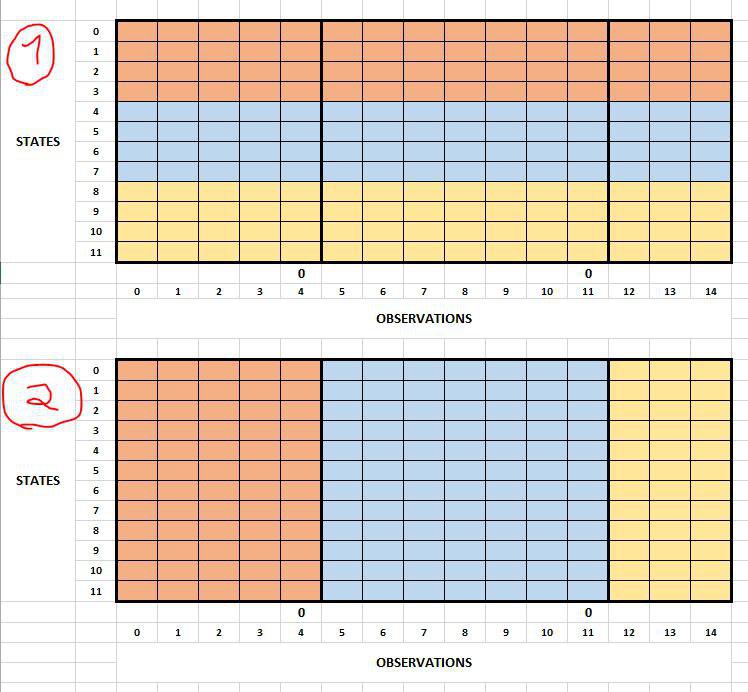
[Parallel computing](https://www.google.co.il/search?espv=2&biw=853&bih=626&q=parallel+computing&spell=1&sa=X&ei=TuFEVdSLGsnxaKGdgNAJ&ved=0CBgQvwUoAA)

Final project

בפרויקט זה ניתן היה להשתמש בכמה דרכי מימוש כאשר היו 2 עיקריים כמו שניתן לראות בתמונה להמחשה הבאה:

כאשר אני בחרתי לממש בשיטה מספר 2.

**שיטה מס' 1:**

השיטה בא להמחיש שכל הסלייבים יעבדו במקביל על מטריצה אחת נתונה כל פעם על ידי חלוקה לשורות וכך כל סלייב מממש את האלגוריתם על השורות החלוקה שמהסטאר קבע לו ללא תלות באחרים כאשר יסיימו את העבודה ימתינו לקבל מהמאסטר את המטריצה השניה וכך האלה עד שהמסטאר יודיע על סיום עבודה.

**שיטה מס' :2**

בשיטה השניה המימוש שונה על ידי כך שכל סלייב בעצם מקבל מטריצה שלמה למימוש האלגורתים עליו כאשר הסלייב סיים מודיע למאטסר כי הוא מחכה למטריצה הבאה בתור שפנויה.

**אלגוריתם הסבר:**

transion

AB

בהתחלה כל תהליך מקצה לו זיכרון למטריצה ולמטריצה

לאחר מכן תהליך מספר 0 (מאסטר) קורא מקובץ את הנתונים ומעדכן את מטריצת ו לאחר מכן ישר מבצע לכל התהליכים

transion

AB

Broad cast

על מנת שכולם יחלקו את אותו מידע כדי שלא יצאו נתונים שגויים.

בנקודה זו כל התהליכים מסונכרנים מבחינת המידע.

transion

בשלב הבא כל תהליך מבצע על מטריצת פעולת לוג על כל הערכים כלומר כל ערך במטריצה יהיה עם ערך חדש של לוג של הערך הקיים וביחד עם מקבול של על מנת לייעל (לזרז) את התהליך כי פעולה של לוג היא פעולה יקרה שלוקחת זמן ובמקום כל פעם לחשב מחדש את הערך הלוגי בפונקציית חישוב כל פעם מחדש יש לבצע זאת פעם אחת ובזאת לסיים.

Open mp

**יש לציין** כי הפונקציית חישוב של המטריצה נעשית על ידי חיבור לוגים כי כפל לוג שווה לחיבור ערכי הלוג וכל החישוב הלוגים נעשה מחוץ לפונקציית חישוב וזאת על מנת לעבור לפעולת חיבור שהיא פעולה פחות יקרה מבחינת זמן עיבוד לעומת פעול כפל.

**בחזרה לאלגוריתם:**

לאחר חישוב הלוג של המטריצה הנ"ל יש הקצאה של שני וקטורים

**Final path :**

משמש את המאסטר כדי לקבל מהסלייב כל פעם "דרך" חדשה ומכניס אותו ל

Final path.

**TEMP:**

משמש כל סלייב בנפרד כאשר כל סלייב מסיים את פונקציית החישוב הוא מחפש "דרך חדשה" על ידי פונקציית שמטרתה היא ללכת לשורה האחרונה למצוא את ההסתברות הכי גדולה ומשם ללכת לאבא שלו וכך הלאה עד שמגיע לשורה הראשונה.

FINAL PATH

FIND PATH

וקטור זה מועבר למאסטר והמאסטר מעתיק את ה הוקטור שקיבל לוקטור

שבסוף הוא נכתב לקובץ.

לאחר מכן מאסטר נכנס לתוך תנאי משם הכול מתנהל, המאסטר מקצה מקום לוקטור וטוען אליו נתונים דרך קריאה של קובץ רק המאסטר בשלב זה רק המאסטר מכיר את הוקטור הנ"ל.

OBSERVTION

לאחר מכן המאסטר ממשיך לרוץ ומקצה מקום לוקטור חדש שנקרא

OBSERVTION

ZERO VECTOR

הוא ממלא אותו לפי הסדר באינדקסים של מקומות שבוקטור יש אפסים ,כלומר:

VECTOR ZERO

נניח שוקטור נתון : " 2 ,4,0,3,0,5" אז בוקטור

OBSERVTION

באדינקס 0 יהיה 1 ובאינדקס 1 יהיה 3 ובאינדקס 2 יהיה את המיקום האחרון של סיום שזה 5.

אחרי זה המאסטר שולח לכולם את הגודל של המערך האינדקסים כדי שאם האלגוריתם יעבוד על מצב (שעליו נרחיב בהמשך ) הסלייבים ידעו למי מותר להיכנס ולמי לא.

SIMPLE MODE

לאחר מכן המאסטר שולח לסלייבים "דגל" שידעו אם יש לעבוד במצב "רגיל" או "מסובך" שעליהם נרחיב עכשיו.

**מצב רגיל :**

)SIMPLE MODE(

במצב זה יש יותר סלייבים מאפסים ואז בעצם כל סלייב מקבל מטריצה מבצע את החישובים שהוא צריך מחזיר תשובה למאסטר שהוא סיים ושולח לו את וקטור ה- "דרך" ובזה מסיים ולכן אם יש רק למשל אפס אחד ויש למשל ארבעה סלייבים

אז הסלייב הראשון יעבוד מהתחלה עד לאפס הראשון והסלייב השני יעבוד מ האפס הראשון ועד הסוף ולכן אין צורך בשאר הסלייבים,ויש לסיים אותם.

)COMPLEX MODE(

מצב מסובך:

מצב זה מוגדר כאשר יש יותר אפסים מסלייבים לכן המאסטר צריך כל פעם שולח לסלייב עבודה,כאשר הסלייב מסיים המאסטר בודק אם נשארה עוד עבודה ואם בנוסף זאת עבודה אחרונה.

אם נשארה עוד עבודה וזאת לא עבודה אחרונה אז הוא שולח לסלייב עבודה ומחכה להבא שיודיע לו שהוא סיים כדי לתת לו עבודה חדשה.

במידה והמאסטר הגיע למצב שנשארה עבודה אחרונה

אז ההוא מחכה לסלייב הראשון ש"יודיע" לו שהוא סיים ואז המאסטר נותן לו עבודה אחרונה וכל סלייב שבא אחריו המאטסר אומר לו לסיים כי אין עוד עבודה לתת

ואחרי שהסלייב סיים עם העבודה האחרונה הוא קיבל "דגל" שאומר לו כי אין עבודה נוספת ואין צורך לדבר עם המאסטר עוד פעם ויש לסיים.

**חזרה לאלגוריתם:**

גם במצב "רגיל" וגם במצב "מסובך" המאסטר עובד על איתו עיקרון המאסטר שולח לסלייב הראשון שמדבר איתו את המקום הראשון והשני שב

VECTOR ZERO

ולסלייב הבא את המקום השני והשלישי וכך הלאה עד שאין יותר מה לשלוח.

הסלייב מקבל את הנקודת התחלה וסוף ועם נתונים אלו הוא מקצה זיכרון למטריצה שאיתה משתמש בפונקצייה שנקראת שמטרתה לאתחל את המטריצה שאיתה הוא הולך לעבוד.

INITTRELLIS

כאשר בפונקציה זו הסלייב מאתחל את המטריצה עם הערכים הנכונים.

המטריצה מחזיקה "מבנה" בכל נקודה כאשר ה"מבנה" מחזיק שלוש תכונות

1.הסתברות

2.הורה

3.

FUNC

הוא מאתחל את ההסתברות עם 0 שזה שווה ערך ללוג של אחד וזאת כדי לעבור לפעולה של חיבור ולא כפל כמו שהוסבר בהתחלה.

את ההורה עם מינוס אחד כי בינתיים אין הורה לאף אחד.

ומס' 3 :

CALCULATE

על מנת לחסוך בפעולה זו ולחשב אותה כל פעם מחדש בתוך פונקציית

שעליה נרחיב בהמשך ,נעשה זאת רק באתחול ועל ידי כך חסכנו זמן ושיפרנו יעילות.

וכמובן פעולת הלוג נעשית כמובן על מנת לעבור לפעולת חיבור בשביל לשפר יעילות.

וכך בעצם כל סלייב מאתחל כל "צומת" במטריצה.

**בחזרה לאלגוריתם**:

OBSERVTION

לאחר האתחול שהסלייב עושה המאסטר שולח לסלייב רק חלק מ

כלומר החלק שהוא צריך (היה ניתן שגם כולם יכירו את וקטור זה אבל העדפתי להכניס את פונקציות זו על מנת לחסוך בזכרון שגם ככה עמוס).

CALCUALTE

ולאחר מכן שולח את המטריצה לפונקציית .

OBSERVTION

פונקציה זו מקבלת את המטריצה שהסלייב איתחל ואת וקטור ה

שסלייב קיבל מהמאסטר לפני האתחול .

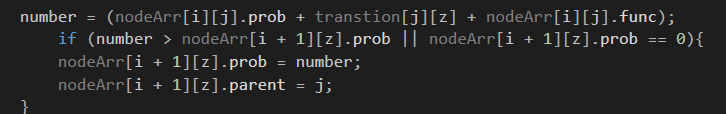
בנוסף מקבלת את גודל מספר השורות שתדע כמה יש לרוץ בתוך הלולאה.ולעשות את החישוב יש לציין שגם הפונקציה משתמשת ב .

OPEN MP

(יש לציין כי מימשתי שורות ועמודות הפוך מהתרגיל)

ולכן לולאה ראשונה רצה על כל השורות , לולאה שניה רצה על כל עמודה ובשביל כל עמודה יש לחשב את ההסתברות מול כל עמודה אחרת ולכן יש לולאה שלישית)

כאשר הפונקציה מחשבת את הדבר הבא :



הפונקציה מחשבת על ידי פעולת חיבור של לוגים כמו שהוסבר את הפונקציה שיש לחשב בצומת שהיא נמצאת כרגע ובודקת אם המספר שיצא גדול יותר מהמספר של הצומת הבא שעליה יש לבדוק במידה והמספר יותר גדול אז הצומת הבאה תקבל את ההסתברות שיצא ואת ההורה החדש אחרת לא יקרה כלום והצומת הבא ישאר עם אותו הסתברות ואותו הורה.

יש לציין שכל סלייב עושה זאת כמובן בשביל כל מטריצה.

**בחזרה לאלגוריתם :**

לאחר שהמטריצה חושבהה בפונקצייה הנ"ל הסלייב שולח את המטריצה לפונקציה שנקראת FINDPATH .

מטרת פונקציה זו בעצם לרוץ על השורה האחרונה ולמצוא את ההסתברות של צומת ומשם היא מגלה מי ההורה שלו חוזרת אחורה ומוצאת את ההורה שלו וככה בעצם עד שמגיעה לפונקציה הראשונה כאשר את הרשימה של כל ההורים מכניסה לוקטור TEMP כמו שהוזכר בהתחלה.

לאחר מכן הסלייב שולח את וקטור TEMP למאסטר והמאסטר מעתיק את תוכן ה TEMP לוקטור FINAL PATH .

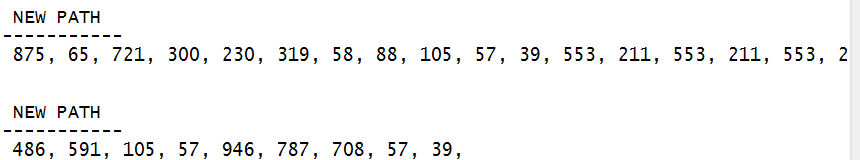
יש לציין כאשר המאסטר עובר על TEMP ומעתיק אותו ל FINAL PATH הוא מעתיק רק את התוכן ששונה ממספר קבוע מראש שאינו יכול להופיע במערך

מערך TEMP הוא בגוגל של מספר ה-שורות כמובן וכל מקום בו מאותחל עם "מספר השורות פלוס אחד" ככה שמספר זה אינו יכול להופיע במערך אחרי פונקציית FINDPATH שהסלייב מבצע בתוך אחד ה MODE .

שהמסטאר מקבל את ה TEMP שואל האם המספר שמופיע הוא "מספר השורות פלוס אחד" אם כן זה אומר שבמקום הזה הסלייב "לא התעסק" ולכן אין סיבה להעתיק אותו,מה שנותן תוצר סופי של וקטור אחד ארוך שמחזיק את סדר המסלולים לפי שאמור להיות.

כמובן שיש שוני מאוד קטן במימוש שלבכל אחד מ לגיב מתי הסלייב שולח חזרה למאסטר את המערך אבל זה אותו יישום.

לאחר שהמאסטר סיים הוא נשלח לפונקציה שכותבת את כל המערך לקובץ שנקרא FINALTRANSTION PATH ובנוסף יש פונקצינליות מעניינת שהוא יודע לזהות מסלול חדש ורושם זאת בקובץ כמו שנראה בתמונה



לאחר מכן כל הזיכרון משוחרר כמקובל וכאן נגמר האלגוריתם .

תודה.