סדנא לפרויקטים 89-385

סריקה מקושרת של רובוטים במבנים

מגישה: הילה זכריה

מנחה : ד"ר נועה אגמון

תוכן עניינים

תיאור כללי של הפרויקט ותוצריו

מטרת הפרויקט :

בהינתן מיפוי של מפת מבנה מסוים לגרף קשיר ללא מעגלים (עץ), נרצה לבצע את סריקתו של המבנה על ידי רובוטים . תהליך הסריקה צריך להתבצע ביעילות מרבית ובזמן הקצר ביותר על ידי הרובוטים תוך שמירה על מרחק תקין ביניהם וכן שמירה על תקשורת תמידית בין הרובוטים לאורך כל תהליך הסריקה .

הפרויקט כולל בתוכו :

- ניתוח ועיבוד המידע מתוך הקלטים לתוכנית .
- בניית הגרף והקמת תשתית לצורך ביצוע הסריקה.
- ביצוע סריקת הגרף על ידי הרובוטים בצורה היעילה ביותר .
- הצגה ויזואלית של הגרף ותהליך הסריקה.

בספר זה אפרט על כל חלק בנפרד כאשר בחלק של ניתוח ועיבוד המידע אפרט כיצד מתקבלים הקלטים ,וכיצד אנו מנתחים אותם לצורך בניית הגרף והקמת התשתית לביצוע הסריקה .

בחלק השני אציג את בניית הגרף וקמת תשתית לצורך ביצוע הסריקה .

לאחר מכן, ארחיב על אופן ביצוע תהליך הסריקה של הגרף על פי האלגוריתם של הגברת מור סיני וכן ביצוע היוריסטיקות לצורך שיפור יעילות הסריקה וביצועה בזמן האופטימלי ביותר.

לבסוף, אתאר כיצד ניתן להציג בצורה ויזואלית את מבנה הגרף ותהליך הסריקה.

התוצר הסופי של התוכנית הוא לכתוב לקובץ טקסט את תהליך הסריקה של הגרף כאשר אנו מציינים עבור כל קודקוד בכל נקודת זמן מי הם הרובוטים שממוקמים אצלו (במידה ויש אצלו רובוטים),ואחרת מציינים כי אין אצלו רובוטים.

מבנה הפרויקט

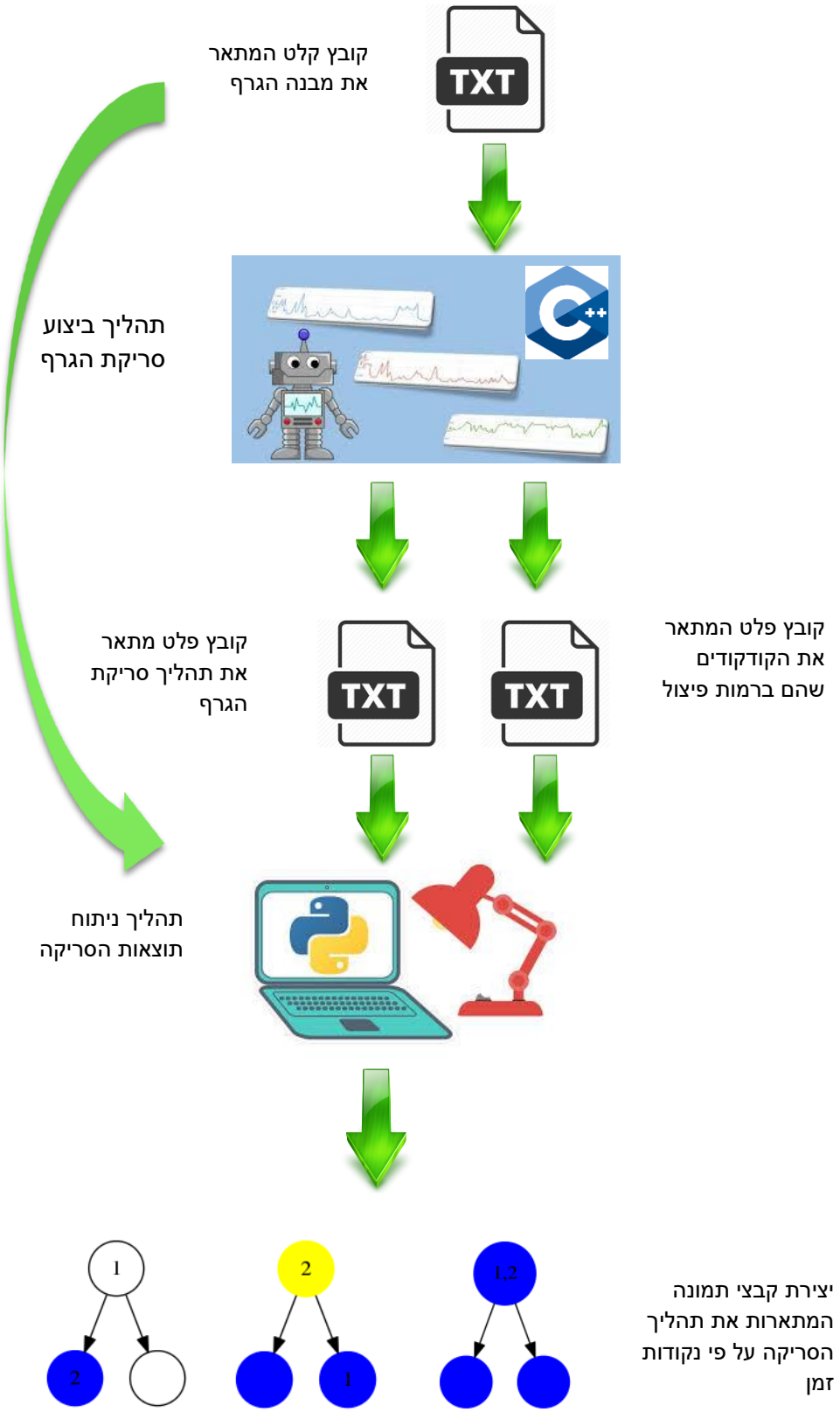
הפרויקט בנוי משני חלקים :

חלק של ביצוע תהליך סריקת הגרף ,חלק זה כתוב בשפת **C++** .אנו מקבלים כקלט קובץ המתאר את בניית הגרף לצורך הסריקה ופרמטרים נוספים (אציגם בהמשך) ומייצר את הקבצים הבאים :

"desplite.txt" – קובץ בו רשומים כל הקודקודים שנמצאים ברמות פיצול. כדי לצור אבחנה בין סוגי הקודקודים, נקבע כי קודקוד פיצול יהיה קודקוד שצורתו מלבנית וקודקוד רגיל יהיה קודקוד שצורתו עגולה .

קובץ **"scanning_for_Visual_presentation.txt"** –בקובץ זה מתואר תהליך הסריקה של הגרף על ידי הרובוטים מפורט לפי זמנים ,כאשר בכל נקודת זמן בקובץ כתובים קודקודי הגרף, ה-**state** שלהם ואיזה רובוטים ממוקמים אצלם .

חלק שני המבצע את הצגת תהליך הסריקה בצורה ויזואלית, חלק זה כתוב בשפת **python** ומקבל כקלט את הקבצים :**"scanning_for_Visual_presentation.txt"**," **"desplite.txt** " שיצרה התוכנית בחלק הראשון וכן את הקובץ המתאר את מבנה הגרף .
הפלט של התוכנית הוא קבצי תמונה **"png"** כאשר כל תמונה מייצגת נקודת זמן מסוימת בתהליך הסריקה.


קובץ קלט המתאר
את מבנה הגרף
TXT
תהליך ביצוע
סריקת הגרף
קובץ פלט המתאר
את הקודקודים
שהם ברמות פיצול
TXT
TXT
קובץ פלט מתאר
את תהליך סריקת
הגרף
תהליך ניתוח
תוצאות הסריקה
יצירת קבצי תמונה
המתארות את תהליך
הסריקה על פי נקודות
זמן
1
2
2
1
1,2

ניתוח ועיבוד המידע

התוכנית מקבלת שלושה קלטים מהמשתמש הניתנים כארגומנטים לפונקצית ה- **main** של התוכנית :

- נתיב (**path**) לקובץ קלט -
  בקובץ זה מתואר מבנה הגרף שנוצר מתהליך מיפוי מפת המבנה לגרף קשיר ללא מעגלים ,יש לתת כקלט לתוכנית את הנתיב בו שמור הקובץ ומהנתיב הזה התוכנית תקרא את תוכן הקובץ.

- מספר שלם **K** - מספר הרובוטים שיסרקו את הגרף .

- רשימת "**רמות פיצול**"-
  רשימה של הרמות בגרף בהן נרצה לבצע פיצול של הרובוטים
  אל הקדקודים הנמצאים ברמות הנמוכות יותר (הבנים ותתי העצים שלהם)
  באופן שווה .

סדר קבלת הארגומנטים :

הארגומנט הראשון הינו הנתיב לקובץ הקלט המתאר את מבנה הגרף, הארגומנט השני הינו מספר שלם שמייצג את מספר הרובוטים שיסרקו את הגרף והארגומנט האחרון הינו רשימה של רמות הפיצול בגרף.

לצורך בניית הגרף, התוכנית קוראת את תיאור הגרף מהקובץ קלט ,ומנתחת את תוכן הקובץ על מנת לצור את הקודקודים והקשתות בגרף .

אחר כך התוכנית יוצרת את הרובוטים שבאמצעותם נבצע את סריקת הגרף. מספר הרובוטים שייווצרו יהיו כמספר הרובוטים שנתנו בקלט ומספרם יתחיל מ-1 ועד **k** . עבור כל רובוט אנו נשמור את הקודקוד הנוכחי שהוא נמצא בו ואת הקודקוד הבא אליו הוא צריך לעבור בנקודת הזמן הבאה של הסריקה .

בנוסף, התוכנית יוצרת רשימה של כל הרמות בהן אנו נבצע פיצול של הרובוטים בגרף על פי רשימת רמות הפיצול שנתנה בקלט .

בניית הגרף והקמת תשתית לצורך ביצוע הסריקה

בהינתן הנתיב לקובץ הקלט התוכנית קוראת את תוכן הקובץ ותוך כדי ביצוע הקריאה , מתחילה לבנות את הגרף על ידי המחלקה "**Graph**" (פירוט המחלקה בחלק של ה"תיעוד למתחזק").

התוכנית יוצרת את הקודקודים כאשר לכל קודקוד יש תווית שם שנתנה לו בקובץ הקלט (מספר שלם) וכן יוצרת את הקשתות בין הקודקודים .

הגרף בתוכנית מיוצג כאוסף של כל קודקודי הגרף כאשר לכל קודקוד יש פוינטר לאבא שלו ורשימה של פוינטרים לבנים שלו (במידה והוא לא עלה) כך שניתן לגשת לשכנים של הקודקוד בתהליך הסריקה ביעילות ובמהירות .

אחר כך אנו מגדירים עבור כל קודקוד באיזה רמה הוא נמצא בגרף . הרמה של כל קודקוד נקבעת על פי העומק המירבי של תת העץ הספציפי של הקודקוד ולא על פי עומק הגרף כולו כמקובל .לצורך כך התוכנית עוברת על כל העץ מהעלים כלפי השורש ברקורסיה
כלומר, אנו מתחילים את המעבר על הקודקודים מהעלים שהרמה שלהם היא 0 ועולים כלפי השורש כאשר תוך כדי המעבר אנו מחשבים לכל קודקוד פנימי מהו עומק תת העץ המקסימלי שלו ומוסיפים 1 עבורו - כך אנו קובעים את הרמה של כל קודקדי הגרף .

בנוסף , עלינו לדעת מהו שורש העץ (לא ניתן כקלט) כיוון שזהו הקודקוד ממנו הרובוטים צריכים להתחיל לבצע את תהליך הסריקה , לצורך כך התוכנית שומרת תוך כדי חישוב הרמות של הקודקודים ,את הקודקוד שהרמה שלו היא הכי גבוהה ומסמנת אותו כשורש העץ.

בגרף כאמור, יש לנו קודקודים שנמצאים "ברמות פיצול" וקודקודים שאינם נמצאים "ברמות פיצול" ,כאשר הרובוטים מגיעים לקודקוד מסויים עליהם לדעת כיצד לנוע מהקודקוד לעבר קודקוד אחר בהתאם לסוג הרמה שבה נמצא הקודקוד וכן בהתאם לעוד מספר פרמטרים שארחיב עליהם בחלק של תהליך הסריקה .אם הרמה של הקודקוד הינה "רמת פיצול", כפי שהמשתמש הכניס כקלט לתוכנית אזי מצוין עבור קודקוד זה כי הוא נמצא ב"רמת פיצול", ואחרת מצוין עבור הקודקוד כי הוא אינו נמצא ב"רמת פיצול" .

מבוא לתהליך סריקת הגרף

בחלק זה אציג ואפרט על תהליך ביצוע סריקת הגרף על ידי הרובוטים . נרצה שתהליך סריקת הגרף יתבצע במהירות וביעילות על פי האלגוריתם "**NCOCTA**" מתוך המאמר : "**Maintaining Communication in Multi-Robot Tree Coverage**" .

לצורך ביצוע הסריקה ניתן לכל קודקוד **state** (מצב סריקה) מתוך שלושת המצבים הבאים (פירוט על המצבים בעמוד הבא) :

Not visited, inhabited, visited

המצבים הללו מהווים את מצב הסריקה של כל קודקוד עד לנקודת הזמן הנוכחית בסריקה . טרם תהליך הסריקה כל הקודקודים מאותחלים להיות במצב **not visited** כיוון שהרובוטים עוד לא החלו בסריקת הגרף וגם כל הרובוטים ממוקמים בקודקוד ההתחלתי (שורש העץ). בסופו של תהליך הסריקה כל הקודקודים יהיו במצב **visited** כיוון שהרובוטים יסיימו לסרוק את הגרף ,וכל הרובוטים יתמקמו חזרה בקודקוד ההתחלתי.

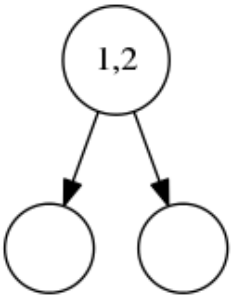
מהירות הסריקה נמדדת לפי משך הזמן שלקח לרובוטים לסרוק את כל הגרף, החל מנקודת הזמן 0 בה כל הקודקודים במצב **not visited** וכל הרובוטים ממוקמים בקודקוד ההתחלתי ועד לנקודת הזמן **t** בה כל הקודקודים במצב **visited** והרובוטים ממוקמים שוב בקודקוד ההתחלתי.

בשביל לייעל את הסריקה בצעתי מספר היוריסטיקות על האלגוריתם אשר שיפרו משמעותית את משך זמן הסריקה של הגרף והביאו לכך שכמות גדולה יותר של קודקודים נסרקה במינימום משאבים ובמספר אופטימלי של רובוטים .

תיאור המצבים שעוברים הקודקודים במהלך תהליך הסריקה

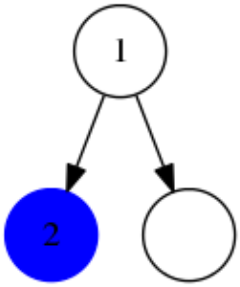
בכל נקודת זמן במהלך הסריקה ה-**state** של כל קודקוד משתנה לאחד מבין שלושת המצבים הבאים :

- **Not visited** – במצב זה יכולים להימצא כל קודקודי העץ אשר אינם נמצאים במצב **inhabited** או **visited**, כלומר קודקודים שתת העץ שלהם עדיין לא נסרק במלואו (ראה איור 1).



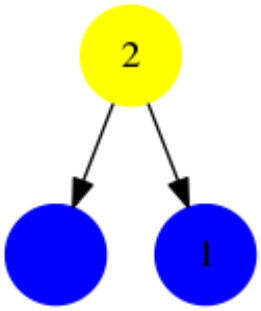
איור 1 : עדיין לא התחלנו לסרוק את העץ ,ישנם שני רובוטים הממוספרים ב1,2 אשר צריכים לסרוק את העץ

- **Visited** – במצב זה נמצאים קודקודי העלים שרובוטים סרקו אותם כבר או קודקודים שהרובוטים כבר סרקו את כל תת העץ שלהם (ראה איור 2).



איור 2 : רובוט מספר 2 ביקר בעלה ולכן העלה שינה את מצבו מ **not visited** ל**visited** שורש העץ נשאר **not visited** כיוון שהרובוטים סיימו לסרוק את כל תת העץ שלו .

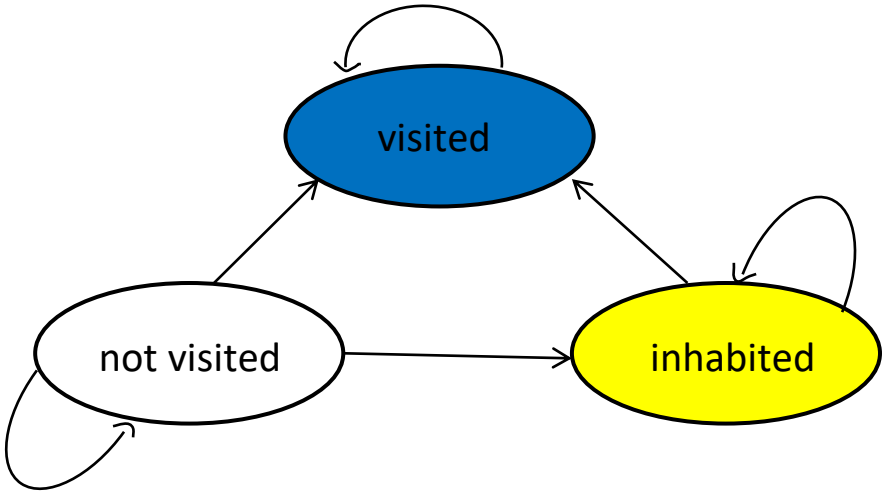
- **Inhabited** – במצב זה יכולים להימצא רק קודקוד השורש או קודקודים פנימיים בעץ אשר תתי העצים שלהם נסרקו על ידי רובוטים בזמן הנוכחי אך עדיין נמצאים רובוטים בתת העץ שלהם (ראה איור 3).



איור 3 : הרובוטים ביקרו בכל תת העץ של השורש ולכן שורש העץ שינה את מצבו מ **not visited** ל **inhabited**

כאמור,בתחילת הסריקה של הגרף כל הקודקודים נמצאים במצב **not visited** ובסיום הסריקה כל הקודקודים נמצאים במצב **visited** .

אוטומט המתאר את אפשרויות המעבר בין במצבים השונים של הקודקודים:



תיאור תהליך סריקת הגרף

תהליך הסריקה של הגרף מתבצע על פי האלגוריתם"**NCOCTA**" שכתבה הגברת מור סיני , באופן הבא:

בתחילת הסריקה הרובוטים נמצאים בשורש העץ. במהלך הסריקה הרובוטים נעים לכיוון העלים ואחר כך חוזרים מעלה אל השורש.

בכל נקודת זמן מתחילתו של תהליך הסריקה ועד סופו, הרובוטים יכולים לבצע מעבר על קשת אחת בעץ לכל היותר בהתאם לתנאי האלגוריתם.על מנת לחשב את אופן מעבר הרובוטים בסריקה, בכל נקודת זמן אנו עוברים על כל הקודקודים $v \in V$ שיש בהם רובוטים, כאשר אנו מתחילים מהקודקודים הנמצאים ברמות הנמוכות אל עבר הקודקודים הנמצאים ברמות הגבוהות יותר. אנו מחשבים לפי המצב הנוכחי של כל $v \in V$ כזה את המעבר של הרובוטים בגרף בנקודת הזמן הבאה .

תהליך הסריקה מתבצע באופן הבא :

אם הקודקוד v נמצא במצב **visited** אזי :

- סיימנו לסרוק את תת העץ של הקודקוד v ולכן כל הרובוטים שנמצאים בקודקוד זה עולים חזרה לאבא של הקודקוד v.
- אם הקודקוד v הוא שורש העץ (אין לו אבא) אזי כל הרובוטים נשארים בקודקוד v וסיימנו את תהליך הסריקה של כל העץ .

אם הקודקוד v נמצא במצב **not visited** וגם יש לו בן u שעדיין לא סיימנו לסרוק את תת העץ שלו אזי :

- אם הקודקוד v נמצא **ברמת פיצול** :

רובוט אחד יישאר בקודקוד האב v לצורך שמירה על תקשורת וסנכרון ושאר הרובוטים יתחלקו שווה בשווה בין הבנים של קודקוד האב v.

אם יש ל-v בן w שאין לו צורך בעוד רובוטים בשל אחת מהסיבות הבאות :

- סיימנו לסרוק את תת העץ של הבן w , קרי סטטוס הקודקוד w הוא **visited** או **inhabited** .
- יש בתת העץ של קודקוד הבן w מספיק רובוטים שיסרקו את תת העץ שלו ולכן אין צורך לקודקוד הבן בעוד רובוטים נוספים.

אזי, קודקוד האב v ייתן את הרובוטים שיש באפשרותו לתת לבן אחר (שאינו w) אשר צריך עוד רובוטים. מספר הרובוטים שצריך כל קודקוד לצורך סריקת תת העץ שלו שווה למספר הקודקודים שנמצאים במצב **not visited** בנקודת הזמן הנוכחית בסריקה.

- אם הקודקוד v **אינו** נמצא ברמת פיצול אזי :

  - אם מספר הרובוטים שקודקוד האב v יכול לתת לקודקוד הבן u גדול מסך הרובוטים שקודקוד הבן u אכן צריך לצורך סריקת תת העץ שלו .קודקוד האב v יעביר לקודקוד הבן u את מספר הרובוטים שהוא צריך , רובוט אחד יישאר אצל v לצורך תקשורת וסנכרון ואת שאר הרובוטים שנשארו (במידה ונשארו עוד רובוטים שניתן לתת) ניתן לבן אחר של קודקוד האב v שצריך עוד רובוטים .

  - אם מספר הרובוטים שקודקוד האב v יכול לתת לקודקוד הבן u קטן או שווה לסך הרובוטים שקודקוד הבן u צריך אזי כל הרובוטים שנמצאים בקודקוד האב v יעברו לקודקוד הבן u.

אחרת, אם הקודקוד v נמצא במצב **inhabited** אזי סיימנו לסרוק את תת העץ של הקודקוד v ,אבל עדיין ישנם רובוטים שנמצאים בתת העץ של v ,לכן רובוט אחד ישאר בקודקוד v על מנת לשמור על תקשורת עם אותם רובוטים שנמצאים בתת העץ וטרם עלו חזרה לקודקוד v , ושאר הרובוטים שהיו בקודקוד v יעברו לאבא של v.

אם הקודקוד v הוא שורש העץ (אין לו אבא) אזי כל הרובוטים נשארים בקודקוד v ויחכו עד שכל הרובוטים שנמצאים בתת העץ יעלו חזרה לשורש v.

היוריסטיקות לאלגוריתם

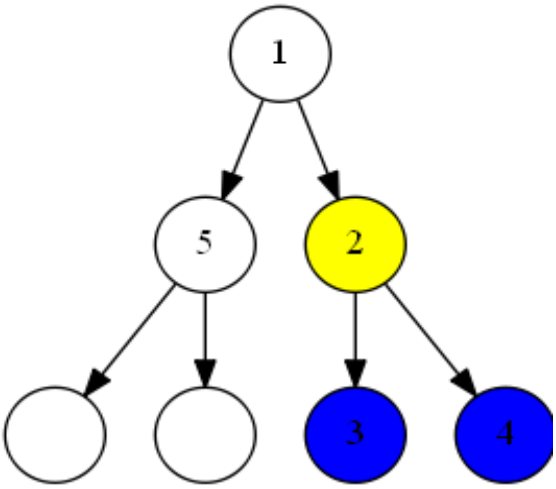
בנוסף למימוש אלגוריתם הסריקה בצעתי מספר היוריסטיקות לאלגוריתם על מנת להקטין את משך זמן הסריקה ולשפר את ביצועי האלגוריתם. בחלק זה אסביר את היוריסטיקות השונות , הבעיות אותן הן מנסות לפתור וכיצד הן מבוצעות .

הבעיה : כייצד לבצע את המעבר על כל הקודקודים בגרף , מאילו קודקודים להתחיל? , על אילו קודקודים כדאי לנו לעבור על מנת לייעל את תהליך הסריקה?

פיתרון : נעבור רק על הקודקודים שיש בהם רובוטים ולא על כל הקודקודים בגרף ,כיוון שרק מהקודקודים הללו הרובוטים יכולים לנוע בנקודת הזמן הבאה לעבר קודקודים אחרים . בנוסף, כשאנו עוברים על כל הקודקודים , נתחיל מהקודקודים שנמצאים מהרמות הנמוכות לעבר הקודקודים שנמצאים ברמות הגבוהות , כאשר עבור כל רמה אנו מתחילים לסרוק קודם את הקודקודים שנמצאים במצב **visited** אחר כך את הקודקודים שנמצאים במצב **inhabited** ולבסוף את הקודקודים שנמצאים במצב **not visited** .

בצעתי את הפתרון על ידי כך ששמרתי רשימה של פוינטרים לכל הקודקודים בהם נמצאים רובוטים , ומיינתי את הרשימה לפי רמות כאשר אם היו לנו שני קודקודים במהלך המיון שהיו באותה הרמה ,מיינתי אותם לפי מצבם על פי מתן העדיפויות שתיארתי לעיל .

שרטוט לצורך הדגמה :



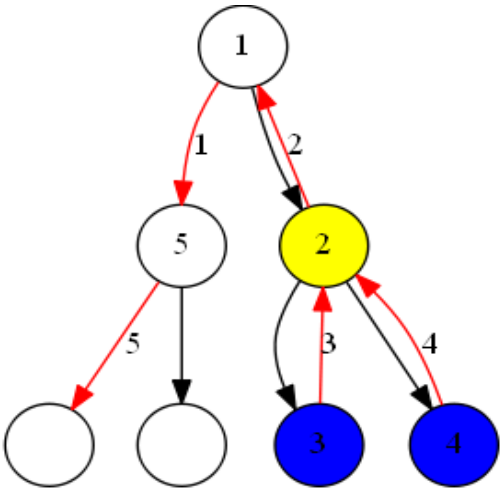
המספר בכל קודקוד מהווה את שמו של הרובוט שנמצא כעת בקודקוד. אנו מתחילים מהעלים שנמצאים במצב **not visited** אחר כך עוברים לרמה אחת מעל וסורקים קודם את הקודקוד שנמצא במצב **inhabited** (הצהוב) לפני הקודקוד שנמצא במצב **not visited** (הלבן) ואחר כך עוברים לרמה מעל וכן הלאה.

<u>הבעיה :</u> ישנו קודקוד אב שיש לו עוד בנים הנמצאים במצב **not visited** , ובקודקוד האב יש רק רובוט אחד. על מנת שהוא יוכל לתת לבנים שלו רובוטים עליו לחכות שהוא יקבל רובוטים מקודקוד האב שלו, אך בתהליך הסריקה אנו סורקים לפי רמות כך שקודם עוברים על הבנים לפני האבות לכן כאשר הבן יקבל את הרובוטים הוא לא מעביר את הרובוטים שהיו אצלו לבן שלו .

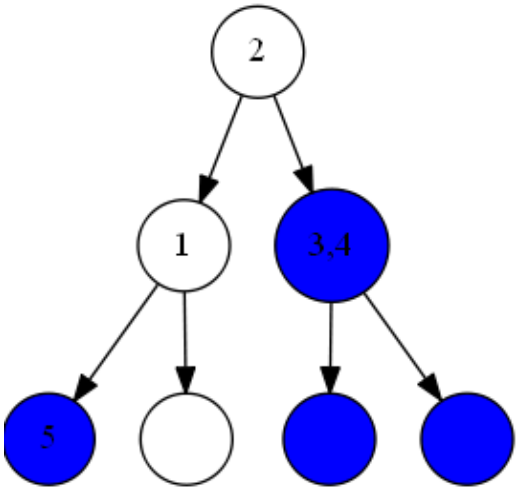
<u>פיתרון :</u> כאשר אנו עוברים על קודקוד האב (אחרי שסרקנו את הבן) וקודקוד האב מעביר רובוטים לבן שלו , אנו נסרוק שוב את הבן על מנת להעביר את הרובוטים הלאה לבנים ברמות הנמוכות יותר ובכך נשפר את משך זמן הסריקה הכולל של הגרף, במקום לחכות שהרובוטים ינועו לבנים בנקודת הזמן הבאה .

בצעתי את הפיתרון על ידי כך שאם נתון לנו המקרה שתיארתי בבעיה במהלך הסריקה ,אנו נבצע סריקה נוספת בצורה רקורסיבית רק עבור הקודקודים שנמצאים בתת העץ של הבן שקיבל את הרובוטים מהאב , ועל ידי כך נצליח לבצע את מעבר הרובוטים בתת העץ של הבן בצורה האופטימלית ביותר .

שרטוט לצורך הדגמה :



בשרטוט החצים האדומים מתארים את המעבר של הרובוטים עבור נקודת הזמן הבאה . אם לא היינו סורקים שוב את הקודקוד שרובוט מספר 5 נמצא בו לאחר שהוא מקבל מאביו את רובוט מספר 1 היינו מקבלים כי בבן היו נמצאים רובוטים 1,5 והיינו מאבדים מהיעילות , לכן כאשר אנו נסרוק שוב את הקודקוד שרובוט 5 נמצא בו הוא יעביר את רובוט מספר 5 לבן שלו וכתוצאה מכך נקטין את משך זמן הסריקה של הגרף. לאחר המעבר נקבל כי סריקת הגרף בנקודת הזמן הבאה תראה כך:

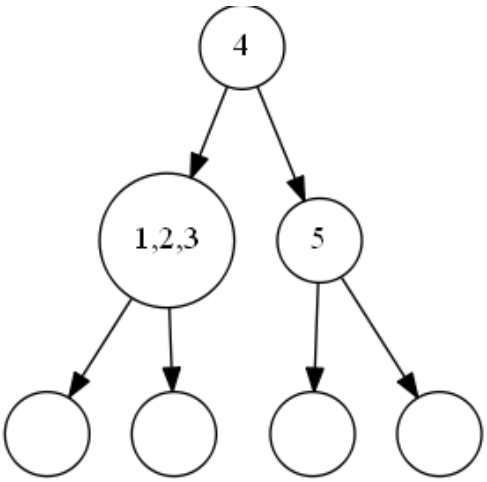


<u>הבעיה :</u> לקודוקד האב יש מספר בנים שהם **not visited** והוא נמצא ברמה שנמצאת מעל רמת פיצול . מספר הרובוטים שקודקוד האב יכול לתת לקודקוד הבן מסויים גדול מסך הרובוטים שקודקוד הבן צריך לצורך סריקת תת העץ שלו ,לכן את שאר הרובוטים הוא יכול להעביר לבן הנוסף שנמצא במצב **not visited** אבל אז אם נעביר את שאר הרובוטים לבן הנוסף ייווצר נתק בין הרובוטים ולא תהיה תקשורת ביניהם .

<u>פיתרון :</u> במקרה שקודקוד אינו נמצא ברמת פיצול, לפני שנעביר את הרובוטים הנותרים לבן הנוסף נוודא שאכן נשאר רובוט אחד לפחות בקודקוד האב לצורך שמירה על התקשורת בין הרובוטים .

בצעתי את הפתרון על ידי בצוע חישוב מקדים בטרם מעבר הרובוטים בגרף ושימוש במשתנה flag שאותחל לfalse , ובמידה ותוצאת החישוב הייתה כי צריך לשמור על רובוט בקודקוד האב לצורך תקשורת ,שיניתי את הflag להיות true.

שרטוט לצורך הדגמה :

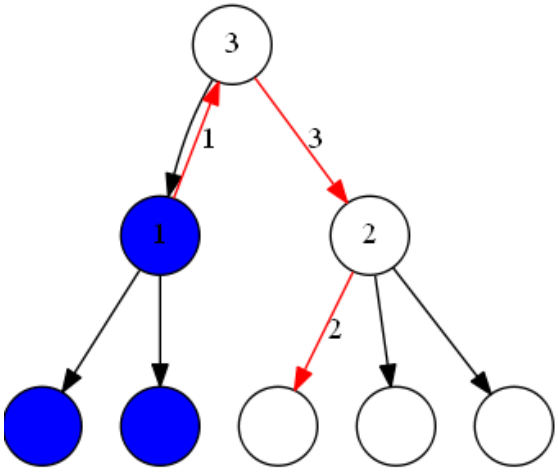


בשרטוט יש לנו 5 רובוטים שסורקים את הגרף .הבן השמאלי של שורש העץ צריך 3 רובוטים לצורך ביצוע הסריקה של תת העץ שלו (מספר הרובוטים הנדרש הוא כמספר הקודקודים בתת העץ שהם במצב **not visited**) וכך גם הבן הימני אך אין לנו מספיק רובוטים לבן הימני ולכן אנו נותנים לו את מספר הרובוטים שנשארו לאחר שהשארנו רובוט אחד לפחות בקודקוד האב (שורש העץ).

<u>בעיה:</u> לקודקוד האב יש רובוטים בתתי עצים שונים ,אבל תת עץ של בן אחד סיימנו לסרוק ותת עץ של בן אחר עוד לא סיימנו לסרוק ,איך נדע האם צריך לשמור רובוט אחד אצל האב לצורך תקשורת או שאין צורך לשמור רובוט אצל האב לצורך תקשורת .

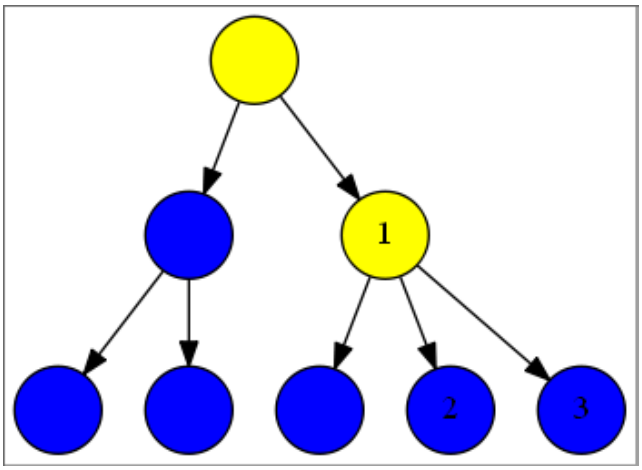
<u>פיתרון וביצוע:</u>
כתבתי פונקציה בשם chackIfNeedToKeepRobotForCommunication שבודקת האם יש רובוטים בתתי עצים של בנים שונים , והאם סיימנו לסרוק את תת עץ של בן מסויים או לא,אם סיימנו אז אנו לא צריכים לשמור רובוט אצל קודקוד האב לתקשורת,ואחרת צריך לשמור רובוט אצל קודוקד האב לצורך תקשורת.

שרטוט לצורך הדגמה :



בשרטוט החצים האדומים מתארים את המעבר של הרובוטים בנקודת הזמן הבאה ,לאחר שסיימנו לסרוק את תת העץ של הבן השמאלי של שורש העץ ולכן אין צורך להמשיך להשאיר רובוט בשורש לצורך תקשורת כיוון שבסריקה הבאה כל הרובוטים יהיו בתת העץ של הבן הימני .

לאחר המעבר וחישוב נקודת הזמן הבאה הגרף יראה כך :

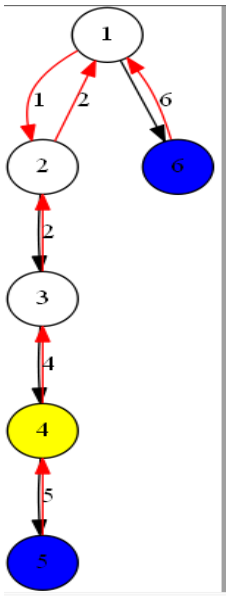


<u>הבעיה</u> : מצב של קודקוד מסויים השתנה ל-**inhabited** כלומר סיימנו לסרוק את תת העץ אבל יש עוד רובוטים בתת העץ שלו .הבעיה היא שהקודקודים שנמצאים ברמות מעליו אינם יודעים ואז הם עלולים להעביר לתת העץ של קודקוד זה עוד רובוטים שלא לצורך ובכך נאבד מהיעילות של הסריקה ונגדיל את משך זמן הסריקה של הגרף כולו.

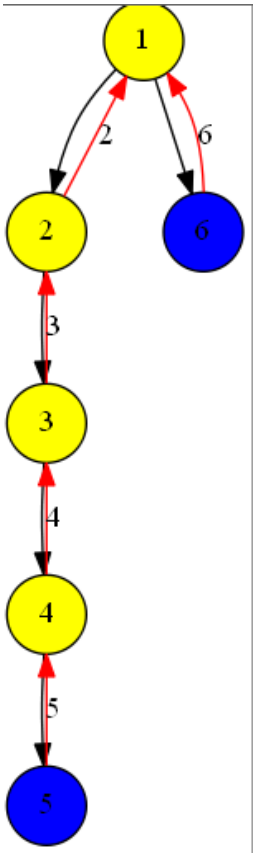
<u>פיתרון</u> : כאשר קודקוד משנה את מצבו ל-**inhabited** נשנה את המצב של כל האבות הקדמונים שלו עד השורש גם כן ל-**inhabited** על מנת שהרובוטים ידעו כי סיימנו לסרוק את תת העץ והרובוטים צריכים כעת לעלות חזרה לשורש העץ.

בצעתי את הפיתרון על ידי כך שברגע שקודקוד מסויים שינה את מצבו ל-**inhabited** שיניתי באופן אטרטיבי אחורה את המצב של כל האבות הקדמונים שלו ל **inhabited** גם כן בהתאם. כזכור לכל קודקוד יש פוינטר לקודקוד האב שלו כך שאנו יכולים לגשת לקודקוד האב בצורה ישירה ובעלות מינימלית .

שרטוט לצורך הדגמה :



בשרטוט החצים באדום מראים את המעבר שיעשו הרובוטים בנקודת הזמן הבאה . ניתן לראות כי סיימנו לסרוק את תת העץ של הבן השמאלי של השורש, וכעת נותר לנו רק להעלות את הרובוטים חזרה ,אבל השורש נתן את רובוט מספר 1 לבן השמאלי שלא לצורך . לכן לפי הפתרון לעיל הגרף יראה כך ויציג שיפור טוב לתהליך הסריקה :



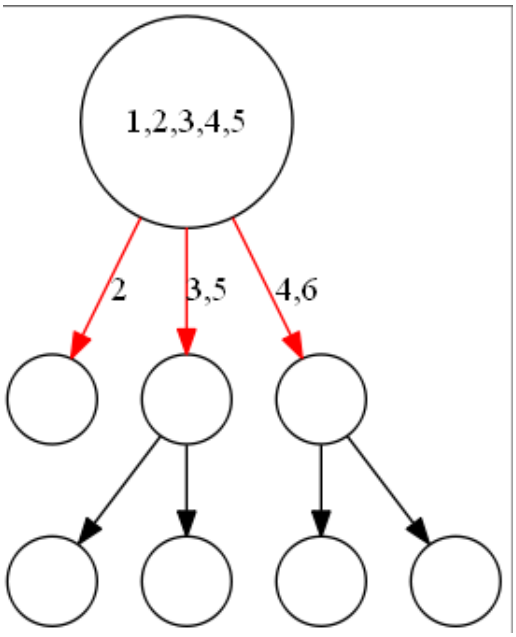
כעת אנו יודעים כי סיימנו לסרוק את תת העץ של הבן השמאלי והרובוטים שנמצאים בו צריכים לעלות חזרה לשורש ,לכן רובוט מספר 6 יעלה לשורש ויישאר בו עד תום ביצוע סריקת כל הגרף ושאר הרובוטים יתחילו לבצע עלייה לכיוון השורש.

<u>הבעיה</u>: ישנו קודקוד שנמצא בנקודת פיצול ,ויש לו בנים שהם **visited** או **inhabited** ובנים שהם במצב **not visited**. הבעיה היא איך קודקוד האב יחלק את הרובוטים בין הבנים שלו בצורה יעילה ככל האפשר,בין אילו בנים יחלק את הרובוטים ?

<u>פיתרון:</u> אם לקודקוד אב שנמצא ברמת פיצול יש רובוטים לחלק הוא יחלק אותם שווה בשווה בין כל הבנים ,אבל אם יש בן שהוא במצב **visited** או **inhabited** והוא אינו צריך עוד רובוטים קודקוד האב יעביר את הרובוטים שהוא רצה לחלק לו לקודקוד בן אחר לפי הצורך.

בצעתי את הפתרון על ידי כך שכל קודקוד אב שנמצא בנקודת פיצול מחלק את הרובוטים אחד אחד כאשר לפני כל חלוקה של רובוט לבן מסויים מתבצעת בדיקה האם הבן אכן צריך את הרובוט שאנו עומדים לתת לו או לא , במידה ולא אנו נעביר את הרובוט לקודקוד אחר שכן צריך .

שרטוט לצורך הדגמה :



נניח כי רמה מספר 2 (הרמה של השורש) היא רמת פיצול . החצים האדומים מתארים את המעבר של הרובוטים בנקודת הזמן הבאה , ניתן לראות כי הבן השמאלי ביותר קיבל רק רובוט אחד (רובוט מספר 2) כיוון שמיד לאחר שנעביר לקודקוד זה את רובוט מספר 2 הוא ישנה את מצבו ל **visited** ולכן לא יצטרך עוד רובוטים, וכתוצאה מכך קודקוד האב יחלק את מספר הרובוטים שנשארו בין שאר הבנים שווה בשווה . הודות לחלוקה הזאת הרווחנו חלוקה יעילה של הרובוטים שמביאה לשיפור במשך זמן סריקת הגרף.

ספר משתמש

הפרויקט מכיל את התוכניות הבאות :

- תוכנית בשפת **C++** אשר בונה את הגרף על פי קובץ הקלט , מבצעת את תהליך סריקת הגרף על ידי הרובוטים ולבסוף כותבת לקובץ טקסט את שלבי הסריקה שביצעה .

- תוכנית בשפת **python** אשר מציגה בצורה ויזואלית את המעבר של הרובוטים בין הקודקודים וכן שינוי המצבים של הקודקודים במהלך הסריקה על פי נקודות הזמן שלקח לנו לבצע את סריקת הגרף כולו.

התוכנית הראשונה מקבלת את הקלטים כארגומנטים לפונקצית ה**main** כאשר :

- הקלט הראשון יהיה הנתיב בו שמור הקובץ הקלט המתאר את מבנה הגרף . יש לוודא שאכן קיים קובץ בשם זה בנתיב ,ולוודא שאכן הנתיב תקין ויש גישה אליו.

- הקלט השני יהיה מספר הרובוטים שיסרקו את הגרף .
יש לשים לב כי קלט זה תקין ומספר הרובוטים הוא מספר שלם טבעי **(integer)** .

- הקלט שלישי הוא רשימה של כל הרמות בגרף בהן יתבצע הפיצול של הרובוטים. יש לשים לב כי הרמות חוקיות ואכן נמצאות בטווח הרמות של העץ כולו (מרמת העלים ועד לרמת השורש) .

<u>סביבת עבודה :</u>

ניתן להריץ את התוכנית במערכת הפעלה **Linux 32 bit / 64 bit**.

לצורך הרצת התוכנית יש להיכנס בטרמינל לנתיב בו שמור קובץ הריצה בשם **exe.out** לרשום את הפקודה **./exe.out** ולאחר מכן ל לתת את הקלטים בסדר שתיארתי לעיל.
לדוגמה :

./exe.out "/home/user/folder/graph_file.txt" 15 1 2 3

נתיב לקובץ קלט

מספר רובוטים

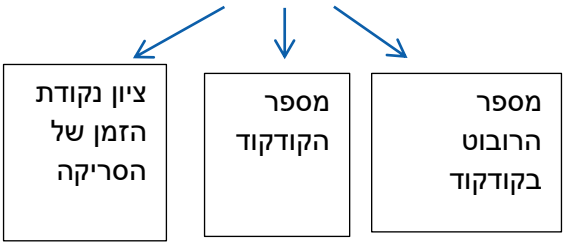
רשימה של רמות פיצול

התוכנית מבצעת את תהליך בניית הגרף על פי קובץ הקלט, מבצעת את תהליך סריקת הגרף וכותבת לתוך קובץ טקסט **"output_scaning.txt"** המתאר את תהליך הסריקה של הגרף על ידי הרובוטים מפורט על פי זמנים ,כאשר בכל נקודת זמן בקובץ כתובים קודקודי הגרף, ואיזה רובוטים ממוקמים אצלם .

בנוסף , התוכנית כותבת קובץ נוסף בשם **"desplite.txt"** בו כתובים שמות כל הקודקודים שנמצאים ברמת פיצול. התוכנית של הצגת הגרף ותהליך הסריקה בצורה ויזואלית משתמשת בקובץ זה כדי לדעת איך להציג את הקודקודים . וכן גם את קובץ הפלט **"scanning_for_Visual_presentation.txt"** בו מתואר תהליך הסריקה של הגרף על ידי הרובוטים מפורט לפי זמנים ,כאשר בכל נקודת זמן בקובץ כתובים קודקודי הגרף,ה**state** שלהם ואיזה רובוטים ממוקמים אצלם .

דוגמה לקובץ הפלט **"output_scaning.txt"**:

- 1 7 1
- 1 8 2
- 1 9 3



דוגמה לקובץ הפלט **"scanning_for_Visual_presentation.txt "**:

$$$$$$$$$$$$$$$$$$$$$
time:1
Node:0#current robots:1#state node:2
Node:1#current robots:2#state node:1
Node:2#current robots:#state node:2
****

כל נקודת זמן מופרדת על ידי רצף תווים של **$** ו* כדי להקל על ניתוח הקובץ בתוכנית הפייתון .
שורה ראשונה : ציון זמן הסריקה .
שורה שניה עד אחת לפני אחרונה : תיאור כל קודקוד בגרף ,תיאור הרובוטים שממוקמים אצלו ותיאור המצב של הקודקוד עד נקודת הזמן המצוינת מעלה .
State node יכול להיות כאמור אחד משלושת המצבים :

Visited - **state node** מספר 1
Inhabited – **state node** מספר 3
Not visited - **state node** מספר 2 .

דוגמה לקובץ הפלט **"desplite.txt "** :

1,15,13

קודקודים אלה נמצאים ברמות פיצול והם מופרדים ביניהם באמצעות פסיקים.

הקבצים **"scanning_for_Visual_presentation.txt "**,**"desplite.txt"** ",וקובץ הקלט המתאר את מבנה הגרף הם הקלטים לתוכנית הצגת תהליך הסריקה של הגרף בצורה ויזואלית שכתבתי בשפת **Python** אשר מציגה באופן ויזואלי את תהליך סריקת הגרף על ידי הרובוטים .

התוכנית משתמשת בתוכנת **open-source "Graphviz"** בעלת יכולת הדמיה לייצוג מידע מבניים כמו גרפים.

את התוכנית ב**Python** ניתן להריץ במערכת הפעלה **Linux 32 bit / 64 bit**.

לצורך הרצה יש לשים באותה תיקיה של הפרויקט את כל קבצי הקלט וקבצי התוכנית, לפתוח טרמינל ולהכנס לנתיב בו שמרנו את הקבצים . בטרמינל יש לכתוב את הפקודה :

python Draw_Graph.py ולאחר מכן תיווצר תיקיה בשם IMG בה ימצאו קבצי הפלט שהם תמונות ה"png" המתארות את תהליך סריקת הגרף.

התוכנית משתמשת במספר קבצים על פיהם היא מציגה את הגרף ותהליך הסריקה :

- קובץ **"input.txt"** -
  בקובץ זה מתואר מבנה הגרף שנוצר מתהליך מיפוי מפת המבנה לגרף קשיר ללא מעגלים ,יש לשמור את הקובץ באותו נתיב שנמצאים הקבצים של התוכנית .
  התוכנית תיגש לקובץ זה ותבצע קריאה ממנו לצורך בניית הגרף בצורה ויזואלית .

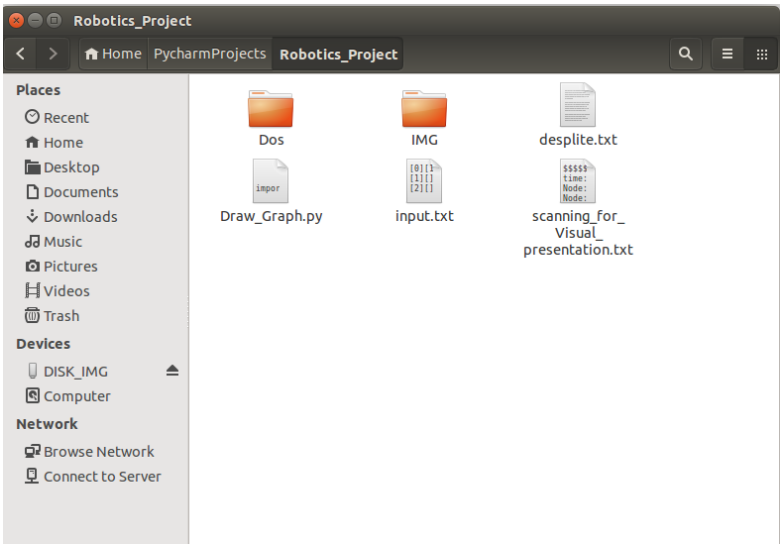
- **"desplite.txt"** – קובץ בו רשומים כל הקודקודים שנמצאים ברמות פיצול. כדי לצור אבחנה בין קודקוד פיצול לקודקוד רגיל , נקבע כי קודקוד פיצול יהיה קודקוד שצורתו מלבנית וקודקוד רגיל יהיה קודקוד שצורתו עגולה .

- קובץ **"scanning_for_Visual_presentation.txt"** – קובץ הפלט של תוכנית הסריקה הכתובה בשפת ה**C++** .בקובץ זה מתואר תהליך הסריקה של הגרף על ידי הרובוטים מפורט לפי זמנים ,כאשר בכל נקודת זמן בקובץ כתובים קודקודי הגרף,ה**state** שלהם ואיזה רובוטים ממוקמים אצלם .

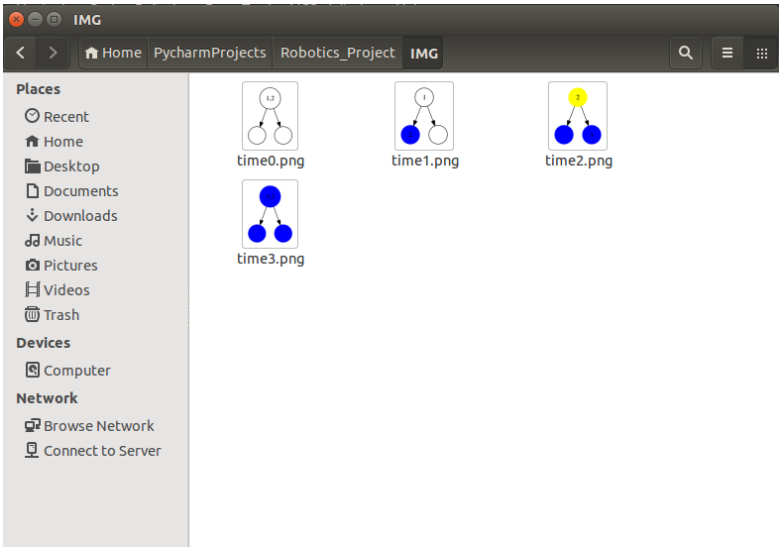
התוכנית בונה את הגרף על פי הקבצים לעיל ומבצעת הדמיה של הגרף ותהליך הסריקה על פי נקודות הזמן בסריקה .

הפלט של התוכנית היא קבצי תמונה "**png**" המתארות את תהליך סריקת הגרף על פי זמני הסריקה , מכאן שמספר קבצי התמונות הוא כמשך הזמן שלקח לנו לסרוק את הגרף .
תמונות ה"**png**" שיצרה התוכנית ממוקמות בתוך התיקייה **IMG** שנמצאת גם כן באותו הנתיב של הקבצים לעיל .

מבנה התיקייה בה שמורים קבצי התוכנית יחד עם קבצי הקלט בהם התוכנית משתמשת :



דוגמה לתוכן התיקייה **IMG** , בתיקייה זו התוכנית שומרים את תמונות ה"**png**" שיצרה , כל תמונה מתארת את מצב סריקה הזמן על פי נקודה מסוימת בתהליך הסריקה :



הוראות התקנה

טרם הרצת התוכנית על המשתמש להוריד את תוכנת ה-"**Graphviz**" באופן הבא :

שלב 1 : להכנס לקישור
http://pkgs.fedoraproject.org/repo/pkgs/graphviz/graphviz-2.38.0.tar.gz/5b6a829b2ac94efcd5fa3c223ed6d3ae/graphviz-2.38.0.tar.gz
ולהוריד משם את ה**package** של ממשק **graphviz** .
בנוסף יש להוריד את ה**patch** מהאתר הזה :
http://www.linuxfromscratch.org/patches/blfs/svn/graphviz-2.38.0-consolidated_fixes-1.patch
שלב 2 : לפתוח טרמינל במערכת ההפעלה **linux** .
שלב 3 :התקנת ה-**graphviz** ע"י הפקודות :

```
patch -Np1 -i ../graphviz-2.38.0-
consolidated_fixes-1.patch &&

sed -i '/LIBPOSTFIX="64"/s/64//' configure.ac &&

autoreconf                                  &&
./configure --prefix=/usr --disable-php   &&
make
```

לאחר מכן להכניס את הפקודה

```
make install
```

ולבסוף להשלים את אופן ההתקנה על ידי הפקודה :

```
ln -v -s /usr/share/graphviz/doc \
         /usr/share/doc/graphviz-2.38.0
```

.

תיעוד למתחזק

התוכנית שמבצעת את תהליך סריקת הגרף כתובה בשפה **C++** .
סביבת העבודה במערכת ההפעלה **Linux** סביבת העבודה היא **Eclipse**.

ספריות מרכזיות שהשתמשתי בהם בתוכנית :

<string> - This header introduces string types, character traits and a set of converting functions**.**

<iostream>- Header that defines the standard input/output stream objects.

<vector> - Vectors are sequence containers representing arrays that can change in size.

<algorithem>- The header <algorithm> defines a collection of functions especially designed to be used on ranges of elements.
A range is any sequence of objects that can be accessed through iterators or pointers, such as an array or an instance of some of the STL containers.

<sstream>- Header providing string stream classes.

<fstream>- Stream class to both read and write from/to files.

<map>- Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order.
In a map, the key values are generally used to sort and uniquely identify the elements, while the mapped values store the content associated to this key.

<list> - Lists are sequence containers that allow constant time insert and erase operations anywhere within the sequence, and iteration in both directions.
List containers are implemented as doubly-linked lists.

תוכנית זו מחולקת למספר חלקים :

- ניתוח ועיבוד המידע מתוך הקלטים לתוכנית:
ניתוח ועיבוד המידע נעשה על ידי המחלקה **readFromFile** ,ועל ידי פונקצית ה**main** שמקבלת את הקלטים מהמשתמש.

- בניית הגרף והקמת תשתית לצורך ביצוע הסריקה:
בניית הגרף נעשה על ידי המחלקות **Graph**, **Node** .המחלקה **Graph** מקבלת את תיאור הגרף מהמחלקה **readFromFile** ובונה על פיו את הגרף לצורך תהליך הסריקה .המחלקה **Node** אחראית על יצירת הקודקודים וכן ביצוע פעולות שונות על הקודקודים במהלך הסריקה שנדרשות לפי האלגוריתם ושמירת מידע רלוונטי לגבי קודקודי הגרף כגון: מהו המצב של הקודקוד כעת, מה יהיה במצב של הקודקוד בנקודת הזמן הבאה, פוינטר לקודקוד האב וכן רשימה של פוינטרים לבנים של הקודקוד ועוד .

- ביצוע סריקת הגרף על ידי הרובוטים :

ביצוע סריקת הגרף נעשה על ידי המחלקה **CalFutureGraph** שנעזרת במחלקה **CalNodeData** לצורך ביצוע חישובים על תתי העצים בגרף כגון : חישוב מספר הרובוטים הנוכחי בתת עץ של קודקוד מסויים בגרף, ספירת מספר הבנים שהם במצב not visited בגרף ,בדיקה האם קודקוד מסויים צריך לקבל עוד רובוטים ועוד .

- כתיבת תוצאות הסריקה לקובץ פלט :
בסופו של תהליך הסריקה אנו כותבים לקובץ טקסט על ידי המחלקה **WriteOutputToFile** את פירוט מהלך הסריקה , כאשר בכל נקודת זמן אנו מפרטים אילו רובוטים ממוקמים באילו קודקודים .

פירוט המחלקה המבצעת את תהליך הקריאה של הגרף מקובץ הקלט

שם המחלקה : "**ReadFromFile** "

תיאור : המחלקה אחראית על ביצוע קריאת מבנה הגרף מקובץ הקלט, וכן בניית הגרף לצורך הסריקה תוך כדי הקריאה .

פונקציות מרכזיות :

```
Void readInputFile(char* path);
```

קלט: path- הנתיב בו שמור קובץ מבנה הגרף.

תיאור : הפונקציה מבצעת את קריאת תוכן הגרף .

```
vector<string> split(string str, char delimiter);
```

קלט: str- מחרוזת אותה אנו רוצים לפרסר,**delimiter**-תו לפיו אנו מפרסרים את המחרוזת

תיאור : הפונקציה מפרסרת את המחרוזת קלט לטוקנים על פי התו **delimiter** ושומרת אותם בוקטור של מחרוזות. לבסוף, מחזירה את הוקטור המחרוזות .

```
vector<int> ReadFromFile::parseSonsString(string son_str);
```

קלט: son_str – מחרוזת של כל קודקודי הבנים

תיאור: הפונקציה מנתחת את מחרוזת הקלט, ומחזירה וקטור שמות של כל קודקודי הבנים .

פירוט המחלקות המבצעות את תהליך בניית הגרף

שם המחלקה : "Graph"

תיאור : המחלקה אחראית על תהליך בניית הגרף וייצוגו בתוכנית.המחלקה שומרת את ייצוג הגרף על ידי רשימת כל הקודקודים בגרף ושמירת שורש העץ, כאשר לכל קודקוד יש פוינטר לקודקוד האב שלו, וכן רשימה של פוינטרים לבנים שלו.

פונקציות מרכזיות :

```
Node* create_New_Node(int name);
```

קלט: **name** – מספר שלם המייצג את תווית שם הקודקוד.

תיאור: הפונקציה יוצרת קודקוד חדש בשם **name** ומחזירה פוינטר לקודקוד זה .

```
void addEdge(Node* v, Node* w);
```

קלט: **v,w** – פוינטר לשני קודקודים שיש ביניהם קשת בגרף .

תיאור: הפונקציה מאתחלת את הפוינטר לאבא של הקודקוד v להיות הקודקוד w .
במידה ולקודקוד v אין אבא עדיין היא מאתחלת אותו להיות שורש העץ .

```
void add_desplit_levels(int l);
```

קלט: **l** – מספר שלם שמייצג את רמת פיצול בגרף.

תיאור: הפונקציה מוסיפה את הרמת פיצול שקבלה לרשימה של "רמות הפיצול" בגרף .

```
void init_levels_in_Graph(Node *root);
```

קלט : **root**- פוינטר לשורש העץ.

תיאור: הפונקציה מחשבת עבור כל קודקוד מה הרמה שלו בעץ ,ושומרת עבור כל קודקוד את הרמה שחשבה עבורו .

```
void checkIfLevelNodeDesplite(Node* n);
```

קלט : **n** – פוינטר לקודקוד בגרף .

תיאור: הפוקציה בודקת האם הרמה בה נמצא הקודקוד היא רמת פיצול או לא , במידה והיא רמת פיצול היא מאתחלת את הממבר isDespliteLevel של הקודקוד n ל-true, אחרת הוא false .

```
vector<Node*> getNodeVector();
```

תיאור : הפונקציה מחזירה וקטור של כל הקודקודים בגרף .

```
Node* findNodeInVec(int name);
```

קלט: name- מספר שלם המייצג את תווית שם הקודקוד.

תיאור: הפונקציה מקבלת את שם הקודקוד שאנו רוצים לקבל פוינטר אליו , מחפשת ברשימת הקודקודים של הגרף את הקודקוד בעל השם הזה ומחזירה פוינטר אליו .

```
void sortNodesVector();
```

תיאור : הפונקציה ממיינת בסדר עולה את רשימת הקודקודים בגרף לפי שמות הקודקודים .

שם המחלקה : "Node"

תיאור: המחלקה אחראית על יצירת הקודוקדים , איתחולם,וכן הכלת מידע אודות הקודקוד כגון : שם הקודקוד, הרמה של הקודקוד, פוינטר לקודקוד האב , רשימת פוינטרים לבנים של הקודקוד (במידה והוא לא עלה) ,רשימת הרובוטים שנמצאים בקודקוד בנקודת הזמן הנוכחית של הסריקה , רשימת הרובוטים שיהיו לקודקוד בנקודת הזמן הבאה בסריקה , מהו המצב של הקודקוד בנקודת הזמן הנוכחית ומה יהיה המצב של הקודקוד בנקודת הזמן הבאה .בנוסף, המחלקה מבצעת כל מיני פעולות וחישובים הנדרשים לביצוע על הקודקודים במהלך הסריקה לפי האלגוריתם.

פונקציות מרכזיות :

```
void add_Son(Node* s);
```

קלט: **s** – פוינטר לקודקוד .

תיאור: הפונקציה מקבלת פוינטר לקודקוד **s** ומוסיפה אותו לרשימת הפוינטרים של הבנים של הקודקוד .

```
void add_father(Node* f);
```

קלט: **f**- פוינטר לקודקוד

תיאור : הפונקציה מקבלת פוינטר לקודוקד **f** ומאתחלת את הפוינטר לאבא של הקודקוד על ידי הפוינטר לקודקוד שקבלה.

```
void print_my_sons();
```

תיאור: הפונקציה מדפיסה את שמות כל קודקודי הבנים של הקודקוד.

```
void initRobotsWhichIcanGiveList();
```

תיאור: הפונקציה מאתחלת את רשימת כל הרובוטים שהקודקוד יכול לתת לבנים שלו , כלומר רשימת כל הקודקודים שממוקמים בקודקוד בנקודת הזמן הנוכחית .

```
bool isLevelDesplite();
```

תיאור: הפונקציה בודקת האם הקודקוד נמצא ברמת פיצול . אם הקודקוד נמצא ברמת פיצול היא מחזירה **true** ואחרת מחזירה **false** .

```
void changeMyLevel();
```

תיאור: הפונקציה מחשבת את עומק תת העץ המושרש מהקודקוד ולפי העומק המרבי שלו היא משנה את הרמה של הקודקוד . אם הקודקוד הוא עלה היא משנה את הרמה של הקודקוד מ 1- ל 0 , ואחרת היא משנה את הרמה של הקודקוד להיות העומק של תת העץ +1 עבור הקודקוד עצמו.

```
int getNextRobotListSize();
```

תיאור : הפונקציה מחזירה את מספר הרובוטים שיהיו ממוקמים בקודקוד בנקודת הזמן הבאה בסריקה .

```
void calMyAcentorsNextState(Node* node);
```

קלט : **node** – פוינטר לקודקוד

תיאור : הפונקציה מקבלת פוינטר לקודקוד ומחשבת בהתאם למצב שלו מה צריך להיות המצב של הקודקודים שנמצאים ברמות מעליו (אבות קדמונים של הקודקוד), לדוגמה : אם הקודקוד נמצא במצב visited היא בודקת האם קודקוד האב שלו צריך להיות במצב inhabited .

```
Node* get_son_who_is_not_visited_yet();
```

תיאור: הפונקציה עוברת על כל קודקודי הבנים של הקודקוד ומחזירה פוינטר לבן הראשון שהיא מוצאת שהוא נמצא במצב not visited .

```
void addRobotToNextRobotList(Robot *r);
```

קלט: **r** – פוינטר לרובוט .
תיאור: הפונקציה מקבלת פוינטר לרובוט ומוסיפה אותו לרשימת הרובוטים שיהיו לקודקוד בנקודת הזמן הבאה .

```
void addRobotToCurrentRobotList(Robot *r);
```

קלט: **r-** פוינטר לרובוט .
תיאור: הפונקציה מקבלת פוינטר לרובוט ומוסיפה אותו לרשימת הרובוטים שיש לקודקוד בנקודת הזמן הנוכחית .

```
void moveAllRobotsToMyfather(Robot *r);
```

קלט: **r-** פוינטר לרובוט.

תיאור : הפונקציה מקבלת פוינטר לרובוט ומעבירה את הרובוט לקודקוד האב .

```
Node* moveRobotsToNotVisitedSon_(Robot *r,Node* son);
```

קלט: **r-** פוינטר לרובוט,**son-** פוינטר לקודקוד בן.

תיאור: הפונקציה מעבירה את הרובוט r לבן son שנמצא במצב not visited וצריך רובוטים .

```
Node* DespliteRobots(Robot* rob,Node* son);
```

קלט: **rob-** פוינטר לרובוט, **son-** פוינטר לקודקוד בן

תיאור: הפונקציה שמעבירה את הרובוט rob לבן son במקרה robו נמצא בנקודת הזמן הנוכחית בקודקוד שנמצא ברמת פיצול .
אם לקודקוד אין עדיין רובוטים בנקודת הזמן הבאה של הסריקה , אז הרובוט rob נשאר אצל הקודקוד –לצורך תקשורת בין הרובוטים, אחרת הרובוט rob עובר לקודקוד הבן son

(כיוון שזה אומר שיש כבר רובוט שנשאר בקודקוד לצורך תקשורת בין הרובוטים) . הפונקציה מחזירה את הפוינטר לקודקוד שאליו rob עבר בחישוב (אחד מהשניים : הקודקוד עצמו או קודקוד הבן) .

```
void moveAllRobotsToMyFatherExceptOne(Robot *r);
```

קלט : **r-** פוינטר לרובוט.

תיאור: הפונקציה מעבירה את כל הרובוטים שנמצאים בנקודת הזמן הנוכחית בקודקוד מלבד רובוט אחד שצריך להשאר בקודקוד לצורך שמירה על התקשורת בין הרובוטים.

```
void changeData();
```

תיאור: הפונקציה מעדכנת בסיום חישוב תהליך הסריקה של נקודת זמן מסויימת, את המצב החדש של הקודקוד , וכן מעדכנת את רשימת הרובוטים שכעת עברו אל הקודקוד. כלומר מחליפה את ה-currentRobotList להיות ה-NextRobotList ואת ה-CurrentState להיות ה-NextState .

```
void calNextState(Node *node);
```

קלט: **node-** פוינטר לקודקוד.

תיאור: הפונקציה מחשבת עבור הקודקוד מה צריך להיות מצבו בנקודת הזמן הבאה של תהליך הסריקה.

```
bool FindRobotinList(Robot* r,vector<Robot*> list);
```

קלט : **list** –רשימת פוינטרים של רובוטים , **r** – פוינטר לרובוט .

תיאור: הפונקציה בודקת האם הרובוט r מופיע ברשימה list או לא , אם הרובוט r מופיע היא מחזירה true , אחרת מחזירה false .

פירוט המחלקות המבצעות את תהליך סריקת הגרף

שם המחלקה : "CalFutureGraph"

תיאור : המחלקה אחראית על ביצוע כל הלוגיקה של סריקת הגרף על פי האלגוריתם . המחלקה מבצעת את סריקת הקודקודים ומחשבת עבור כל רובוט שנמצא בקודקוד מסויים לאיזה קודקוד הוא צריך לעבור בנקודת הזמן הבאה בסריקה,וכן אחראית על קריאת הפונקציות הרלוונטיות לצורך ביצוע תהליך הסריקה . במחלקה זו יש לנו רשימה **NodeWithRobots** שזוהי רשימה של פוינטרים לכל הקודקודים שממוקמים אצלם רובוטים,ובנוסף שומרים משתנה **time** שמחשב את משך הזמן שלקח לנו לבצע את סריקת הגרף מתחילתה ועד סופה.

פונקציות מרכזיות :

```
void initRobotList(int numOfRobots,Node* n);
```

קלט : **numOfRobots**- מספר שלם שמהווה את מספר הרובוטים שיש לנו לצורך ביצוע הסריקה, **n**- פוינטר לקודקוד בגרף.

תיאור: הפונקציה יוצרת את האובייקטים של הרובוטים , מספר האובייקטים יהיה כערך שהעברנו במשתנה קלט **numOfRobots** , וממקמת את כל הרובוטים שיצרה בקודקוד **n** שהוא הקודקוד ההתחלתי ממנו אנו מתחילים את הסריקה.

```
void runAlgo(Graph *g,Node *root);
```

קלט : **root**-פוינטר לקודקוד השורש של העץ ,**g**- פוינטר לגרף (לעץ).

תיאור : הפונקציה אחראית על הרצת האלגוריתם לביצוע הסריקה מתחילתו ועד סופו , ומבצעת בכל נקודת זמן בסריקה קריאות לפונקציות האחראיות על ביצוע לוגיקת האלגוריתם .

```
void scanNodeWithRobotsOnly(Node *currentNodeOfRobot);
```

קלט : currentNodeOfRobot – פוינטר לקודקוד שממוקמים אצלו רובוטים בנקודת הזמן הנוכחית בתהליך הסריקה.

תיאור: הפונקציה אחראית על ביצוע כל לוגיקת האלגוריתם הכוללת את ביצוע החישובים לאיזה קודקוד יעברו הרובוטים בנקודת הזמן הבאה ,בהסתמך על מצב הקודקוד ונתונים נוספים שהאלגוריתם עושה בהם שימוש לצורך חישוב המעבר של הרובוטים.

```
void setStateToRobotsNodes(Robot *r);
```

קלט : **r**- פוינטר לרובוט

תיאור : הפונקציה מבצעת חישוב של המצב החדש של הקודקוד שהרובוט **r** נכח בו בנקודת הזמן הנוכחית , וחישוב המצב החדש של הקודקוד שהרובוט **r** יהיה נוכח בו בנקודת הזמן הבאה .

```
void moveRobots();
```

תיאור: הפונקציה מבצעת את המעבר של הרובוטים מקודקוד אחד לשני לקודקוד בתהליך הסריקה , כלומר ממקמת את הרובוטים להיות בקודקוד אליו הם עוברים .

```
void  print_graph(Graph *g);
```

קלט : **g**- פוינטר לגרף .

תיאור : הפונקציה מדפיסה את הגרף,

פלט: הפלט מציג את הקודקודים בגרף ,את הרובוטים שממוקמים אצלם (במידה ויש) ואת המצב של הקודקודים בתהליך הסריקה עד כה .

```
void initNodeListForNextSacn();
```

תיאור: פונקציה המאתחלת את רשימת הקודקודים עליה אנו צריכים לעבור בנקודת הזמן הבאה בתהליך הסריקה (רשימת הקודקודים שיש בהם רובוטים).

```
Node* returnSonToGiveRobotTo(Node* currentNodeOfRobot);
```

קלט : **currentNodeOfRobot** – פוינטר לקודקוד שאינו נמצא ברמת פיצול .

תיאור : הפונקציה מקבלת פוינטר לקודקוד **currentNodeOfRobot**, ובודקת עבורו לאיזה קודקוד בן של **currentNodeOfRobot** צריך לתת רובוטים. הפונקציה מחזירה פוינטר לקודקוד הבן שמצאה .

```
bool chackIfNeedToKeepRobotForCommunication(CalNodeData
&data,Node* son,Node* currentNodeOfRobot);
```

קלט : **data**- רפרנס לאובייקט שמחשב עבור קודקוד איזה מהבנים שלו צריכים צריכים לקבל רובוטים על מנת שתתי העצים שלהן יסרקו .

Son – פוינטר לקודקוד בן .

currentNodeOfRobot- פויטר לקודקוד אב .

תיאור: הפונקציה בודקת האם עלינו להשאיר רובוט אחד אצל קודקוד האב לצורך שמירה על תקשורת בין הרובוטים או לא לפני שהיא נותנת את הרובוטים לקודקוד הבן (ראה בחלק של ההיוריסטיקות הסבר למקרים שבהם אנו עושים שימוש בפונקציה זו) .אם צריך להשאיר רובוט אצל קודקוד האב לצורך תקשורת הפונקציה מחזירה **true** אחרת מחזירה **false** .

```
bool checkConnectivity();
```

תיאור : פונקציה שבודקת אחרי כל תזוזה של הרובוטים בתהליך הסריקה האם הם עדיין נמצאים בתקשורת זה עם זה ,אם כן מחזירה **true** אחרת מחזירה **false** .

```
Node* DespliteReturnSonToGiveRobotTo(Node*
currentNodeOfRobot);
```

קלט: currentNodeOfRobot – פוינטר לקודקוד (שנמצא ברמת פיצול)

תיאור : הפונקציה מקבלת פוינטר לקודקוד **currentNodeOfRobot** שנמצא ברמת פיצול, ובודקת עבורו לאיזה קודקוד בן של **currentNodeOfRobot** צריך לתת רובוטים על מנת שנקבל חלוקה שווה של הרובוטים בין הבנים. הפונקציה מחזירה פוינטר לקודקוד הבן שמצאה .

```
void SortRobotList();
```

תיאור : פונקציה שעוברת על רשימת כל הרובוטים בגרף ,וממיינת אותה לפי רמות הקודקודים בהם נמצאים מהרמה הנמוכה לרמה הגבוהה ביותר , וכאשר יש רובוטים שנמצאים באותה הרמה היא ממיינת לפי מצב הקודקוד כאשר ניתנת עדיפות לקודקוד במצב **visited** אחר כך עדיפות לקודקוד במצב **inhabited** ולבסוף לקודקוד במצב **not visited** . במידה והקודקודים נמצאים גם באותה הרמה וגם באותו המצב אנו ממינים לפי שם הקודקוד בסדר עולה.

שם המחלקה : "CalNodeData"

תיאור : המחלקה אחראית על ביצוע חישובי עזר למחלקה **CalFutureGraph** עבור חלוקת הרובוטים של קודקוד אב לקודקודי הבנים . בחישובים אלו המחלקה מחשבת ובודקת אילו מקודקודי הבנים צריכים רובוטים לצורך ביצוע תהליך הסריקה של תתי העצים המושרשים מהם ולכמה רובוטים .בנוסף, המחלקה שומרת רשימה של כל קודקודי הבנים הללו .

פונקציות מרכזיות :

```
void initVectorSonsWhoNeedRobots(Node* node);
```

קלט: node - פוינטר לקודקוד אב .

תיאור: הפונקציה מאתחלת ושומרת רשימה של כל קודקודי הבנים של הקודקוד **node** שצריכים רובוטים לצורך ביצוע סריקה של תתי העצים שלהם.

```
int CalHowMuchRobotsInMySubTree(Node* n);
```

קלט : n- פוינטר לקודקוד .

תיאור : הפונקציה מחשבת כמה רובוטים יש בתת העץ של הקודקוד n בנקודת הזמן הנוכחית של הסריקה ומחזירה את מספר הרובוטים שחישבה .

```
int CalHowMuchNotVisitedSonsInMySubTree(Node* n);
```

קלט: n- פוינטר לקודקוד

תיאור : הפונקציה מחשבת כמה קודקודי בנים שנמצאים במצב not visited בנקודת הזמן הנוכחית יש בתת העץ של הקודקוד n . הפונקציה מחזירה את מספר הקודקודים שחישבה .

```
int CalHowMuchRobotsIWillHaveNextTimenMySubTree(Node* n);
```

קלט : n – פוינטר לקודקוד.

תיאור: הפונקציה מחשבת כמה רובוטים יהיו בתת העץ של הקודקוד n בנקודת הזמן הבאה של תהליך הסריקה ומחזירה את מספר הרובוטים שחישבה .

```
Node*  getSonToGiveRobotForHim();
```

תיאור: הפונקציה בודקת איזה מקודקודי הבנים ששמרה ברשימה בשלב האיתחול צריך עדיין עוד רובוטים . הפונקציה מחזירה פוינטר לקודקוד בן שמצאה .

```
bool CheckIfVectorIsEmpty();
```

תיאור: הפונקציה בודקת האם הרשימה של הקודקודי בנים שיצרה ריקה או לא ,אם היא לא ריקה היא מחזירה true אחרת היא מחזירה false והמשמעות היא שיש לכל קודקודי הבנים מספיק רובוטים שיסרקו את תתי העצים שלהם , לכן אפשר להעלות את הרובוטים שיש בקודקוד האב מעלה כלפי השורש פרט לקודקוד אחד שישמור על תקשורת עם הרובוטים שנמצאים בתתי העצים של קודקודי הבנים .

```
bool checkIfSonNeedsMoreRobots(Node *son);
```

קלט : son – פוינטר לקודקוד בן

תיאור : הפונקיה בודקת האם קודקוד הבן צריך עוד רובוטים לצורך סריקת תת העץ שלו או לא , במידה וצריך היא מחזירה true אחרת היא מחזירה false.

```
int CalHowMuchNotVisitedSonsInMySubTreeWithOutRobots(Node* n);
```

קלט : n- פוינטר לקודקוד .

תיאור : הפונקציה מחשבת לכמה קודקודים שנמצאים בתת העץ של קודקוד n ובמצב not visited אין רובוטים .
הפונקציה מחזירה את מספר הקודקודים שחשבה .

פירוט המחלקה היוצרת את קובצי הפלט

שם המחלקה : **"WriteOutputToFile"**

תיאור: המחלקה אחראית על כתיבת תוצאות הסריקה לקובץ טקסט בשם "output.txt" . לאחר מכן מתבצעת קריאה של קובץ הפלט ועל פיו מוצג תהליך הסריקה באופן ויזואלי. בנוסף על פי קובץ הפלט המתאר את תהליך הסריקה הרובוטים האמתיים (מדגם hamster) ינועו בשטח המבנה .

פונקציות מרכזיות :

```
void writeNodeDespliteToFile(string s);
```

קלט: **s-** מחרוזת המכילה את שמות כל הקודקודים שנמצאים ברמות פיצול .

תיאור ופלט: הפונקציה כותבת לתוך קובץ "Node_desplite.txt" את שמות כל הקודקודים שנמצאים ברמות פיצול, הקובץ נשמר באותו path שבו נמצאים קבצי התוכנית וקובץ הריצה של התוכנית .

```
void WriteToFile(string s);
```

קלט:s- מחרוזת המתארת את פלט תהליך הסריקה .

תיאור ופלט: הפונקציה כותבת לתוך קובץ "output.txt" את תוצאות תהליך הסריקה של הגרף לפי נקודות זמן בסריקה ותיאור בכל נקודת זמן באיזה קודקוד היו ממוקמים הרובוטים . הקובץ נשמר באותו path שבו נמצאים קבצי התוכנית וקובץ הריצה של התוכנית .

```
void writeRandomGraph(string str);
```

קלט: str- מחרוזת המתארת את תהליך בניית הגרף באופן רנדומי

תיאור ופלט : הפונקציה כותבת לתוך קובץ "Graph.txt" את תיאור מבנה הגרף שנוצר באופן רנדומי (הגרף הרנדומי מסייע לביצוע טסטים ובדיקות של התוכנית ותהליך הסריקה). הקובץ נשמר באותו path שבו נמצאים קבצי התוכנית וקובץ הריצה של התוכנית .

פירוט מחלקות נוספות בתוכנית

שם המחלקה: "Robot"

תיאור המחלקה : המחלקה אחראית על יצירת האובייקט של הרובוט , הרובוטים עוברים בין הקודקודים בתהליך הסריקה כך שבכל מעבר הם שומרים על תקשורת וסנכרון מידע עם שאר הרובוטים . המחלקה שומרת עבור כל רובוט את שמו , ומי הקודקוד הנוכחי שהוא ממוקם בו ולאיזה קודקוד הוא יעבור לנקודת הזמן הבאה בסריקה .

פונקציות מרכזיות :

```
void setCurrentNode(Node *n);
```

קלט: n – פוינטר לקודקוד.

תיאור : הפונקציה מאתחלת את הקודקוד הנוכחי שהרובוט נמצא בו להיות הקודקוד n .

```
void setNextNode(Node* n);
```

קלט : n –פוינטר לקודקוד .

תיאור : הפונקציה מאתחלת את הקודקוד שהרובוט צריך לעבור אליו בנקודת הזמן הבאה להיות הקודקוד n .

```
int getName();
```

תיאור : הפונקציה מחזירה את שמו של הרובוט .

```
int getCurrentNodeLevel();
```

תיאור : הפונקציה מחזירה את הרמה שבו נמצא הקודקוד הנוכחי של הרובוט (הקודקוד שבו הרובוט נמצא כעת) .

```
int getCurrentNodeState();
```

תיאור : הפונקציה מחזירה את המצב של הקודקוד הנוכחי שהרובוט נמצא בו .
ערכי החזרה :
1 –הקודקוד נמצא במצב visited
2- הקודקוד נמצא במצב not visited
3- הקודקוד נמצא במצב inhabited

```
int getCurrentNodeName();
```

תיאור : הפונקציה מחזירה את שמו של הקודקוד הנוכחי שהרובוט נמצא בו .

```
Node* getCurrentNode();
```

תיאור : הפונקציה מחזירה פוינטר לקודקוד הנוכחי שהרובוט נמצא בו .

```
Node* getNextNode();
```

תיאור : הפונקציה מחזירה פוינטר לקודקוד שהרובוט יהיה ממוקם בו בנקודת הזמן הבאה בסריקה .

שם המחלקה : **"CreateRandomGraph"**

תיאור המחלקה : המחלקה אחראית על יצירת גרף דמוי עץ באופן רנדומלי , לצורך ביצוע טסטים ובדיקת תקינות תהליך הסריקה על עצים גנריים בעלי brunch factor שונים, עומקים שונים וכו'.

פונקציות מרכזיות :

```
Node* createRoot();
```

תיאור: הפונקציה יוצרת את קודקוד השורש של העץ ומחזירה פוינטר אליו .

```
void CreateRandomGraph::createSonsRandomly(Node *n);
```

קלט : n- פוינטר לקדקוד.

תיאור: הפונקציה יוצרת קודקודי בן לקודקוד הקלט n באופן רנדומלי תחת הגבלת ה brunch-factor והעומק הרצוי של הגרף .

```
void CreateRandomGraph::BuildRandomGraph(Node *node,int
MaxDeep,int currentDeep)
```

קלט : node – פוינטר לקדקוד , **MaxDeep** –העומק המקסימלי הרצוי של הגרף הרנדומי, **currentDeep** –העומק הנוכחי של הגרף בזמן הבניה .

תיאור: הפונקציה מבצעת את תהליך בניית תת העץ של הקודקוד n בגרף הרנדומלי לפי אילוצי הDeep max כלומר העומק המירבי של הגרף ,ומספר הבנים שמקסימלי שרצוי שיהיה לכל קודקוד בגרף (brunch factor).

פירוט מבנה התוכנית להצגת ויזואלית של סריקת הגרף

תיאור : התוכנית מנתחת את הגרף על פי קובץ הקלט ,ומייצרת קבצי תמונה שמציגים את תהליך סריקת הגרף על פי נקודות הזמן בסריקה . בתוכנית יש לנו **class Graph** שבו מיוצג הגרף כמילון כאשר ה-**key** הוא הקודקוד וה-**value** הוא הבנים של הקודקוד .

לאחר מכן התוכנית קוראת את קובץ הפלט **"scanning_for_visual_presention.txt"** , וכן את **"desplite.txt"** על מנת לדעת אילו קודקודי נמצאים ברמות פיצול ואילו לא ,את אלו שנמצאים ברמות פיצול היא מייחדת על ידי כך שצורתם היא מלבנית ולא עגולה כשאר הקודקודים, בנוסף התוכנית צובעת כל קודקוד בהתאם ל**state** שלו בכל נקודת זמן.

<u>פונקציות מרכזיות :</u>

def getNeighborsList(neighbors_str)

<u>קלט :</u> neighbors_str – מחרוזת של המתארת את שכני הקודקוד.

<u>תיאור:</u> הפונקציה מנתחת את המחרוזת ומחזירה רשימה של כל שכני הקודקוד.על פי רשימה זו נוכל להציג את הקשתות בגרף.

def getTime(line)

<u>קלט:</u> line –שורה שקראנו מהקובץ.

<u>תיאור :</u> הפונקציה שומרת את נקודת הזמן שהיא מנתחת כרגע מתהליך סריקת הגרף .התוכנית משתמשת בערך המוחזר על מנת לתת לקובץ התמונה את שמה ,ושנדע לזהות לאיזה זמן בסריקה התמונה משויכת .

def getNode(node)

<u>קלט :</u> node – מחרוזת של שם הקודקוד אותו אנו רוצים לבנות

תיאור: הפונקציה מחלצת את שם הקודקוד ויוצרת אותו, אם הקודקוד הוא קודקוד נמצא ברמת פיצול היא מציירת אותו בתור מלבן , אחרת היא מציירת אותו בעיגול .

def getNodeRobots(robots)

<u>קלט :</u> robot –מחרוזת של הרובוטים שנמצאים בקודוקד

<u>תיאור :</u> הפונקציה מחלצת את שמות הרובוטים וממקמת אותם בקודקוד.

def getNodeState(state)

<u>קלט :</u> state –מחרוזת המתארת את מצב הקודקוד

<u>תיאור :</u> הפונקציה מחלצת את מצב הקודקוד וצובעת אותו בהתאם למצבו .

אתגרים ופתרונות

במהלך ביצוע הפרויקט נתקלתי בקשיים הנוגעים לאופן תכנון חלקי הפרויקט כך שהם יהיו תקינים ויביאו ליעילות מרבית של תהליך הסריקה . כחלק מתהליך התכנון והביצוע.

בחירת שפת תכנות ומימוש התוכנית :

תחילה נתקלתי בקושי באיזו שפת תכנות לממש את תכנית הסריקה , תחילה התחלתי לתכנת בשפת python ואחר כך בחרתי לשנות לשפת C++ כיוון שבC++ ישנן סיפריות של מבני נתונים שונים שהיו לי יותר נוחות לשימוש מאשר הספריות של שפת python .
בנוסף ,רציתי ליצור הפרדה למספר מחלקות וקבצים בין החלקים השונים של התוכנית כמו בניית הגרף , ביצוע תהליך הסריקה וכו' ואילו הרגשתי שבpython לא הייתי מצליחה לצור הפרדה שכזו.

בחירת ייצוג הגרף:

אחר כך הייתי צריכה לחשוב איך לייצג את הגרף בצורה יעילה ונכונה ככל האפשר לצורך תהליך הסריקה , בחנתי מספר אפשרויות כמו מטריצת שכנויות, ייצוג על פי רשימת סמיכויות אבל ראיתי כי המעבר על ייצוגים אלה לא יעילה במיוחד כאשר אנו רוצים לחפש קודקוד או לדעת האם יש קשר של שכנו בין קודקודים שונים ,מה גם שברוב המקרים הייצוג הזה שמר יותר מידי מידע שלא היה לנו צורך בו בהכרח . לכן , בחרתי לייצג את הגרף כוקטור של קודקודי הגרף כאשר לקודקוד יש גישה ישירה (פוינטר) לקודקוד האב שלו ולקודקודי הבן , ובכך אנו מייעלים את תהליך המעבר על הקודקודים בשלב תהליך הסריקה וחוסכים בזיכרון ומשאבים.

ביצוע תהליך הסריקה ומעבר הרובוטים :

שלב ביצוע תהליך סריקת הגרף והעברת הרובוטים מקודקוד לקודקוד לווה בלא מעט קשיים אשר נדרשו ממני לבצע מספר מימושים והיוריסטיקות לצורך מציאת הפתרון היעיל ואופטימלי ביותר לביצוע סריקת הגרף על פי האלגוריתם. בשלבי חלוקת הרובוטים מקודקוד שנמצא ברמת פיצול לבנים שלו , הייתי צריכה לחשוב על דרך יעילה ונכונה לחלוקת הרובוטים באופן שווה ככל הניתן, בחרתי עבור מקרה כזה לחלק את הרובוטים באופן הבא: כאשר עוברים על כל הבנים שצריכים לקבל עוד רובוטים אנו נחלק להם בכל פעם רובוט ונעבור לקודקוד הבן הבא וכך נצליח לבצע חלוקה שווה של הרובוטים, במקרה שאני מפסיקים לחלק בקודקוד בן מסויים ואחר כך אנו יכולים לחלק עוד רובוטים כדי שנמשיך לחלק מאותו קודקוד בן שהפסקנו ולצור שיוון בחלוקה ,שמרתי במשתנה את הקודקוד ממנו הפסקתי לחלק על מנת שאחר כך אוכל להתחיל מאותו קודקוד את החלוקה שוב .

כדי להימנע ממצב שבו אנו מחלקים את אותו הרובוט למספר קודקודים שונים ,שמרנו רשימה נפרדת של כל הרובוטים שאנו יכולים עוד לחלק (כלומר שעוד לא חולקו לקודקודים אחרים) ועל פיה בצעתי את חלוקת הרובוטים, בכל פעם שרובוט עבר לקודקוד אחר הוצאתי אותו מהרשימה ובכך נמנעתי מבעיות טכניות שעלולים לצור שיבושים בתהליך הסריקה והתקשורת .

קושי נוסף היה בעיית **DeadLock** בתהליך הסריקה שנבע מכך שהתחילה חלוקת רובוטים לתת עץ של בן חדש בטרם סיום חלוקת הרובוטים לתת עץ של בן קודם שהיה צריך עוד רובוטים ,כתוצאה מכך שני תתי העצים חיכו לעוד רובוטים לעד ותהליך הסריקה נתקע .כדי להימנע מבעיית ה**deadLock** דאגתי לסיים קודם את תהליך העברת הרובוטים לבן מסויים ורק אחרי שהיו לבן זה מספיק רובוטים על מנת לסרוק את תת העץ שלו העברתי את שאר הרובוטים לתת עץ של בן חדש .

<u>בחירת אופן הצגת הגרף בצורה ויזואלית :</u>

בחרתי להשתמש בממשק **Graphviz** לצורך הצגת תהליך הסריקה בצורה ויזואלית כיוון שזהו כלי שהוא **open source** וניתן להריץ אותו על פלטפורמות שונות ללא תלות במערכת ההפעלה או סביבת העבודה בה אנו עובדים , בנוסף הוא מאוד ידידותי למשתמש ונוח להצגת הגרפים . **Graphviz** יוצר קבצי תמונה **png.** על פי קבצי **DOT.** שבאמצעותם אנו מתארים את מבנה הגרף .

כיוונים עתידיים

אני רואה בפרויקט אבן דרך משמעותית בתחום הביטחוני והצבאי בהם יש צורך לבצע סריקת של מבנים ממולכדים .בזכות ביצוע סריקת המבנים על ידי הרובוטים ושמירה על התקשורת והסנכרון ביניהם גופים בטחונים יוכלו לבצע סריקות של מבנים שיש חשש לכך שהם ממולכדים בצורה יעילה ומהירה ללא מעורבות של חיילים בשטח ,כך שלא יהיה חשש מפגיעה בחייהם של הלוחמים. אפשר לצור מעין יחידת בקרה על הרובוטים שמריצה את המימוש של תהליך הסריקה ונותנת הוראות לכל רובוט לאיזה כיוון עליו לנוע בשטח ,וכך לחסוך את העבודה עם הקבצים שלוקחת זמן ולפעמים מסרבלת את העניינים , כמובן שבמקרה זה עלינו לצור מודל כזה שישמור על התקשורת בין הרובוטים.
אני חושבת שעבור ביצוע סריקות במבנים מסוימים נדרשת לפעמים התערבות חיצונית לצורך שינוי מעבר הרובוטים על פי אסטרטגיה אחרת , לכן הייתי מוסיפה אפשרות סנכרון למערכת על מנת לבצע את הסריקה גם במקרים בהם נדרשת התערבות חיצונית .