תמונה שמכילה טקסט, שלט

התיאור נוצר באופן אוטומטי

הפקולטה להנדסה

המעבדה לעיבוד אותות

פירוק מהיר של רשת מבוזרת לרכיבים קשירים

שהם זינר

פרויקט שנה ד' לקראת תואר ראשון בהנדסה

מנחה אקדמי: ד"ר רן גלס

אוקטובר 2021

תוכן עניינים

[תקציר 2](#_Toc108946681)

[הרעיון לפרויקט 2](#_Toc108946682)

[מטרת הפרויקט: 2](#_Toc108946683)

[שלבי הפרויקט: 2](#_Toc108946684)

[עכעכע 4](#_Toc108946685)

[תודות 5](#_Toc108946686)

[רקע עיוני 6](#_Toc108946687)

[הצגת הבעיה 7](#_Toc108946688)

[הבעיה 7](#_Toc108946689)

[מוטיבציה 7](#_Toc108946690)

[תיאור של חלופות לפתרון 8](#_Toc108946691)

[אלגוריתמים דטרמיניסטיים ידועים 8](#_Toc108946692)

[תיאור מפורט של השיטה הנבחרת והסיבות לבחירתה 9](#_Toc108946693)

[השיטה שנבחרה 9](#_Toc108946694)

[תיאור האלגוריתם 9](#_Toc108946695)

[מימוש האלגוריתם 10](#_Toc108946696)

[בדיקה מקיפה של השיטה. תוצאות וניתוחן 11](#_Toc108946697)

[החסמים 11](#_Toc108946698)

[התוצאות 11](#_Toc108946699)

[מסקנות וסיכום – האם התוצאות מתאימות לניתוח הראשוני? האם הרעיון מוצלח? יתרונות וחסרונות של הפתרון 12](#_Toc108946700)

[רשימה ביבליוגרפית 13](#_Toc108946701)

# תקציר

## הרעיון לפרויקט

הפרויקט עוסק ברשתות מבוזרות ואלגוריתמי גרפים המבוצעים על-ידי רשתות בעלות מאות ואלפי יחידות חישוב עצמאיות.

## מטרת הפרויקט:

מטרת הפרויקט היא לפתח מערכת לחישוב מהיר של network decomposition - פירוק הרשת לרכיבים קשירים כך שרדיוס כל רכיב קשיר הוא O(log n).

במאי 2020 התגלתה פריצת דרך במאמר של Rozhon and Ghaffari המתאר אלגוריתם המאפשר ביצוע פירוק רשת בזמן פולי-לוגריתמי בגודל הרשת. כלי חשוב זה מאפשר ביצוע מהיר ויעיל של משימות רשת שונות ומגוונות כגון צביעה או MIS.

## שלבי הפרויקט:

* 1. לימוד החומר והאלגוריתם של Rozhon Ghaffari.
  2. פיתוח מערכת המדמה רשת מבוזרת, והפעלת אלגוריתם הפירוק עליה.
  3. מדידת הביצועים ושיפור היעילות.

# תודות

ברצוני להודות מעומק-לב לד"ר רן גלס – על שהסכים להכניס אותי תחת הנחייתו, עזר לי בבחירת הנושא לפרויקט, עודד, תמך ועזר לי להיחלץ כאשר נקלעתי למבויים סתומים במהלך העבודה. הפרויקט הזה לא היה יוצא לפועל ללא עזרתו ודחיפתו.

לספיר אשתי – על הסבלנות וההבנה כאשר הייתי צריך להשקיע זמן בעבודה על הפרויקט.

לפקולטה להנדסה באוניברסיטת בר אילן – על יצירת סביבה המאפשרת ומעודדת מחקר. על שהקנתה בי ידע וחשפה אותי לעולם המחקר.

# רקע עיוני

***חישוב מבוזר*** - שיטה שבמסגרתה חלקים שונים של משימת חישוב כלשהי המוטלת על תוכנית מחשב מתבצעים במחשבים נפרדים המקושרים ביניהם באמצעות רשת. לדוג': רשתות ניתוב, רכבים אוטונומיים, blockchain, שיתוף קבצים, אבטחת קצה-לקצה.

***רשת*** - גרף קשיsר בעל צמתים ו- קשתות.

- המרחק הגדול ביותר (בקשתות) בין שני צמתים בגרף.

כל צומת בגרף מייצג מחשב נפרד ברשת, וכל קשת מייצגת 'חיבור' שעל גביו המחשבים מחליפים הודעות.

*מודל* ***LOCAL*** - מודל של חישוב מבוזר בו החישוב מבוצע בצורה סינכרונית ומחולק ל***סיבובים***.  
בכל סיבוב, כל צומת בגרף יכול לבצע את הפעולות הבאות:

1. שליחת הודעות לצמתים שכנים, ללא מגבלה על גודל ההודעות.
2. קבלת הודעות מצמתים שכנים, במידה ונשלחו על-ידם בסיבוב הקודם.
3. ביצוע חישובים.

לכל צומת קיים מזהה () ובהתחלה הוא לא מכיר את הטופולוגיה של הגרף (), אלא רק את השכנים שלו .

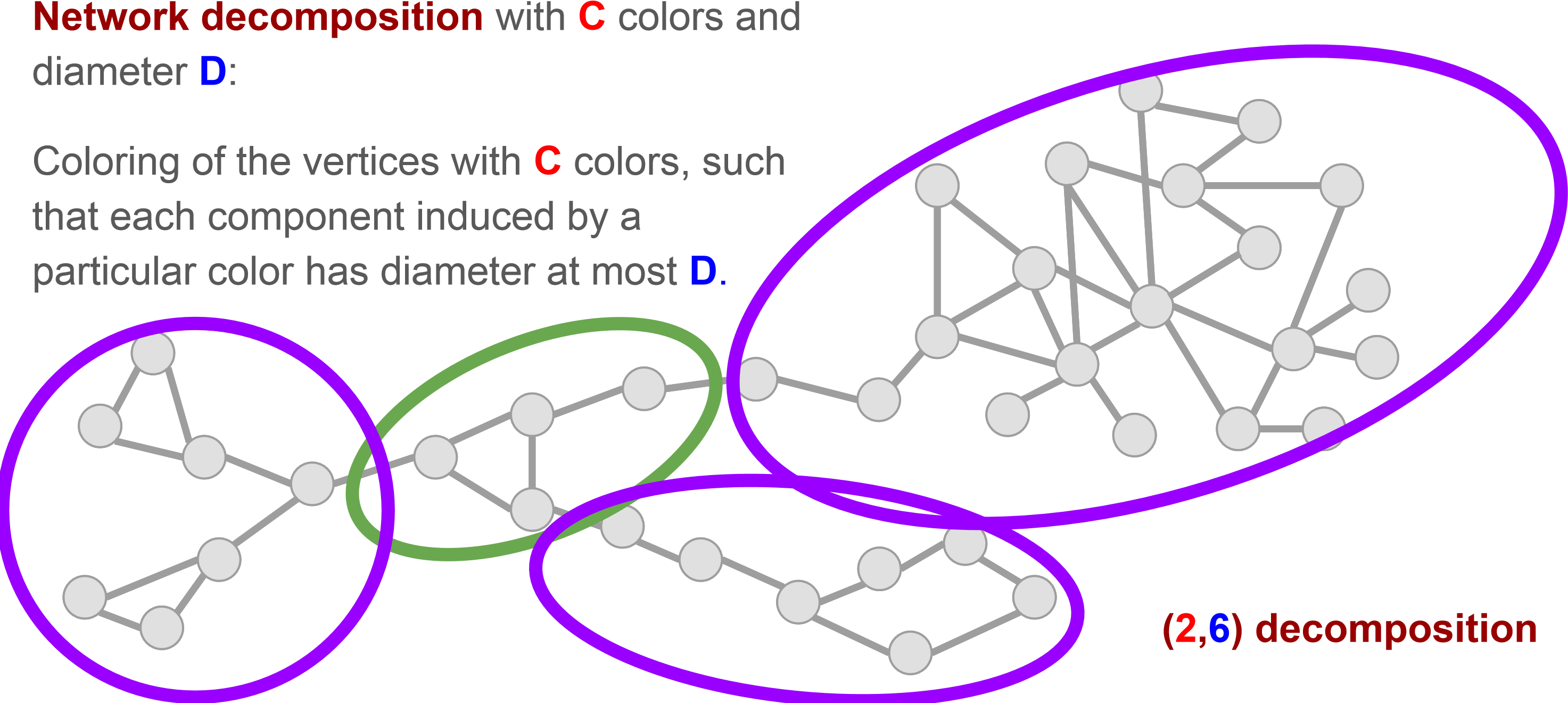
סיבוכיות הזמן של אלגוריתמים לחישוב מבוזר נמדדת ע"י מספר הסיבובים הנדרשים עד לסיום החישוב ע"י כל הצמתים בגרף. מדד אחר לביצועי אלגוריתם מבוזר היא *סיבוכיות ההודעות* – כמות ההודעות שנשלחו ע"י צמתי הגרף במהלך החישוב.

# הצגת הבעיה

## הבעיה

***פירוק-(C,D) של רשת*** - צביעה של ב- צבעים, כך שלכל רכיב-קשירות בגרף המושרה ע"י צבע מסוים יש קוטר של לכל היותר .

דוגמה לפירוק-(2,6) של גרף:



## מוטיבציה

טענה - לאחר O(D) סיבובי תקשורת, כל צומת יכול ללמוד את G בשלמותו.

אם כל צומת ישלח לשכנים שלו את המזהה שלו ואת , ויפיץ הלאה כל הודעה שהוא מקבל, לאחר סיבובים, כל צומת ידע את הטופולוגיה של הגרף עד למרחק קשתות ממנו. מכאן שלאחר סיבובים, כל הצמתים ידעו את בשלמותו.

מסקנה - ניתן לפתור כל 'בעיית-גרף' ב- סיבובים

אם ידוע, כל צומת יכול לבצע את כל החישובים בעצמו בצורה מרוכזת.

מסקנה נוספת - אם נוכל לבצע פירוק-(C,D) יעיל (יעיל = סיבוכיות פולי-לוגריתמית) של , עם ו- מספיק 'קטנים', נוכל לפתור בעיות גרף כגון MIS, (Δ+1)-צביעה, בצורה יעילה.

דוגמה לאלגוריתם דטרמיניסטי יעיל עבור MIS: ראשית נבצע פירוק-(C,D) של הרשת ב- סיבובים. לאחר-מכן, נעבור צבע-צבע. עבור כל צבע, נקבל גרף מושרה המורכב מרכיבי קשירות זרים בקוטר D. כל רכיב קשירות יאסוף את המידע הטופולוגי שלו, יחד עם הצבעים של הצמתים שסמוכים אליו, לצומת אחד, ב- סיבובים. הצומת יפעיל אלגוריתם ריכוזי לחישוב (Δ+1)-צביעה על רכיב-הקשירות ויפיץ את התוצאה חזרה אל הצמתים האחרים ב- סיבובים. סה"כ נקבל סיבוכיות של .

# חלופות לפתרון

## אלגוריתמים דטרמיניסטיים ידועים

ברוך אברבוך ושות'(1989)‎ [1]סיפקו אלגוריתם דטרמיניסטי שמחשב פירוק (C,D) ב- סיבובים, כאשר

פנקונסי וסטריניווסאן(1992)‎2 סיפקו גרסה של האלגוריתם הזה ששיפרה את הנתונים שלו ל-

אך במשך זמן רב לא נמצא שיפור לתוצאה הזאת, מה שחסם גם יעילות של אלגוריתמים דטרמיניסטיים אחרים שמסתמכים על פירוק-רשת.

# תיאור השיטה הנבחרת והסיבות לבחירתה

## השיטה שנבחרה

במאי 2020, Rozhoň and Ghaffariפרסמו מאמר ובו הציגו אלגוריתם שמאפשר לבצע פירוק- בסיבוכיות זמן של סיבובים במודל **LOCAL**. ‎[3]  
זמן קצר לאחר מכן, ביולי 2020, Ghaffari, Rozhoň and Grunau פרסמו גרסה משופרת של האלגוריתם, שמשיגה פירוק- בסיבוכיות זמן של סיבובים במודל **LOCAL**.

בחרתי לבסס את הפרויקט על הגרסה המשופרת, למרות שהיא מוסיפה מעט מורכבות לאלגוריתם, על מנת שהתוצאות יהיו כמה שיותר רלוונטיות.

## תיאור האלגוריתם

ריצת האלגוריתם בנויה משלבים, כאשר כל שלב מורכב מצעדים. בתחילת הריצה כל צומת בגרף הוא רכיב בפני עצמו (והמזהה שלו זהה למזהה של הצומת). בתחילת כל צעד, כל צומת עשוי לבקש לעבור לרכיבים שכנים. אם רכיב מקבל מספיק בקשות, הוא מוסיף לעצמו את הצמתים המבקשים. אם הוא לא מקבל מספיק בקשות, הוא 'הורג' את הצמתים המבקשים ומפסיק לגדול עד תחילת השלב הבא. צמתים שנהרגו אינם משתתפים יותר בפעולת האלגוריתם מלבד העברת הודעות. בסוף הריצה, לכל הפחות חצי מהצמתים נותרו בחיים, והם מחולקים לרכיבי קשירות מנותקים שקוטרם לא עולה על .

# מימוש האלגוריתם

בחרתי לממש את האלגוריתם בשפת C++, מפני שזו שפה שאני מכיר ומחבב, וכי היא מאפשרת ביצועים טובים.

אומנם מדובר באלגוריתם לחישוב מבוזר, אך כדי לא לחרוג מגבולות יכולותיי בחרתי, בעצתו של המנחה שלי ד"ר רן גלס, לכתוב תוכנית למחשב אחד שתדמה ריצה של האלגוריתם על רשת אקראית.

בכתיבת הקוד ניתן דגש על שימוש יעיל במשאבים ושימוש במבני-נתונים cache-friendly.

בקובץ NetworkDecomposition.hpp נמצא מבנה נתונים Graph שנועד לייצג את הרשת המבוזרת ולהריץ את אלגוריתם הפירוק. הבנאי של Graph מקבל כקלט את מספר הצמתים ברשת - n, את רשימת ה- Idsשל הצמתים (אופציונלי. בהיעדר רשימה כזו, הבנאי יבנה רשימה אקראית) ואת מטריצת השכנויות של הגרף, מיוצגת ע"י וקטור בגודל n^2. לחלופין, במקום מטריצת שכנויות, ניתן לספק לבנאי מחרוזת עם מילה אחת מתוך {chain, ring, clique, random} על מנת ליצור גרף אקראי בטופולוגיה המתאימה. בהינתן קלט random, הבנאי יבנה גרף קשיר עם קשתות אקראיות (ההסתברות של כל קשת אפשרית להימצא בגרף היא בערך 0.5).

את הקוד המלא ניתן למצוא בכתובת ה-GitHub: <https://github.com/sziner/NetworkDecomposition>

# בדיקת האלגוריתם, תוצאות וניתוחן

על מנת לבדוק את האלגוריתם, כתבתי קובץ הרצה בשם Test.cpp המבצע מספר ריצות של האלגוריתם על כל אחת מהטופולוגיות chain, ring, clique, random עבור ערכי n של 1000, 10,000, 100,000, 500,000 ו- 1,000,000.  
עקב מגבלות זיכרון של המחשב האישי שלי, לא התאפשר לי להריץ את האלגוריתם בהצלחה על ערכי n גדולים מ-10,000 עבור הטופולוגיה clique, או על ערכי n גדולים מ-100,000 עבור שאר הטופולוגיות, מבלי להקפיץ חריגת bad\_allocation.

## התוצאות

את תוצאות הבדיקה ניתן למצוא ב[תיקיית הפרויקט](https://github.com/sziner/NetworkDecomposition) בגיטהאב. הפלט מתמקד בנתונים שקל לעקוב אחריהם גם עבור כמויות גדולות של צמתים – מספר רכיבי הקשירות הפעילים בכל שלב, והרדיוס המקסימלי מבין הרכיבים. נתונים אלו מלמדים על מידת ואופן הפיזור של הצמתים בין רכיבי הקשירות השונים.  
מה שהפתיע אותי במיוחד בתוצאות ההרצה היתה העובדה שכל הפעלה של האלגוריתם הניבה פלט שבו כמעט כל הצמתים בגרף נצבעו באותו צבע, נקרא לו צבע 0, מלבד אולי מספר צמתים בודדות שנצבעו בצבע 1. למרות שהאלגוריתם מבטיח חסם עליון של על מספר הצבעים, בפועל התוצאות אפילו לא התקרבו לכך.

התוצאה הזו עשויה להיראות משונה בהתחלה. למשל, עבור n=100,000 בטופולוגית שרשרת, רק 125 צמתים נצבעו ב-1 בעוד שכל שאר הצמתים נצבעו ב-0. אבל כשבוחנים את התוצאה אל מול החסמים המובטחים ע"י האלגוריתם, הפירוק שמתקבל עומד בכל התנאים: זמן הריצה באותה הדוגמה הוא 14,114,557 סיבובים, בעוד שהאלגוריתם מבטיח חסם עליון של כ-61 מיליון סיבובים. עומק האשכול המקסימלי הוא 1295 לעומת חסם של 64,736.

# מסקנות וסיכום

בכל הבדיקות שנעשו האלגוריתם סיפק פירוק עם אשכולות בקוטר המובטח ובסיבוכיות הזמן המובטחת, הן בטופולוגיה אקראית והן במקרי-קצה כגון שרשרת או קליקה.

בגלל הקבועים הגדולים שבחסמי זמן הריצה והקוטר, ייתכן שעבור רשתות קטנות ניתן יהיה להגיע לתוצאות ברות-השוואה בסיבוכיות זמן טובה יותר בדרכים אחרות. כדי להביא לידי ביטוי את יתרונותיו של האלגוריתם על פני אלגוריתמים אחרים, ולהפוך את הקבועים שבניתוח סיבוכיות הזמן וגדלי האשכולות לזניחים, יש לבחון את ביצועיו על רשתות בסדרי-גודל גדולים מאוד. גדולים יותר מאלו שהתאפשר לי לספק תוצאות עבורן.

עם זאת, המטרה הראשית של עבודתי היתה מימוש האלגוריתם ואימות הניתוח התאורטי שלו ע"י תוצאות בשטח, ותוצאות הריצה אכן מאששות זאת.

# רשימה ביבליוגרפית

1. Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In Proc. 30th IEEE Symp. On Foundations of Computer Science (FOCS), pages 364–369, 1989.
2. Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In Proc. 24th ACM Symp. on Theory of Computing (STOC), pages 581–592, 1992.
3. V´aclav Rozhoˇn and Mohsen Ghaffari. Polylogarithmic-time deterministic network decomposition and distributed derandomization. In Proc. Symposium on Theory of Computation (STOC), 2020.

<https://github.com/sziner/NetworkDecomposition.git>