אוניברסיטת אריאל בשומרון

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ
zvimints@gmail.com

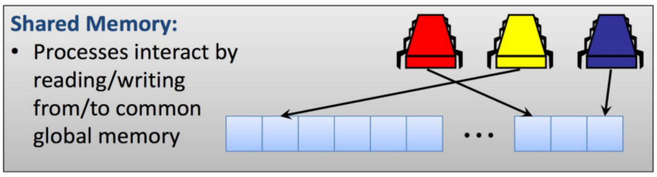
מערכות מבוזרות - Distributed System

אוסף של יחידות חישוב אשר יכולים לתקשר אחד עם השני
הערה: גם מחשב אחד הוא מבוזר, היות ויש כמה מעבדים וכמה תהליכים.

יתרונות של מערכות מבוזרות:

1. לשתף מידע בין קדקודים שונים
2. לטפל בכמות מידע גדולה יותר (במקום שכל המידע יהיה במקום כלשהו, אז ניתן להחזיק מידע בכל קודקוד)
3. עיבוד מקבילי (Parallelism)
4. Redundancy and Resiliency (אם מתוך 100 מכשירים אז 30 נפלו אז נבנה אותם ככה שהם יעבדו רק בעומס)
5. Scaling

מודלים למערכות מבוזרות



מערכות סינכרוניות: לכל המכשירים ישנו "קו שעון" משותף ולכן אפשר לדבר על סיבובים, שליחת ההודעות מתרחשת בתחילת הסיבוב ולקראת סוף הסיבוב ההודעה מגיעה. זמן הסיבוב ידוע מראש

במודל "Message Passing"

בראונד r : בזמן $r-1$, כל תהליך שולח הודעות. ההודעות מגיעות בזמן r

במודל "Shared Memory"

בכל ראונד, כל תהליך יכול לגשת לתא זיכרון בודד.

מערכות אסינכרוניות: אין "קו שעון", זמן הגעת ההודעה הוא מספר **סופי** כלשהו **לא ידוע** אשר יכול להשתנות בין הודעות.

במודל "Message Passing"

ההודעות תמיד מגיעות
זמן הגעת ההודעה הוא לא ידוע מראש ותמיד נסתכל על המקרה הגרוע ביותר

במודל "Shared Memory" (יש זיכרון אחד)

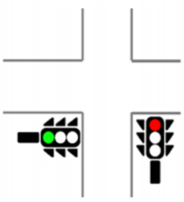
כל התהליכים בסופו של דבר יבצעו את הצעד הבא
זמן העיבוד לא ידוע מראש ותמיד נסתכל על המקרה הגרוע ביותר

תקלות Failures:

- **Crash Failure**
  קודקוד אשר מפסיק לעבוד בזמן מסוים, במערכת סינכרונית זה יכול לקרות באמצע השעון.
- **Byzantine Failure**
  קודקודיים שרוצים לפגוע בחישוב, "האקרים".
- **Omission Failure**
  קודקוד אשר מפסיק לעבוד לזמן מסויים
- **Resilience**
  התמודדות עם מספר מסויים של תקלות (מודל שמתמודד לדוגמא עם 50 תקלות כן ו51 לא)

תכונות נכונות של מערכת מבוזרות:

- **Safety** - שום דבר רע לא קורה ויקרה
  דוגמא: בכל נקודת זמן, לכל היותר רמזור אחד בצבע ירוק
  על מנת להוכיח Safety נוכיח בעזרת Invariant, המצב ההתחלתי היה בטוח, וכל מצב משאיר את המערכת בטוחה
- **Liveness** - משהו טוב יקרה למשהו
- דוגמא: יש תמיד אור ירוק אחד דלוק ברמזור
  דוגמא: במערכת אשר מספקת אוכל לאנשים, אז משהו מקבל אוכל.
  דוגמא: במערכת החלפת הקשר, אז משהו יכול לגשת למקור מידע.
- **Fairness** - משהו טוב יקרה לכולם
  דוגמא: במערכת אשר מספקת אוכל לאנשים, אז כולם מקבלים מספיק אוכל.
  דוגמא: במערכת החלפת הקשר, אז כל אחד יכול לגשת למקור מידע כל הזמן.
  דוגמא: שני האורות ירוקים ברמזור



קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
zvimints@gmail.com

מערכת בסיסית מכילה:
1. n קודקודים אשר מיוצגים ב- $v_1, \dots, v_n$
באופן לא מפורש - Implicit יש n קודקודים ממוספרים מ-1 עד n כאשר n הוא מספר ידוע.
2. בכל זמן, לכל קודקוד v_i יש מצב פנימי Q_i

Schedule: רצף של שליחות וקבלות המהוות את האירועים במערכת.
Admissible Schedule: רצף הגיוני, כלומר אין קבלת הודעה לפני שהיא נשלחת.

קונפיגורציה C זה וקטור של n כניסות אשר מתאר את מצב המערכת.
כאשר לאחר סדרת אירועים נקבל את הריצה הבאה:
$C_0, \Phi_1, C_1, \Phi_2, \dots$
כאשר Φ_i הוא אירוע כלשהו.

Local View **– מה כל קודקוד רואה מה-Schedule מסויים.**
בהינתן מתזמן S נגדיר $S|i$ להיות נקודת המבט של קודקוד v_i לאורך המערכת.
לדוגמא, בהינתן רצף $S = s_{13}, s_{23}, s_{31}, r_{13}, s_{32}, r_{31}, r_{23}, s_{13}, s_{21}, r_{31}, r_{12}, r_{32}$
$S|1 = s_{13}, r_{13}, s_{13}, r_{12}$
$S|2 = s_{23}, r_{23}, s_{21}$
$S|3 = s_{31}, s_{32}, r_{31}, r_{31}, r_{32}$
אבחנה: $S|i$ רואה את כל מה ש-i בהתחלה

תרגיל לדוגמא
יהיו $p_1, \dots, p_n$ קודקודים במערכת שליחת הודעות המריצים את האלגוריתם הבא:
אם $i = 1$:
שלח '1' ל- p_2
המתן לקבל הודעה ממנו
אם קיבלת $5 \le$ החזר 1
אחרת, החזר 0.
אחרת:
המתן לקבל הודעה מ– p_{i-1}
אם $i = n$ דלג על השלבים הבאים (תהליך אחרון)
שלח הודעה ל – p_{i+1}
המתן לקבל הודעה מ – p_{i+1}
שלח אותה ל– p_{i-1} בתוספת 1
אם המתנת ל – p_{i+1} יותר מ-$2n \cdot T$ יחידות זמן, החזר 0 ושלח ל – p_{i-1} את '1', אחרת החזר '1'
עבור השאלה הזו: (מערכת סינכרונית)
Safety – קודקוד p_1 מחזיר '1'
Liveness – משהו שאינו p_1 מחזיר '1'
Fairness – כולם מחזירים '1'
זמן הסיבוב – T

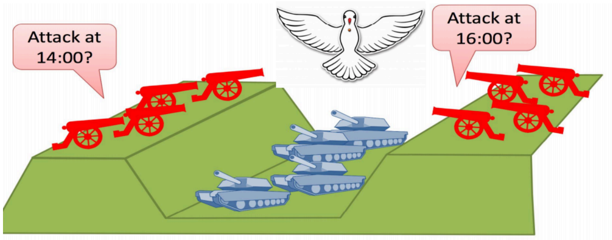
מודל הרצה לאלגוריתם:
1. מערכת סינכרונית, הודעה אחת הולכת לאיבוד.
Safety – אם הודעה אבדה בדרך מהקודקוד הראשון לקודקוד החמישי אז הקודקוד הראשון יקבל $5 >$ ולכן ואין *Safety*
Liveness – אם ההודעה מהקודקוד הראשון לשני אבדה אז כולם תקועים
הערה: אם *Liveness* לא מתקיים אז *Fairness* לא מתקיים

2. מערכת אסינכרונית – כל ההודעות מגיעות.
Safety – לא, כי אם הטיימר של p_2 פקע , אז הוא ישלח לקודקוד הראשון '1' ו- $1 < 5$
Liveness – גם אם הטיימרים של כולם יפקעו אז p_n לא יחכה ולכן יחזיר 1.
Fairness – כמו *Safety*

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב

נכתב ע"י צבי מינץ
zvimints@gmail.com

בעיית 2 הגנרלים



מודל: שני יחידות חישוב דטרמיניסטיות אשר מדברות במערכת סינכרונית מעל מערכת שליחת הודעות לא אמינה (הודעות יכולות ללכת לאיבוד)

קלט: כל יחידת חישוב מתחילה עם קלט מהקבוצה $\{0,1\}$

פלט: כל אחד מיחידות החישוב צריך להוציא פלט $\{0,1\}$

הסכמה: שני יחידות החישוב צריכות להסכים על **אותו** פלט

נכונות: אם הקלט זהה אז הפלט צריך להיות זהה לקלט במידה והודעות לא הולכות לאיבוד, אחרת, להחזיר 0 או 1 מוסכמים.

תקלות: הודעה יכולה ללכת לאיבוד

סיום: שני יחידות החישוב צריכים להסכים **בזמן סופי**

אבחנה: ללא נכונות, אפשר לבצע אלגוריתם פשוט אשר הינו 'החזר 0'

$\Downarrow$

הבעיה אינה פתירה

שיטת הוכחה: להניח שקיים אלגוריתם ולהראות סדרת הרצה הזהות זו לזו שתביא לסתירה

סימונים:

הרצה E לא ניתנת להבדל מהרצה E' עבור קודקוד v אם v רואה את אותו הקלט ואת אותם ההודעות ב-E ו-E'

אם E לא ניתנת להבדל עבור קודקוד v, אזי v מבצע את אותה סדרה של פעולות (בגלל שזה דטרמניסטי) בשני ההרצאות, ולכן נסמן $E|v = E'|v$ או לחלופין $E\sim_v E'$

אבחנה: אם נשקול סדרה של הרצאות $E_0, E_1, E_2, \dots E_k$ כך ש- $E_{i-1}\sim_v E_i$ עבור $i \in \{1, \dots, k\}$ עבור קודקוד v כלשהו אזי מתקיים כי כל הקודקודיים מוצאים אותו פלט בכל ההרצאות (בגלל שזה דטרמיניסטי)

הוכחה:

נניח בשלילה כי קיים אלגוריתם סופי דטרמיניסטי הפותר את הבעיה ב-T סיבובים

נשקול את רצף ההרצות *Executions* הבא:

Nodes always decide after exactly T rounds

Execution E_0: Both inputs are 0, no messages are lost.

Execution E_1: One of the messages in round T is lost.

...

Execution E_{2i}: Both messages in round T+1-i are lost.

Execution E_{2i+1}: One of the messages in round T-i is lost.

...

Execution E_{2T}: Both inputs are 0, no messages are delivered. All outputs are 0 due to similarity.

Execution E_{2T}: Both inputs are 0, no messages are delivered. All outputs are 0 due to similarity.

Execution E_{2T+1}: Input of v_1 is 0 but input of v_2 is 1. No messages are delivered.

Execution E_{2T+2}: Both inputs are 1, no messages are delivered.

...

Execution $E_{2T+2i+1}$: Exactly one of the messages in round i is delivered.

Execution $E_{2T+2i+2}$: Both messages in round i are delivered.

...

Execution E_{4T+2}: Both messages in round T are delivered. Decision must be 1 - a contradiction.

לסיכום:

- We start with an execution in which both nodes have input 0 and no messages are lost: by validity both nodes must decide 0.
- We remove messages one by one to obtain a sequence of executions such that consecutive executions are similar.
- From an execution with no messages delivered and both inputs 0, we can get to an execution with no messages delivered and both inputs 1 (in two steps).
- By adding back messages one by one, we obtain an execution in which both nodes have input 1 and no messages are lost: by validity both nodes must decide 1 $\Rightarrow$ contradiction!
- Not hard to generalize to an arbitrary number n>1 of nodes.

הרצאה 3 – בעיית 2 הגנרלים רנדומלי

תזכורת: בעיית 2 הגנרלים יכולה להיות פתירה אם:

- נאפשר לאחד מהצדדים להטיל מטבע
- נקבל גם פתרון עם הסתברות $1-\epsilon$ להסכמה

אלגוריתם "הרמות" - $\boldsymbol{Level\ Algorithm}$:

1. שני הרמות מאותחלות ל-0
2. בכל ראונד: שני הקודקודים שולחים את הרמה שלהם לשני
3. כאשר קודקוד u עם רמה l_u מקבל הודעה מקודקוד v אשר עם רמה l_v אזי u מעדכן את הרמה שלו ל- $l_u =$ $\max\{l_u, l_v+1\}$

אבחנה: הרמות בחיים לא יורדות

תכונות

טענה: בכל הזמנים, כמות הרמות שונה לכל היותר ב-1
הוכחה: באינדוקציה על מספר הסבבים
בהתחלה $l_u = l_v = 0$ ולכן הבסיס מתקיים.
נניח כי הטענה נכונה עבור סיבוב t ונשקול את הסיבוב ה-$t+1$
אם $l_u = l_v$ אזי הטענה מתקיימת כי הרמה של כל קודקוד עולה לכל היותר ב-1 ובחיים לא יורדת.
ולכן נניח בה"כ כי $l_u = l_v + 1$, לא משנה מה קורה בסיבוב $t+1$ אזי l_u לא משתנה, אם v מקבל הודעה l_u מקודקוד u אזי הרמה עולה ל-$l_v = l_u + 1$

טענה: אם כל ההודעות מתקבלות אזי הרמה של שני הקודקודיים זהה ושווה למספר הסיבובים
הוכחה: באינדוקציה על מספר הסיבובים
עבור שלב הבסיס זה מתקיים באופן טריוויאלי
נניח כי הטענה נכונה $l_u = l_v = t$ לאחר סיבוב ה-t ונשקול את הסיבוב ה-$t+1$
יהי l'_u ו- l'_v להיות הרמה של הקודקודים u, v לאחר הסיבוב ה-$t+1$, לפי ההגדרה של האלגוריתם אזי

$$l'_u = l_u + 1 = t + 1$$
$$l'_v = l_v + 1 = t + 1$$

כנדרש

טענה: הרמה l_u של קודקוד u שווה ל-0 אם"ם u לא מקבל אף הודעה.
הוכחה: אם u לא מקבל אף הודעה אזי הרמה לא משתנה ולכן זה נשאר 0, ולכן נניח כי בנקודה מסויימת הוא מקבל הודעה עם הרמה l_v מקודקוד v.
בבגלל שהרמה בחיים לא קטנה ומתחילה ב-0 אז היא לפחות 0, כלומר $l_v \geq 0$ ולכן מהגדרת האלגוריתם הוא מעדכן את הרמה $l_u = \max\{l_u, l_v+1\} > 0$ ובגלל שהרמה בחיים לא יורדת $l_u > 0$ לכל אורך האלגוריתם

לסיכום:

1. בסוף האלגוריתם ההפרש רמות לכל היותר 1
2. אם כל ההודעות מגיעות, אזי כל הרמות שוות למספר הסיבובים r
3. הרמה של קודקוד u היא לפחות 1 אם"ם u מקבל לפחות הודעה אחת

<u>אלוגריתם רנדומי לפתרון בעיית הגנרלים:</u>

- נניח כי u יכול להשתמש ברנדומזציה
- נניח שהקלטים יכולים להיות רק 0 או 1

1. קודקוד u בוחר מספר $t \in \{1, \dots, r\}$ באופן אחיד
2. שני הקודקודים מריצים את האלגוריתם "הרמה" (Level Algorithm") עבור r צעדים
בזמן ריצת האלגוריתם, בכל הודעה שני הצדדים מוסיפים גם כן את הקלט שלהם וקודקוד u מציין גם כן את t
3. כל קודקוד פולט 1 אם:
   1. הקודקוד יודע מה הערך של t ויש להם קלט זהה
   2. שני הקלטים זהים ל-1
   3. הרמה של הקודקוד הוא לפחות t
4. הקודקוד פולט 0

טענה: אם אחד הקלטים הוא 0 אז שני הקודקודים פולטים 0
הוכחה: נובע ישירות מהאלגוריתם

טענה: נניח שאחד הקלטים הוא 1
1. אם אף הודעה לא הולכת לאיבוד אזי שני הקלטים יפלטו 1
2. שני הקודקודים פולטים את אותו הערך אלא אם כן $\{l_u, l_v\} = \{t-1, t\}$

הוכחה: נניח תחילה כי $\{l_u, l_v\} = \{t-1, t\}$ בהתאמה, לפי ההגדרה של האלגוריתם, u מחליט 0.
בגלל ש-$l_v = t > 0$ אזי v מכיר את t ואת 2 הקלטים אשר הם 1, ולכן הוא מחליט 1.
כעת, נניח כי $\{l_u, l_v\} \neq \{t-1, t\}$. אם $l_u, l_v < t$ אזי שניהם פולטים 0, אחרת $l_u, l_v \geq t$, מכיוון ש-$t > 0$ אזי שני מכירים את t וראו את הקלטים, ולכן שניהם יפלטו 1. קבלנו כי אם אף הודעה לא הולכת לאיבוד אזי $l_u = l_v = r \geq t$ ולכן יפלטו 1.

טענה: האלגוריתם מגיע להסכמה עם הסתברות של לפחות $1 - \frac{1}{r}$
הוכחה:
תזכורת: ההפרש רמות הוא לכל היותר 1
בגלל שהרמה נקבעת רק לפי כמות ההודעות שמגיעות/הולכות לאיבוד, אזי ה"אויב" יכול לקבוע שהם יהיו שונות, ולכן $\{l_u, l_v\} = \{i-1, i\}$ עבור $i \in \{1, \dots, r\}$
בגלל שה"אויב" לא מכיר את t אזי ההסתברות $\{l_u, l_v\} = \{t-1, t\}$ הינה $\frac{1}{r}$ בגלל התפלגות אחידה
לפי שני הטענות הקודמות, הקודקודים מגיעים להסכמה בכל שלב אחר ולכן נוכל להסיק כי ההסתברות לכשלון הינה $\frac{1}{r}$

<u>חסם תחתון:</u>
דרישה: אם אחד מהקלטים הוא 0, אזי שני הקודקודים פולטים 0
אבחנה: האלגוריתם שלנו קיים את הדרישה הזאת.
על מנת להוכיח חסם תחתון, נניח כי שני הקלטים הינם 1 ולכן:
- אם אף הודעה לא הולכת לאיבוד, אזי שני הפלטים חייבים להיות 1
- אחרת, הקודקודים צריכים לפלוט את אותו הערך עם הסתברות $1 - \epsilon$

טענה: נראה כי $\epsilon \geq \frac{1}{r}$ ולכן זהו חסם הדוק.
הוכחה: צריך להשלים (היה בתרגול 3)

טענה (טענה יותר חלשה): נראה כי $\epsilon \geq \frac{1}{2r}$
כלומר, אם יש r סיבובים אז $\epsilon \geq \frac{1}{2r}$, אם נגדיל את כמות הסיבובים אז הטעות קטנה.
הוכחה: כל סיבוב מורידים 2 הודעות, ההסתברות לטעות בכל הודעה הינה ϵ ולכן נקבל כי אם בסוף נחליט את הקלטים ל0 ו1 אז נקבל כי $1 = q_v \leq 2\epsilon r \Rightarrow \epsilon \geq \frac{1}{2r}$

הרצאה 4 – בעיית 2 הגנרלים רנדומלי

בהינתן גרף, שני קודקודים יכולים לדבר אחד עם השני רק אם יש קשת בינהם.
נתעסק במערכת אסינכרונית – כל ההודעות מגיעות בזמן סופי **כלשהו**
בניגוד למודל הסינכרוני, אין סיבובים. כל פעם שמקבלים הודעה נוצר "מאורע" Event .

הנחות:
1. כל קודקוד בגרף הוא יחודי, יש לו מזהה יחודי כך שכל קודקוד יכול להבדיל מקודקוד אחר.
2. הגרף קשיר, כי אחרת ההודעות לא יכולות לעבור לכולם

אלגוריתם Flooding

קודקוד v שולח הודעה לכל השכנים שלו, השכנים מקבלים הודעה ומעבירים גם אותה לשכנים וכו׳
במידה ואני קודקוד u שכבר קיבל הודעה – מיותר להעביר אותה שוב לכל השכנים, בנוסף לא אשלח הודעה לשכן ממנו קבלתי את ההודעה

נכונות האלגוריתם נובעת מההנחה שהגרף קשיר, וידוע כי כל ההודעות מגיעות בזמן סופי כלשהו.

ניתוח הסיבוכיות עבור <u>מערכת סינכרונית</u>:

בזמן 0 $\Leftarrow$ רק v יודע את ההודעה
בזמן 1 $\Leftarrow$ כל מי שבמרחק 1 מ- v יודע את ההודעה
בזמן 2 $\Leftarrow$ כל מי שבמרחק 2 מ- v יודע את ההודעה
$\vdots$
בזמן r $\Leftarrow$ כל מי שבמרחק r מ- v יודע את ההודעה

הגדרות:

אורך מסלול בגרף – מספר הקודקודים פחות 1 (מספר הקשתות)
אורך $dist(u,v)$ – המסלול **הקצר ביותר** בין u ל-v
כל קודקוד יקבל את ההודעה שלו בסיבוב ה-t במידה ו-t זה המרחק שלו מקודקוד v

נגדיר **רדיוס** של v בתוך G:

$$rad(G,v) \coloneqq \max\{dist_G(u,v) : u \in V\}$$

הרדיוס של G באופן כללי:

$$rad(G) \coloneqq \min\{rad(G,v) : u \in V\}$$

קוטר הגרף (Diameter)

$$\mathrm{diam(G)} \coloneqq \max\{\mathrm{dist_G}(u,v) : u,v \in V\} = \max\{rad(G,v) : v \in V\}$$

אבחנה:

$$\frac{\dim(G)}{2} \leq rad(G) \leq rad(G,v) \leq diam(G)$$

ההוכחה לזה תהיה שאלה במטלה.

ניתוח סיבוכיות <u>במערכת אסינכרונית</u>:

באלגוריתם Flooding יש Addversory שיכול לעקב אותנו, ישנו עיקוב מקסמלי אבל סופי
הנחות:

1. העיקוב של ההודעות הוא מספר בין 0 ל-1
2. כל חישוב על אחד הקודקודים לוקח 0 זמן
3. סדר האירועים נשמר – קודם יש שליחת הודעה ואז קבלת הודעה (לא יתכן הפוך)

סיבוכיות של מערכת אסינכרונית לוקחת מקסימום זמן ריצה של **כל** הרצה אפשרית

טענה: באלגוריתם Flooding הסיבוכיות במערכת אסינכרונית וסינכרונית זהה.
טענה: סיבוכיות הריצה של אלגוריתם Flooding ממקור v בתקשורת אסינכרונית הוא $rad(G,v)$
הוכחה: ע״י עיקוב של כל ההודעות ב-1 נקבל כי הזמן שלוקח להודעה M להגיע לקודקוד האחרון הוא לפחות $rad(G,v)$
הערה: ניתן להסיק שכל אלגוריתם אי שפעם שנכתוב, אזי הסיבוכיות זמן ריצה המקסמלי שלו במערכת אסינכרוני הוא לפחות כמו במערכת סינכרונית (כי תמיד אפשר לעקב ב-1)
הוכחה חסם תחתון, כעת נוכיח חסם עליון באינדוקציה על המרחק

צ״ל שאם קודקוד נמצא במרחק t מ-v אזי הקודקוד יקבל את ההודעה בזמן $\leq t$
עבור שלב **הבסיס**, v יודע את ההודעה בזמן 0, ורוצה לעביר לשכן שלו, השאר יקבל את ההודעה שלו בזמן ≤ 1 ו-t שווה ל-1.

צעד: נניח שיש קודקוד u שמעביר ל-$u+1$, ו-u נמצא במרחק t מ-v, כלומר לפי הנחת האינדוקציה, u מקבל את ההודעה בזמן $\leq t$, מכאן ש-$u+1$ נמצא במרחק 1 יותר מ-u ולכן $u+1$ יקבל את ההודעה בזמן $\leq t+1$
אולם, יכול להיות ש-$u+1$ יקבל את ההודעה בזמן קצר יותר ממסלול אחר, שהוא מהיר יותר (אבל עם יותר קשרים) בדרך מ-v ולכן הוא קיבל את ההודעה בזמן פחות מ-t ולכן הטענה מתקיימת.
הערה: יכול להיות ש-u לא ישלח את ההודעה ל-$u+1$ אם $u+1$ שלח לו הודעה קודם.

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב

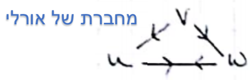
נכתב ע"י צבי מינץ
zvimints@gmail.com

סיבוכיות ההודעות:

כמה הודעות נשלחות? אנחנו מנסים לייעל את מספר ההודעות גם כי:

1. לא מחזירים למי ששלח
2. אם כבר קבלתי הודעה, לא מעבירה אותה שוב

לכן, מספר ההודעות שנשלחות הוא בסדר גודל של מס׳ הקשתות בגרף.
הכי הרבה הודעות שיכולות לעבור על קשת זה 2, במקרה הבא:



ולכן המקסימום יש $2|E|$ הודעות ולכן $O(|E|)$

<u>מעכשיו, נרצה שגם האלגוריתם יעבוד מהר וגם שישלחו כמה שפחות הודעות.</u>

תזכורת: עץ – גרף קשיר חסר מעגלים

בעץ שלנו נגדיר שורש (קודקוד כלשהו שנבחר), זה פשוט אומר שהוא הראשון ואין לו אבא.
האלגוריתם נשאר אותו דבר אלא רק בתוספת של 2 דברים:

1. הקודקוד שורש – אין לו אבא
2. מי שקיבלתי ממנו את ההודעה נשמר כאבא שלי.

באופן יותר פורמלי:

Source node v:

initially do

parent = NIL (i.e. v is the root of the tree).

send M to all neighbours.

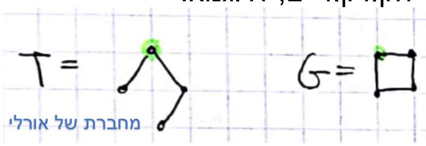
Non-source node u:

upon receiving M from some neighbor w: if M has not been received before, then

parent = w.

send M to all neighbors except w.

העץ T הוא עץ פורש של G אם הוא נוגע בכל הקודקודיים, לדוגמא:

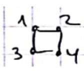


בהעברה הזאת יהיו קשתות בגרף המקור שיעלו מגרף העץ הפורש.
מדוע? כי אם יש לי 2 אבות, כלומר 2 קודקודים ששלחו לי הודעה אז אנחנו בוחרים באופן שרירותי אבא, והצע שהיא לא האבא שלי נמחקת מהעץ.

איך נבנה את העץ במערכת סינכרונית?

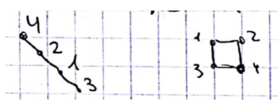
$$dist(G) = dist(T)$$

מקודקוד

העץ הוא תת גרף של G, באופן כללי המרחקים בעץ יכולים לגדול למשל  אז אחרי הפריסה של העץ המרחק בין 3 ל-4 יהיה 3 ולא 1 כמו ב-G

העץ שנפרש בצורה טובה ביותר, ישמור על המרחקים המקוריים מ-v לאחרים. עץ זה נקרא עץ BST.

דוגמא לפריסה **לא** טובה:



פריסה **טובה** זה 4 עץ בינארי ש-4 הוא השורש, 2 ו-3 בנים ו-1 הוא אחד הבנים של 2 או 3 (לא משנה)

במערכת סינכרונית – מה שלא יקרה, בטוח נבנה עץ BFS

In synchronous systems, a node u is reached in round t if and only if $dist_G(u,v)$ = t.

Hence, in the constructed tree: distance of a node u to the root equals the round in which u is reached.

Hence, the constructed tree preserves the distances to the root of the graph G.

Such trees are called Shortest Path trees or Breadth First Search trees.

אוניברסיטת
אריאל
בשומרון

קורס מערכות מבוזרות – סמסטר א' 2020 מדעי המחשב
נכתב ע"י צבי מינץ
zvimints@gmail.com

למה זה יוצא עץ BFS? כי זה בדיוק האלגוריתם לייצרת עץ BFS, הנה תיאור:

תיאור אינטואיטיבי [עריכת קוד מקור | עריכה]

האלגוריתם משתמש במבנה נתונים מסוג תור על מנת לקבוע מהו הצומת הבא בו הוא עומד לבקר. בכל פעם שהוא מבקר בצומת הוא מסמן אותו ככזה שנבדק, ואז בודק את כל הקשתות שיוצאות ממנו. אם קשת מובילה לצומת שטרם נבדק, צומת זה מתווסף לתור. בדרך זו מובטח כי האלגוריתם יסרוק את הצמתים בסדר שנקבע על פי מרחקם מהצומת ההתחלתי (כי צומת שנכנס לתור יצא ממנו רק לאחר שכל הצמתים שהיו בו קודם יצאו).

שימושים: בדיקת המסלולים הקצרים בגרף לא ממושקל.

במערכת אסינכרונית זה לא בטוח יקרה..
אבחנה: במערכת אסינכרונית, **כל** עץ פורש יכול להווצר ע״י flooding אלגוריתם, כלומר העץ שנבנה יכול להיות באורך $n-1$ גם אם הרדיוס של הגרף המקורי הוא 1, היות וניתן לשחק עם הזמנים של ההודעות ניתן ליצור כל עץ אפשרי.

אלגוריתם Convergecast

Leaf node u:

Initially do: send a message to parent (e.g., send input value).

Internal node w:

Upon receiving a message from child node x: **if** w has received messages from all children, **then** send message to parent (e.g., send sum of all inputs in the subtree whose root is w).

Root node v:

Upon receiving a message from child node x: **if** v has received messages from all children, **then** convergecast terminates.

זה בעצם ההפך מ-Broadcast, לכולם יש הודעה ורוצים להודיע בחזרה לקודקוד v שקיבלתם ממנו את ההודעה.
דוגמא: חישוב סכום של ערכים ולהודיע לשורש העץ

כל עלה שולח הודעה להורה עם הערך שלו, כל עלה יודע שהוא עלה כי אין לו ילדים.
כל קודקוד יודע מי האבא שלו וכמה ילדים יש לו

כל **קודקוד** מחכה שיקבל את ההודעה מכל הילדים שלו, מוסיף את הערך שלו ושולח לאבא
האלגוריתם נגמר כאשר השורש מקבל את ההודעות מכל הבנים שלו
הסיבוכיות – העומק של העץ
סיבוכיות ההודעות - מספר הקשתות בעץ (אחד פחות ממספר הקודקודים) כי על כל צלע עוברת הודעה אחת בלבד.

שימוש חשוב – שילוב עם אלגוריתם Flooding לקבל מענה.

אלגוריתם Flooding/Echo

Given a root, flooding and convergecast can be used together as follows:

1. Use flooding to construct a tree (when receiving a message, inform the sender if its your parent or not).
2. Use convergecast (echo) to report back to the root when done (leaves begin reporting back up immediately).

1. הולכים ושולחים הודעה לכולם ותוך כדי בונים את העץ Flooding
2. משתמשים בעץ כדי לשלוח הודעות למעלה כדי לאשר ל-v שכולם קבלו את ההודעה.

הסיבוכיות המשותפת: רדיוס + עומק. **Flooding/Echo Algorithm**

במערכת **סינכרונית** העומק שווה לרדיוס ולכן זה $O(diam(G)) = 2 \cdot rad(G)$

במערכת **אסינכרונית** העומק הוא כמספר הצלעות/קודקודים פחות 1 (המקרה הגרוע ביותר, קו ישר) + לכל היותר הרדיוס ולכן פה $O(|V(G)|)$ שזה גרוע יותר.