מערכות מבוזרות

הגדרות כלליות:

מודלים:

1. **שליחת הודעות:** כל מחשב או שרת הוא קודקוד בגרף כאשר לכל קודקוד יש מצב פנימי (קונפיגורציה שלו) והתקשורת היא על ידי שליחת הודעות בין מחשבים. כל ערוץ שליחת הודעות מיוצג על ידי צלע בגרף. (לרוב כולם יכולים לשלוח לכולם – כלומר גרף שלם).
2. **זיכרון משותף:** התקשורת בין המחשבים היא על ידי כתיבה וקריאה מתאי זיכרון המשותפים לכולם.

בכל מודל נדון במצב סינכרוני ואסינכרוני:

- **סינכרוני:** ביצוע האלגוריתם מחולק לסיבובים כאשר בכל סיבוב כל מחשב מבצע פעולות פנימיות (יכול להיות מספר פעולות) ושולח הודעה אחת לכל אחד מהאחרים (במודל 1) כותב\קורא תא זיכרון אחד בלבד (מודל 2). לאחר מכן, כל ההודעות מתקבלות (מגיעות ליעד) ורק אז עוברים לסיבוב הבא.

- **אסינכרוני:** הכל קורה במקביל. אין סיבובים.
  - <u>במודל שליחת הודעות –</u> זמן הגעת ההודעות לא ידוע. בזמן הזה המחשבים יכולים לבצע פעולות ואף לשלוח עוד הודעות.
    במקרה כזה מחלקים ל-2 ערוצי תקשורת:
    - FIFO – לא ייתכן שההודעה מאוחרת מאותו מחשב תגיע לפני הודעה מוקדמת.
    - כללי – הכל ייתכן.
  - <u>במודל זיכרון משותף –</u> הכתיבה\קריאה של כולם מתבצעת במקביל ולכן ייתכן דריסת נתונים.

תקלות (האלגוריתם לא מתבצע בשלמותו):

בכל מודל יתכנו מספר סוגי תקלות:

1. **קריסה –** אחד המחשבים מפסיק לתפקד באמצע האלגוריתם.
   - <u>במודל שליחת הודעות</u> – זה מתבטא בכך שהוא לא שולח יותר הודעות.
   - <u>במודל זיכרון משותף</u> – זה מתבטא בכך שהוא לא כותב שום דבר לזיכרון המשותף.

2. **תקלה זמנית –** אחד המחשבים לא מתפקד באופן זמני.
   - <u>במודל שליחת הודעות</u> – זה יכול להתבטא בכך שחלק מההודעות לא מגיעות ליעדן. (הולכות לאיבוד).

- <u>במודל זיכרון משותף</u> – זה יכול להתבטא בכך שבחלק מהזמן המחשב לא כותב כלום לזיכרון המשותף.

3. **<u>ביזנטי –</u>** אחד המחשבים מתנהג רנדומלית (מבחינתנו – "נגד" האלגוריתם).
- <u>במודל שליחת הודעות</u> – שולח הודעות עם תוכן אקראי (לאו דווקא התוכן שאמור להיות לפי האלגוריתם).
- <u>במודל זיכרון משותף</u>– כותב בתאי הזיכרון תוכן אקראי.

<u>תכונות של פרוטוקולים (אלגוריתמים)</u>

- **<u>בטיחות (safety) –</u>** לא קורה משהו רע.
- **<u>תגובה\ חיות (liveness) –</u>** משהו טוב קורה מתישהו למישהו.
- **<u>הגינות (fairness) –</u>** משהו טוב קורה מתישהו לכולם.

<u>מודל שליחת ההודעות</u>

<u>מושגים:</u>

- **<u>קונפיגורציה :</u>** המצב הכולל של כל המערכת. כלומר: רשימה של המצבים של אחד מהמחשבים במערכת.
- **<u>מצב:</u>** רשימת המשתנים וערכיהם, באיזה שלב\שורה של האלגוריתם המחשב נמצא בביצוע וכו.

***קונפיגורציה ומצב לא מכילים את התקשורת בין המחשבים: סטטוס ההודעות.*
- **<u>הרצה:</u>** סדרת קונפיגורציות עם האירועים ביניהם, הקלט של כל מחשב נתון עם ההרצה. כאשר:
  - אירוע = שליחה או קבלה של הודעה. דוגמא: $C_0, s_{12}, C_1, r_{21}, C_2, ...$
  - קלט = הזנה התחלתית של כל אחד מהמחשבים באופן מקומי, אין לו קשר להודעות כי הוא נקבע בתחילת הריצה (כמובן שאפשר לשלוח את הקלט בהודעות)

<u>סימון:</u> s_{ij} - מחשב i שולח הודעה s למחשב j.

<u>סימון:</u> r_{ij} - מחשב i מקבל את ההודעה r ששלח מחשב j.

- **<u>לוח זמנים:</u>** הרצה ללא הקונפיגורציות אלה רק האירועים. (גם כאן הקלט כלול).
- **<u>צמצום לוח זמנים לקודקוד v:</u>** לוח זמנים המוצגים בו רק האירועים שהתרחשו במחשב v.

דוגמא: $S = s_{12}, s_{13}, r_{31}, s_{31}, s_{32}, r_{23}, r_{21}, r_{13}$

אז:

$S|1 = s_{12}, s_{13}, r_{13}$.

$S|2 = r_{23}, r_{21}$.

$S|3 = r_{31}, s_{31}, s_{32}$.

<u>שימו לב –</u> יש חשיבות לסדר האירועים בהרצה, בלוח הזמנים ובצמצום.

<u>הערה:</u>

- **מחשב דטרמיניסטי –** זה מחשב הפועל בצורה אחידה ביחס לאותם אירועים. לדוגמה – אם הוא קיבל את אותם קלטים ואותם הודעות אז הוא מתנהג אותו דבר בדיוק.
- **מחשב אי-דטרמיניסטי –** מחשב שעובד עם אקראיות או\ו יכול להתנהג באופן שונה גם אם קיבל קלטים זהים והודעות זהות.

משפט: אם S, S' הם 2 לוחות זמנים עם אותו קלט כך ש: $S|1 = S'|1$ אז מחשב 1 מבצע את אותם פעולות בדיוק ב 2 ההרצות (בהנחה שהוא דטרמיניסטי).

בעיית 2 הגנרלים

הבעיה הכללית: נתונים 2 מחשבים ברשת, כל אחד מהם מקבל קלט 0 או 1 והמטרה היא ששניהם בסוף יוציאו את אותו הפלט - בעיית הסכמה (קונצנזוס).

תכונות של בעיות הסכמה:

1. **הסכמה** = כל המחשבים תמיד יוציאו את אותו הפלט.
2. **תקפות** = אם הקלט של שניהם זהה ולא אבדו הודעות אז הפלט חייב להיות זהה לקלט.
   לדוגמא: אם גם מחשב 1 וגם מחשב 2 קיבלו את הקלט 0 אז שניהם חייבים להוציא את הפלט 0 ולא.
3. **סיום** = האלגוריתם חייב לעצור לאחר זמן סופי.

המודל:

- 2 מחשבים (דטרמיניסטיים).
- מודל שליחת הודעות (מודל 1).
- סינכרוני (מחולק לסיבובים).
- תקלות: הודעות יכולות ללכת לאיבוד ולא להגיע.

השאלה: האם קיים אלגוריתם המקיים את 3 התכונות (הסכמה, תקפות, סיום) לעיל במודל הנ"ל.
תשובה: לא קיים אלגוריתם כזה.

הוכחה: ע"י סדרת הרצות דומות.
נניח בשלילה שקיים אלגוריתם המקיים את 3 התכונות לעיל. ולכן הוא מבצע לכל היותר T סיבובים (כי הוא מקיים את תכונת הסיום).
נגדיר את ההרצות הבאות:

- E_1: קלט - (0,0). לא אבדו הודעות. מכיוון שהאלגוריתם מקיים תקפות אז הפלט בהכרח $(0,0)$.

- E_2: קלט - (0,0). בסיבוב T (האחרון) ההודעה מ v_1 ל v_2 הלכה לאיבוד. כל שאר ההודעות הגיעו.
  נשים לב שההרצה E_2 דומה להרצה E_1 עבור v_1, כלומר: $E_1 \sim_{v_1} E_2$. ולכן הפלט של v_1 יהיה 0 ומכיוון שהאלגוריתם מקיים הסכמה אז גם הפלט של v_2 יהיה 0, כלומר הפלט הוא: $(0,0)$.

- E_3: קלט - (0,0). כמו E_2 פרט לכך שבסיבוב T (האחרון) גם ההודעה מ v_2 ל v_1 הלכה לאיבוד. כל שאר ההודעות הגיעו.
  נשים לב שההרצה E_3 דומה להרצה E_2 עבור v_2, כלומר: $E_2 \sim_{v_2} E_3$. ולכן הפלט של v_2 יהיה 0 ומכיוון שהאלגוריתם מקיים הסכמה אז גם הפלט של v_1 יהיה 0, כלומר הפלט הוא: $(0,0)$.

- E_4 : קלט - (0,0). כמו E_3 פרט לכך שבסיבוב T-1 ההודעה מ v_1 ל v_2 הלכה לאיבוד. כל שאר ההודעות הגיעו. באותו אופן כמו מקודם: $E_3 \sim {}_{v_1}E_4$.

…

- E_{2T+1} : קלט - (0,0). כל ההודעות אבדו. $E_{2T} \sim {}_{v_2}E_{2T+1}$. ולכן כמו מקודם הפלט יהיה $(0,0)$.
- E_{2T+2} : קלט - (0,1). כל ההודעות אבדו. $E_{2T+1} \sim {}_{v_1}E_{2T+2}$. ולכן כמו מקודם הפלט יהיה $(0,0)$.
- E_{2T+3} : קלט - (1,1). כל ההודעות אבדו. $E_{2T+2} \sim {}_{v_2}E_{2T+3}$. ולכן כמו מקודם הפלט יהיה $(0,0)$.
- E_{2T+4} : קלט - (1,1). רק ההודעה בסיבוב מספר 1 מ v_1 ל v_2 הגיעה. $E_{2T+3} \sim {}_{v_1}E_{2T+4}$. ולכן כמו מקודם הפלט יהיה $(0,0)$.

…

- E_{4T+3} : קלט - (1,1). לא אבדו הודעות. $E_{4T+2} \sim {}_{v_2}E_{4T+3}$. ולכן כמו מקודם הפלט יהיה $(0,0)$.

וזו סתירה לתקפות כיוון שלפי התקפות הפלט צריך להיות זהה לקלט שהוא $(1,1)$.

בעיית 2 הגנרלים במודל אי-דטרמיניסטי

הבעיה זהה לבעייה הקודמת עם אותו מודל פרט לכך שהמחשבים הם אי דטרמיניסטיים.
במקרה כזה הבעיה פתירה בהסתברות קטנה לטעות.

אלגוריתם הרמות - עבור 2 מחשבים

לכל קודקוד v יש משתנה L_v המייצג את הרמה שלו.
אתחול: $L_v = 0$ לכל v .
עבור i מ 1 עד T :

- שלח הודעה עם L_v למחשב השני.
- אם קיבלת הודעה ממחשב u עם הרמה L_u אז עדכן: $L_v = max(L_v, L_u + 1)$.

2 המחשבים מעלים את הרמה אחד לשני.

מסקנות:

- אם כל ההודעות הלכו לאיבוד הרמה של כל אחד תהיה 0.
- אם כל ההודעות הגיעו הרמה של כל אחד מהם תהיה T.
- ובכל מקרה, ההפרש בין הרמות של 2 המחשבים לא יעלה על 1. $|lvl_u - lvl_v| \le 1$.

דגש: באמצעות אלגוריתם זה, ניתן ליצור אלגוריתם לבעיית 2 הגנרלים.

אלגוריתם לבעיית 2 הגנרלים אסינכרוניים

אלגוריתם ל v : קלט x

- הגרל מספר t בין 1 ל r
- בצע למשך r סיבובים את אלגוריתם הרמות כך שבמקום לשלוח רק את הרמה, צרף להודעה גם את x וגם את t . (כלומר סה"כ v שולח L_v, x, t)

אלגוריתם ל u : קלט x

- בצע למשך r סיבובים את אלגוריתם הרמות כך שבמקום לשלוח רק את הרמה, צרף להודעה גם את x . (כלומר סה"כ u שולח L_u, x)

כל אחד מהמחשבים פולט 1 אם ורק אם מתקיימים כל התנאים הבאים:

1. המחשב יודע את t ואת 2 הקלטים.

2. 2 הקלטים הם 1.
3. הרמה שלי היא לפחות t.

בכל מקרה אחר - המחשב יפלוט 0.

Validity: מתקיימת כי אם 2 הקלטים הם 0 וכל ההודעות הגיעו אז לפי התנאי ה 2 שלא מתקיים בשניהם - שניהם יפלטו 0.
אם 2 הקלטים הם 1 וכל ההודעות הגיעו תנאי 1 מתקיים כי ההודעות הגיעו והן מכילות את t ואת x (הקלט). תנאי 2 מתקיים כי הקלטים הם 1. תנאי 3 מתקיים כי לפי אלגוריתם הרמות, הרמה תהיה r אצל שניהם כי כל ההודעות הגיעו ומתקקים r גדול או שווה ל t. לכן הפלטים יהיו 1.

הסכמה: מתקיימת תמיד אלא אם כן:
הקלט של שניהם הוא 1, הרמה של אחד מהם היא t בדיוק והשנייה היא t-1.
במקרה כזה אחד מהם יוציא 0 והשני 1.
כל מה שהיריב (המערכת) יכול לזמן לנו - לא תלוי בהסתברות (מחשיבים אותו שיקרה בוודאות).
היריב לא יודע את t. ולכן הוא יכול לנחש את t ולגרום למצב בו נקבל רמה t לאחד ורמה t-1 לשני.
ההסתברות שהיריב ינחש את ה t הנכון היא: $1/r$.
ולכן ההסתברות להסכמה היא: $1-1/r$. (כל זה במקרה והקלטים הם 1,1 בכל מקרה אחר תמיד תהיה הסכמה על 0)

סיום: מתקיים לאחר r סיבובים.

חסם תחתון על טעות באלגוריתם רנדומלי הפותר את בעיית 2 הגנרלים.

טענה: כל אלגוריתם רנדומלי הפותר את בעיית 2 הגנרלים ומקיים:

1. מסיים לאחר r סיבובים
2. תקפות: אם 2 הקלטים זהים ולא אבדו הודעות אז 2 הפלטים זהים לקלט.
   בנוסף: אם לפחות אחד מהקלטים הוא 0, הפלט של שניהם 0, לא משנה מה קורה.
3. הסכמה בהסתברות לפחות $1-\varepsilon$. **(כלומר שההסתברות לטעות היא לכל היותר ε)**

אז: $\varepsilon \geq 1/2r$ או $2\varepsilon \geq 1/r$

דגש: אם תנאי 2 מתקיים, אסור לטעות. אחרת (בתנאי 3), מותר טעות עד ε .

רעיון ההוכחה: דומה לשיטת ההוכחה שלא קיים אלגוריתם לבעיה הרגילה. ע"י הרצות דומות.
E_0: קלט $(1,1)$. לא אבדו הודעות, פלט: $(1,1)$ לפי תקפות (2) בהסתברות 1.
(כלומר, ההסתברות לטעות ולפלט 0 היא 0)
E_1: קלט $(1,1)$. ההודעה בסיבוב r מ v_1 ל v_2 אבדה. v_1 לא מבחין ולכן: $P(v_1\ output\ \ 0)=0$ כמו בהרצה הקודמת. v_2 מבחין ולכן: $P(v_2\ output\ 0)\leq\varepsilon$ לפי ההנחה.
(לפי (3) - ההסתברות לטעות לכל היותר ε)
E_2: קלט $(1,1)$. כל ההודעות בסיבוב r אבדו. v_2 לא מבחין ולכן: $P(v_2\ output\ 0)\leq\varepsilon$ כמו בהרצה הקודמת. v_1 מבחין ולכן: $P(v_1\ output\ 0)\leq 2\varepsilon$ לפי ההנחה.
(לפי (3) - ההסתברות לטעות לכל היותר $\varepsilon+\varepsilon$)
E_3: קלט $(1,1)$. כל ההודעות בסיבוב r אבדו וגם ההודעה בסיבוב $r-1$ מ v_1 ל v_2 אבדה.
v_1 לא מבחין ולכן: $P(v_1\ output\ \ 0)\leq 2\varepsilon$ כמו בהרצה הקודמת. v_2 מבחין ולכן: $P(v_2\ output\ 0)\leq 3\varepsilon$ לפי ההנחה.

E_4: קלט $(1,1)$. כל ההודעות בסיבוב r אבדו וגם ההודעה בסיבוב $r-1$ אבדו.
v_2 לא מבחין ולכן: $P(v_2\ output\ \ 0) \le 3\varepsilon$ כמו בהרצה הקודמת. v_1 מבחין ולכן: $P(v_1\ output\ 0) \le 4\varepsilon$ לפי ההנחה.
...
E_{2r}: קלט $(1,1)$. אף הודעה לא מגיעה. v_1 לא מבחין ולכן: $P(v_1\ output\ \ 0) \le 2r\varepsilon - 1$
v_2 מבחין ולכן: $P(v_2\ output\ \ 0) \le 2r\varepsilon$.
E_{2r+1}: קלט $(0,1)$. אף הודעה לא מגיעה. v_2 לא מבחין ולכן: $P(v_2\ output\ \ 0) \le 2r\varepsilon$.
מצד שני, מתקיימת תקפות (סעיף 2) ולכן: $P(v_2\ output\ \ 0) = 1$ כי אם אחד מהקלטים הוא 0 אז שניהם חייבים לפלוט 0 בהסתברות 1 (ללא טעות).
קיבלנו: $1 \le 2r\varepsilon$ ולכן: $\varepsilon \ge 1/2r$. מש"ל.

<u>טענה חזק יותר:</u> אם הצלחנו להוכיח את הנ"ל אז ניתן להגדיר עוד הרצה $\tilde{E}$ כך שבהרצה ה r ההודעה מ v_2 ל v_1 לא הגיעה ולכן זה יהיה $E_2 \sim \tilde{E}$ אבל הסיכוי של $P(v_1 output\ 0) \le \varepsilon$ ולא 2ε ולכן ניתן להוריד את החסם ל: $\varepsilon \ge 1/r$.

<u>אלגוריתמי שליחת הודעות במודל שליחת ההודעות</u>

<u>מודל:</u> שליחת הודעות (גרף עם קודקודים שהצלעות הן ערוצי התקשורת) סינכרוני ואסינכרוני.
ללא תקלות.

<u>יעילות אלגוריתם:</u>

סיבוכיות זמן (זמן ההודעות ללא החישובים המקומיים):

- <u>סינכרוני</u>: במודל זה יש סיבובים. בכל סיבוב, ייתכן שכולם שולחים הודעה ומקבלים הודעה. (בסיום כל סיבוב, כל ההודעות שנשלחו מגיעות ליעדן)
כל סיבוב הוא יחידת זמן אחת והסיבוכיות תהיה מספר הסיבובים עד לסיום האלגוריתם.

- <u>א-סינכרוני</u>: במודל זה, זמן הגעת ההודעות לא ידוע. המודל הוא ללא תקלות ולכן כל הודעה תגיע, רק לא ידוע מתי.
במודל זה, ניקח את ההודעה שהתעכבה הכי הרבה זמן ונקבע לה שהיא יחידת זמן אחת. בהתאם אליה, נחלק זמנים מנורמלים לכל ההודעות.

סיבוכיות הודעות:
מספר ההודעות (בסדר גודל) הכולל שעבר ברשת מתחילת האלגוריתם ועד סופו.

<u>האלגוריתמים:</u>

1. **<u>אלגוריתם Flooding:</u>**

<u>מטרה:</u> קודקוד v צריך לשלוח הודעה M לכל שאר הקודקודים.
- קודקוד v: שלח את M לכל שכניך.
- שאר הקודקודים:
  - ❖ בקבלת הודעה M מקודקוד w: אם זו הפעם הראשונה שקיבלת את M,
    - ➢ שלח את M לכל שכניך פרט ל w.

<u>סיבוכיות זמן:</u>
- סינכרוני: $O(rad(G,v))$

- א-סינכרוני: $O(rad(G,v))$

סיבוכיות הודעות:

- סינכרוני: $O(|E|)$
- א-סינכרוני: $O(|E|)$

2. אלגוריתם ***Flooding spanning tree – flooding***

מיצר עץ בכך שכל קודקוד קובע את האבא להיות מי ששלח אליו את *M* בפעם הראשונה.
סינכרוני: מתקבל עץ *BFS*. כי האבא הוא זה שבמסלול הכי קצר מהשורש *v* אלי.
א-סינכרוני: מתקבל **כל** עץ פורש. כולל עץ שעומקו *n-1*.

הערה: עץ BFS הוא העץ הנמוך ביותר כך ש v הוא שורשו.

3. אלגוריתם convergeCast:

מטרה: שליחת הודעה בעץ פורש נתון בחזרה לשורש.

- עבור עלה v:
  - שלח את M לאבא.
- עבור שאר הקודקודים שאינם השורש:
  - בקבלת M, אם קיבלת את ההודעה מכל הילדים - שלח לאבא.
- עבור השורש:
  - בקבלת M, אם קיבלת הודעה מכל הילדים - סיים.

סיבוכיות זמן: עומק העץ

- סינכרוני/ואסינכרוני: אותה סיבוכיות.

סיבוכיות הודעות:

- $O(|E|) = O(|V|)$.

סינכרוני/ואסינכרוני: אותה סיבוכיות. בכל צלע עוברת הודעה אחת.

4. הפעלת flooding ואחריו convergeCast על העץ המתקבל:

סיבוכיות זמן:

- סינכרוני: $O(rad(G,v))$
- א-סינכרוני: $O(|V|)$.

סיבוכיות הודעות: $O(|E|)$.

אלגוריתמים למציאת עץ BFS במערכת א-סינכרונית:

1. אלגוריתם Dijkstra:

דגש: צריך להיות ללא משקלים על הצלעות.

מטרה: מציאת עץ BFS במערכת א-סינכרונית.

הרעיון: בניית העץ רמה אחרי רמה. עד שלא רואים בוודאות שכיסינו את כל הקודקודים ברמה i, לא עוברים לקודקודים ברמה i+1.

קודקוד v (השורש) שולח "רמה 1" לכל שכניו. כל שכן שקיבל את ההודעה, קובע שהוא ברמה 1 והאבא הוא v ושולח ACK ל v בחזרה.

כאשר v מקבל תשובה **מכל** שכניו, הוא שולח "רמה 2" לכל שכניו.

כל שכן שקיבל את ההודעה והוא ברמה 1, הוא מעביר אותה הלאה לכל שכניו.

בפעם השנייה שהוא קיבל את ההודעה, הוא מחזיר הודעת NACK לשולח.
כל שכן שקיבל את ההודעה "רמה k" והוא לא סווג ברמה, קובע את עצמו ברמה k, קובע את האבא להיות השולח ומחזיר ACK לשולח.
ברגע שקודקוד קיבל תשובה מכל שכניו, הוא מחזיר תשובה לאבא.
<u>סיבוכיות זמן:</u> $O(1+2+3+...+rad(G,v)) = O(diam(G)^2)$
<u>סיבוכיות הודעות:</u> $O(|E|+|V|diam(G))$

2. Bellman-Ford

<u>מטרה:</u> מציאת עץ BFS במערכת א-סינכרונית
<u>הרעיון:</u> לא להמתין אלא לעדכן ולעדכן עד שאין כבר מה לעדכן.
כל קודקוד u מאתחל את המרחק שלו d_u מ v להיות ∞ כאשר v מעדכן $d_v = 0$ (מרחק מעצמו).

- עבור קודקוד v: שלח את המרחק שלך + 1 לכל שכניך.
- עבור שאר הקודקודים u: בקבלת הודעה d מקודקוד w:
  - אם $d < d_u$ אז:
    - עדכן $d_u = d$
    - עדכן את האבא להיות w
    - שלח את $d_u + 1$ לכל השכנים פרט ל w.

כאשר מסתיימות כל ההודעות ברשת - האלגוריתם עוצר.
<u>סיבוכיות זמן:</u> $O(rad(G,v)) = O(diam(v))$.
<u>סיבוכיות הודעות:</u> $O(|E|\cdot|V|)$

<u>זמנים לוגיים:</u>

<u>מודל:</u> שליחת הודעות א-סינכרוני.
<u>המטרה:</u> לתת "חתימת זמן" לכל אירוע כך שהזמן ייצג את הסדר הכרונולוגי של האירועים.

<u>ערבוב סיבתי - Causal Shuffle:</u> בהינתן 2 לוחות זמנים S, S' יקראו ערבוב סיבתי אחד של השני אם $S|v = S'|v$ לכל v.
במקרה כזה לא נוכל להבחין בין לוחות הזמנים ולכן לא נוכל לקבוע מה קרה בוודאות לפני מה.

<u>הגדרה Happens Before:</u>

אירוע e קרה בוודאות לפני אירוע e' ($e \Rightarrow e'$) ב S **אם ורק אם** הוא חייב להופיע לפניו בכל ערבוב סיבתי של S.
פורמאלית: היחס "קרה לפני" מוגדר באופן הבא: $e \Rightarrow_S e'$. כלומר אירוע e קרה בוודאות לפני אירוע e' אם:

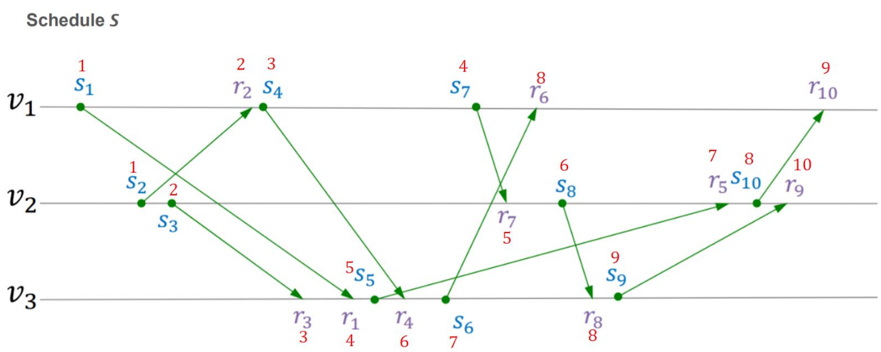
1. e אירוע שליחה, e' אירוע קבלה.
2. e, e' קרו באותו מחשב והזמן של e קטן יותר.
3. קיים אירוע e'' כך ש: $e \Rightarrow_S e''$ וגם $e \Rightarrow e'' \Rightarrow e'$.

<u>אלגוריתמים לחתימת זמן:</u>

1. <u>שעוני למפורט:</u>

אלגוריתם לכל מחשב v:
- אתחל: $c_v = 0$
- לכל אירוע e שאינו קבלה, קבע: c_v ++ וקבע: $\tau(e) = c_v$.

- לכל אירוע שליחה e , צרף להודעה את c_v .
- לכל אירוע קבלה e מקודקוד u . ההודעה מכילה את c_u . קבע: $c_v = max(c_v, c_u) + 1$ וקבע את $\tau(e) = c_v$.
-



דגש: האלגוריתם של שעוני למפורט לא מספק בדיוק את היחס של "קרה לפני".

משפט: אם $e \Rightarrow e'$ אז זה **מחייב** ש: $\tau(e) < \tau(e')$.

אם $\tau(e) < \tau(e')$ אז זה **לא מחייב** ש $e \Rightarrow e'$.

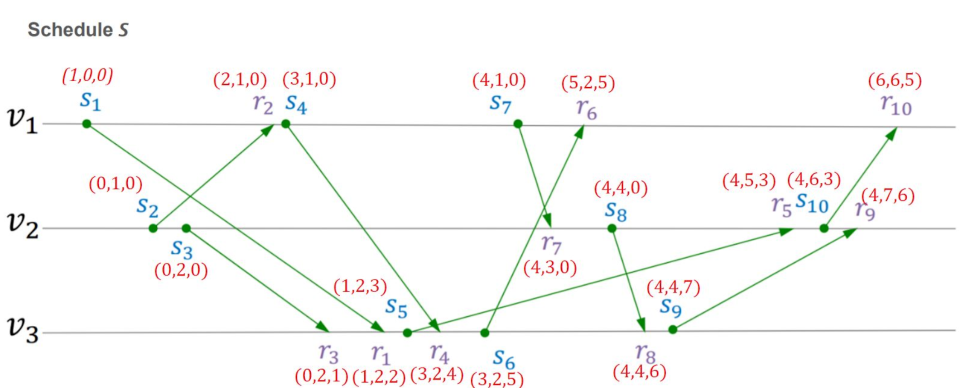
2. Vector - Clock

נועד כדי לפתור את הבעיה מלמפורט.

אלגוריתם למחשב v:

- אתחל וקטור: $VC(v) = (0, 0, ..., 0)$ לכל קודקוד. (בגודל כמות הקודקודים)
- עבור כל אירוע e שליחה, קבע: $VC_v(v)$ ++ וקבע: $VC(e) = VC(v)$.
- עבור כל אירוע שליחה, שלח בנוסף גם את הוקטור.
- עבור כל אירוע קבלה e מקודקוד u :
  - קבע: $VC_w(v) = max(VC_w(u), VC_w(v))$ לכל $w \neq v$.
    (להשוות בין שני הוקטורים ולהשוות איבר איבר ולקחת את ה max)
  - קבע: $VC_v(v)$ ++ .
  - $VC(e) = VC(u)$.

$VC_w(v)$ זה אומר וקטור – במיקום ה w שלו – על קודקוד v .



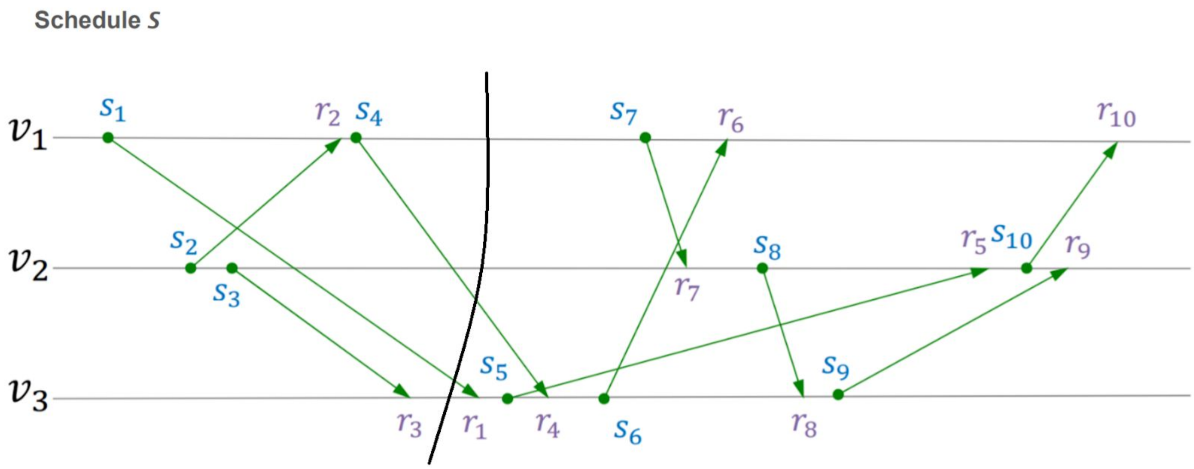
איך יודעים מי קרה לפני מי?

טענה: $e \Rightarrow_S e'$ אם ורק אם $VC_w(e) \le VC_w(e')$ לכל w וקיימת לפחות קואורדינטה אחת בה זה קטן ממש. אם יש קואורדינטות שאחד מהם גדול יותר וקואורדינטות שהשני גדול יותר אז הם קרו במקביל וניתן להחליף ביניהם.

חתכים:

בהינתן לו"ז S. חתך C ב S הוא קבוצה של אירועים מתוך S. כך שלכל 2 מאורעות שהתרחשו באותו קודקוד: e, e'. אם $e \in C$ וגם e' קרה באותו קודקוד אבל לפני e אז גם $e' \in C$.

לדוגמא:



החתך הוא: $C = \{s_1, r_2, s_4, s_2, s_3, r_3\}$.

חתך עקבי:

חתך שבו אם $e \in C$ וגם $e' \Rightarrow_S e$ אז גם $e' \in C$.

משפט: החתך הוא עקבי **אם ורק אם** לכל אירוע קבלה $r_M \in C$ מתקיים: $s_M \in C$.

הוכחה: נניח כי קיים חתך C עקבי וקיים הודעה M עם אירוע שליחה s_M ואירוע קבלה r_M כך ש $r_M \in C$.

על פי הגדרת Happens Before אנחנו יודעים כי $s_M \Rightarrow r_M$ ולכן $s_M \in C$ לפי הגדרה של חתך עקבי.

באופן כללי צריך להראות הוכחה על מספר מצבים:

- אם e ו e' על אותו קודקוד - נכון באופן טריוויאלי לכל חתך.
- אם e ו e' הם אירועי קבלה ושליחה של אותה הודעה, זה מוכח לפי הגדרה.
- אחרת, צריך לעשות אינדוקציה על המאורעות "בין" e ו e'.

החתך C לעיל הוא עקבי כי הוא חתך ולכל אירוע קבלה שיש בו גם אירוע השליחה נמצא.

לעומת זאת, $C' = \{r_3\}$ אינו חתך עקבי (למרות שהוא כן חתך).

חישוב תמונת מצב עקבית - חישוב חתך עקבי:

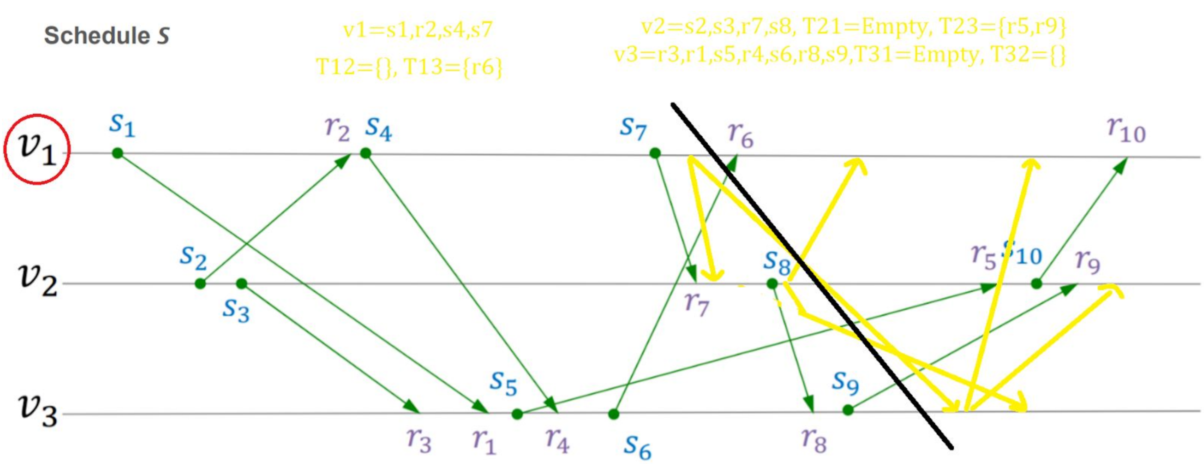
השיטה 1: להשתמש באלגוריתם שעוני למפורט.

- בוחרים זמן t.
- לוקחים את קבוצה C להיות כל האירועים שהזמן שניתן להם קטן או שווה ל t.
- הקבוצה הזו היא בהכרח חתך עקבי.

הוכחה: יהא $C \ni e$ ויהא $e' \Rightarrow_S e$ אז לפי נכונות שעוני למפורט: $\tau(e') < \tau(e) \le t$ ולכן: $e' \in C$.

שיטה 2 (יותר יעילה): אלגוריתם קנדי-למפורט, u רוצה תמונת מצב של המערכת.

- **הנחה: יש FIFO**
- אתחול: u מתחיל להקליט את תמונת המצב שלו ושולח הודעת סימון לכל שכניו.
- כאשר w מקבל מ u הודעת סימון:
  - אם זו הפעם הראשונה,
    - w מתחיל להקליט את המצב שלו.
    - קבוצת ההודעות מ u ל w היא ריקה.
    - אתחל: $S = \Phi$ כאשר S היא קבוצת כל ההודעות ש w עלול לקבל בזמן ההקלטה מכל אחד משכניו.
    - לאחר סיום ההקלטה, w שולח הודעת סימון לכל אחד משכניו (פרט ל u)
  - אם זו לא הפעם הראשונה אז w סוגר את ההקלטה עם u.



קונצנזוס

קונצנזוס - בעיית הסכמה. מספר מחשבים רוצים להסכים על אותו פלט.

המודל: זיכרון משותף א-סינכרוני. ישנם תאי זיכרון (רגיסטרים) משותפים כך שכל מחשב/מעבד יכול לגשת לכל אחד מהם לקריאה או לכתיבה.

תכונות שיש לקיים:

1. הסכמה -תמיד אותו פלט.
2. תקפות - אם הקלט זהה אז הפלא זהה לקלט
3. סיום - כל המחשבים שעובדים צריכים לסיים בזמן סופי.

תקלות: קריסה זמנית או טוטאלית.

הבעיה: לא ניתן לחכות למישהו שימשיך כי אולי הוא קרס ומצד שני, אם לא נמתין, עלול לקרות מצב שבו נדרוס אחד את השני בתאי הזיכרון.

הגדרות של סוגי קודקודים בעץ ההרצה:

<u>מצב התחלתי -</u> שורש העץ.
<u>מצב סופי - פלט -</u> העלים של העץ.
<u>קודקוד יוניוולנטי \ חד ערכי -</u> קודקוד שכל המסלולים החל ממנו ולמטה יובילו לאותו פלט. לדוגמא 0.
<u>קודקוד ביוולנטי \ דו-ערכי -</u> קודקוד שיש מסלולים תחתיו שיובילו לפלט 0 ויש מסלולים שיובילו לפלט 1.
<u>קודקוד קריטי -</u> קודקוד ביוולנטי ששני בניו הם יוניוולנטיים.

<u>טענה:</u> אם יש רק פעולות של קריאה אטומית וכתיבה אטומית (כל אחת בנפרד) אז לא קיים אלגוריתם שיוביל להסכמה.

<u>הוכחה:</u> נניח בשלילה שיש אלגוריתם המביא להסכמה.
הרעיון הוא להשתמש בעץ ההרצות ולהראות שיש בעץ מסלולים שמנקודה מסוימת הם בלתי ניתנים להבחנה ואז הפלט יהיה זהה.
נוכיח תחילה שבהכרח יש בעץ קודקוד קריטי ואז ממנו נראה שכל אפשרות לקריאה/כתיבה של אחד התהליכים או שניהם לא יובילו לקונצנזוס.
ההנחה היא שהפעולות היחידות שהן מותרות הן קריאה וכתיבה.

נניח בשלילה שיש אלגוריתם המשיג את 3 התכונות ומביא להסכמה.
בעץ יש מסלול המוביל לפלט 1 עבור כולם וגם יש מסלול המוביל לפלט 0 עבור כולם.
כי לפי תקפות, אם הקלט של שניהם יהיה 0 אז הפלט יהיה 0. ואם הקלט של שניהם יהיה 1 אז הפלט יהיה 1.

<u>שלב 1:</u> נראה כי יש קודקוד ביוולנטי: נשים לב שעבור הקלט $(0,1)$. אם רק התהליך הראשון ירוץ והשני יקרוס אז מכיוון שאסור להיות תלוי בזמן של האחר, התהליך הראשון חייב להחזיר 0. לפי תקפות.
באופן סימטרי, על אותו קלט, התהליך השני חייב להחזיר 1.
לכן מהשורש יש מסלול ל 0 ויש מסלול ל 1 ולכן הוא ביוולנטי.

<u>שלב 2:</u> יהיה הקודקוד הביוולנטי ברמה הגבוהה ביותר (כמה שיותר למטה בעץ) - קיים כי האלגוריתם סופי.
נראה שהוא קריטי.
נניח בשלילה שלא ולכן יש לו בן שהוא ביוולנטי - סתירה לכך שלקחנו את הביוולנטי ברמה הכי גבוהה.

מהקודקוד הקריטי יוצאות קשתות המתארות בקשת שמאל פעולה מהאלגוריתם של הקודקוד הראשון.
ובקשת ימין פעולה מהאלגוריתם של הקודקוד השני.

פעולות אפשרויות:
קודקוד 1: קריאה מתא זיכרון A
קודקוד 2: קריאה מתא זיכרון B

קודקוד 1: קריאה מתא זיכרון A
קודקוד 2: קריאה מתא זיכרון A

קודקוד 1: קריאה מתא זיכרון A
קודקוד 2: כתיבה לתא זיכרון B

קודקוד 1: קריאה מתא זיכרון A
קודקוד 2: כתיבה לתא זיכרון A

קודקוד 1: כתיבה לתא זיכרון A
קודקוד 2: כתיבה לתא זיכרון B

קודקוד 1: כתיבה לתא זיכרון A
קודקוד 2: כתיבה לתא זיכרון A

<u>**שלב 3:**</u> בכל אחד מקומבינציית המקרים, נראה שני מסלולים בלתי ניתנים להבחנה מ 2 צידי הקודקוד הביוולנטי וזה יוביל לסתירה כי אז 2 המסלולים יובילו לאותו פלט אחיד (כי מקיימים הכסמה) סתירה לכך שזה קודקוד קריטי ובפרט ביוולנטי.

אם באחד מהצדדים יש כתיבה של קודקוד 1 לתא A אז אם הכתיבה תתבצע לאחר הפעולה של קודקוד 2 ואז קודקוד 1 יסיים לבד או שפעולת קודקוד 2 בכלל לא תתבצע כעת ורק קודקוד 1 יסיים לבדו נקבל 2 מסלולים בלתי ניתנים להבחנה ולכן הפלא של קודקוד 1 יהיה זהה ולכן גם זהה לקודקוד 2 (בגלל ההסכמה) - קיבלנו שיש מ 2 הצדדים של קודקוד קריטי את אותו הפלט וזו סתירה.

אם שניהם קוראים אז אם קודקוד 1 קורא ואז ממשיך לבד או שקודקוד 2 קורא ראשון ואז 1 והוא ממשיך לבד לאחר מכן, שוב נקבל בשני הצדדים 2 מסלולים בלתי ניתנים להבחנה עבור קודקוד 1 ובאותו אופן נגיע לסתירה.

<u>**משפט: (הוכחה לפי דן)**</u> לא קיים אלגוריתם wait free אסינכרוני שמשיג הסכמה עם פעולה קריאה או כתיבה אטומיות.
<u>**הוכחה:**</u>
שלב 1: יהי 2 תהליכים A,B שמקבלים קלט בינארי. נניח בשלילה שקיים אלגו' המשיג קונצנזוס.
האלגו' A,B עושים צעד (קריאה או כתיבה) נבנה הרצות.
שלב 2: נוכיח שיש בחירה של קלטים כך ששורש העץ הוא ביוולנטי.
נסתכל על הרצות כאלו: שאחד עושה הכל ואז השני מתעורר. אחד רץ לבד ולכן מוציא את הקלט שלו בה"כ 0.
והשני אם הוא ירוץ לבד הוא יוצא את הקלט שלו בה"כ 1. ולכן השורש ביוולנטי.
שלב 3: יש קודקוד קריטי.
התחלנו מקודקוד ביוולנטי אם אין קודקוד קריטי אז העץ יהיה ביוולנטי לעד. נוכיח שהעץ לא יכול להישאר ביוולנטי לעד. כל שלב נבחן את הקודקוד אם קריטי סיימנו, אחרת יש לו קודקוד ביוולנטי נלך אליו, אם קריטי סיימנו אם לא נלך לילד הביוולנטי וככה נמשיך, לא נוכל להמשיך לאינסוף כי האלגוריתם סופי והעלה יוניוולנטי ולכן קיים קודקוד קריטי.

- <u>**הערה:**</u> נשים לב שעד כאן לא השתמשנו בזה שהוא קורא או כותב לתא אז אפשר להגיד לכל שלב.

שלב 4: נסתכל על האפשרויות שיש לנו צומת בעץ שאמור להיות קריטי.

- **אם אחד מהם קורא:**
 בה"כ B קורא בצומת, בצד ימין, אם A רץ לבד בצד שמאל הוא מוציא 0. אם בצד שמאל B קורא אז A רץ לבד הוא לא יכול להבדיל ויוצא 1 למרות שאמור להוציא 0 לפי קריטי ולכן לא יכול להיות קריטי.
- **שניהם כותבים לתאים שונים:**
אם A כותב ל y ו B כותב ל x בה"כ: אם A כותב לy (בחרנו את המסלול הזה) עדין B רוצה לכתוב ל B והוא יכול להבחין בין אם A כותב לy או לא כי זה לא קשור אליו. והפוך אם B כותב לx משהו.
ולכן יוצא אותו פלט ב 2 המסלולים ולכן לא קריטי.
- **שניהם כותבים לאותו תא**:
אם נלך למסלול השמאלי בריצה הזו, היה output 1 ואם ניתן קודם לשני לרשום ב x ואז לראשון לכותב ל x ורץ לבד אז הוא לא יכול להבדיל בין זה וזה כי הוא לא קרא את השינוי. ישר כתב.

ולכן יפלוט אחד בסתירה לקריטי.

לסיכום - קיבלנו שאין צומת שיכולה להיות קריטית בסתירה לזה שחייב להיות קודקוד קריטי.

שלבי הוכחה:

1. נניח בשלילה שיש אלגו wait free ולכן יש לו עץ הרצות.
2. נוכיח שיש בחירה של קלטים כך שהשורש ביוולנטי.
3. על סמך 2 יש קודקוד קריטי (השורש ביוולנטי והעלים יוניוולנטיים)
4. נוכיח שלא יכיל קודקוד קריטי בעץ - סתירה ל 3.

<u>הסכמה כאשר הפעולה המותרת היא פעולת קריאה/כתיבה אטומית (הכל ביחד בפעולה אחת).</u>

<u>במקרה כזה יש פתרון:</u>

משאב משותף: תא זיכרון C. מאותחל ל $?$.

- $r = RMW(C,x)$
- אם $r=?$ אז החזר x.
- אחרת החזר r.

<u>קיימים מספר סוגים של פונקציות RMW:</u>

1. $RMW(C,x)$
   - $old = read(C)$
   - $write(C, f(x))$
   - return old
2. $Test\ And\ Set$:
   - $old = read(C)$
   - $write(C,1)$
   - return old
3. $Fetch\ \&\ Inc$
   - $old = read(C)$
   - $write(C, old+1)$
   - return old
4. $Fetch\ \&\ Add(x)$
   - $old = read(C)$
   - $write(C, old+x)$
   - return old
5. $Compare\ \&\ Swap(x,y)$
   - $old = read(C)$
   - $if(old\ ==\ x)\ write(C,y)$
   - return old

<u>מספר קונצנזוס:</u> המספר n הגדול ביותר עבור קבוצת פונקציות A כך שבעזרת הפונקציות הללו ניתן לספק קונצנזוס (הסכמה) ל n תהליכים.

דוגמא: $A = \{read, write\}$ אז $n=1$.

ואם $A = \{RMW\}$ אז $n=2$.

משפט: אם יש אפשרות לממש אלגוריתם X מ Y והמספר קונצנזוס של X הוא m אז המספר קונצנזוס של Y הוא לפחות m.

משפט הקונצנזוס:

1. לכל $f(x) \neq ID(x)$ ב RMW, יש מספר קונצנזוס n כך ש $n \geq 2$.
2. תהי F קבוצה של פונקציות כך ש: (לכל i,j).
   $f_i(f_j(x)) = f_j(f_i(x))$ **- קומוטטיביות. או** $f_i(f_j(x)) = f_i(x)$ **- ניתנות לדריסה**.
   אז יש לה מספר קונצנזוס n כך ש: $n \leq 2$.

קונצנזוס במודל שליחת הודעות סינכרוני

(גרף שבו כל מחשב הוא קודקוד ושולחים הודעות אחד לשני)

רוצים לקיים:

1. הסכמה: כל הפלטים הם אותו הדבר.
2. תקפות: אם כל הקלטים זהים אז כל הפלטים זהים לקלט.
3. סיום: לאחר זמן סופי.

תקלות:

1. **קריסת מחשב:**

אלגוריתם f - התאוששות: אלגוריתם שמצליח להשיג הסכמה כאשר נתון מראש שיהיו לכל היותר f קודקודים תקולים. כאשר ההסכמה היא רק על הקודקודים שאינם תקולים.

משפט: במערכת שליחת הודעות סינכרונית עם $n \geq f+2$ קודקודים, כל אלגוריתם f- התאוששות דורש לפחות $f+1$ סיבובים כדי להשיג הסכמה.

(כי אם יש f תקולים אז שיישארו לאחר התקולים לפחות 2 קודקודים שתהיה ביניהם הסכמה כי אחרת אין דרישה להסכמה כי לא יישארו קודקודים עובדים במערכת)

2. **קודקודים ביזנטיים במודל שליחת הודעות סינכרוני:**

קודקוד ביזנטי הוא מחשב תקול ששולח הודעות באופן שרירותי ללא תלות באלגוריתם.
הוא יכול במקרה הגרוע לשלוח הפוך מהאלגוריתם, הוא נחשב בעינינו קודקוד "רע" שרוצה להפיל את האלגוריתם.

רוצים לקיים הסכמה:

- הסכמה: כל הפלטים הם אותו הדבר. עבור הקודקודים הרגילים.
- תקפות: אם כל הקלטים זהים אז כל הפלטים זהים לקלט עבור הקודקודים הרגילים.
- סיום: לאחר זמן סופי.

ידוע מראש שיהיו m קודקודים תקולים ביזנטית לכל היותר.

משפט: לא קיים אלגוריתם m - התאוששות (שיכול להתמודד עם m תקולים ביזנטית) כאשר $m \geq n/3$
(n = כמות הקודקודים במערכת)

כלומר אם לפחות שליש מכלל הקודקודים תקולים ביזנטית אז לא ניתן להגיע להסכמה בכל הרצה.

אלגוריתם המלכה:

מגיע להסכמה עבור m תקולים תוך $m+1$ שלבים כאשר לכל סיבוב ב2 חלקים.
ועובד רק כאשר $m < n/4$.

<u>האלגוריתם: קלט x</u>
בכל אחד מ $m+1$ השלבים בצע:
<u>שלב 1:</u>
- שלח לכולם את הקלט שלך.
- בקבלת הקלטים מכל האחרים בחר את הקלט שהופיע ברוב הפעמים. אם יש שוויון, בחר את המינימום.
- אם הקלט הנבחר הופיע יותר מ $n/2+m$ - תמוך בו. אחרת - אל תתמוך (אלא רק שמור את מה שבחרת).

<u>שלב 2:</u>
- בוחרים מלכה (לפי סבב של אינדקס הקודקודים או לפי בחירה אחרת)
- המלכה שולחת את הקלט שלה לכולם (הקלט = מה שבחרה בשלב 1)
- כל קודקוד שלא תומך בקלט שלו משלב 1. משנה את ערכו לקלט של המלכה.

לאחר כל השלבים, פלוט את הערך הסופי שלך.

נראה למה בהכרח תהיה הסכמה:
תקפות: אם כולם קיבלו את אותו קלט אז כולם שולחים את אותו הדבר ולכן כל קודקוד יקבל את הקלט הזהה לפחות $n-m > n/2+m$ (מתקיים כי: $m < n/4$) ולכן כולם ייתמכו בקלט הזה (וכמובן ייבחרו בו) ולכן תהיה הסכמה ללא תלות במלכה.

הסכמה: הטענה היא שלאחר הסיבוב בו המלכה היא תקינה (ולא ביזנטית) תהיה הסכמה.
אם אף קודקוד לא תמך בקלט אז המלכה תשנה את כולם אליה. מכאן, המשך הסיבובים יהיה כמו בתקפות ולכן ללא תלות במלכות הבאות נקבל הסכמה.

אם עד לאותו סיבוב היה קלט שהוא יותר מ $n/2+m$ פעמים אז חלק יתמכו בו וחלק לא.
אבל בכל מקרה, כולם יקבלו את הקלט הזה (כולל המלכה) יותר מ $n/2$ פעמים ולכן ישנו את הקלט שלהם אליו.
בשלב 2 המלכה תשלח את אותו הדבר ולכן גם אם הם לא תמכו, הקלט לא ישתנה.

<u>סד"פ -</u>

<u>שאלות גנרלים:</u>

- **בעיית הגנרלים דטרמיניסטית - לא פתירה.**
  - מניחים בשלילה שיש אלגוריתם כזה
    - **לרשום מלל:** "נניח בשלילה שקיים אלגוריתם סופי דטרמיניסטי שפותר את הבעיה - אם האלגוריתם לא מחייב שליחת הודעה באיזשהו סיבוב אז הוא שולח ε ".
  - אם יש יותר מ2 קודקודים - נרצה לבחון מתי קודקוד אחד מבחין בשינוי אבל זה לא משנה את הפלט.

- **לשים לב:** לפעמים ניתן לעשות שימוש באלגוריתם הרמות כדי **כן** לפתור את הבעיה!
(אם יש הבטחה שזמן מוגדר **כל** קודקוד מקבל הודעה 1 לפחות - ניתן לעשות זאת ע"י אלגוריתם שמזכיר את הרמות. אם יש התחייבות ל1 הפרכה רגילה.)
- ***הערה:*** יש לשים לב אם צריך להוכיח משהו לפני כדי להראות שהרצות צמודות הן זהות. (עיין ערך מבחן 2020)
- נריץ הרצות דומות:
  - הרצה ראשונה E_0 - קלט X כל ההודעות נשלחו. (ולידיות והסכמה)
  - נעשה עוד xT הרצות (כאשר x = למס' הקודקודים) - להוריד הודעות עד שמגיעים שבסיבוב הראשון כל ההודעות נופלות. הרצה אחרונה E_{xT} הרצה ראשונה E_1.
בכל הרצה צריך "בגלל הסכמה - קודקוד v_i ....".
    - לואה ראשונה $1 \le i \le T$:
    - $E_{xi-(x-1)}$ דומה ל $E_{xi-(x)}$ - רק שבסיבוב $T-(i-1)$ ההודעה ל v_1 לא הגיעה.
    - $E_{xi-(x-2)}$ דומה ל $E_{xi-(x-1)}$ - רק שבסיבוב $T-(i-1)$ ההודעה ל v_1 לא הגיעה.
    - ...
    - E_{xi-1} דומה ל E_{xi-2} - רק שבסיבוב $T-(i-1)$ ההודעות ל v_j לא הגיעה.
    - E_{xi} דומה ל E_{xi-1} - רק שבסיבוב $T-(i-1)$ ההודעות ל v_k לא הגיעה.
  - לאחר מכן נעשה עוד x הרצות להחליף את הקלטים. (כאשר אף הודעה לא נשלחה)
    - E_{xT+1}: דומה ל E_{xT} אבל הקלט של v_1 שונה.
    - נמשיך עד E_{xT+x} ככה נחליף את כל הקלטים.
  - ואז עוד xT הרצות - כדי להגיע מכל ההודעות לא מגיעות להכל מגיע. סה"כ $2xT+x$.
    - לולאה שניה $1 \le i \le T$:
    - $E_{xT+x+xi-(x-1)}$ דומה ל $E_{xT+x+xi-(x)}$ - רק שבסיבוב i ההודעות ל v_1 כן הגיע.
    - ...
    - $E_{xT+x+xi}$ דומה ל $E_{xT+x+xi-(x-(x-1))}$ - רק שבסיבוב i ההודעות ל v_k כן הגיע.

<u>שאלות של אלגוריתם הרמות:</u>

- **שאלות הוכחה הפרכה:**
  - **לזכור ש<u>ניתן</u> להפיל הודעות!**
  - הפרכה: אם u קיבל יותר הודעות מ v - $l_u > l_v$. - דוגמת הרצה.
  - הפרכה: אם u קיבל יותר הודעות מ v - $l_u < l_v$. - דוגמת הרצה.
  - הפרכה: אלגוריתם הרמות עם קבלה של 2 הודעות לא מתנהג אותו דבר.
  - שיטות הוכחה: לעשות שימוש ב W.O.P עם הנחה בשלילה.
נגדיר :נניח בשלילה שההנחה מתקיימת\לא הפעם הראשונה שקורה "משהו", נראה שזה לא קורה ולכן צריך ללכת אחורה אבל סתירה ל W.O.P.
  - להוכיח $|l_u - l_v| \le 1$ - אינדוקציה על כמות הסיבובים, ולפצל למקרים:
(בכל אחד: אם לא התקבלו הודעות - אם רק הודעה אחת התקבלה - אם שניהם.)
    - $l_u = l_v = r$.
    - בה"כ $l_v = r,\ l_r = r+1$.

<u>שאלות מערכת אסינכרונית:</u>

- **עיכוב הודעות:**
  - **חסם מינימלי:** עיכוב של ההודעה הכי גדולה הכי קצת זמן. (אם רוצים מקס')

- **חסם מקסימלי:** נוסיף עיכוב לכל קודקוד לפי המספר שלו (שגדל כמו n^i) ככה שהקודקוד הכי גדול מתעכב הכי הרבה. (כאן לשים לב כמה הודעות כל אחד שולח וגורם לאלו שאחריו לשלוח. יהיה לנו Σ כלשהי).
- **בקליקה:** חסם מינ' $\Theta(n^2)$ חסם מקס' $\Theta(n^3)$. וניתן להגיע לכל דבר בין לבין.
  - כדי להגיע ל**"בין לבין"** החסמים - נגדיר התקדמות הדרגתית כמו במקס' עד קודקוד c כלשהו ואז נקבל ... $\sum_{i=1}^{c}$ מהסיגמה נקבל את c בחזקה כלשהי וננסה לשנות עד שיגיע לסדר גודל הרצוי. **חשוב לזכור שאם יש** $c^x \cdot n$ **בסיגמה זה =** $O(n \cdot c^x)$
    לאחר שנעדכן עד ל c הזה בצורה הדרגתית את היתר ישר נעדכן בערך הכי גדול.
- **במעגל:** חסם מינ' $\Theta(n)$ חסם מקס' $\Theta(n^2)$.

<u>שאלות אלגוריתמים לשליחת הודעות:</u>

- **ב flooding -**
  - אם אין לנו מעגל יהיה לנו בדיוק $|E|$ הודעות.
  - אם רוצים להגיע לכמעט $2|E|$ פשוט נחבר את כולם לכולם.

<u>שאלות לוחות זמנים:</u>

- **הוכחה\הפרכה ש** S' **הוא קוזאל שאפל של** S**:**
  - נראה local view של כל קודקוד נתון ונשווה בין לוחות הזמנים.
    (local view = כל קודקוד בהינתן S נראה איזה אירועים יש על הציר שלו).
- **שאלות של Happens Before:**
  - אם שואלים איזה מאורעות **לא מקיימים** את היחס בהינתן e מסוים:
    - אם יש לנו S ו S' קוזאל שאפל שלו, נבדוק איפה e ממוקם ומה מגיע אחריו:
      - אם e הוא שליחה אירוע הקבלה מגיע אחרי.
      - כל מה שבאותו קודקוד ונמצא אחרי e.
      - נבדוק את ה local view של כל קודקוד וננסה להסיק יחס, אם לא ידוע אז גם לא ניתן להגיד שזה מקיים. (כדי לנסות לבחון אירועים הקשורים לקודקוד שאנחנו עליו).
    - לרשום למה כל אחד **"לא ביחס"** וכדאי לרשום גם למה היתר **"כן ביחס"**.
  - אם שואלים **כן מקיימים** אז זה בדיוק הפוך.
  - אם שואלים **בין לבין** נעשה את החיתוך בין הקבוצות.
- **בניית לוז וקוזאל שאפל בהינתן local views:**
  - טריק: לקחת קודם את כל השליחות (אם צריך להכניס קבלה בין לבין נכניס) ואז את מה שנשאר אפשר לערבב בשני סידורים שונים.
- **שעוני למפורט:**
  - **ייצוג לוז לפי שעוני למפורט:**
    - טבלה שבשורה הראשונה יש את הקודקודים ובעמודה הראשונה יש את האירועים.
    - בכל אירוע שליחה נקדם ובכל אירוע קבלה נבדוק $max(c_v, c_u) + 1$, ונמלא בטבלה.
  - **נקודות שצריך לזכור:**
    - $e \Rightarrow e'$ <-- $\tau(e) < \tau(e')$ <u>אבל</u> $\tau(e) < \tau(e')$ -\-> $e \Rightarrow e'$.
    - S' הוא קוזאל שאפל של S $\Leftrightarrow$ $\forall e \in S' \ \tau_S(e) = \tau_{S'}(e')$.

- בין שני אירועים רציפים על אותו קודקוד יכול להיות הפרש גדול ממש ב τ שלהם עם מאורעות בין לבין.

- **Vector Clocks:**
  - **נקודות שצריך לזכור:**
    - שני מאורעות סמוכים $e_1 \Rightarrow e_2$ על אותו קודקוד $VC_v(e_1) + 1 = VC_v(e_2)$.
    - $VC_u(u) > VC_u(v)$ לכל שני קודקודים.

      (הווקטור שלי במיקום שלי תמיד גדול או שווה לווקטור של קודקוד אחר במיקום שלי).

      ההוכחה של זה צריכה להיות באינדוקציה על כמות המאורעות. (נגדיר עד אירוע $k-1$ ואז נגיד שהוא בקודקוד w ונציג את כל האפשרויות מה w יכול להיות).

__שאלות חתכים:__

- **חתך עקבי:**
  - **נקודות שצריך לזכור:**
    - **להראות קודם שזה חתך!!**

      אם יש שני אירועים e_1, e_2 באותו קוד' ו $e_1 \Rightarrow e_2$ $e_2 \in C$ צריך להראות שגם $e_1 \in C$.
    - להראות שזה עקבי: $r_M \in C$ וצ"ל $s_M \in C$.
    - בהוכחת C עקבי - לזכור לעשות שימוש ב e'' באינדוקציה על כמות מאורעות.
    - $C_1 \cap C_2$, $C_1 \cup C_2$ זה עדיין עקבי.

      בהוכחה נראה e_1, e_2 כך ש $e_1 \Rightarrow e_2$ ו $e_2 \in \cap / \cup$ צריך להראות שגם $e_2 \in \cap / \cup$.
    - C עקבי ב S וגם ב CS שלו. **אבל** אם הוא עקבי ב S, S' זה לא אומר שהם קוזאל שאפל.
    - חתך ריק Φ הוא גם חתך עקבי.
  - **כדי לקבוע האם חתך הוא עקבי בהינתן S :**
    - חתך - נעשה את ה local view של כל קודקוד ונעבור על זה ונראה מה נמצא שם.
    - עקבי - לכל קבלה יש את השליחה שלהם.

__שאלות קונצנזוס:__

- **מודלים זיכרון משותף אסינכרוני Wait Free Read or Write:**
  - צריך להראות שאין המתנה - אמינות - הסכמה כלשהי -סיום.
  - בהסכמה צריך לזכור: הם לא ממתינים אחד לשני ולכן יכולים לעשות הכל במקביל, לכן האלגוריתם צריך לעבוד גם בצורה דיפולטיבית.
  - בדרך כלל צריך לעשות באלגוריתם קודם על כתיבה ואחרי זה לקריאה - בקריאה ליצור מצב שמגיע לאמינות, אחרת מצב שמביא להסכמה.
  - מצבים שצריך לבדוק בשאלות על אוגרים \ רגיסטרים (מתייחס ל 3 אפשר גם ל 2):
    - אחד קורא לפני שאחרים כתבו.
    - אחד קורא כשעוד אחד כתב.
    - כל הכתיבות לפני הקריאות.
- **מודלים Read Modify Write:**
  - כדי להפריך האם קיים אלגוריתם f שנתון לנו שמספק הסכמה לכמות מסוימת, נעשה הנחה בשלילה שקיים כזה ונראה אלגוריתם כללי.
  - כדי להגיע לכל מספר קונצנזוס שנרצה צריך לעשות קומוטטיביות בין כמה איברים שנרצה. (פשוט בניה על בניה וככה נמשיך...)
- **קודקודים ביזנטיים:**
  - בעץ מספיק ביזנטי יחיד כדי להרוס את העץ. - שורש.
- **אלגוריתם המלכה:**

- לזכור - יש $m+1$ שלבים **בכל שלב** יש 2 חלקים.
  (m = כמות ביזנטים. בכל שלב מלכה אחרת).
- שאלות שרוצים שיפול - כדי לשים את הביזנטי אחרון כדי לקבל פלט לא תקין.