעצמו את D_i^{h+1} מתוך ערכי D_j^h שקבל משכניו וידיעת אורכי הקשתות היוצאות ממנו d_{ij}
 - כאשר כל הצמתים מסיימים החשוב עבור h מסוים, מעביר כל צמת את הערך שחשב לשכניו.
- חסרונות:

- דרוש סנכרון כדי לבצע את האלגוריתם.
- דרוש מנגנון לעצירה מוקדמת של האלגוריתם במקרה של שנוי באורך קשת במשך פעולת האלגוריתם
- התחלה מחדש דורשת (לכאורה) תנאי התחלה חדשים \Leftarrow נצול לא יעיל של ידע קודם.

בעית הספירה לאינסוף

A	B	C	D	E
∞	∞	∞	∞	∞
1	∞	∞	∞	∞
1	2	∞	∞	∞
1	2	3	∞	∞
1	2	3	4	∞
3	2	3	4	∞
3	4	3	3	∞
5	4	5	4	∞
5	6	5	6	∞
7	6	7	6	∞
:	:	:	:	∞
∞	∞	∞	∞	∞

- מעונינים במרחק ל-A.
- ברשת שבדוגמא קשת AB לא פעלה זמן רב והחלה לפעול.
- לאחר זמן מה נופלת קשת AB:
- מספר המחזורים בלתי מוגבל.
- \Leftarrow יש להגביל את מושג האינסוף.
- ניתן להתגבר על הבעיה בדרכים שונות:
- Time-out (Age).
- מספרים סדוריים להודעות.
- אלגוריתמים יותר חכמים.

אלגוריתם B-F מבוזר אסינכרוני (המשך)

האלגוריתם:

כל צמת i מאחסן:

- D_i - הערכת המרחק לצמת 1.
 - D_j - הערכת המרחק של צמת j שכן j אל צמת 1 כפי שדווח לצמת i .
- בצוע:** בכל פעם שיש שנוי ב- D_j או ב- d_{ij} בצע את משוואות B-F ועדכן את שכניך.

משפט: בהנתן מצב התחלתי כלשהו ובהפסק השנויים ב- d_{ij} קיים זמן שאחריו הגודל D_i בכל צמת הוא המרחק הקצר לצמת i .

הערה:

- מתוך המרחקים הקצרים ניתן למצוא את המסלול עצמו.
- יש לבצע את באלגוריתם בנפרד לכל יעד, אך ניתן להעביר את המרחקים לכל היעדים בהודעה אחת.

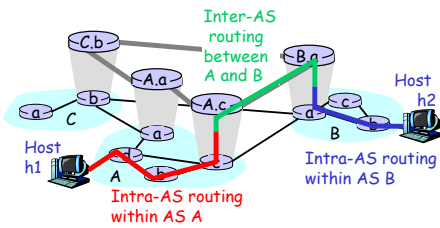
RIP

- פרוטוקול distance vector, כלומר מבוסס על אלגוריתם Bellman Ford.
- פותח בשנות השמונים ע"ב פרוטוקול של Xerox.
- בשמוש נפוץ בשל פשטותו. כיום RIP.
- המטריקה: minimum hop.
- ב- Arpanet נעשה שמוש בעייתי במטריקה הרגילה לעמס בקוים.

נתוב באינטרנט

- הנתוב באינטרנט נעשה בשלוש רמות:
 - ברשתות מקומיות ברמת ה-MAC: עץ פורש לאינטרנט, נתוב מקור לרשתות טבעת
 - ברשתות אוטונומיות: RIP, OSPF, IS-IS, (E)IGRP
 - בין רשתות: BGP

נתוב היררכי באינטרנט



OSPF

- פרוטוקול link state – כל צמת רואה מפה טופולוגית של הרשת ומנתב בעזרתה (Dijkstra).
- מאפשר שתי רמות היררכיה
- בטיחות (auth.)
- מורכב