מרתון 1 מערכות מבוזרות 8.11

מודלים:

1. שלחית הודעות: כל מחשב או שרת הוא קודקוד בגרף כאשר לכל קודקוד יש מצב פנימי (קונפיגורציה שלו) והתקשורת היא על ידי שליחת הודעות בין המחשבים. 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 וא-סינכרוני:

- **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.
  כלומר, *סיבוב במודל 1 = חישוב פנימי, שליחת הודעות לכולם, קבלת ההודעות מהאחרים.*
  סיבוב במודל 2 = חישוב פנימי, גישה לזיכרון המשותף (רק פעולה אחת – או קריאה או כתיבה), חישוב פנימי.
  מספר הסיבובים בכל אלגוריתם הוא סופי T.

- **א-סינכרוני:** הכל קורה במקביל. אין סיבובים.
  - במודל 1 – זמן הגעת ההודעות לא ידוע (אין חסם עליון על זמן עיכוב ההודעות). בזמן הזה המחשבים יכולים לבצע פעולות ואף לשלוח עוד הודעות.
    במקרה כזה מחלקים ל2 ערוצי תקשורת:
    - FIFO – לא ייתכן שהודעה מאוחרת מאותו מחשב תגיע לפני הודעה מוקדמת.
    - כללי – הכל ייתכן.
  - במודל 2 – הכתיבה\קריאה של כולם מתבצעת במקביל ולכן ייתכן דריסת נתונים (אם לדוגמה מחשב 1 כותב 1 לתא 3 ומחשב2 כותב 2 לתא 3 אז לא ידוע בסיום הריצה מה יהיה בתא 3, הכל לפי התזמון).

תקלות (האלגוריתם לא מתבצע בשלמותו):

בכל מודל ייתכנו מספר סוגי תקלות:

1. קריסה – אחד המחשבים מפסיק לתפקד באמצע האלגוריתם (יכול לקרות בכל שלב).
   - במודל 1 – זה יתבטא בכך שהוא לא שולח יותר הודעות.
   - במודל 2 – זה יתבטא בכך שהוא לא כותב שום דבר לזיכרון המשותף.

2. תקלה זמנית – אחד המחשבים לא מתפקד באופן זמני.
   - במודל 1 – זה יכול להתבטא בכך שחלק מההודעות לא מגיעות ליעדן. (הולכות לאיבוד).
   - במודל 2 – זה יכול להתבטא בכך שבחלק מהזמן המחשב לא כותב כלום לזיכרון המשותף.

3. ביזנטי – אחד המחשבים מתנהג רנדומלית (מבחינתנו – נגד האלגוריתם)
   - מודל 1 – שולח הודעות עם תוכן אקראי (לאו דווקא התוכן שאמור להיות לפי האלגוריתם)
   - במודל 2 – כותב בתאי הזיכרון תוכן אקראי.

תכונות של פרוטוקולים (אלגוריתמים)

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

מודל שליחת ההודעות

מושגים:

- קונפיגורציה :
המצב הכולל של כל המערכת. כלומר: רשימה של המצבים של אחד מהמחשבים במערכת.

- מצב:
רשימת המשתנים וערכיהם, באיזה שלב\שורה של האלגוריתם המחשב נמצא בביצוע וכ'ו.

***קונפיגורציה ומצב לא מכילים את התקשורת בין המחשבים: סטטוס ההודעות.*

- הרצה:
סדרת קונפיגורציות עם האירועים ביניהם. כאשר אירוע = שליחה או קבלה של הודעה.
הקלט של כל המחשב נתון עם ההרצה, קלט = הזמנה התחלתית של כל אחד מהמחשבים באופן מקומי, אין לו קשה להודעות כי הוא נקבע בתחילת הריצה (כמובן שאפשר לשלוח את הקלט בהודעות).
דוגמה: .... $C_0, s_{12}, C_1, r_{21}, C_2$. *(קונפ' אירוע קונפ' אירוע.....).*
<u>סימון:</u> s_{ij} הכוונה היא שמחשב i שולח הודעה s למחשב j.
<u>סימון:</u> r_{ij} מחשב i מקבל את ההודעה r ששלחת מחשב j.

- לוח זמנים:
הרצה ללא הקונפיגורציות אלה רק האירועים. (גם כאן הקלט כלול).
דוגמה: .... s_{12}, r_{21}.

- צמצום לוח זמנים לקודקוד v:
לוח זמנים שמוצגים בו רק הארועים שהתרחשו במחשב v.
דוגמה: $S = s_{12}, s_{12}r_{31}, s_{31}, s_{32}, r_{23}, r_{21}, r_{13}$
אז:
$S|1 = s_{12}, s_{13}, r_{13}$
$S|2 = r_{23}, r_{21}$
$S|3 = r_{31}, s_{31}, s_{32}$
בעצם אני רואה את האירועים רק של מחשב יחיד.

שימו לב – יש חשיבות לסדר האירועים בהרצה, בלוח הזמנים ובצמצום.

הערה:

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

<u>משפט:</u> אם S, S' הם 2 לוחות זמנים עם אותו קלט כך ש: $S|1 = S'|1$ אז מחשב 1 מבצע את אותם פעולות בדיוק ב2 ההרצות (בהנחה שהוא דטרמיניסטי).

בעיית שני הגנרלים

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

<u>לבעיית ההסכמה יש תכונות שצריך לקיים :</u>

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.
דוגמה: אם מחשב 1 וגם מחשב 2 קיבלו את הקלט 0 אז שניהם חייבים להוציא את הפלט 0 ולא 1.
3. סיום = האלגוריתם חייב לעצור לאחר זמן סופי.

מודל:

שני מחשבים (דטרמיניסטיים).

מודל שליחת הודעות.

סינכרוני (מחולק לסיבובים).

תקלות הודעות יכולת ללכת לאיבוד ולא להגיע.

<u>השאלה –</u> האם קיים אלגוריתם המקיים את 3 התכונות לעיל במודל הנ"ל?

<u>תשובה –</u> לא קיים אלגוריתם כזה.

<u>הוכחה –</u> על ידי 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 שהאלגו מקיים תקפות אז הפלט בהכרח (0,0).

E_2 קלט – *(0,0)*. בסיבוב *T* האחרון ההודעה מv_1 לv_2 הלכה לאיבוד. כל שאר ההודעות הגיעו.

<u>נשים לב</u> שההרצה E_1 דומה להרצה E_2 עבור v_1 כלומר: $E_1 \sim_{v_1} E_2$, ולכן הפלט של v_1 יהיה 0 ופלט של v_2 מכיוון שהאלגו' מקיים הסמכה יהיה 0 כלומר הפלט הוא *(0,0)*.

E_3 קלט – *(0,0)*. כמו E_2 ובנוסף בסיבוב *T* האחרון גם ההודעה מv_2 לv_1 הלכה לאיבוד. כל שאר ההודעות הגיעו.

<u>נשים לב</u> שההרצה E_3 דומה להרצה E_2 עבור v_2 כלומר: $E_3 \sim_{v_2} E_2$, ולכן הפלט של v_2 יהיה 0 ופלט של v_1 מכיוון שהאלגו' מקיים הסמכה יהיה 0 כלומר הפלט הוא *(0,0)*.

E_4 קלט – *(0,0)*. כמו E_3 ובנוסף בסיבוב *T-1* האחרון גם ההודעה מv_1 לv_2 הלכה לאיבוד. כל שאר ההודעות הגיעו. באותו אופן כמו מקודם $E_3 \sim_{v_1} E_4$.

....

E_{2T+1} הקלט – *(0,0)*. כל ההודעות אבדו. אבל $E_{2T} \sim_{v_2} E_{2T+1}$ ולכן כמו מקודם הפלט יהיה *(0,0)*.

E_{2T+2} קלט – *(0,1)*. כל ההודעות אבדו. $E_{2T+1} \sim_{v_1} E_{2T+2}$ (לא שיניתי לv_1 את הקלט לכן הכל תקין). 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 הראשונה מv_1 לv_2 הגעיה. $E_{2T+3} \sim_{v_1} E_{2T+4}$ ולכן כמו מקודם הפלט יהיה *(0,0)*.

.....

E_{4T+3} קלט *(1,1)*. לא אבדו הודעות. $E_{4T+3} \sim_{v_2} E_{4T+2}$ זאת סתירה לתקפות כיוון שלפי התקפות הפלט צריך להיות זהה לקלט שהוא *(1,1)*.

<u>שאלת מבחן:</u>

(בסיכום של המרתון)

בעיית 4 גנרלים אבל פה צריך הסכמה חלשה זאת אומרת שהפלט של כל קודקוד צריך להיות בינארי וצריך שיהיה רוב מוחלט לאחד הפלטים.

פתרון:

נניח בשלילה שקיים אלגו המקיים את 3 התכונות מהשאלה. מכיוון שהוא מקיים את הסיום אז האלגו מסיים לכל היותר לאחר T סיבובים.

נגדיר סדרה של הרצות דומות כך שברצה הראשונה הקלט יהיה 0,0,0,0 וכל ההודעות יגיעו ומכיוון שהאלגו' מקיים וולידיטי אז הפלט יהיה 0,0,0,0.

לכל 2 הרצות סמוכות בסדרה נשמור ש3 מתוך 4 קוד' לא מבחינים בהבדל, כלומר ההרצות דומות עבורן, ולכן שלושה פלטים תמיד ישארו זהים להרצה הקודמת.

אם 3 הפלטים היו 0-ים אז הרביעי יכול לשנות ל1 ולקבל רוב אפסים או שהוא ישאר גם כן בפלט 0 ויקבל רוב אפסים.

אם שנים מהם הם 0 ואחד מהם הוא 1 אז הרביעי הוא חייב להסכים איתם כי האלגו' מקיים הסכמה חלשה ולכן נקבל רוב אפסים.

לכן לאורך כל ההרצות נשמור על פלט של 3 אפסים לפחות.

בהרצה האחרונה נדאג שהקלט יהיה 1,1,1,1 וכל ההודעות יגיעו אבל הפלט יהיה לפחות 3 אפסים וזו תהיה סתירה לוולידיטי שלפיו הפלט צריך להיות 1,1,1,1

פורמאלית: נוכיח תחילה טענת עזר" לאורך כל סדרת ההרצות הדומות תמיד יהיו לפחות 3 פלאים שהם 0.

באינדוקציה על אינדקס ההרצה:

בסיס – בהרצה הראשונה כל ההודעות מגיעות והקלט הוא 0,0,0,0 ולכן הפלט הוא 0,0,0,0.

צעד – נניח שהטענה נכונה עד הרצה k ונוכיח עבור הרצה $k+1$

מכיוון שהרצה E_k דומה ל $E_{k+!}$ ב3 קוד' אז...

להשלים.

E_1קלט 0,0,0,0 כל ההודעות מגיעות ולכן כל ההודעות מגיעות ולכן הפלט 0,0,0,0 לפי ולידיטי.

E_2 קלט 0,0,0,0 ההודעות שנשלחות לv_1 בסיבוב T לא מגיעות אליו. $v_2 v_3 v_4$ לא מבחינים ולכן לפי טענת העזר בפלט יהיו לפחות 3 אפסים.

E_3 קלט 0,0,0,0 כמו E_2 ובנוסף ההודעות שנשלחות לv_2 בסיבוב T לא מגיעות אליו. $v_1 v_3 v_4$ לא מבחינים ולכן לפי טענת העזר בפלט יהיו לפחות 3 אפסים.

....

E_{4T+1} הקלט 0,0,0,0 אף הודעה לא מגיעה. הרצה זו דומה לקודמת עבור 3 קוד' ולכן לפי טענת העזר בפלט יהיו לפחות 3 אפסים.

E_{4T+2} 1,0,0,0 אף הודעה לא מגיעה. $v_2v_3v_4$ לא מבחינים ולכן לפי טענת העזר בפלט יהיו לפחות 3 אפסים.

...

E_{4T+5} הקלט 1,1,,1,1 ....

...

E_{8T+4} הקלט 1,1,1,1 כל ההודעות מגיעות 3 לא מבחינים ולכן לפי טענת העזר בפלט יהיו לפחות 3 אפסים.

סתירה לולידיטי כיוון שבהרצה האחרונה הפלט היה צריך להיות 1,1,1,1.

בעיית 2 הגנרלים במודל אי-דטרמיניסטי.

הבעיה זהה לבעיה הקודמת עם אותו מודל פרט לכך שהמחשבים הם אי דטרמיניסטיים.

במקרה כזה הבעיה פתירה בהסתברות קטנה לטעות.

אלגוריתם הרמות – עבור 2 מחשבים

לכל קוד' v יש משתנה L_v המייצג את הרמה שלו.

אתחול: $L_v = 0$ לכל v .

עבור i מ1 עד T:

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

2 המחשבים מעלים את הרמה אחד לשני.

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.

באמצעות אלגוריתם זה, ניתן ליצור אלגוריתם לבעיית 2 הגנרלים:

אלגו' לבעיית 2 הגנרלים

אלגוריתם לv: קלט x

- הגרל מספר t בין 1 ל r.
- בצע למשך r סיבובים את אלגו' 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 לu: קלט x

- בצע למשך r סיבובים את אלגו' הרמות כך שבמקום לשלוח רק את הרמה צרף להודעה גם את x וגם את t. כלומר סה"כ u שולח L_u, x.

כל אחד מהמחשבים פולט 1 אמ"מ מתקיימים כל התנאים הבאים:

1. המחשב יודע את t ואת שני הקלטים.
2. שני הקלטים הם 1.
3. הרמה שלי היא לפחות t.

בכל מקרה אחר המחשב המחשב יפלוט 0.

תכונת הולידיטי מתקריימת כי אם שני הקלטים הם 0 וכל ההודעות הגיעו אז לפני התנאי השני שלא מתקיים בשניהם – שניהם יפלטו אפס.

אם שני הקלטים הם 1 וכל ההודעות הגיעו אז תנאי 1 מתקיים כי ההודעות הגיעו והן מכילות את t ואת x.

תנאי 2 מתקיים כי הקלטים הם 1 ותנאי 3 מתקיים כי לפי אלגו' הרמות הרמה תהיה r אצל שניהם כי כל ההודעות הגיעו ומתקיים r גדול או שווה לt. לכן הפלטים יהיו 1.

הסכמה: מתקיימת תמיד אלא אם כן: הקלט של שניהם הוא 1,

הרמה של אחד מהם היא t בדיוק והשניה היא $t-1$.

במקרה כזה אחד מהם יוציא 0 והשני 1.

כל מה שהיריב (המערכת) יכול לזמן לנו – לא תלוי בהסתברות (כלומר 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 הt היא $\frac{1}{r}$.

ולכן ההסתברות להסכמה $1-\frac{1}{r}$. (כל זה במקרה והקלטים הם 1,1 , בכל מקרה אחר תמיד תהיה הסכמה על 0)

סיום – מתקיים לאחר r סיבובים.

מרתון 2 – 29.11.20

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

אנחנו רוצים לחפש אלגוריתם שעובד על *r* סיבובים אבל יותר מהרמות.

טענה: **כל אלגוריתם רנדומאלי** הפותר את בעיית שני הגנרלים ומקיים:

1. מסיים לאחר *r* סיבובים.
2. מקיים תקפות: אם שני הקלטים זהים ולא אבדו הודעות אז שני הפלטים זהים לקלט.
   בנוסף: אם לפחות אחד מהקלטים הוא 0 אז הפלט של שניהם 0.
3. הסכמה בהסתברות לפחות $1-\varepsilon$. כלומר שהטעות היא לכל היותר ε .

אז: $\varepsilon \geq \frac{1}{2r}$.

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

(זה מראה שזו הטעות הכי קטנה שאנחנו יכולים לעשות).

רעיון ההוכחה: דומה לשיטת ההוכחה שלא קיים אלגוריתם לבעיה הדטרמיניסטית הרגילה. ע"י הרצות דומות.

הרצה 1: קלט 1,1 לא אבדו הודעות פלט: 1,1. לפי (2).

כלומר – ההסתברות לטעות ולפלט 0 היא 0.

הרצה 2: קלט 1,1. הודעה אחרונה מ$v_1 to\ v_2$ אבדה. v_1 לא מבחין ולכן: $P(v_1 out\ 0) = 0$ כמו בהרצה הקודמת.

v_2 מבחין ולכן $P(v_2 out\ 0) \leq \varepsilon$ לפי ההנחה.

הרצה 3: קלט 1,1.כל ההודעות בסיבוב *r* אבדו.. v_2 לא מבחין ולכן: $P(v_2 out\ 0) \leq \varepsilon$ כמו בהרצה הקודמת.

v_1 מבחין ולכן $P(v_1 out\ 0) \leq 2\varepsilon$ לפי ההנחה.

ההסתברות לטעות: $p(v_1 out\ 0) - p(v_2 out 0) \leq p(wrong) \leq \varepsilon$. ההסתברות לטעות היא ההפרש שv_1 לבין ההסתברות שv_2 יוציא 0 חסומה ב ε אבל עדיין ייתכן שההתסברות של אחד מהם להוציא 0 תהיה גדולה בε מההסתברות של השני להוציא 0 ולכן בכל פעם שמישהו מבחין ההתסברות שלו להוציא 0 גדלה כמו של האחר + ε .

הרצה 4: קלט 1,1.כל ההודעות בסיבוב *r* אבדו וגם ההודעה בסיס *r-1* מv_1 לv_2 אבדה. v_1 לא מבחין ולכן: $P(v_1 out\ 0) \leq 2\varepsilon$ כמו בהרצה הקודמת.

v_2 מבחין ולכן $P(v_2 out\ 0) \leq 3\varepsilon$ לפי ההנחה.

הרצה 5: 1,1 כל ההודעות בסיבוב *r* אבדו וגם ההודעות בסיבוב *r-1* אבדו.v_2 לא מבחין ולכן $P(v_2 out\ 0) \leq 3\varepsilon$

....

הרצה *2r+1* : קלט 1,1. אף הודעה לא מגיעה. v_1 לא מבחין ולכן: $P(v_1 out\ 0) \leq 2r\varepsilon - \varepsilon$. v_2 מבחין ולכן:$P(v_2 out\ 0) \leq 2r\varepsilon$.

הרצה *2r+2* : קלט *0,1* . אף הודעה לא מגיעה. v_2 לא מבחין ולכן:$P(v_2 out\ 0) \leq 2r\varepsilon$. מצד שני, מתקיימת תקפות (ס' 2) ולכן : $P(v_2 out\ 0) = 1$ כי אם אחד מהקלטים הוא 0 אז שניהם חייבים לפלוט 0 בהסתברות 1 (ללא טעות).

קיבלנו: $1 \leq 2r\varepsilon$ ולכן:$\varepsilon \geq \frac{1}{2r}$. מש"ל.

טעות צריך להעתיק מחדש

אלגוריתמי שליחת הודעות במודל שליחת ההודעות

מודל: שליחת הודעות, גרף עם קודקוד שהצלעות הן ערוצי התקשורת.

סינכרוני וא-סינכרוני.

ללא תקלות.

יעילות אלגוריתם:

- סיבוכיות זמן (זמן ההודעות ללא החישובים המקומיים):

סינכורני : במודל זה יש סיבובים. בכל סיבוב ייתכן שכולם שולחים הודעה ומקבלים הודעה. (כלומר בסיום כל סיבוב, כל ההודעות שנישלחו מגיעות ליעדן)
כל סיבוב הוא יחידת זמן 1 והסיבוכיות תהיה מספר הסיבובים עד לסיום האלגוריתם.

א-סינכרוני: 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 מנורמלית לכל ההודעות.
(דוגמה: אם ההודעה הראשונה הגיע לאחר 1000 שניות אז היא יחידה אחת ואם הודעה שהגיע לאחר 500 שניות אז היא חצי יחידת זמן)
המקרה הגרוע הוא לאו דווקא כאשר וכלן הגיעו בזמן הכי ארוך כי ייתכן שדווקא הודעה אחת מגיעה מהר יותר מאחרת זה גורם לעיכוב בסיום האלגוריתם.
הסיבוכיות תהיה: מספר יחידות הזמן הכולל במקרה הגרוע. (שבו יש יותר מידי שליחת הודעות כתוצאה מעיכוב וזירוז של חלק מההודעות)

השאלות בנושא זה יהיו בדרך כלל חלוקת זמנים להודעות כך שנגיע למקרה הגרוע.

- סיבוכיות הודעות:

מספר ההודעות הכולל (בסדר גודל) שעבר ברשת מתחילת האלגו' ועד סופו.

האלגוריתמים:

1. *Flooding* –

המטרה: 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 שכנייך פרט ל w .

סיבוכיות זמן:

סינכרוני: $O(rad(G,v))$

לאחר סיבוב אחד, כל הקודקודים במרחק 1 מv יקבלו את M .
לאחר סיבוב 2, כל הקודקודים במרחק 1 מv שלחו לשכניהם האחרים והם קיבלו את M.
מכאן שמספר הסיבובים יהיה לפי הקודקוד u שהכי רחוק מv.
המרחק הוא: $d_G(v,u)$ או במילים אחרות הרדיוס של v בגרף.
סה"כ סיבוכיות זמן תהיה: $rad(G,v)$.

א-סינכרוני: $O(rad(G,v))$

כיוון שכל קודקוד ממתין ל*M* ורק בפעם הראשונה הוא שולח לכל שכניו.
הודעה יכולה להגיע לכל היותר לאחר יחידת זמן אחת ולכן הקודקוד במרחק *k* מ*v* יקבל את *M* לאחר לכל היותר *k* יחידות זמן (גם אם בזמן הזה הוא יקבל דרך יותר מ*k* קודקודים).

<u>סיבוכיות הודעות:</u>
סינכרוני: $O(|E|)$
בכל צלע תעבור לכל היותר 2 הודעות. פעם אחת מכל כיוון כי כל קודקוד שולח את *M* רק בפעם הראשונה שהוא מקבל את ההודעה.

א-סינכרוני : $O(|E|)$
מכיוון שלא משנה ממי הקודקוד יקבל הוא ישלח את *M* לכל שכנוי פרט לאחד ורק בפעם הראשונה שהוא מקבל את זה.

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 *converge cast* :
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 \ א-סינכרוני: אותה סיבוכיות.

סיבוכיות הודעות: $O(|E|) = O(|V|)$
סינכרוני \ א-סינכרוני: אותה סיבוכיות.

4. הפעלת *flooding* ואחריו *CC* על העץ המתקבל:

סיבוכיות זמן:
סינכרוני – $O\big(rad(G,v)\big)$
א-סינכרוני – $O(|V|)$

סיבוכיות הודעות: $O(|E|)$

אלגוריתמים למציאת עץ *BFS* במערכת א-סינכרונית:

1. דייקסטרה:
מטרה: למצוא עץ *BFS* במערכת א-סינכרונית.

רעיון: 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 שקיבלת את ההודעה והוא ברמה 1 הוא מעביר אותה הלאה לכל שכניו.
בפעם השניה 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* לשלוח.
ברגע שקודקוד קיבל תשובה מכל שכניו הוא מחזיר תשובה לאבא.

סיבוכיות זמן: $O\big(1+2+3.+\cdots+rad(G,v)\big)=O(rad(G,v)^2)$
בשלב *k* יקח *k* יחידות זמן לשליחת ההודעה מ*v* לחדשים ועוד *k* יחידות זמן בחזרה ל*v*.
מספר השלבים נקבע לפי הקודקוד הכי רחוק מ*v*.

סיבוכיות הודעות: $O($
בכל צלע יעברו 2 הודעות עבור הפעם הראשונה שנתקלים בצלע זו.
נניח שנתייעל ולא נשלח הודעות לצלע שקיבלנו *NACK* ממנה.
לכן ההודעות יעברו רק בצלעות העץ ולכן מכיוון שיש $O\big(diam(G)\big)$ שלבים ובכל שלב שולחים הודעות במקרה הגרוע בכל העץ נקבל:$O(diam(G)\cdot|E_T|)=O(diam(G)\cdot|V|)$
סה"כ: $O(|E|+|V|diam(G))$

2. בלמן פורד:
מטרה: למצוא עץ *BFS* במערכת א-סינכרונית.
הרעיון: לא להמתין אלא לעדכו ולעדכן עד שאין כמה מה לעדכן.
כל קודקוד*u* מאתחל את המרחק שלוd_u מ*v* להיות אינסוף כאשר *v* מעדכן את d_vלהיות 0.
   - עבור קודקוד *v*: שלח את המרחק שלך +1 לכל שכניך.
   - עבור שאר הקודקודים *u*, בקבלת *d* מקודקוד *w*:
     - אם $d<d_u$ אז :
       - עדכן את $d_u=d$
       - עדכן את האבא להיות *w*
       - שלח את d_u+1 לכל השכנים פרט ל*w*.
כאשר מסתיימות כל ההודעות ברשת האלגו' עוצר.

סיבוכיות זמן: $O\big(rad(G,v)\big)=O\big(diam(v)\big)$
לאחר *K* יחידות זמן קודקוד במרחק *k* יקבל את העדכון הסופי שלו למרחק האמיתי מ*v*.

סיבוכיות הודעות: $O(|E|\cdot|V|)$
בכל צלע יכולים לעבור במקרה הגרוע *n-1* עדכונים.
המקרה הוא:
אחד מהשכנים של *v* מקבל את *1* ושולח *2* לשכנים שלו, 2 מספיק להגיע לפני ה1 לשאר השכנים והם מעדכנים ושולחים 3 ואז 4 וכ'ו..
לאחר מכן ההודעה עם 1 מגיעה לזה שקיבל את 2, הוא מעדכן ל1 ושולח 2 למי שהיה 3 וכך הלאה.
לכן במקרה הגרוע בכל צלע יעברו $O(|V|)$ עדכונים.

תרגילים:

מבחן 2018 א' א'

שאלה 3:

שאלה 3 (25 נקודות):

נתון גרף שלם על n הקודקודים $u_1, \dots, u_n$. לכל $1 \le i \le n$ הקלט של u_i הוא i. הקודקודים מתקשרים זה עם זה במערכת העברת הודעות אסינכרונית ללא תקלות. כל קודקוד מריץ את האלגוריתם הבא:

1. אתחל ערך עצמי לקלט שלך ושלח אותו לכל שכניך.
2. בכל פעם שמתקבלת הודעה עם ערך של שכן, בדוק אם הוא קטן ממש מערכך העצמי הנוכחי. אם כן, בצע את שתי הפעולות הבאות:
   א. עדכן את הערך העצמי שלך לזה שהתקבל מהשכן.
   ב. שלח את הערך העצמי החדש שלך לכל שכניך.

הציגו שתי הרצות (כלומר בחירת delays להודעות) כך שבראשונה מספר ההודעות הכולל שנשלח הוא $O(n^2)$ ובשניה $\Omega(n^3)$.

פתרון:

עבור $O(n^2)$ הודעות:

נבחר עיכוב של $\frac{1}{2}$ להודעה 1 ועיכוב של 1 לכל שאר ההודעות.

מכאן, לאחר $\frac{1}{2}$ יחידת זמן כל הקודקודים יעדכנו את ערכיהם ל1 ולכן כל שאר ההודעות שיגיעו לא יגרמו לעדכון נוסף ולכן גם לא תהיה שליחת הודעות נוספת.

סה"כ כמות ההודעות שתהיה שווה לכמות ההודעות שנשלחו בשלב 1 שהוא: $n(n-1)$ כלומר כל קודקוד שולח את הערך העצמי שלו לכל שכניו.

+ כמות ההודעות שכל קודקוד ישלח לאחר שקיבל את ההודעה 1 ועדכן את הערך שלו: $(n-1)(n-2)$

סה"כ : $O\big(n(n-1)+(n-1)(n-2)\big) = O(n^2)$

עבור $\Omega(n^3)$ הודעות:

נבחר עיכוב של $\frac{1}{n}$ להודעה ה*n* .

מכאן, לאחר $\frac{1}{n}$ יחידות זמן, כולם יקבלו הודעה מ*n* ולא יעדכנו כי הוא המספר הגדול ביותר.

לאחר $\frac{1}{n-1}$ יחידות זמן כולם יקבלו את ההודעה *n-1* וקודקוד u_n יעדכן וישלח $n-2$ הודעות לכל שכניו פרט לשולח.

לאחר $\frac{1}{i}$ יחידות זמן כולם יקבלו את ההודעה *i* והקודקודים $u_n, \dots u_{i+1}$ יעדכנו ל*i* ויישלחו הודעות לכל שכניהם פרט לשולח. סה"כ: $(n-i)(n-2)$

כלומר מכיוון שההודעות מגיעות בסדר מהספר הגדול לקטן כל קודקוד *i* יעדכן את ערכו *i-1* פעמים ובכל פעם ישלח *n-2* הודעות.

סה"כ כמות ההודעות שתהיה :

$$n(n-1) + \sum_{i=1}^{n-1}(i-1)(n-2) = n(n-1) + (n-2)\cdot\Omega(n^2) = \Omega(n^3)$$

מבחן 2019 א א

שאלה 3:

שאלה 3 (25 נקודות):

נתון מסלול על n הקודקודים $u_1, \dots, u_n$ (כלומר קשתות הגרף הן $u_i u_{i+1}$ לכל $1 \le i < n$). לכל $1 \le i \le n$ הקלט של u_i הוא 0. הקודקודים מתקשרים זה עם זה במערכת העברת הודעות אסינכרונית ללא תקלות ועם FIFO (כלומר, לכל $1 \le i \ne j \le n$, אם u_i שולח שתי הודעות ל-u_j, אז הראשונה בהכרח מגיעה לפני השנייה). כל קודקוד מריץ את האלגוריתם הבא:

1. אתחל ערך עצמי לקלט שלך ושלח אותו לשכן מימינך (כלומר, u_i שולח הודעה ל-u_{i+1} לכל $1 \le i \le n-1$ ו-u_n אינו שולח הודעות כלל).
2. בכל פעם שמתקבלת הודעה עם ערך של שכן, בדוק אם הוא זהה לערכך העצמי הנוכחי. אם כן, בצע את שתי הפעולות הבאות:
   א. הפוך את הערך העצמי שלך (כלומר, אם הוא 0 שנה אותו ל-1 ואם הוא 1 שנה אותו ל-0).
   ב. שלח את הערך העצמי החדש שלך לשכן מימינך.

א. (10 נקודות) הוכיחו שכאשר האלגוריתם עוצר, הערך של u_i שונה מהערך של u_{i+1} לכל $1 \le i \le n-1$.

ב. (10 נקודות) הוכיחו שבכל הרצה של האלגוריתם הנ"ל u_i הופך את ערכו לכל היותר $i-1$ פעמים לכל $1 \le i \le n$.

ג. (5 נקודות) הוכיחו שבכל הרצה של האלגוריתם הנ"ל מספר ההודעות הכולל שנשלח הוא $O(n^2)$.

פתרון:

נוכיח תחילה את הטענה הבאה: לכל $1 \le i \le n$ קודקוד *i* החליף את ערכו בדיוק *i-1* פעמים.

באינדוקציה על *i*:

עבור *i=1* קודקוד 1 מתחיל עם ערך 0 ולא מקבל שום הודעה כי אין קודקוד משמאלו ולכן מחליף את ערכו *0* פעמים. (*1-1*).
צעד: נניח שהטענה נכונה עבור *i* כלשהו ונוכיח שקודקוד *i+1* מחליף את ערכו בדיוק *i* פעמים.
לפי ההנחה, *i* החליף את ערכו *i-1* פעמים ולפי האלגוריתם, הוא שלח לקודקוד *i+1* את ההודעה הראשונה + הודעה כל כל עדכון. סה"כ: *i= i-1+1* הודעות בסדר *0,1,0,1....*
לפי *FIFO*: מכיוון שכולם מתחילים בערך עצמי 0, ההודעות יגיעו ל*i+1* בסדר הזה וכל הודעה תשנה את ערכו.
סה"כ קיבלנו שקודקוד *i+1* משנה את ערכו בכל הודעה שיקבל ולכן הוא ישנה את ערכו בדיוק *i* פעמים.

א. לפי הטענה, u_i שינה את ערכו *i-1* פעמים. u_{i+1} שינה את ערכו *i* פעמים.
מכיוון שאלו מספרים עוקבים, אחד מהם זוג ואחד מהם איזוגי. לכן מכיוון שההחלפה היא בין 0 ל1 לסיגורין נקבל שהערך בסיום האלגוריתם של אחד מהם הוא 0 ושל השני הוא 1.

ב. לפי הטענה, u_i הופך את ערכו בדיוק *i-1* פעמים ולכן זה גם לכל היותר *i-1* פעמים.

ג. לפי הטענה, כל קודקוד *i* הפך את ערכו בדיוק *i-1* פעמים ולכן שלח בדיוק *i* הודעות.

פרט לקודקוד u_n שלא שלח כי אין שכן מימינו.
סה"כ כמות ההודעות שנשלחה בכל הרצה :

$$\sum_{i=1}^{n-1} i = O(n^2)$$

תרגיל: 2017 א א

שאלה 3:

שאלה 3 (25 נקודות):

n קודקודים $u_0,\dots,u_{n-1}$ מסודרים במעגל (לכל $0 \le i \le n-1$ שכני u_i במעגל הם u_{i-1} ו- u_{i+1} כאשר החיבור והחיסור הם מודולו n). לכל $0 \le i \le n-1$ הקלט של u_i הוא i. הקודקודים מתקשרים זה עם זה במערכת העברת הודעות אסינכרונית ללא תקלות. כל קודקוד מריץ את האלגוריתם הבא:

1. אתחל ערך עצמי לקלט שלך ושלח אותו לשני שכניך.
2. בכל פעם שמתקבלת הודעה עם ערך של שכן, בדוק אם הוא קטן ממש מערכך העצמי הנוכחי. אם כן, בצע את שתי הפעולות הבאות:
   א. עדכן את הערך העצמי שלך לזה שהתקבל מהשכן.
   ב. שלח את הערך העצמי החדש שלך לשני שכניך.

הציגו שתי הרצות (כלומר בחירת delays להודעות) כך שבראשונה מספר ההודעות הכולל שנשלח הוא $\theta(n)$ ובשניה $\theta(n^2)$.

פתרון:

עבור $\Theta(n)$ הודעות:

נבחר את העיכוב של ההודעות עם 0 להיות $\frac{1}{n}$.

ונבחר את העיכוב של שאר ההודעות להיות 1.

מכאן, ההודעה 0 תגיע לשני השכנים של u_0 לפני ששאר ההודעות יגיעו. הם יעדכנו ל0 וישלחו לשני שכניהם (אחד מהם הוא u_0 שלא יעדכן כי ערכו כבר = 0).

השכנים במרחק 2 מu_0 יקבלו את ההודעה 0 (לאחר $\frac{2}{n}$ יחידות זמן), יעדכנו ל0 ויישלחו.

...

השכנים במרחק $\frac{n}{2}$ (מעוגל למטה) מu_0 יקבלו את ההודעה 0 (לאחר $\frac{1}{2}$ יחידות זמן), יעדכנו ל0 ויישלחו.

כעת, הערך של כולם הוא 0 ולכן לא יתבצעו יותר עדכונים וגם לא שליחת הודעות.

רק לאחר מכן, יגיעו שאר ההודעות ולא יעדכנו את הערכים.

סה"כ כמות ההודעות: $2n$ הודעות לפי שלב 2 באלגוריתם

+ $2(n-1)$ הודעות עבור כל קודקוד שעדכן ל0.

לכן ישלחו בדיוק $2n + 2(n-1) = \Theta(n)$ הודעות.

עבור $\Theta(n^2)$:

נבחר להודעה i עיכוב של : $\frac{1}{2^i}$

מכאן: נשים לב שכל שההודעה מכילה מספר גדול יותר היא תגיע לפני הודעה עם מספר קטן יותר ובנוסף הודעה עם מספר i תתעכב יותר מהזמן של ההודעות $i+1, i+2, \dots n-1$ ביחד.

כי: $\frac{1}{2^i} = \frac{\frac{1}{2^{i+1}}}{1-\frac{1}{2}} > \frac{1}{2^{i+1}} + \frac{1}{2^{i+2}} + \dots + \frac{1}{2^n}$ לפי סכום סדרה הנדסים אינסופית.

מכאן לפי העיכובים הנ"ל כל קודקוד u_i יעודכן i פעמים ובכל פעם ישלח 2 הודעות. (לא יעדכנו אף אחד כיוון שסדר הגעת ההודעות הוא מהגדול לקטן).

סה"כ ההודעות : $2n + \Sigma_{i=0}^{n-1} 2i = 2n + \Theta(n^2) = \Theta(n^2)$

כי כל קודקוד שולח בהתחלה 2 הודעות ועל כל עדכון עוד 2 הודעות.

מרתון 3 –

זמנים לוגיים

תזכורת: לוח זמנים, סדרה של האירועים שהתרחשו בהרצה בדרך כלל אירועי שליחה וקבלה של הודעות.

$$S = s_{12}, r_{21}, s_{12}, s_{31}, r_{31}, r_{13}$$

המטרה: לתת חתימת זמן לכל אירוע כל שהזמן ייצג את הסדר הכרונולוגי של האירועים. (מה היה לפני ומה היה אחרי).

נשתמש באלגוריתמים לשעונים לוגיים: מה היה קודם לפי ההיגיון ולא לפי הזמן האמיתי.

ערבול סיבתי: קזואל שאפל

שני לוחות זמנים S, S' יקראו עירבול סיבתי אם אחד של השני אם $S|v = S'|v$ לכל v. (ז"א לכל קודקוד הכל יראה זהה).

במקרה כזה לא נוכל להבחין בין לוחות הזמנים ולכם לא נוכל לקבוע מה קרה בוודאות לפני מה.

דוגמה: עבור $S = s_{12}, r_{21}, s_{12}, s_{31}, r_{31}, r_{13}$ הציגו ערבוב סיבתי שלו:

אסור להפוך בין שליחה וקבלה של אותה הודעה ואסור בין כל 2 אירועים שקרו באותו מחשב.

וכמובן יש גם טרנזיטיביות.

$$S' = s_{31}, s_{12}, r_{21}, s_{31}, r_{31}, r_{13}$$

הגדרה: אירוע e קרה בוודאות לפני e' בS אם ורק אם הוא חייב להופיע לפניו בכל ערבוב סיבתי של S.

פורמלית: היחס "קרה לפני" מוגדר באופן הבא : $e \rightarrow_S e'$. כלומר : אירוע e קרה בוודאות לפני e' אם:

1. e אירוע שליחה וe' קבלה.
2. e, e' קרו באותו מחשב והזמן של e קטן יותר.
3. קיים אירוע e'' כל ש: $e \rightarrow_S e''$ וגם $e'' \rightarrow_S e'$ e

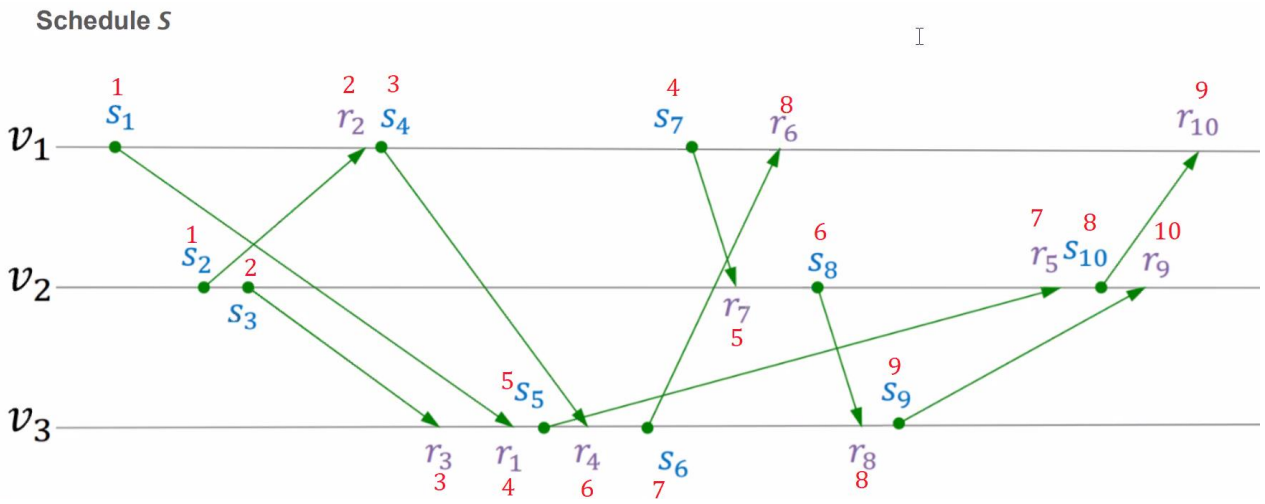
אלגוריתמים לחתימת זמן:

1. **שעוני למפורט:**

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 את $c_v = \max(c_v, c_u) + 1$
- וקבע את $\tau(e) = c_v$.

דוגמה:



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

אם e קרה בוודאות לפני e' אז זה מחייב ש: $\tau(e) < \tau(e')$.

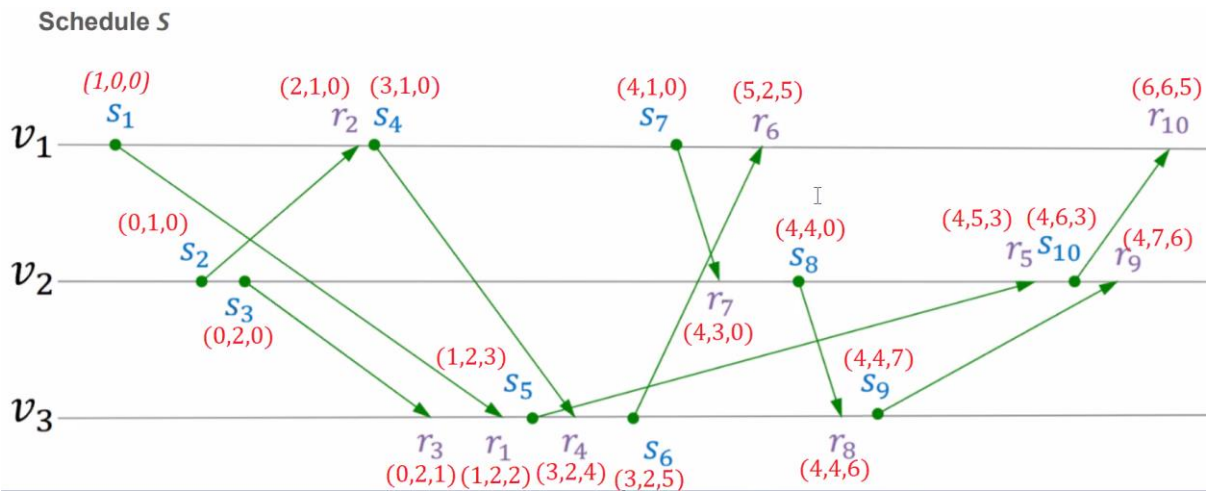
אם $\tau(e) < \tau(e')$ אז זה לא מחייב ש e קרה בוודאות לפני e'.

כדי לפתור את הבעיה משתמשים באלגוריתם $Vector\ Clock$.

2. אלגוריתם $Vector\ Clock$

- אתחל וקטור: $VC(v) = (0,0,0,0 \dots)$. (הווקטור מאותחל עם כמות המחשבים שיש לי)
- עבור כל אירוע e שאינו קבלה, קבע: $VC_v(v) + +$ וקבע: $VC(e) = VC(C)$.
- עבור כל אירוע שליחה, שלח בנוסף גם את הווקטור.
- עבור כל אירוע קבלה e מקודקוד u:
  - קבע $VC_w(v) = \max(VC_w(u), VC_w(v))$. לכל $w \neq v$.
  - קבע: $VC_v(v) + +$.
  - $VC(e) = VC(u)$.

- $VC_v(v)$ זה אומר ווקטור – במיקום הv שלו – על קודקוד v.



איך יודעים מי קרה לפני מי?

טענה: $e \rightarrow_s e'$ אם ורק אם $VC_w(e) \leq VC_w(e')$ לכל w וקיימת לפחות קורדינטה אחת בה זה קטן ממש.

אם יש קור' שאחד מהם גדול יותר וקור' שהשני גדול יותר אז הם קרו במקביל וניתן להחליף בינהם.

תרגילים:

2020 א' א'

שאלה 3 (25 נקודות):

יהי S לוח זמנים (schedule) המכיל אך ורק מאורעות של שליחה וקבלה של הודעות. נריץ עליו את אלגוריתם ה-Vector clocks (האלגוריתם מופיע בסוף המבחן).

א. (15 נקודות) הוכיחו כי בכל שלב במהלך ריצת האלגוריתם, לכל שני קודקודים u ו-v מתקיים $VC_u(u) \geq VC_u(v)$.

ב. (10 נקודות) הוכיחו כי $VC(e) \neq VC(e')$ לכל שני מאורעות שונים $e, e' \in S$.

פתרון:

א. נוכיח באינדוקציה על מספר השלבים (האירועים שקרו)
בסיס: לאחר 0 אירועים כל ווקטור מאותחל ל0-ים ולכן $VC_u(u) = VC_u(v)$
צעד: נניח שהטענה נכונה לאחר $n-1$ אירועים במערכת. נוכיח שהטענה נכונה עבור אירוע n.
נחלק למקרים:

1. אם האירוע התחרש בקודקוד u .
אז לפי האלגו' בשלב n. $VC_u(u)^n = VC_u(u)^{n-1}$. v לא עושה כלום ולכן לפי הנחת האינדוקציה:
$$VC_u(u)^n = VC_u(u)^{n-1} + 1 \geq VC_u(v)^{n-1} + 1 = VC_u(v)^n + 1 \geq VC_u(v)^n$$
2. אם האירוע התחרש בקודקוד $v \neq u$.
אם זה אירוע שליחה אז לפי האלגוריתם, קודקוד v לא מעדכן את $VC_u(v)$ ולכן לפי ההנחה:
$$VC_u(v)^n = VC_u(v)^{n-1} \leq VC_u(u)^{n-1} = VC_u(u)^n$$
אם זה אירוע קבלה מקודקוד w אז לפי האלגוריתם לפי האלגוריתם:
$$VC_u(v)^n = \max(VC_u(v)^{n-1}, VC_u(w)^{n-1})$$
לפי ההנחה שניהם קטנים יותר לכן גם המקסימלי :
$$VC_u(v)^n = \max(VC_u(v)^{n-1}, VC_u(w)^{n-1}) \leq VC_u(u)^{n-1}$$

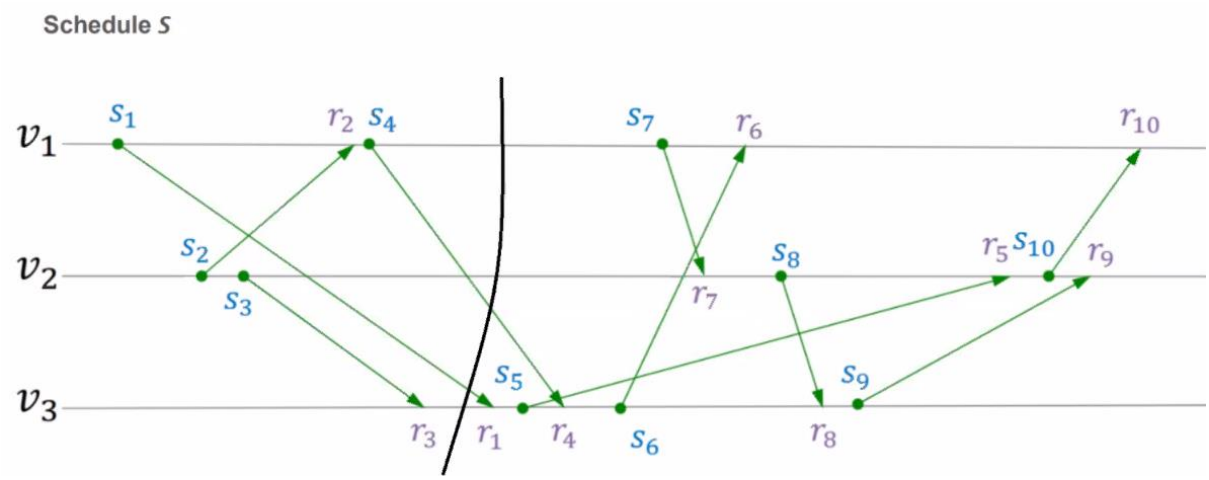
ב. נניח בשלילה שקיימים e, e' מאורעות כך ש$VC(e) = VC(e')$.
נחלק למקרים:

1. אם e, e' התחרשו באותו קודקוד אז מכיוון שלפי האלגוריתם, כל אירוע מקדם את המונה של הקודקוד אזי בהכרח : $VC_u(e) \neq VC_u(e')$. – סתירה.
2. אם התחרשו בקודקוד שונים u, v. נניח שe התרחש בv וגם e' לא קרה אחרי e. לפי ס' א, לפני אירוע e מתקיים:
$VC_v(v) \geq VC_v(u)$.
ומכאן, לאחר ביצוע e מתקיים : $VC_v(v) > VC_v(u)$.
אבל לאחר e' היה $VC(e) = VC(e')$ בפרט זה אומר : $VC_v(v) = VC_v(u)$ – אבל לאחר e זה לא מתקיים.

חתכים:

בהיתן לו"ז S. חתך C בS 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 $e' \in C$.

לדוגמה:



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

בדוגמה כאן ניתן גם לקחת רק את r_3 לבד כי הוא ההתחלה של v_3 וביתר הקוד' לא קרה כלום.

זה יהיה בניגוד לעקבי.

חתך עקבי:

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

משפט: חתך הוא עקבי אם ורק אם לכל אירוע קבלה $r_M \in C$ מתקיים ש$s_M \in C$. כל אירוע קבלה חייב שיהיה לו את האירוע שליחה אבל לא להפך.

האם הדוגמה היא חתך עקבי? – כן .

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 חתך.

חישוב תמונת מצב עקבית

חישוב חתך עקבי:

<u>השיטה 1 – להשתמש באלגוריתם שעוני למפורט.</u>

ואז בוחרים זמן t ולוקחים את קבוצה C להיות כל האירועים שהזמן שניתן להם קטן או שווה לt.

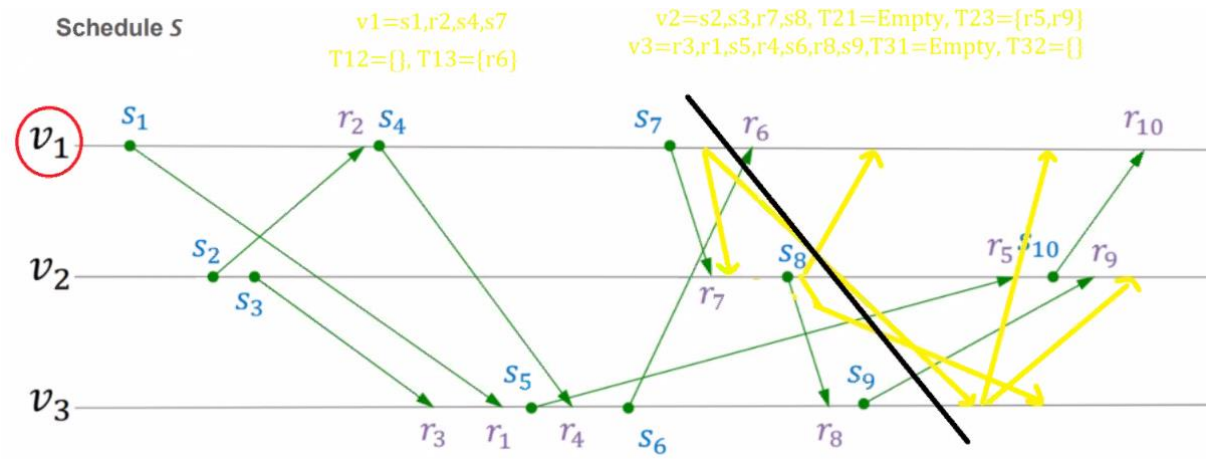
הקבוצה הזו היא בהכרח חתך עקבי.

הוכחה: יהא $e \in C$ ויהא $e' \rightarrow_S e$ אז לפי נכונות שעושני למפורט: $\tau(e') < \tau(e) \leq t$ ולכן $e' \in C$.

<u>השיטה 2 – (יותר יעילה) אלגוריתם קנדי-למפורט.</u>

u רוצה תמונת מצב של כל המערכת. יש הנחה בשיטה זו שיש *FIFO*.

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



קונצנזוס

בעיית הגנרלים היא סוג אחד של קונצנזוס.

קונצנזוס זו בעיית הסכמה. מספר מחשבים רוצים להסכים על אותו פרט.

המודל: זיכרון משותף א-סינכרוני. ישנם תאי זיכרון שנקראים רג'יסטרים 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 הפלט 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.

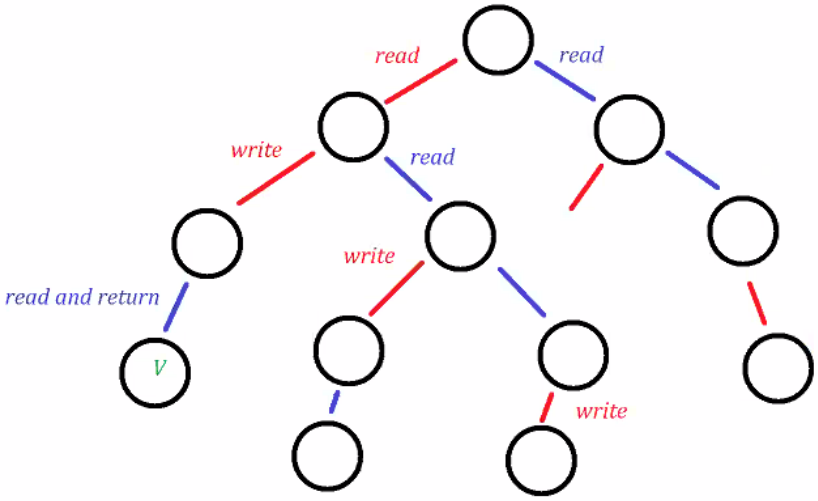
אנו מחפשים אלגוריתם *wait – free* : לא מחכים לאף אחד.

דוגמה לאלגו' שלא טוב:

משאב משותף: תא זיכרון C. מאותחל ל ?

- $r = read\ (C)$
- אם $r = ?$ אז $write(C, x)$ והחזר x.
- אחרת החזר r.

עץ הרצה לדוגמה הנ"ל:



עץ הרצה – עץ שיש בכל מסלול שלו חישוב מסונכרן אחרת.

<u>הגדרות של סוגי קודקודים בעץ ההרצה:</u>

1. מצב התחלתי - שורש העץ.
2. מצב סופי - פלט - עלים של העץ.
3. קודקוד יוניוולנטי – קודקוד שכל המסלולים החל ממנו ולמטה יובילו לאותו פלט. לדוגמה 0.
4. קודקוד ביוולנטי – קודקוד שיש מסלולים תחתיות שיובילו לפלט 0 ויש מסלולים שיובילו לפלט 1.
5. קודקוד קריטי – קודקוד ביוולנטי ששני בניו הם יוניוולנטים.

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

- דרך ההוכחה: 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 שמותרות 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.

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 ל0 ויש מסלול ל1 ולכן ביוולנטי.

יהא הקודקוד הביוולנטי ברמה הגבוהה ביותר – 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 אפשריות:

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*

בכל אחד מקומבינציית המקרים נראה שני מסלולים בלתי ניתנים להבחנה מ2 צידי הקודקוד הביוולנטי וזה יוביל לסתירה כי אז 2 המסלולים יובילו לאותו פלט אחיד (כי מקיימים הסכמה) וסתירה לכך שזה קודקוד קריטי ובפרט ביוולנטי.

אם באחד הצדדים יש כתיבה של קודקוד 1 לתא *A* אז אם הכתיבה תתבצע לאחר הפעולה של קודקוד 2 ואז קודקוד 1 יסיים לבד או שפעולות קודקוד 2 בכלל לא תתבצע כעת ורק 1 יסיים לבדו נקבל 2 מסלולים בלתי ניתנים להבחנה ולכן הפלט של קודקוד 1 יהיה זהה ולכן גם זהה לקודקוד 2 בגלל ההסכמה.

קיבלנו שיש מ2 הצדדים של קודקוד קריטי את הפלט וזו סתירה.

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

הסכמה כאשר הפעולה המותרת היא פעולת קריאה\כתיבה אטומית. (הכל ביחד בפעולה אחת)

במקרה כזה יש פתרון:

משאב משותף: תא זיכרון C. מאותחל ל ?

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

קיימים מספר סוגים של פונקציות RMW:

1. $RMW(C, x)$
   a. $old = read(C)$
   b. $write(C, f(x))$
   c. $return\ old$
2. $Test\ and\ Set$:
   a. $old = read(C)$
   b. $write(C, 1)$
   c. $return\ old$
3. $Fetch\ and\ inc$
   a. $old = read(C)$
   b. $write(C, old + 1)$
   c. $return\ old$
4. $Fetch\ and\ add(x)$
   a. $old = read(C)$
   b. $write(C, old + x)$
   c. $return\ old$
5. $Compare\ and\ Swap(x, y)$
   a. $old = read(C)$
   b. $if\ old == x \rightarrow write(C, y)$
   c. $return\ old$

מספר קונצנזוס: המספר n הגדול ביותר עבור קבוצת פונקציות A כך שבעזרת הפונקציות הללו ניתן לספק קונצנזוס (הסכמה) לn תהליכים.

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

$A = \{RMW\}$ אז $n = 2$.

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

משפט: תהי *F* קבוצה של פונקציות כך ש: (לכל i, j)

1. $f_i\left(f_j(x)\right) = f_j\left(f_i(x)\right)$ קומוטטיביות.

   או

2. $f_i\left(f_j(x)\right) = f_i(x)$ ניתנות לדריסה.

אז המספר הקונצנזוס שלהן לא יכול לעלות על 2.

תרגילים:

מבחן 2019 א א

שאלה 4 (25 נקודות):

הציגו אלגוריתם wait-free הפותר את בעית הקונצנזוס הבאה עבור **שלושה** קודקודים ע"י שימוש במספר כלשהו של רגיסטרים אטומיים לקריאה/כתיבה (atomic read/write registers). כל הרגיסטרים מאותחלים לערך 1-. התקלות המותרות היחידות הן קריסת קודקוד (crash failure).
הוכיחו את נכונות האלגוריתם שלכם.
קלט ופלט: הקלט והפלט של כל קודקוד שייך לקבוצה $\{0,1,2\}$.
הסכמה חלשה: הערך המוחלט של ההפרש בין **כל** שני פלטים הוא 1 לכל היותר.
סיום: כל קודקוד תקין חייב להוציא פלט אחרי זמן סופי.
Validity: אם הקלטים של שלושת הקודקודים זהים, אז הפלט של כל קודקוד תקין צריך להיות זהה לקלט שלו.

פתרון: יהיו 3 רג'יסטרים: C_1, C_2, C_3

קודקוד i קלט x:

- $write(C_i, x)$
- $y = read(C_{i+1mod3})$
- $z = read\ (C_{i+2mod3})$
- אם $y = x\ or\ y = -1$ וגם $z = x\ or\ z = -1$ החזר x.
- אחרת החזר 1.

הוכחה: לפי הגדרת האלגוריתם, הוא *wait free* (לא תלויים בקודקודים אחרים)

סיום: האלגוריתם מסתיים לאחר זמן סופי.

תקפות: נניח ששלושת הקודקודים קיבלו את אותו קלט x. מכאן תוכן הריג'יסטרים יהיה *1-* או x ולכן התנאי בשורה הרביעית של האלגוריתם יתקיים תמיד ולכן כל אחד יפלוט את הקלט שלו.

הסכמה חלשה: אם כולם קיבלו את האותו הקלט הוכחנו בתקפות. אם לפחות אחד מהקלטים שונה.

נניח בשלילה שהאלגוריתם לא סיפק הסכמה חלשה, לכן מישהו פלט 0 ומישהו פלט 2. נניח בה"כ ש0 היה מהיר יותר וכתב את הערך שלו ל רג'יסטר לפני ש2 כתב.

לכן 2 יראה את הקלט של 0 שהוא שונה ממנו ולכן יחזיר 1 בסתירה לכך שהוא החזיר 2.

מרתון 4 –

קונצנזוס במודל שליחת הודעות סינכרוני (גרף שבו על מחשב הוא קודקוד ושולחים והדעות אחד לשני)

רוצים לקיים:

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

בבעיית 2 הגנרלים התקלה הייתה: איבוד הודעות.

כאן יש לנו 2 תקלות אחרות:

1. קריסת מחשב.
2. קודקודים ביזנטיים.

קריסת מחשב:

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

משפט: במערכת שליחת הודעות סינכרונית עם $n \geq f+2$ קודקודים (כי אם יש f תקולים אז שיישארו לאחר התקולים לפחות 2 קודקודים שתהיה בניהם הסכמה כי אחרת אין דרישה להסכמה כי אל יישארו קודקודים עובדים במערכת). כל אלגוריתם f – התאוששות דורש לפחות

$f+1$ סיבובים כדי להשיג הסכמה.

הסיבה (רעיון ההוכחה)

דוגמה לאלגוריתם: קלט x

1. בכל סבב שלח את הקלט לכולם
2. בקבלת קלט ממישהו אחר, בחר את המינימאלי וקבע אותו לקלט החדש שלך.

המטרה שכולם יסכימו על המינימאלי.

נניח שיש לנו לפחות $f+2$ קודקודים כאשר לעל היותר f מהם תקולים אז המקרה הגרוע הוא:

תיאור הקלט: יש $f+2$ קודקודים לפחות. קודקוד i קיבל את קלט i .

המינימאלי 0 הספיק לשלוח את הקלט שלו רק ל1 ואז קרס. כעת רק 1 יודע על המינימאלי.

בסיבוב 2: 1 שולח ל2 את המינימאלי (0) ואז קורס. כעת 2 יודע על המינימאלי (וכל השאר אחריו לא).

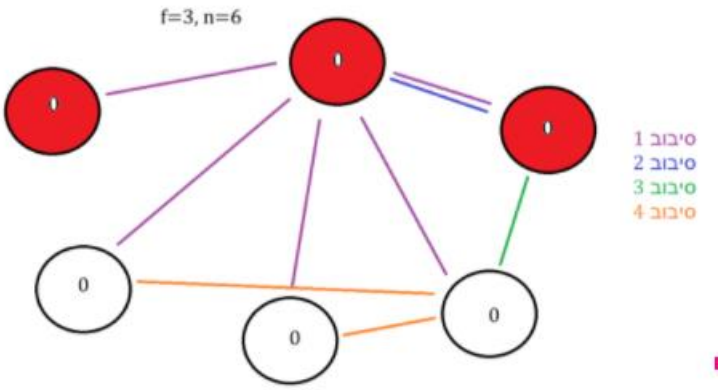
...

בסיבוב f : קודקוד f שולח את המינימאלי ל $f+1$ ואז קורס, ולכן $f+1$ יודע על המינימול אבל קודקוד

$f+2$ לא יודע (וגם הבאים אחריו אם יש...)

בסיבוב $f+1$: לפי ההנחה כבר לא יהיו תקולים ולכן $f+1$ לכולם את המינימום ובכך נגיע להסכמה.

המחשה של הרעיון:



<u>קודקודים ביזאנטיים במודל שליחת הודעות סינכרוני:</u>

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.

<u>רוצים לקיים:</u>

1. הסכמה: כל הפלטים הם אותו הדבר. עבוד הקודקודים הרגילים.
2. תקפות: אם כל הקלטים זהים אז על הפלטים זהים לקלט עבור רגילים.
3. 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 $m \geq \frac{n}{3}$

(n = כמות הקודקודים במערכת). כלומר אם לפחות שליש מכלל הקודקודים תקולים ביזנטית אז לא ניתן להגיע להסמכה בכל הקצה.

רעיון ההוכחה: מתחילים בהוכחה עבור 3 קודקודים כאשר 1 תקול ביזנטית.

נניח שהקודקודים הם A, B, C כאשר C תקול ביזנטית.

נניח בשלילה שיש אלגוריתם שמספק הסכמה, תקפות וסופיות.

נניח שהקלט של A, B הוא 0 לכן הפלט הוא 0 ללא תלות בC לפי התקפות.

נניח שהקלט של A, B הוא 1 לכן הפלט הוא 1 ללא תלות בC לפי התקפות.

מה קורה כאשר הקלט של A הוא 0 ושל B הוא 1?

אם C אומר ל A שהקלט שלו הוא 0, אז A חייב להחליט 0 כי זה לא ניתן להבחנה מהמקרה בו B תקול.

אם C אומר ל B שהקלט שלו הוא 1, אז B חייב להחליט 1 כי זה לא ניתן להבחנה מהמקרה בו A תקול.

יוצא שקיבלנו חוסר הסכמה כיוון שA פולט 0 וB פולט 1.

עד כאן הוכחנו למקרה פרטי ש$n = 3 , m = 1$ (3 קודקודים ו1 תקול).

במקרה הכללי של $m = \frac{n}{3}$ נבצע סוג של רדוקציה למקרה הפרטי.

ניקח את $\frac{n}{3}$ התקולים ונתייחס אליהם כקודקוד אחד (נגדיר הרצה בה כולם יתנהגו אותו דבר).

את התקינים נחלק ל2 קבוצות של $\frac{n}{3}$ כל אחד וגם שם נתייחס לכל קבוצה כאשר קודקוד אחד (כל חברי הקבוצה יקבלו את אותו הקלט ואת אותם ההודעות מהתקולים).

ולכן באותו אופן כמו במקרה הפרטי, נקבל חוסר הסכמה בין הקבוצות.

כאשר $m < \frac{n}{3}$ (כלומר פחות משליש תקולים) אז יש אלגוריתמים שכן פותרים את הבעיה.

האלגוריתם הפשוט:

מספר סיבובים - $m + 1$. (m מספר התקולים)

בסיבוב 1: כל קודקוד שולח לכל האחרים את הקלט שלו.

לאחר הסיבוב כל קודקוד מרכיב מערך של כל הקלטים שקיבל. (בתא הi הקלט ששלח לו הקודקוד הi)

בסיבוב 2: כל קודקוד שולח לכל האחרים את המערך קלטים שלו שקיבל בסיבוב 1.

לאחר הסיבוב כל קודקוד מרכיב מערך דו מימדי שבעמודה הi יש את כל הקלטים שכולם קיבלו על קודקוד i.

(בשורה הi יש את המערך מהסיבוב הראשון של קודקוד i)

בסיבוב 3: כל אחד שולח את מה שקיבל בויצר בסיבוב 2 ואז כל אחד מרכיב מערך תלת מימדי.

....

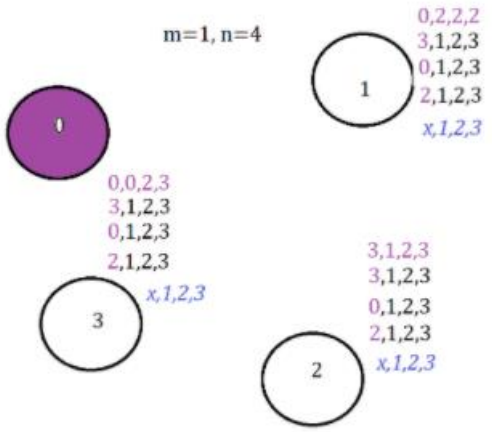
בסיבוב $m + 1$: לאחר הרכבת כל הקלטים. בוחרים מכל קודקוד את הקלט שהופיע אצלו רוב הפעמים.

מרכיבים מכאן מערך בגודל n שבתא בi יש את הקלט (לפי בחירת הרוב) של קודקוד i.

אם יש קודקוד i שהקלט שלו לא עיקבי (אין רוב) מתעלמים ממנו.

מתוך המערך הזה, בוחרים את האיבר שהופיע ברוב הפעמים והוא הפלט. אם אין איבר כזה – מחליטים על המינימום.

דוגמת הרצה: $m = 4\ n = 1$



האלגוריתם לוקח מינימום סיבובים ($m + 1$ הוא החסם התחתון על אלגוריתן עם m תקולים) אבל גודל כל הודעה גדל אקספוננציאלית מסיבוב לסיבוב – חיסרון גדול!

לכן יש אלגוריתמים יעילים יותר:

אלגוריתם המלכה:

מגיע להסכמה עבור m תקולים תוך $m+1$ סיבובים כאשר לכל סיבוב 2 חלקים.

ועובד רק כאשר $m < \frac{n}{4}$.

האלגוריתם: קלט x

בכל אחד מ$m+1$ הסיבובים בצע:

שלב 1:

- שלח לכולם את הקלט שלך.
- בקבלת הקלטים מכל האחרים בחר את הקלט שהופיע ברוב הפעמים. אם יש שיוון אז בחר את המינימום.
- אם הקלט הנבחר הופיע יותר מ $m+\frac{n}{2}$- תמוך בו. אחרת – אל תתמוך באף קלט. (אלא רק שמור את מה שבחרת).

שלב 2:

- 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 הסיבובים, 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 $n-m > \frac{n}{2}+m$ (מתקיים כי: $m < \frac{n}{4}$) ולכן כולם ייתמכו בקלט הזה (וכמובן ייבחרו בו) ולכן תהיה הסכמה ללא תלות במלכה בכלל.

הסכמה: הטענה היא שלאחר הסיבוב בו המלכה היא תקינה (ולא ביזנטית) תהיה הסמכה.

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 $m+\frac{n}{2}$ פעמים אז חלק יתמכו בו וחלק לא.

אבל בכל מקרה, כולם יקבלו את הקלט הזה (כולל המלכה) יותר מ$\frac{n}{2}$ פעמים ולכן ישנו את הקלט שלהם אליו.

בשלב 2 תשלח את אותו הדבר ולכן גם אם הם לא תמכו , הקלט לא ישתנה.

אלגוריתם המלך:

מגיע להסכמה עבור m תקולים תוך $m+1$ סיבובים כאשר לכל סיבוב 3 חלקים.

ועובד רק כאשר $m < \frac{n}{3}$.

האלגוריתם: קלט x

בכל אחד מ$m+1$ הסיבובים בצע:

שלב 1:

- שלח לכולם את הקלט שלך.

שלב2:

- בקבלת הקלטים מכל האחרים:
  - אם ערך x הופיע לפחות $n-m$ פעמים, שלח הודעה לכולם: "תמכו בx ".
- אם הודעה "תמכו בx" התקבלה יותר מ m פעמים, שנה את הערך לx.
- אם הודעה "תמכו בx" התקבלה לפחות $m-n$ פעמים, תמוך בx.

שלב 3:

- בוחרים מלך (לפי סבב של אינדקס הקודקודים או לפי בחירה אחרת)
- המלך שולח את הקלט שלו לכולם (הקלט = מה שבחרה בשלב 2)
- כל קודקוד שלא תומך בערך שלו משנה את ערכו לערך של המלך.

(יש דוגמה בסרטון והסבר)

נראה למה בהכרח תהיה הסכמה:

** להעתיק שוב

תקפות: אם כולם קיבלו את אותו קלט אז בכל סיבוב כולם שולחים את אותו הדבר ולכן כל קודקוד יקבל את הקלט הזהה לפחות n-m ולכן כולם יישלחו לתמוך בקלט הזה.
לכן גם כולם יקבלו הודעת: "תמוך בקלט" לפחות n-m פעמים ולכן כולם ייתמכו בו ולא ישנו לערך של המלך ללא תלות במלך.

הסכמה: הטענה היא שלאחר הסיבוב בו המלך הוא תקין (ולא ביזנטי) תהיה הסכמה.
אם אף קודקוד לא תמך בקלט אז המלך ישנה את כולם אליו. מכאן, המשך הסיבובים יהיה כמו בתקפות ולכן ללא תלות במלכים הבאים נקבל הסכמה.

אם יש מישהו שכן תומך בקלט x אז הוא בהכרח קיבל הודעה לתמוך ב x מלפחות n-m קודקודים (כולל עצמו) אבל אז גם ללא הביזנטיים, היו לפחות m<n/3<n-m-m=n-2m קודקודים אמיתיים ששלחו לתמוך ב x (*). כעת, כולם קיבלו הודעות אלו ולכן כולם (גם אם לא ייתמכו) ישנו את ערכם ל x (כולל המלך).
בשלב 3 המלך ישלח את אותו הדבר ולכן גם אם הם לא תמכו, הקלט לא יישתנה.

(*) אם קודקוד אמיתי שולח הודעה לתמוך בx אז אף קודקוד אמיתי אחר לא יישלח לתמול ב $y \neq x$.

כי אם מישהו שלח לתמוך ב x אז הוא קיבל את x מלפחות $n-m$ קודקודים.

ואם מישהו שלח לתמול ב y. אז הוא קיבל את y מלפחות $n-m$ קודקודים.

כאשר m הביזנטיים יכלו לשלוח x לראשון וy לשני.

סה"כ כמות הקודקודים השונים בגרף הוא לפחות: $n-m+n-m-m > n-\frac{n}{3}+n-\frac{n}{3}-\frac{n}{3}=n$ סתירה כי יש n קודקודים שונים בגרף.

שאלה 4 ממבחן 2020 א'

אלגוריתם המלך עבור n קודקודים כאשר אחד מהם ביזנטי:

- האלגוריתם מורכב משני שלבים כאשר כל שלב מורכב משלושה סיבובים.
- בכל שלב יש מלך אחר. שני המלכים נבחרו מראש וזהותם ידועה לכל הקודקודים.
- לכל קודקוד ערך עצמי המאותחל לקלט שלו.

כל אחד מהקודקודים מריץ את האלגוריתם הבא בשלב ה- i עבור $i=1,2$:

סיבוב 1:

- שדר את הערך העצמי שלך לכולם (גם לעצמך).

סיבוב 2:

- אם ערך כלשהו שודר לך בסיבוב 1 לפחות $n-1$ פעמים, הצע אותו לכולם (גם לעצמך).
- אם ערך כלשהו x הוצע לך בסיבוב זה לפחות פעמיים, עדכן את הערך העצמי שלך ל-x.

סיבוב 3:

- המלך משדר את הערך העצמי שלו לכולם.
- אם הערך העצמי הנוכחי שלך הוצע לך בסיבוב 2 לפחות $n-1$ פעמים, שמור עליו. אחרת אמץ את הערך ששלח המלך בסיבוב זה כערך העצמי שלך.

סיום: הפלט של קודקוד הוא הערך העצמי שלו.

שאלה 4 (25 נקודות):

נתבונן במערכת שליחת הודעות סינכרונית על גרף שלם עם n קודקודים כאשר בדיוק אחד מהם ביזנטי (Byzantine).

א. (12 נקודות) הוכיחו שאם מריצים את אלגוריתם המלך (האלגוריתם מופיע בסוף המבחן) עם $n=3$ והקלט של שני הקודקודים התקינים הוא 0 אז הם **מגיעים להסכמה**.

ב. (13 נקודות) תנו דוגמה להרצה של אלגוריתם המלך עם $n=3$ כך שהקלט של קודקוד תקין אחד הוא 0 והקלט של הקודקוד התקין השני הוא 1 והם **אינם מגיעים להסכמה**.

פתרון:

א. בשלב 1 של כל אחד מ2 הסיבובים, 2 הקודקודים התקינים ישלחו 0 ולכן גם הערך יגיע לכל אחד מהם לפחות פעמיים: $3-1$. ולכן שניהם יציעו את 0 לכולם. ובפרט שניהם יקבלו את הצעת 0 לפחות פעמיים ולכן ישמור עליו גם בשלב 3 ללא תלות במלך.
ולכן הערך 0 יהיה הפלט של שניהם בסיום האלגוריתם.

ב. דוגמת הרצה:
בכל אחד מהסיבובים נבצע את אותו הדבר:
שלב 1 :
קודקוד תקין 0 ישלח 0 לכולם. (כולל לעצמו)
קודקוד תקין 1 ישלח 1 לכולם. (כולל לעצמו)
הקודקוד הביזנטי ישלח 0 לקודקוד 0 וישלח 1 לקודקוד 1.
שלב 2:

כעת, קודקוד 0 קיבל פעמיים 0 ולכן ישלח לכולם, כולל לעצמו, הצעה ל0.
קודקוד 1 קיבל פעמיים 1 ולכן ישלח לכולם, כולל לעצמו, הצעה ל1.
הביזנטי ישלח ל0 הצעה ל0 וישלח ל1 הצעה ל1.
שלב 3:
0 קיבל 2 הצעות ל0 ולכן שומר על הערך הזה ללא תלות במלך.
1 קיבל 2 הצעות ל1 ולכן שומר על הערך הזה ללא תלות במלך.

סה"כ הפלט של 0 יהיה 0 ושל 1 יהיה 1 – אין הסכמה.

מבנה מבחן:

שאלה 1 – בעיית הגנרלים – וריאציה.

שאלה 2 – לוחות זמנים, חתכים, קוזאל שאפל.

שאלה 3 – מתעסקת אלגוריתמי תזמון (למפורט, *VC*) או אלגוריתמי שליחת הודעות מבוזרים. (*BFS*, אסינכרונים, *delay*)

שאלה 4 – קונצנזוס (תאי זיכרון, ביזאנטי ....)

מרתון אחרון

מבחן 2020 א ב

שאלה 1 (25 נקודות):

א. (8 נקודות) מצאו שני לוחות זמנים (schedules) שונים S_1 ו-S_2 למערכת עם שלושה קודקודים הרואים את המאורעות הבאים (כלומר השורה ה-i למטה היא ה-restricted view של הקודקוד ה-i):

1: s_{12}, s_{13}, r_{13}

2: $r_{21}, s_{23}, r_{23}, r_{23}$

3: $s_{32}, s_{31}, r_{32}, s_{32}, r_{31}$

ב. (8 נקודות) תנו דוגמה לחתך קונסיסטנטי ב-S_1 ולחתך לא קונסיסטנטי ב-S_2. הוכיחו שאכן החתך הראשון קונסיסטנטי והשני לא.

ג. (9 נקודות) הדגימו הרצה של אלגוריתם שעוני Lamport (האלגוריתם מופיע בסוף המבחן) על S_1.

ך

א. נעבוד בשיטה של כל עוד אני יכול להכניס אני מכניס! פשוט להכניס שליחה כמה שאני יכול ואם יש לי קבלה אני מחכה שהשליחה שלו תהיה קודם.

$S_1 = s12, s13, r21, s23, s32, s31, r32, s32, r31, r23, r23, r13$

$S_1 = s12, s13, r21, s23, s32, s31, r32, s32, r31, r23, r13, r23$

בכללי אפשר להפוך שני מאורעות שהם לא שליחה וקבלה והם בקודקודים שונים.

ב. חתך עקבי בS_1: $C_1 = \{s12\}$

חתך לא עקבי בS_2: $C_2 = \{r21\}$ (יש קבלה ואין שליחה)

שני הקבוצות הם חתכים כיוון שהראשונה היא המאורע הראשון בקודקוד 1 והשניה היא המאורע הראשון בקודקוד בקודקוד 2.

נשתמש במשפט: חתך הוא עקבי אם ורק אם לכל אירוע קבלה $r_M \in C$ מתקיים $s_M \in C$.

בקבוצה הראשונה אין אירוע קבלה ולכן זה נכון באופן ריק.

בשני יש את $r21$ אבל אין את $s12$ ולכן הוא לא עקבי.

ג. שלב ראשון: מאתחלים מונה $c_1c_2c_3 = 0$.

קודקוד 1 שולח ל2: $c_1 + +$ ומכאן השעון שמוצמד $\tau(s_{12}) = 1$.

קודקוד 1 שולח ל3: $c_1 + +$ ומכאן השעון שמוצמד $\tau(s_{13}) = 1$.

קודקוד 2 מקבל מ1: $c_2 = \max(c_2, \tau(s_{12})) + 1 = 2$ ומכאן $\tau(r_{21}) = 2$.

קודקוד 2 שולח ל3: $c_2 + +$ ומכאן השעון שמוצמד $\tau(s_{23}) = 3$.

קודקוד 3 שולח ל2: $c_3 + +$ ומכאן השעון שמוצמד $\tau(s_{32}) = 1$.

קודקוד 3 שולח ל1: $c_3 + +$ ומכאן השעון שמוצמד $\tau(s_{31}) = 2$.

קודקוד 3 מקבל מ2: $c_2 = \max(c_3, \tau(s_{23})) + 1 = 4$ ומכאן $\tau(r_{32}) = 4$.

.... (להשלים מהמרתון)

שאלה 2 (25 נקודות):

א. (13 נקודות) הוכיחו שאם C ו-C' חתכים קונסיסטנטים בלוח זמנים כלשהו, אז $C \cap C'$ חתך קונסיסטנטי באותו לוח זמנים.

ב. (12 נקודות) יהי A אלגוריתם wait-free המשיג קונצנזוס של שני תהליכים עם קלט ופלט בינאריים במערכת זכרון משותף ללא תקלות. יהי T עץ הרצות (execution tree) עבור A. הוכיחו שאם יש ב-T קודקוד ביוולנטי (bivalent) אז יש בו גם קודקוד קריטי.

א. צריך להוכיח שהוא חתך קודם ואז שהוא עקבי.
נוכיח תחילה ש$C \cap C'$ הוא חתך.
אם יש לי אירוע בחתך אז כל האירועים שלפניו **באותו קודקוד** יהיו גם בחתך.
יהי $e \in C \cap C'$ שהתחרש בקודקוד v ויהיה e' אירוע שקרה לפני e באותו קודקוד.
צ"ל: $e' \in C \cap C'$.
לפי ההנחה, $e \in C$ וגם $e \in C'$ ומכיוון שהם חתכים אזי: $e' \in C \; and \; e' \in C'$ ולכן: $e' \in C \cap C'$.
עכשיו נוכיח שהחתך עקבי, נשתמש במשפט: חתך הוא עקבי אם ורק אם לכל אירוע קבלה $r_M \in C$ מתקיים $s_M \in C$.
יהי $r_M \in C \cap C'$ מכאן: $r_M \in C \; and \; r_M \in C'$ ומכיוון שהם חתכים עקביים אזי: $s_M \in C \; and \; s_M \in C'$ ולכן: $s_M \in C \cap C'$.

ב. תזכורת:
עץ הרצה לשני תהליכים: עץ בינארי שבו כל מסלול מהשורש עד העלה הוא ביצוע של שני התהליכים בכל סדר ביצוע אפשרי עד הפלט. אם האלגוריתם משיג **הסכמה** אז הפלט בכל עלה הוא זהה.
קודקוד יוניוולנטי: קודקוד שכל המסלולים ממנו לכל העלים תחתיו מגיעים לפלט זהה.
קודקוד ביוולנטי: קודקוד שיש ממנו שני מסלולים שכל אחד מהם מגיע לפלט אחר.
קודקוד קריטי: קודקוד ביוולנטי ששני בניו הם יוניוולנטיים (כל אחד לפלט אחד).

הוכחה: נניח שיש בעץ קודקוד ביוולנטי. יהי b הקודקוד הביוולנטי העמוק ביותר בעץ (רמה הגבוהה ביותר..). קיים כזה כי האלגוריתם הוא סופי ולכן עץ ההרצות הוא סופי.
הוא אינו עלה כי בעלים יש הסכמה ולכן הם יוניוולנטיים.
מכיוון שהוא ברמה העמוקה ביותר אז שני הבנים שלו (קיימים כי הוא ביוולנטי אז יש ממנו לפחות 2 מסלולים) הם יוניוולנטיים ולכן הוא קודקוד קריטי.
דגש:
לקחת את הביוולנטי האחרון העמוק ביותר.
להדגיש למה קיים כזה – סופיות של האלגו' וההסכמה של העלים.

שאלה 3 (25 נקודות):

נתון מעגל על n הקודקודים $u_1, \dots, u_n$ (כלומר קשתות הגרף הן $u_i u_{i+1}$ לכל $1 \le i < n$ ו-$u_n u_1$).
לכל $1 \le i \le n$ הקלט של u_i הוא i. הקודקודים מתקשרים זה עם זה במערכת העברת הודעות אסינכרונית ללא תקלות. כל קודקוד מריץ את האלגוריתם הבא:

1. אתחל ערך עצמי לקלט שלך ושלח אותו לכל שכניך.
2. בכל פעם שמתקבלת הודעה עם ערך של שכן, בדוק אם הוא גדול ממש מערכך העצמי הנוכחי. אם כן, בצע את שתי הפעולות הבאות:
   א. עדכן את הערך העצמי שלך לזה שהתקבל מהשכן.
   ב. שלח את הערך העצמי החדש שלך לכל שכניך.

הציגו שתי הרצות (כלומר בחירת delays להודעות) כך שבראשונה מספר ההודעות הכולל שנשלח הוא $O(n)$ ובשניה $\Omega(n^2)$. נמקו את תשובתכם.

פתרון:
הרצה עם $O(n)$ הודעות:
כל הודעה עם ערך עצמי n תשלח בעיכוב של $\frac{1}{n}$.
כל הודעה אחרת תתעכב ב1.
נראה שאכן בהרצה זו יהיו $O(n)$ הודעות.
בשלב הראשוני, על קודקוד שולח את הערך שלו ל2 שכניו, סה"כ: $2n$ הודעות.
כל ההודעות פרט ל2 שנשלחו מ*n* מתעכבות ב1.
לאחר $\frac{1}{n}$ יחידות זמן, קודקודים *1,n-1* מקבלים את ההודעה מעדכנים את הערך העצמי שלהם ל*n* ושולחים את *n* לשני שכניהם.
אחד מהשכנים הוא *n* עצמו ולכן הוא לא עושה דבר בקבלת ההודעה.
לאחר $\frac{2}{n}$ יחידות זמן מתחילת האלגוריתם, קודקודים *2,n-2* מקבלים את ההודעה מעדכנים את הערך העצמי שלהם ל*n* ושולחים את *n* לשני שכניהם.
וככה ממשיכים .....
לאחר $\frac{i}{n}$ יחידות זמן יחידות זמן מתחילת האלגוריתם, קודקודים *i,n-i* מקבלים את ההודעה מעדכנים את הערך העצמי שלהם ל*n* ושולחים את *n* לשני שכניהם.
לכל היותר, לאחר $\frac{n}{2}$ צעדים, הערך העצמי של כל הקודקודים הוא *n*. בכל צעד נשלחו 4 הודעות לכל היותר.
לאחר מכן, לא ישלחו יותר הודעות כי כולם כבר בערך המקסימלי.
לפי בחירת העיכובים רק לאחר שכולם מעודכנים ל*n* יגיעו שאר ההודעות ולא יקרה דבר.
סה"כ: מספר ההודעות הכולל הוא לכל היותר : $2n + \frac{n}{2} \cdot 4 = 4n = O(n)$.

הרצה עם $\Omega(n^2)$ הודעות:
הודעה עם 1 תתעכב: 1
הודעה עם 2 תתעכב: n.
הודעה עם *3* תתעכב: n^2.

הודעת עם ערך x תתעכב: $\frac{n^{x-1}}{n^n}$.
הרעיון הוא לקחת עיכוב ככה שההודעה כי קטנה יכולה לעשות את כל הגרף עד שההודעה הבאה תשלח.
נראה שאכן בהרצה זו יהיו $\Omega(n^2)$ הודעות.
בשלב הראשוני, על קודקוד שולח את הערך שלו ל2 שכניו, סה"כ: $2n$ הודעות.
נשים לב שרק הודעות מ*i* ל*i-1* מעדכנות ערך.
לאחר מכן, נתבונן על קודקוד *i* :
קודקוד 1,3: יקבל את 2 לאחר $\frac{1}{2^{n-2}}$ יחידות זמן ויישלח 2 הודעות. (שלא ישנו)
קודקוד *2,4*: יקבל את *3* לאחר $\frac{1}{2^{n-3}}$ יחידות זמן ויישלחו 2 הודעות
קודקוד 1: יקבל את 3 לאחר $\frac{1}{2^{n-4}} = \frac{2}{2^{n-3}}$ יחידות זמן מתחילת האלגוריתם. יעדכן וישלח 2 הודעות שלא יעשו כלום.
באותו זמן, 3 יקבל את ההודעה 4 מקודקוד 4 וישלח ל2 שכניו.

נשים לב שלפי בחירת העיכובים, הודעה עם ערך קטן תספיק לעבור בכל הגרף לפני שהודעה עם ערך גדול תגיע למישהו. כי: $\frac{n^{x-1}}{n^n} \cdot n = \frac{n^x}{n^n}$
לכם, כל קודקוד *i* יעדכן את ערכו *n-i* פעמים.
הוכחה: ההודעה עם 2 תגיע ל1 ותעדכן אותו.
לאחר מכן הודעה עם 3 תגיע ל2 תעדכן ואת הוא יישלח ל1 ויעדכן
לאחר מכן ההודעה עם 4 מעדכן את 3 *u*אז את 2 ואז את 1...
...
בשלב ה*i* ההודעה עם *i+1* יעדכן..... *i-1,....1* בעצם *i-1* עדכונים.
כל עדכון הוא שליחת עוד 2 הודעות סה"כ מספר ההודעות הוא לפחות:
$2n + \Sigma_{i=2}^{n} 2(i-1) = 2n + 2 \cdot \frac{n(n-1)}{2} = \Omega(n^2)$.

שאלה 4 (25 נקודות):

הציגו אלגוריתם wait-free הפותר את בעית הקונצנזוס הבאה עבור **שלושה** קודקודים ע"י שימוש במספר כלשהו של רגיסטרים אטומיים לקריאה/כתיבה (atomic read/write registers). כל הרגיסטרים מאותחלים לערך 1-. הניחו כי המערכת פועלת ללא תקלות. הוכיחו את נכונות האלגוריתם שלכם.

קלט ופלט: הקלט והפלט של כל קודקוד שייך לקבוצה $\{1,2,\dots,100\}$.

הסכמה חלשה: אסור שכל שלושת הפלטים יהיו שונים זה מזה (כלומר צריך רוב לאחד הפלטים).

סיום: כל קודקוד חייב להוציא פלט אחרי זמן סופי.

Validity: אם הקלטים של שלושת הקודקודים זהים, אז הפלט של כל קודקוד צריך להיות זהה לקלט שלו.

פתרון:

האלגוריתם ישתמש ב3 רגיסטרים $R_1R_2R_3$.

אלגוריתם קודקוד i : קלט x_i:

- כתוב את x_i לתוך R_i .
- קרא את $R_{(i+1)mod3}$ לתוך y ואת $R_{(i+2)mod3}$ לתוך z.
- אם $z,y \in \{-1,x_i\}$ אז החזר x_i.
- אחרת החזר:100

הוכחת נכונות:

סיום: האלגורית הוא *wait free* (לא ממתינים לאף אירוע) ומסתיים לאחר זמן סופי.

פלט הוא תמיד בין 1 ל100 כי או שמחזירים את הקלט (שהוא בין 1 ל100) או שמחזירים את המקסימום מבין 2 מספרים שלפחות אחד מהם הוא בין 1 ל100 והשני יכול להיות -1.

תקפות: נניח שכל הקלטים היו x. נשים לב שמכיוון שהרגיסטרים מאותחלים ב-1 וכותבים את הקלט שהוא רק *x* אז כל הקודקודים יקראו מינוס 1 או *x* ולכן יחזירו כולם את הקלט של עצמם שהוא *x*.

הסכמה: אם כל הקלטים זהים הוכחנו בתקפות שתהיה הסכמה.

אחרת, יהא *i* הקודקוד הראשון שכתב את הקלט שלו. לכן 2 האחרים יקראו את הקלט של הראשון.

אם כל הקלטים שונים: אז ה2 האיטיים יותר יחזירו 100 כי הקלט של הראשון לא שווה לשלהם.

אם קיימים 2 קודקודים בעלי קלט זהה: יהא *i* הקודקוד האחרון מבין ה2 הנ"ל שכתב את הקלט שלו ולכן הוא רואה את הקלט של השני (עם הקלט הזהה).

אם הקודקוד בעל הקלט השונה כותב ראשון אז שני האחרים יחזירו 100 כי הם יקראו קלט שלא שווה אליהם.

אחרת, אם השלישי כותב לפני *i* אז גם השלישי וגם *i* יחזירו 100.

אחרת, *i* רואה רק מינוס 1 ואת הקלט של הראשון (שזהה אליו) ולכן גם הראשון וגם השני יחזירו את הקלט שלהם שהוא זהה.

מבחן 2020 א' א'

שאלה 1 (25 נקודות):

הוכיחו שבעיית "ארבעת הגנרלים" הבאה אינה פתירה על ידי אלגוריתם דטרמיניסטי. נתונים ארבעה קודקודים המתקשרים זה עם זה במערכת שליחת הודעות סינכרונית עם תקלות (כלומר יתכן שהודעה נשלחת אך לא מגיעה ליעדה). לכל קודקוד קלט בינארי (כלומר 0 או 1). המטרה היא לכתוב אלגוריתם המקיים את שלוש התכונות הבאות:

1. **הסכמה חלשה:** הפלט של כל קודקוד צריך להיות בינארי וצריך שיהיה רוב מוחלט לאחד הפלטים (כלומר אסור שבדיוק שני גנרלים יפלטו 0 ובדיוק שני גנרלים יפלטו 1).
2. **סיום:** לכל קלט, האלגוריתם צריך לסיים לרוץ ולהוציא פלט אחרי מספר סופי של צעדים.
3. **Validity:** אם הקלטים של ארבעת הקודקודים זהים וכל ההודעות מגיעות, אז הפלט של כל קודקוד צריך להיות זהה לקלט שלו.

עשינו במרתון.

שאלה 2 (25 נקודות):

נתונים שני לוחות הזמנים (schedules) הבאים:

$$S: \quad s_{32}, s_{12}, r_{21}, s_{43}, r_{34}, r_{23}, s_{24}, r_{42}, s_{41}, r_{14}$$

$$S': \quad s_{43}, s_{12}, r_{21}, s_{32}, r_{23}, s_{24}, r_{42}, s_{41}, r_{14}, r_{34}$$

א. (7 נקודות) הוכיחו או הפריכו: S' הוא causal shuffle של S.

ב. (10 נקודות) הוכיחו שחתך C הוא קונסיסטנטי אם ורק אם $r_M \in C \implies s_M \in C$ מתקיים לכל הודעה M בעלת מאורע שליחה s_M ומאורע קבלה r_M.

ג. (8 נקודות) האם $\{s_{12}\} \cup \{r_{21}, r_{23}, s_{24}\} \cup \{s_{32}\} \cup \{s_{43}, r_{42}\}$ מהווה חתך קונסיסטנטי? נמקו את תשובתכם.

א. נבדוק את כל נקודות המבט של קודקודי המערכת:

$$S|v_1 = s12, r14$$
$$S'|v_1 = s12, r14$$
$$S|v_2 = r21, r23, s24$$
$$S'|v_2 = r21, r23, s24$$
$$S|v_3 = s32, r34$$
$$S'|v_3 = s32, r34$$
$$S|v_4 = s43, r42, s41$$
$$S'|v_4 = s43, r42, s41$$

קיבלנו שלכל v מתקיים ש$S|v = S'|v$ ולכן S' הוא CS של S.

ב. הוכחה:

כיוון 1: נניח כי C חתך עקבי.

יהי $r_M \in C$ ומכאן: לפי הגדרת חתך עקבי, כל אירוע e שקרה לפני r_M שייך לC.

בפרט s_M קרה לפני r_M (כי אלו אירועי שליחה וקבלה) ולכן $s_M \in C$.

כיוון 2: נניח שלכל $r_M \in C$ מתקיים: $s_M \in C$. צ"ל:C הוא חתך עקבי.

יהי $e \in C$ ויהי e' אירוע כך ש: $e' \Longrightarrow_S e$ מקיים: $e' \in C$.

נוכיח באינדוקציה על מספר האירועים שקרו בין e' לe שכל אירוע כזה שייך לחתך:

בסיס: $k = 0$ אז e' באותו קודקוד אירוע אחד לפני ואז מהגדרת חתך: $e' \in C$.

או ש$e = r_M, e' = s_M$ ולפי ההנחה, $e' \in C$.

נניח שהטענה נכונה לכל אירוע e' כך שהיו בינו לבין e לכל היותר k אירועים ונוכיח שהטענה נכונה עבור e' כזה במרחק $k+1$ אירועים.

לפי ההנחה: קיים e'' שהוא אירוע שקרה אחרי e' במרחק 0 מ e' ולפני e במרחק k מe ולכן שייך לחתך לפי הנחת האינדוקציה. מכאן:e' באותו קודקוד של e'' אירוע אחד לפני ואז מהגדרת חתך: $e' \in C$.

או ש$e'' = r_M, e' = s_M$ ולפי ההנחה מהכיוון השני, $e' \in C$.

ג. תחילה נראה שזה חתך: נתבונן בנקודת המבט של כל הקודקודים:

$$S|v_1 = s12, r14$$
$$S|v_2 = r21, r23, s24$$
$$S|v_3 = s32, r34$$
$$S|v_4 = s43, r42, s41$$

ואכן האירועים שהם בחתך הם הרישא של כל קודקוד.

כעת, לכל אירוע קבלה בחתך הנ"ל, אירוע השליחה גם נמצא ולכן הוא עקבי:

(צריך סה"כ להראות את זה)

מבחן 2019 ב ב

שאלה 1 (25 נקודות):

הוכיחו שבעיית "ארבעת הגנרלים" הבאה אינה פתירה על ידי אלגוריתם דטרמיניסטי. נתונים ארבעה קודקודים המתקשרים זה עם זה במערכת שליחת הודעות סינכרונית עם תקלות (כלומר יתכן שהודעה נשלחת אך לא מגיעה ליעדה). לכל קודקוד קלט בינארי (כלומר 0 או 1). המטרה היא לכתוב אלגוריתם המקיים את שלוש התכונות הבאות:

1. **הסכמה חלשה:** הפלט של כל קודקוד צריך להיות בינארי וסכום כל ארבעת הפלטים חייב להיות מספר זוגי.
2. **סיום:** לכל קלט, האלגוריתם צריך לסיים לרוץ ולהוציא פלט אחרי מספר סופי של צעדים.
3. **Validity:** אם הקלטים של ארבעת הקודקודים זהים וכל ההודעות מגיעות, אז הפלט של כל קודקוד צריך להיות זהה לקלט שלו.

פתרון:
נניח בשלילה שקיים אלגוריתם דטרמיניסטי A המקיים את כל התכונות ומסיים תוך T סיבובים.
נגדיר את סדרת ההרצות הבאה:
E_0 קלט 0000 כל ההודעות הגיעו
E_1: קלט 0000 ההודעות בסיבוב הT לv_0 לא מגיעות.
E_2: קלט 0000 כמו E_1 ההודעות בסיבוב הT לא מגיעות גם לv_1 .
...
E_{4i+j} : קלט 0000 כמו E_{4i+j-1} כאשר בנוסך ההודעות בסיבוב הT-i לא מגיעות גם לv_j.
כאשר: $0 \le i \le T-1$, $0 \le j \le$.

E_{4T}: קלט *0000* אף הודעה לא מגיעה.
E_{4T+1}: : קלט *1100* אף הודעה לא מגיעה.
E_{4T+2}: : קלט *1110* אף הודעה לא מגיעה.
E_{4T+3}: : קלט *1111* אף הודעה לא מגיעה.
E_{4T+4}: : קלט *1111* אף הודעה לא מגיעה.

$E_{4T+4i+j}$: קלט 1111 כמו $E_{4T+3+4i+j-1}$ כאשר בנוסך ההודעות בסיבוב הi מגיעות גם לv_j.

בהרצה 0 מתקיים תקפות ולכן הפלט של כולם 0000
נשים לב שעבור כל 2 הרצות סמוכות יש רק קודקוד אחד שמבחין בין ההרצות ושאר 3 הקודקודים לא מבחינים ולכן הפלט שלהן יהיה זהה להרצה הקודמת שהוא 000 אבל מכאן לפי קיום הסכמה גם הרביעי חייב להוציא 0 כי אחרת הסכום יהיה אי זוגי ולכן הפלט הוא 0000.
מכאן, בהרצה האחרונה לפי הדמיון של ההרצות הקודמות נקבל שהפלט עדיין 000 בסתירה לתקפות.

שאלה 3:
אותו דבר כמו הקודמת.
עיכובים:
עבור $O(n)$:
נבחר עיכוב של $\frac{n^i}{n^n}$ לכל הודעה עם i.
ואז 1 יגיע ל2 שיעדכן וישלח ל3 ....עד n לפני ששאר ההודעות יגיעו.
כל קודקוד יתעדכן פעם אחת ובעדכון ישלח 2 הודעות לכן לכל היותר ישלחו $2n+2n=4n=O(n)$.
עבור $\Omega(n^2)$:
נבחר עיכוב של $\frac{1}{n^i}$ לכל הודעה עם i.
לכן הודעה i תגיע לכולם לפני שהודעה *i+1* תגיע למישהו ולכן כל קודקוד i יעדכן את ערכו $i-1$ פעמים.
בכל עדכון תשלח לפחות הודעה אחת.
סה"כ: $2n+\Sigma_{i=1}^{n}(i-1)=\Omega(n^2)$.

שאלה 4 (25 נקודות):

נתבונן במערכת שליחת הודעות סינכרונית על גרף שלם עם n קודקודים כאשר בדיוק אחד מהם ביזנטי (Byzantine).

א. (12 נקודות) תנו דוגמה להרצה של אלגוריתם המלכה (מופיע בסוף המבחן) עם $n = 4$ כך שהקלט של שלושת הקודקודים התקינים הוא 1 אך הם **אינם מגיעים להסכמה**.

ב. (13 נקודות) הוכיחו שאם $n = 5$ והקלט של ארבעת הקודקודים התקינים הוא 0 אז הם **כן מגיעים להסכמה**.

פתרון:

א. קלט: $(1,1,1,b)$.
שלב 1:
סיבוב 1:
כולם שולחים את הקלט שלהם (1) וגם הביזאנטי שולח 1. לכן הערך התקבל יותר מ3 פעמים ולכן כולם תומכים בו.
בסיבוב 2: המלכה היא קודקוד תקין אבל אף אחד לא משנה את הערך שלו.
שלב 2:
סיבוב 1:
כולם שולחים את הקלט שלהם (1) וגם הביזאנטי שולח 0. לכן הערך התקבל 3 פעמים ולכן אף קודקוד לא תומך בערך.
בסיבוב 2: המלכה היא ביזנטי והיא תשלח 0 לחלק מהקודקודים ו1 לאחרים.
נקבל שלאחר 2 השלבים אין הסכמה.

ב. הקלט של כולם הוא 0 ולכן הערך 0 יתקבל בסיבוב 1 יותר מ3.5 פעמים (4 פעמים) ומכאן כולם יתמכו בו ולא ישנו את ערכם גם אם המלכה תהיה ביזנטית.
בשלב השני הערך נשאר אותו דבר ולכן שוב הם יתמכו ב0 ללא תלות במלכה.
ומכאן הפלט של כולם יהיה 0.

מבחן 2018 ב ב

שאלה 1 (25 נקודות):

הוכיחו או הפריכו את הטענות הבאות הנוגעות לאלגוריתם הרמות לשלושה קודקודים שמופיע בסוף המבחן:

א. (12 נקודות) בתחילת כל סיבוב של ריצת האלגוריתם ההפרש בין הרמות של כל שני קודקודים מתוך השלושה הוא 1 לכל היותר.

ב. (13 נקודות) בתחילת כל סיבוב של ריצת האלגוריתם קיימים שני קודקודים שההפרש בין רמותיהם הוא 1 לכל היותר.

א. לא נכון,
נגדיר את ההרצה הבאה עבור 3 סיבובים:
ההודעות מu_3 ולu_3 לא מתקבלות.
בתחילת סיבוב ראשון: $l_1 = l_2 = l_3 = 0$.
ההודעת מu_1 לu_2 ולהיפך מתקבלות ולכן $l_1 = l_2 = \max(0,0+1) = 1$
בתחילת סיבוב 2: $l_1 = l_2 = 1, l_3 = 0$.
ההודעת מu_1 לu_2 ולהיפך מתקבלות ולכן $l_1 = l_2 = \max(1,1+1) = 2$
בתחילת סיבוב 3: $l_1 = l_2 = 2, l_3 = 0$.
ההודעת מu_1 לu_2 ולהיפך מתקבלות ולכן $l_1 = l_2 = \max(2,2+1) = 3$.

ב. נכון, נוכיח באינדוקציה על מספר הסיבוביםn.
בסיס: $n = 0$ כולם מאותחלים ל0 לכן ההפרש בין כל 2 הוא לכל היותר 1.
צעד: נניח שהטענה נכונה עד סיבוב n ונוכיח עבור סיבוב $n + 1$:
לפי הנחת האינדוקציה, בה"כ בתחילת סיבוב n הרמות היו (x, y, z).
כאשר: $|x - y| \leq 1$.
אם בסיבוב הn לא הגיעו הודעות לx, y ההפרש נשאר אותו דבר ולכן מתקיימת הטענה.
אם הגיעה רק הודעה אחת בין שני הקודקודים אז או שהערך של הקודקוד המעדכן נשאר אותו דבר ואז הטענה מתקיימת לפי הנחת האינדוקציה, או שהוא שווה לערך של השולח +1 ואז ההפרש ביניהם הוא 1 ומקיים את הטענה.
אחרת, אם ההודעה מz מגיעה למישהו.
מקרה 1: $z > x, y$ ואז: z לא יתעדכן. ואם מישהו נוסף קיבל את ההודעה מz אז ערכו שווה ל$z + 1$.
מקרה 2: $z < x, y$ ואז ההודעה מz לא מעדכנת. ולכן נותרנו עם מקרה 4.
מקרה 3: $|z - x| \leq 1$ וגם $|z - y| \leq 1$. דומה למקרה 4.
מקרה 4: אם שניים שהם כבר בהפרש 1 מקבלים הודעות זה מזה אז:
$x = \max(x, y + 1)$, $y = \max(y, x + 1)$ ומכאן:
אם $x < y$ אז: $x = y + 1$, $y = y$.
$x > y$ אז: $x = x$, $y = x + 1$.
$x = y$ אז: $y = x + 1$, $x = y + 1$ או שניהם.
בכל מקרה ההפרש הוא לכל היותר 1.

שאלה 2 (25 נקודות):

נתונים שני לוחות הזמנים (schedules) הבאים:

$$S: \quad s_{12}, r_{21}, s_{21}, s_{13}, s_{32}, r_{12}, r_{31}, s_{23}, r_{23}, r_{32}, s_{31}, r_{13}$$

$$S': \quad s_{12}, r_{21}, s_{13}, s_{32}, r_{31}, s_{21}, s_{23}, r_{12}, r_{32}, s_{31}, r_{13}, r_{23}$$

ג. (7 נקודות) הוכיחו או הפריכו: S' הוא causal shuffle של S.
ד. (8 נקודות) רשמו את כל המאורעות $e \in S$ המקיימים $e \Rightarrow_S r_{13}$ וגם $s_{23} \Rightarrow_S e$ (נמקו בקצרה את תשובתכם).
ה. (10 נקודות) הוכיחו שאם C ו-C' חתכים קונסיסטנטים בלוח זמנים כלשהו, אז $C \cup C'$ חתך קונסיסטנטי באותו לוח זמנים.

ג. נרשום את הנק' מבט של כל קודקוד.
ד. $S|v_1 = s_{12}, s_{13}, r_{12}, r_{13}$
$S|v_2 = r_{21}, s21\ s23\ r23.$
$S|v_3 = s32\ r31\ r32 s31.$
הרעיון הוא למצוא את המסלול מ$s23$ ל$r13$.
$., r32, s31$

ה. יהא $e \in C \cup C'$ ויהא e' מאורע כלשהו כך ש$e' \backslash \Rightarrow_s e$ צ"ל: $e' \in C \cup C'$.
ו. מכאן: $e \in C$ או $e \in C'$ ולכן מכיוון ששניהם חתכים עיקביים אז: $e' \in C$ או $e' \in C'$ ולכן באיחוד.

שאלה 3 (25 נקודות):

נתון גרף שלם על n הקודקודים $u_1, \dots, u_n$. לכל $1 \le i \le n$ הקלט של u_i הוא i. הקודקודים מתקשרים זה עם זה במערכת העברת הודעות אסינכרונית ללא תקלות. כל קודקוד מריץ את האלגוריתם הבא:

1. אתחל ערך עצמי לקלט שלך ושלח אותו לכל שכניך.
2. בכל פעם שמתקבלת הודעה עם ערך של שכן, בדוק אם הוא קטן ממש מערכך העצמי הנוכחי. אם כן, בצע את שתי הפעולות הבאות:
   א. הקטן את הערך העצמי שלך באחד.
   ב. שלח את הערך העצמי החדש שלך לכל שכניך.

הוכיחו שבכל הרצה של האלגוריתם הזה (כלומר לכל בחירה חוקית של delays להודעות) מספר ההודעות הכולל שנשלח הוא $\Omega(n^3)$.

פתרון:
הוכחה:
הטענה – כל קודקוד יעדכן את ערכו לפחות $\frac{i-1}{2}$ פעמים לכל בחירת עיכוב להודעות.
הוכחת הטענה: בשלב הראשון, יש $n-1$ הודעות שנשלחו לכל קודקוד.
מתוכן רק אלו הקטנות מערכו יעדכנו אותו ויגרמו לשליחה נוספת ולכן יש $i-1$ כאלה.
כאשר מגיעה הודעה $x \le i-1$ לקודקוד i אז הוא מוריד את ערכו ב1 ושולח עוד $n-1$ הודעות.
כעת, ההודעה x כבר הגיעה ובנוסף אם $x \ne i-1$ ההודעה $i-1$ כבר לא תעדכן את קודקוד i כי היא שווה לערכו העצמי.
ולכן כעת יש במעבר רק $i-3$ הודעות שיכולות לעדכן את קודקוד i.
לכן לאחר כל הודעה יש 2 הודעות פחות שיכולות לעדכן את i, ומכאן שהוא יעודכן לפחות $\frac{i-1}{2}$.
לכן קודקוד i לאחר קבלת ההודעות של השלב הראשון ישלח לפחות עוד $\frac{i-1}{2} \cdot (n-1)$ הודעות.
סה"כ ישלחו לאחר הודעות השלב הראשון לפחות: $\Sigma_{i=1}^{n} \frac{i-1}{2} \cdot (n-1) \ge \frac{n-1}{2} \Sigma(i-1) = \Omega(n^3)$ הודעות.

שאלה 4 (25 נקודות):

הציגו אלגוריתם wait-free הפותר את בעית הקונצנזוס הבאה עבור **שלושה** קודקודים ע"י שימוש במספר כלשהו של רגיסטרים אטומיים לקריאה/כתיבה (atomic read/write registers) ואוגר RMW אחד המשתמש בפונקציה compare and swap (תיאור הפונקציה הנ"ל מופיע בסוף המבחן) המאותחל לערך 1-. הניחו כי לכל קודקוד יש ID יחודי הידוע לכל הקודקודים. התקלות המותרות היחידות הן קריסת קודקוד (crush failure). הוכיחו את נכונות האלגוריתם שלכם.
קלט ופלט: הקלט והפלט של כל קודקוד שייך לקבוצה $\{0,1\}$.
הסכמה: כל הפלטים צריכים להיות שווים זה לזה.
סיום: כל קודקוד תקין חייב להוציא פלט אחרי זמן סופי.
Validity: אם הקלטים של שלושת הקודקודים זהים, אז הפלט של כל קודקוד תקין צריך להיות זהה לקלט שלו.

פתרון:
אלגוריתם: קלט x_i:

- $y = R.CAS(-1, x_i)$
- אם $y = -1$ החזר x_i.
- אחרת, החזר y.

הפלט הוא קלט של מישהו או של מישהו אחר ולכן הוא 0-1.
סיום: מתקיים.

הסכמה + תקפות: מכיוון שCAS היא אטומית אז יהי v הקודקוד הראשון שמבצע אותה ולכן יש באוגר -1 ומכאן שהערך יוחלף לx_v וזה גם הפלט של v.
כל השאר שיקראו אחר כך, לא ישנו את הערך כי לא היה מינוס 1 אבל יקראו את x_v ויחזירו אותו ולכן הפלט של כולם יהיה x_v.
לכן תמיד הפלא של כולם זהה לקלט של הראשון שביצע CAS ובפרט אם כל הקלטים זהים אז כל הפלטים זהים לקלטים.

מבחן 2017 א ב

שאלה 1 (25 נקודות):

יהיו C ו-C′ שני חתכים בלוח זמנים (schedule) כלשהו S. יהי $D = C \cup C'$.

א. (8 נקודות) הוכיחו ש-D חתך.
ב. (8 נקודות) הוכיחו שאם C ו-C′ קונסיסטנטים ב-S אז D קונסיסטנטי ב-S.
ג. (9 נקודות) תנו דוגמה ללוח זמנים S וחתכים C ו-C′ כך ש-D **אינו** קונסיסטנטי ב-S (הוכיחו את טענותיכם).

ג. $S = s12, r21$ כאשר $C = \Phi, C' = \{r21\}$.
באיחוד יש מאורע קבלה ללא מאורע השליחה שלו ולכן האיחוד טינו חתך עקבי.

שאלה 2 (25 נקודות):

א. (10 נקודות) תנו דוגמה ללוח זמנים S כך שבהרצה של אלגוריתם שעוני Lamport (האלגוריתם מופיע בסוף המבחן) על S מתקבלת קפיצה בגודל 10 (כלומר קיים קודקוד u ושני מאורעות e_1 ו-e_2 רצופים על u כך שמתקיים $\tau(e_2) = \tau(e_1) + 10$).
ב. (15 נקודות) הוכח או הפרך את הטענה הבאה: יהי S לוח זמנים ויהיו e_1 ו-e_2 שני מאורעות רצופים על קודקוד u כלשהו כך שבהרצה של אלגוריתם שעוני Lamport על S מתקיים $\tau(e_2) = \tau(e_1) + k$ לשלם חיובי k כלשהו. אזי ב-S מופיעים לפחות $k - 1$ מאורעות בין e_1 ו-e_2.

א. ...
ב. לא נכון.

שאלה 3 (25 נקודות):

נתונים שני מעבדים A ו-B במערכת מבוזרת עם זכרון משותף. לכל מעבד קלט מן הקבוצה $\{0,1,2\}$ והוא מחזיר פלט מאותה הקבוצה. כתבו אלגוריתם ללא המתנה (wait-free) הפותר את בעיית ההסכמה הבאה:

1. **הסכמה חלשה:** אם y_A הוא הפלט של A ו-y_B הוא הפלט של B אז $|y_A - y_B| \leq 1$.
2. **Validity:** אם $x_A = x_B$ כאשר x_A הוא הקלט של A ו-x_B הוא הקלט של B, אז $y_A = y_B = x_A$ כאשר y_A הוא הפלט של A ו-y_B הוא הפלט של B.
3. **סיום:** שני הקודקודים צריכים להוציא פלט אחרי זמן סופי.

לרשותכם שני אוגרים של קריאה/כתיבה אטומית (atomic read/write registers) המאותחלים לערך -1. תארו את האלגוריתם שמריץ כל אחד מהמעבדים והוכיחו שהוא מקיים את שלוש התכונות לעיל.

דומה לשאלות קודמות.

מבחן 2017 א א

שאלה 4 (25 נקודות):

היו S ו-S' שני לוחות זמנים (schedules) המכילים את אותם מאורעות (כלומר $e \in S'$ אם ורק אם $e \in S$). נניח שכל המאורעות ב-S הם שליחה וקבלה של הודעות. נריץ על שניהם את אלגוריתם שעוני Lamport (האלגוריתם מופיע בסוף השאלה) ונסמן את השעונים המתקבלים ב-τ ו-τ' בהתאמה.

א. (10 נקודות) הוכח או הפרך: אם S' הוא causal shuffle של S אז $\tau'(e) = \tau(e)$ לכל $e \in S$.

ב. (15 נקודות) הוכח או הפרך: אם $\tau'(e) = \tau(e)$ לכל $e \in S$ אז S' הוא causal shuffle של S.

א. הוכחה: מכיוון שS' הוא קוזאל של S אז מנקודת מבט של כל אחד מהקודקודים סדרת האירועים זהה ולכן לא ניתן להבחין בין ההרצות ומכאן שהתנהגות האלגוריתם בכל קודקוד תהיה זהה ולכן גם התוצאה תהיה זהה.

ב. הוכחה: נניח בשלילה שS' הוא לא CS של S ולכן קיים קודקוד u כך שסדרת האירועים בנק' שלו בS שונה מזו שבS' ולכן קיימים שני מאורעות e, e' בu כך ש: e היה לפני e' בS, e' היה לפני e בS'.
לפי שעוני למפורט, אם אירוע התחרש לפני אירוע אחר על אותו קודקוד אז חתימת הזמן שלו קטנה יותר.
לכן: $\tau(e) < \tau(e')$ וגם $\tau'(e') < \tau'(e)$. ומכאן לפי הנתון: : $\tau(e) < \tau(e')$ וגם : $\tau(e') < \tau(e)$ וזו סתירה.