Instruction Manual

הקדמה:

הסימולטור הינו מערכת לביצוע חישובים ברשתות מבוזרות.

רשת חישוב מבוזרת היא רשת המורכבת ממספר רב של מחשבים נפרדים אשר מבצעים פעולת חישוב כלשהי בצורה מבוזרת, כלומר ללא גורם מרכזי המנהל את החישוב.

ישנו קושי בבדיקת אלגוריתמים המפותחים לרשתות מבוזרות בעקבות הצורך במספר רב של מחשבים בשלב הבקרה, ולכן לעיתים קרובות הבדיקה מבוצעת רק לאחר ההתקנה על המכשירים עצמם, מה שעלול לגרום לעלות גבוהה במידה והאלגוריתם שגוי ומתרחשת שגיאה.

לצורך כך, נבנה סימולטור המדמה את סביבת העבודה הדרושה- רשת בעלת מספר רב של מחשבים. הסימולטור יאפשר למשתמש להכניס כקלט אלגוריתם מסוים אותו ברצונו להריץ על רשת מבוזרת יחד עם רשימת פרמטרים המתארת את הרשת עליה האלגוריתם ירוץ בהתאם לדרישותיו והסימולטור ידמה את ריצת האלגוריתם על הרשת שנקבעה.

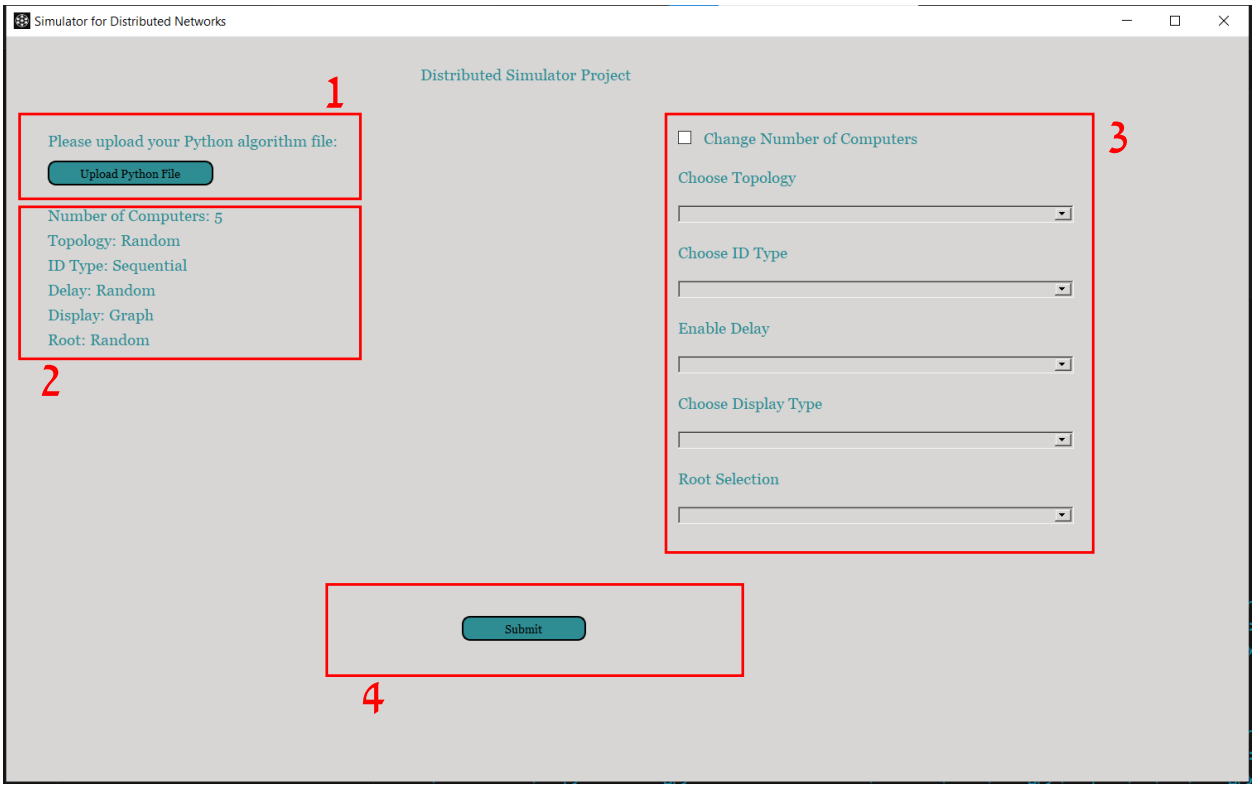
למי נועדה המערכת:

המערכת נועדה לחוקרים שרוצים לבדוק אלגוריתמים על רשתות מבוזרות בסביבה שהם מעוניינים לקבוע עם הפרמטרים הרצויים.

המערכת תסמלץ את האלגוריתם לפי הפרמטרים שלהם ותאפשר לחוקרים להוציא תצוגה גרפית/טקסטואלית של הריצה של האלגוריתם לבחירתם.

סקירת התהליכים במערכת:

לאחר הרצת הסימולטור ופתיחת החלון הראשי יוצג למשתמש בחירת הפרמטרים לרשת ואזור להעלאת האלגוריתם שהוא מעוניין לסמלץ, מצורפת תמונה של המסך הראשי להסבר:



1. <u>אזור העלאת האלגוריתם:</u> לאחר לחיצה על הכפתור יפתח עבורכם סייר הקבצים ושם תוכלו לבחור את קובץ האלגוריתם שלכם שכתוב לפי ממשק המערכת (רשום בהמשך הקובץ).
2. <u>ערכי ברירת המחדל של הרשת:</u> יוצגו לכם ערכי ברירת המחדל שנקבעו במידה ורק תרצו להעלות את האלגוריתם שלכם ולהריץ אותו על הרשת שאנחנו קבענו.
3. <u>אזור בחירת פרמטרי הרשת:</u> למען הרצת האלגוריתם עליכם לקבוע פרמטרים שעליהם תרצו לבחון את האלגוריתם שלכם, לאחר סימון ריבוע יפתח עבורכם שורת כתיבה לערך המתאים לפרמטר. במידה ולא תסמנו את התיבה יינתן ערך ברירת מחדל שנקבע מראש (ערכי ברירת המחדל רשומים בהמשך).
4. <u>אזור ההרצה:</u> לאחר לחיצה על הכפתור תועברו לדף חדש בו יוצגו התוצאות, בין אם בחרתם בצורה טקסטואלית ובין אם בגרפית.

<u>ברירת המחדל של המערכת:</u>

כאשר אתם ניגשים לאזור פרמטרי הרשת במסך הפתיחה ברשותכם האופציה לבחור פרמטרים מסל הפרמטרים שמוצעים:

1. <u>מספר המחשבים ברשת:</u> ברשותכם לבחור כמה מחשבים יפעלו על האלגוריתם, שימו לב כי ברירת המחדל היא חמישה מחשבים.
2. <u>טופולוגיית הרשת:</u> ברשותכם לבחור את צורת הרשת (קו, קליקה, אקראי), שימו לב כי ברירת המחדל היא טופולוגיה אקראית.
3. <u>סוג מזהה הרשת:</u> ברשותכם לבחור האם מזהה הרשת יקבע בצורה אחידה או בצורה אקראית, שימו לב כי ברירת המחדל היא אחידה.
4. <u>בחירת תצוגת הרשת:</u> ברשותכם לבחור האם תצוגת הרשת תהיה טקסטואלית או גרפית, שימו לב כי ברירת המחדל היא תצוגה גרפית.
5. <u>בחירת מנהיג:</u> ברשותכם לבחור root למערכת, שימו לב כי ברירת המחדל היא בחירה אקראית של מי המנהיג.

סוגי התצוגה של המערכת:

למערכת שתי אופציות לתצוגה: תצוגה גרפית ותצוגה טקסטואלית.

נסביר לכם כעת ונצרף דוגמה לכל צורת תצוגה.

תצוגה טקסטואלית:

במידה והרשת גדולה וכרגע לא ניתן לייצג אותה בצורה גרפית, ניתן לבחור בתצוגה טקסטואלית שתציג את הרשת בצורה ברורה.

לאחר לחיצה על הרצת האלגוריתם תקבלו את ההודעה הבאה שתייצג את הפרמטרים שהכנסתם (במידה ולא תקבלו את ערכי ברירת המחדל של המערכת), לדוגמה:

```
Number of Computers: 5
Topology: Random
ID Type: Sequential
Display Type: Graph
Delays: {(0, 1): 5, (0, 2): 8, (0, 3): 10, (0, 4): 8, (1, 2): 10, (1, 4): 1, (3, 4): 4}
```

בקטע ה$delay$ קיבלנו מילון שמסביר בצורה הבאה, על הקשת $(0,1)$ יהיה 5 $delay$ ועל הקשת $(0,2)$ יהיה 8 $delay$. לאחר מכן יציגו לכם את הרשת ואיך היא בנויה:

```
Computers:
id = 0
connected edges = [1, 2, 3, 4]
delays = [5, 8, 10, 8]

id = 1
connected edges = [0, 2, 4]
delays = [5, 10, 1]

id = 2
connected edges = [0, 1]
delays = [8, 10]

id = 3
connected edges = [0, 4]
delays = [10, 4]

id = 4
connected edges = [0, 1, 3]
delays = [8, 1, 4]

3  is root
```

מוצג לנו כיצד כל מחשב מיוצג הid שלו, הקודקודים שמחוברים אליו וה$delay$ של כל קשת ביניהם.

לאחר מכן תתחיל ההצגה של ההרצה של האלגוריתם,

כל שליחת הודעה מוצגת באופן הבא:

```
message received:  {'source_id': 3, 'dest_id': 4, 'arrival_time': 4, 'content': 'running a broadcast'}
```

מופיע המידע הבא:

- מחשב המקור שממנו נשלחה ההודעה.
- מחשב היעד שאליו שולחים את ההודעה.
- זמן ההגעה של ההודעה.
- תוכן ההודעה.

בדוגמה שצורפה, אנחנו מריצים את אלגוריתם $broadcast$ וההודעה נשלחת ממחשב 3 למחשב 4 והיא מגיעה בזמן 4.

בסוף ההרצה של האלגוריתם יופיע זמן הריצה שלקח להריץ את האלגוריתם באופן הבא:

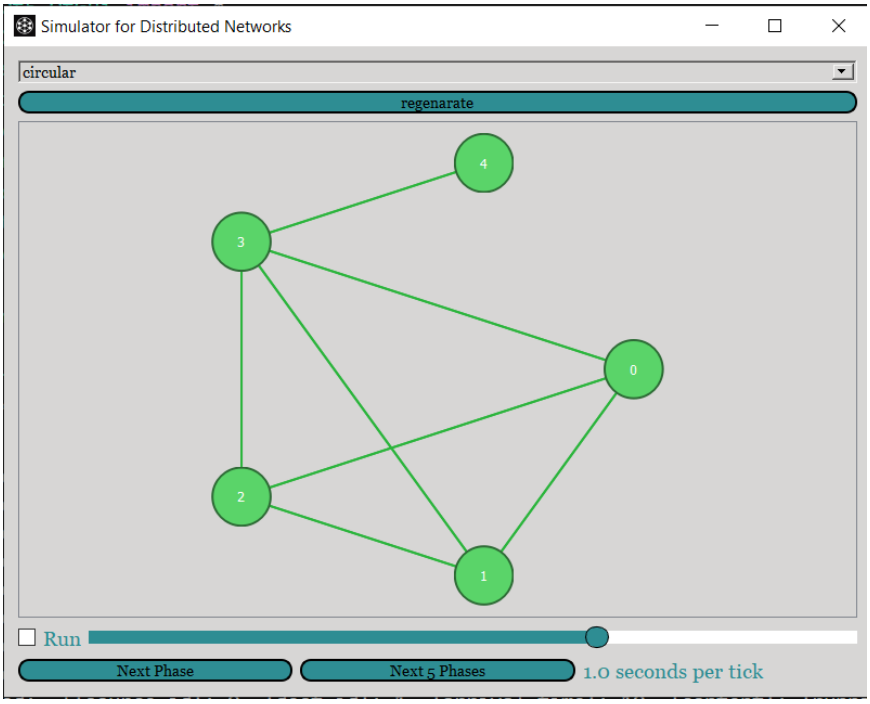
```
--- 0.5632135868072 51 seconds ---
```

לדוגמה, עבור אלגוריתם $broadcast$ לקח לנו 0.56 שניות להריץ אותו ברשת שהצגנו בהתחלה.

תצוגה גרפית:

כאשר משתמש בוחר בתצוגה גרפית, לאחר לחיצה על כפתור הריצה יועבר המשתמש לחלון חדש שבו תיפתח לו תצוגה של האלגוריתם והוא יוכל לעבור בין השלבים של הסימולציה.

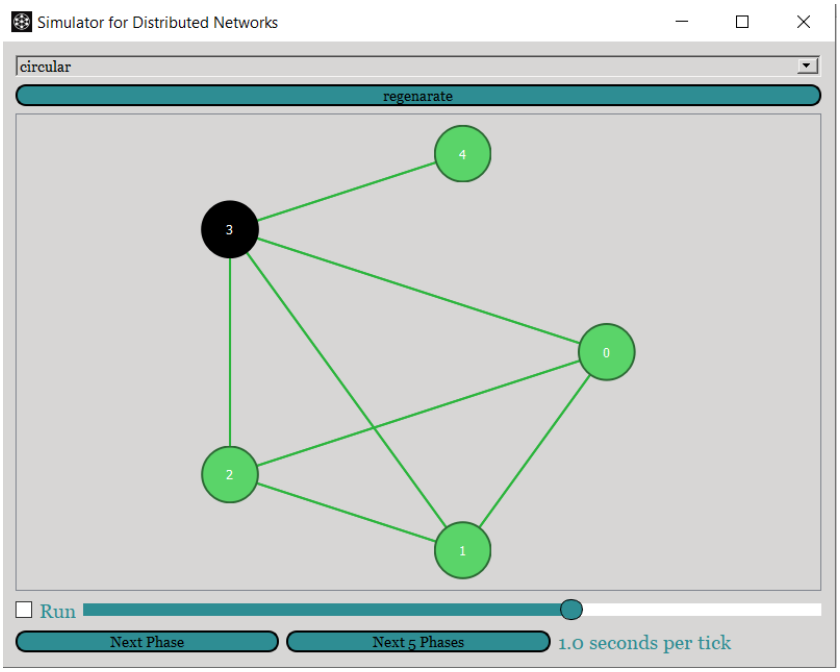
לדוגמה, עבור אלגוריתם $broadcast$ שהצגנו לפני יתקבל החלון הבא:



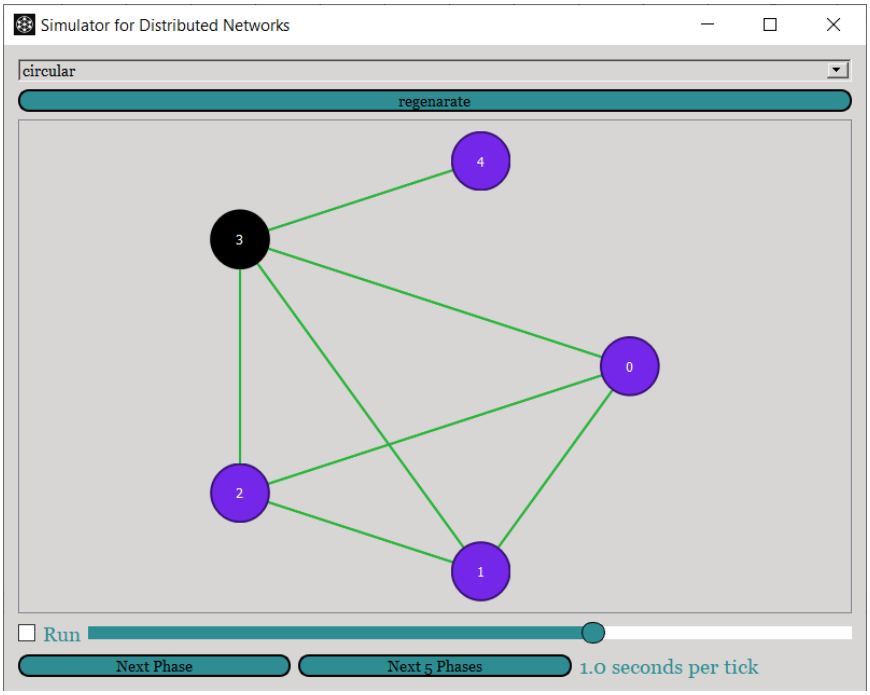
כעת נשים לב כי ניתן לדלג בין מצב אחד קדימה לבין חמישה מצבים קדימה.

במידה והיינו בוחרים טופולוגיה $random$ בתפריט בראש החלון היינו יכולים ללחוץ על $regenerate$ ולסדר את הרשת מחדש.

כאשר נלחץ על $next\ phase$ נתחיל את הרצת האלגוריתם ונקבל את התצוגה הבאה:



נוכל להמשיך ככה עד שבעצם נגיע לסיום האלגוריתם:



ניתן לראות כאן הרצה של אלגוריתם $broadcast$ כאשר קודקוד 3 הוא ה$root$ שלנו.

Class Description

עבור $compute\ class$:

נתחיל מלהסביר את המשתנים שזמינים לכם לשימוש במערכת,

הסבר על הפונקציות זמין בהמשך הקובץ

$member$	$description$
id	מזהה רשת שהוא ייחודי לכל מחשב, ניתן לבחור במסך הפתיחה אם הid יהיו בסדר עולה אחיד או שיוגרלו באופן רנדומלי
$connectedEdges$	רשימה המכילה את הid של כל מחשב ברשת שמחובר למחשב הנוכחי. לדוגמה, הרשימה $[2,5]$ עבור מחשב עם מזהה 1 אומר לנו כי קיימות ברשת הקשתות $(1,5), (1,5)$ **שימו לב!** התקשורת על הקשתות היא דו כיוונית והגרף הוא לא מכוון
$delays$	רשימה המכילה את ה$delay$ שיש על קשת, היא תהיה תואמת לרשימת ה$connectedEdges$ ובעצם האיבר הראשון ברשימה הוא העיכוב על הקשת שבין המחשב שלנו למזהה שנמצא במקום הראשון ברשימת ה$connectedEdges$
$state$	מפרט על המצב של המחשב במהלך ריצת האלגוריתם ברשת, זהו משתנה מסוג מחרוזת. **שימו לב!** ניתן לבצע השמה למשתנה זה לערך $terminated$ שבעצם יסיים את ההשתתפות שלו באלגוריתם ברשת
$root$	משתנה בוליאני שאומר לנו האם המחשב הוא ה$root$ שמתחיל את ריצת האלגוריתם. **שימו לב!** בכל המחשבים הפרמטר מאותחל להיות $False$ ובמידה ותרצו לשנות את זה תוכלו בפונקציית ה$init$ שלכם לשים ערך $True$

משתנה זה יהיה רלוונטי בקטע הקוד רק במצב של תצוגה גרפית! משתנה מסוג מחרוזת שמכיל את צבע הקודקוד, שימו לב כי ניתן לשנות את הקודקוד במהלך הצביעה.	$color$
משתנה מסוג מספר שלם (int), מייצג את המרחק לפי בחירתכם (לדוגמה ניתן לבצע איתו פעילות למרחק מהשורש שלנו) **שימו לב!** משתנה זה מאותחל לאינסוף עבור כלל המחשבים.	$dist$
משתנה מסוג מספר שלם (int), מייצג את מזהה הרשת של ההורה של הקודקוד, כלומר, מי קיבל ממנו הודעה.	$parent$
שימו לב! משתנה זה לא נגיש למשתמש משתנה זה הינו תור ההודעות של אותו מחשב	$messageQueue$
שימו לב! משתנה זה לא נגיש למשתמש משתנה זה יכיל את קובץ הקוד שהמשתמש כתב ויריץ אותו	$algorithm\_file$

נעבור כעת לפונקציות שזמינות לכם בקלאס זה, לכל משתנה שצוין לפני כן קיימת פונקציית get .
שימו לב! פונקציות set אשר מקבלות את הערך החדש ומעדכנות אותו, קיימות אך ורק למשתנים הבאים:

- $state$
- $root$
- $color$
- $dist$
- $parent$

שאר המשתנים לא ניתנים לשינוי!

עבור $class\ communication$:

$method\ name$	$parameters$	$description$
$send\_message$	$source\ ,destination\ ,meesage$	פונקציה לשליחת הודעה, מקבלת את הפרמטרים הבאים: • $source$: מזהה המחשב ששולח את ההודעה (הid שלו) • $destination$: כתובת היעד שלנו (הid שלו) • $message$: תוכן ההודעה שאנחנו מעבירים
$send\_to\_all$	$source\ ,message$	פונקציה זו משתמשת בפונקציה $send\_message$ ושולחת הודעה מה$source$ לכל השכנים של המחשב ברשת. באותו אופן $source$ הוא הid של המחשב שמפיץ את ההודעה ו$message$ הוא תוכן ההודעה שנעביר
$recieve\_message$	$message\ ,communication$	**שימו לב! פונקציה זו אינה זמינה למשתמש** פונקציה זו מחזירה את ההודעה למחשב הרלוונטי ואת התוכן שלה. המערכת בעצם משתמשת בפונקציה זו בכך שהיא בעצם מפעילה אותה בכל סבב של המערכת ואנחנו מקבלים את ההודעה שהתקבלה באחת מהפונקציות $mainAlgorithm$ או $init$ שאנחנו רושמים

אלגוריתם לדוגמה למערכת:

בקוד הבא נצרף את אלגוריתם הbroadcast וכיצד עליכם לכתוב אותו.

someAlgorithm.py > ...

```python
import computer
import communicationModule

''' user implemented code that runs a broadcast algorithm'''

def mainAlgorithm(self: computer.Computer, communication : communicationModule.CommunicationModule, message):
    if  self.state != "terminated":
        communication.send_to_all(self.id, "running a broadcast")
        self.setColor("#7427e9")
        self.setState("terminated")


def init(self: computer.Computer, communication : communicationModule.CommunicationModule):
    if (self.getRoot()):
        print(self.id, " is root")
        communication.send_to_all(self.id, "running a broadcast")
        self.setColor("#000000")
        self.setState("terminated")


def main():
    pass

if __name__=="__main__":
    main()
```

כאשר, שתי הספריות מיובאות בראש הקוד.

לאחר מכן אנו ניצור שתי פונקציות:

- פונקציית init: פונקציה זו מאתחלת את הרשת שלנו ותפעיל אותה.
- פונקציית mainAlgorithm: פונקציה זו תהווה את האלגוריתם שאנחנו רוצים להריץ (במקרה שלנו זה אלגוריתם broadcast אשר מפיץ לכל המחשבים שלא סיימו את ההודעה), והיא מקבלת את המחשב, קלאס התקשורת ואת ההודעה (במידה ולא תתקבל הודעה המחרוזת תהיה ריקה ואין שימוש בה, באלגוריתמים הדורשים קבלת מידע ולהסתמך עליו ניעזר בזה).

הערה:

נשים לב כי ישנן פקודות של setcolor שהינן באחריות המשתמש לקבוע.

משמעות הצבעים הינן להבדלה בין המצבים השונים שמחשב יכול לעבור ביניהם במהלך ריצת האלגוריתם.

בדוגמה, בפונקציית האתחול קבענו את הקודקוד להיות בצבע שחור על מנת לסמן אותו.

לאחר מכן שאר הקודקודים יצבעו בצבע כחול כהה בעת קבלת ההודעה.