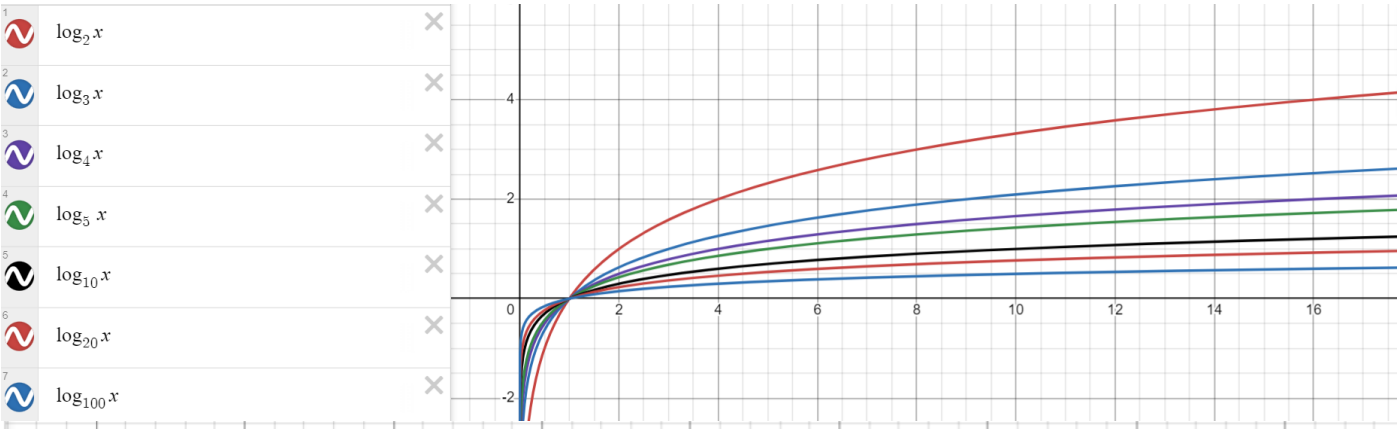
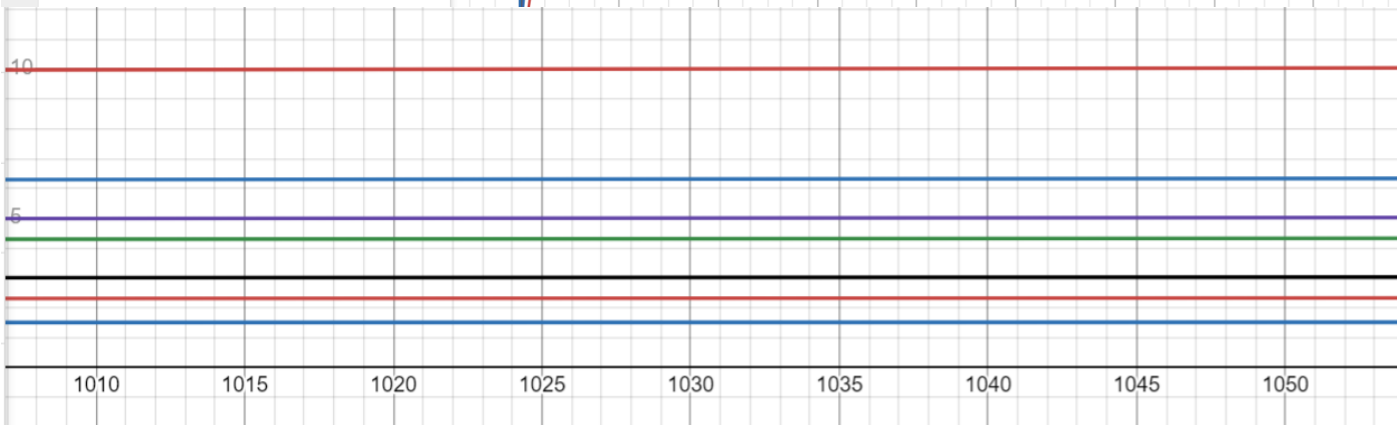
**נוסחת האב:** בכל פעם חוזרים על החישוב על חלק קטן יותר – זה מוגדר ע"י B, ועושים את החישוב הזה A פעמים. ככל שA יותר גדול מB, צריך לעשות יותר חישובים וכל חישוב הוא על חלק יחסית גדול של הN המקורי. מנגד, אם ככל שB גדול יותר מA, בכל פעם עושים את החישוב על חלק קטן יותר, וצריך פחות חישובים. לכן בודקים את לוג B של A.
בנוסף, אם עושים פעולה שמוגדרת ע"י N בחזקת C, זה מוסיף לזמן.
בפועל, מה שיותר גדול – לוג B של A, או מ בחזקת C – זה מה שקובע את הסיבוכיות. N בחזקת מה שגדול.
אם הם שווים, זה N בחזקה הזאת (בחזקת C או לוג B של A, הם שווים) כפול לוג N.

**שיטת ההצבה החוזרת:** נפתח את הנוסחה עד שנבין כמה רמות יהיו עד שנגיע לבסיס ומה קורה בכל רמה.

**שיטת הניחוש:** אינדוקציה: נניח שהפונקציה קטנה מN כפול קבוע כלשהו עבור כל קלט עד N, (או קטן מביטוי כלשהו שתלוי בN) ואז נוכל להציב את זה בנוסחה (איפה שיש N חלקי משהו) ולפתוח ככה את הנוסחה ולמצוא את הקבוע.

**חוקי לוג:** כדי לעבור מבסיס X לבסיס Y של לוג N, מכפילים בלוג X של Y. כלומר, כל מעבר בין בסיסים סופיים של לוג זה פשוט הכפלה בקבוע. אז לוג N של כל בסיס סופי זה אותה סיבוכיות.




אפשר לראות, שבמספרים גדולים אמנם יש הבדל בין הלוגים, אבל הם הבדל של הוספת מספר כמעט קבוע.

X בחזקת לוג Y של Z שווה Z בחזקת לוג Y של X.

**סכום סדרה:** כשפותחים נוסחה בשיטת הצבה חוזרת, נקבל ביטוי של סדרת הנדסית. אם האיבר קטן מ1 נקבל סדרה מתכנסת. אם הוא גדול מ1, כל עוד הסדרה סופית הסיבוכיות תהיה כמו האיבר הכי גדול, כי בנוסחה של סכום סדרה זה האיבר הגדול פחות 1, חלקי A-1 שזה דברים קבועים אז זה לא משפיע על הסיבוכיות.

**כפל בינארי:** הפיצול לC,B,A מאפשר לעשות רק 3 חישובים שכל אחד הוא כפל על חצי ממספר הביטים. זה מה שנותן סיבוכיות קטנה יותר.

**חציון:**
נבחר פיבוט טוב: נחלק את המערך לקבוצות קטנות ככה שאפשר למיין כל אחת בזמן קבוע, אז המיון כולו הוא O(N).
ונשתמש בSELECT כדי לבחור חציון מתוך אלה. הזמן של זה יהיה T של (N חלקי 5).
מחלקים את המערך לפי מה שגדול או קטן מהפיבוט שבחרנו (זה O(N). בגלל שבחרנו חציון של החציונים, זה מוריד לנו לפחות N3 חלקי 10 איברים, ולכן הקריאה הרקורסיבית היא על N7 חלקי 10 איברים.
זה מה שקובע את הנוסחה.

**אלגוריתמים חמדניים:**
**אלגוריתם הופמן:** טענות עזר:
העץ יהיה מלא (כי אם יש ילד לבד, אפשר פשוט לאחד אותו עם אבא שלו)
שני המשקלים ברמה הכי נמוכה (כי יש שם שני קודקודים לפחות, ואם הם לא שם אפשר להחליף מקומות ולקבל קוד יותר טוב)
שני המשקלים יכולים להיות אחים (כי אם נחליף דברים באותה רמה זה לא משפיע)
אז באינדוקציה, אם ניקח את העץ של N איברים, השלב הראשון בהופמן מאחד את השניים הקטנים, ומשם העץ שנייצר שווה לעץ שהומפן מייצר עבור N-1 איברים, שהוא אופטימלי לפי הנחת האינדוקציה.

**תכנות דינאמי:** במקום להתחיל מהסוף ולבדוק כל פעם דברים אחורה כדי לקבל מידע שצריך, נתחיל מהבסיס ונתקדם עד שנגיע לתשובה.

**תת-מטריצה מקסימלית:** בונים את H בתכנות דינאמי בעזרת הכלה והדחה. O(N^2). ואז בודקים את כל תתי-המטריצות של A על ידי הכלה והדחה של תאים מH. זה O(N^4).

**סכום תת-קבוצה:** כדי לבדוק אם אפשר להגיע לסכום: נבדוק למה אפשר להגיע בלי האיבר החדש (השורה הקודמת בטבלה):
אם אפשר להגיע לסכום בשורה הקודמת, ברור שאפשר גם עם האיבר החדש.
ואם אפשר להגיע בשורה הקודמת ל"סכום פחות האיבר החדש", אז נגיע לזה ונוסיף את האיבר החדש.

**מרחק עריכה:** השורה ועמודה הראשונה: כדי להגיע ממחרוזת ריקה למחרוזת באורך N, צריך N פעולות.
אם התו האחרון שווה, אז המרחק עריכה הוא אותו מרחק כמו שהיה בלי התו הזה בשתי המילים. לכן הולכים אחד אחורה בשורה ובעמודה. (אלכסון למעלה ושמאלה).
אם התו האחרון שונה, אז נצטרך 1+ כדי לשנות את התו הזה, ועוד המרחק של אחד המצבים:
או בלי התו הזה (אלכסון למעלה ושמאלה),
או בלי התו האחרון רק באחת המילים (שורה למעלה או עמודה למעלה)
ניקח את המינימלי מבין שלושת המצבים האלה.

**BFS:**
מתחילים מתור ריק, מכניסים את הקודקוד הראשון.
עד שהתור ריק:
מוציאים את הראשון ומכניסים לסוף התור את השכנים שלו שעוד לא ביקרנו בהם.
בקודקוד שהוצאנו, מעדכנים את המרחק של השכנים שלו, וקובעים שהוא אבא שלהם במסלול.

הוכחת נכונות – טענות עזר:
אי שוויון המשולש – הדלתא של V זה פחות מדלתא של U ועוד המרחק מU לV.
לפי הגדרת האלגוריתם, D של V לא יכול להיות פחות מדלתא של V.
הקודקודים בתור מסודרים לפי D בסדר עולה, (כי מכניסים קודקוד רק כשמבקרים אותו מאבא שלו),
ויש הפרש של לכל היותר 1 בין הראשון והאחרון (באינדוקציה, כשנוציא קודקוד (הוא ראשון אז הוא עם D קטן מהאחרון) כל מה שייכנס הוא עם D שגדול ב1.

בסוף הBFS, הD של כל קודקוד שווה לדלתא שלו:
נב"ש שיש קודקודים שבהם D שונה מדלתא. ניקח את ההוא עם דלתא מינימלי. נקרא לו V.
מטענת עזר 2, D לא יכול להיות פחות מדלתא, אז הוא יותר.
לכן, V לא S (כי לS מראש יש D שווה דלתא), אז יש לו אבא במסלול. נקרא לאבא U.
נסתכל על U. אצלו מתקיים שD שווה דלתא (כי לקחנו את הקודקוד הכי מוקדם שבו הם שונים).
אז D של V הוא יותר (ממש) מדלתא של U ועוד 1.
נסתכל על הרגע שבו הגענו לקודקוד V דרך האבא U, נחשוב מה הצבע של V:
אם הוא היה לבן, היינו קובעים לו D תקין.
אם הוא היה אפור, זה אומר שהגענו אליו כבר דרך קודקוד אחר (נקרא לו W) שהוצאנו מהתור לפני U. בגלל שהוא יצא קודם מהתור, יש לו D קטן יותר, אז כשקבענו את D של V דרך W מקבלים שD של V הוא לכל היותר D של U ועוד 1, שזה תקין.
אם הוא היה שחור, הוא כבר יצא מהתור לפני U אז יש לו D קטן יותר.

כל אחת מהאופציות היא סתירה לכך שD יותר מדלתא של U ועוד 1.

**DFS:**
נרוץ על כל הקודקודים. בכל אחד שלבן, נעשה VISIT על כל השכנים שלו. בתוך VISIT קובעים זמן התחלה, ואז עושים VISIT רקורסיבי לכל השכנים, ואז קובעים זמן סיום.

**משפט הסוגריים** אומר שאם V הוא צאצא ל U, זה אומר שמתחילים ומסיימים את V בתוך הבדיקה של U.
ההוכחה נובעת מזה שהזמנים נקבעים בVISIT שהיא רקורסיבית.
אם אף אחד לא צאצא של השני, אז אחד מתחיל ונגמר לפני שהשני מתחיל.

**משפט המסלול הלבן** אומר שV צאצא של U בעץ אם ורק אם בזמן שגילינו את U היה מסלול לבן לV.
בכיוון הראשון – אם V צאצא ל U, בהכרח היה מסלול לבן כי האבא של קודקוד נקבע רק כשמגלים קודקוד לבן.
בכיוון השני – באינדוקציה על אורך המסלול הלבן, נניח שיש מסלול לבן ונוכיח שV יהיה צאצא של U.
עבור מסלול באורך 1, זה אומר שנגלה את V בבדיקה של השכנים של U אז הוא יהיה צאצא.
עבור אורך N: נסתכל בW, הקודקוד הקודם לV במסלול. מהנחת האינדוקציה, הוא צאצא של U כי המסלול ביניהם באורך N-1.
אם V לבן ברגע שבודקים את W, הוא צאצא של U דרך W.
אם V אפור או שחור: אז גילינו את V לפני W, אז נסיים אותו לפני W. ונסיים את W לפני U (מהנחת האינדוקציה). אז נסיים את V לפני U. ואנחנו כבר יודעים שגילינו את V אחרי U. אז ממשפט הסוגרים, V צאצא של U.

**סוגי צלעות:** הסוגים של הצלעות מבטאים מידע לגבי המצב של כל צלע בזמן שבדקנו אותה.
כשבודקים את השכנים של U: הצלע (U,V):
אם V לבן, זה צלע **עץ.** (tree edge). הצלע הזאת מתארת חלק מהמסלול שעשינו בחיפוש.
אם V אפור, זה צלע **אחורה.** (back edge). כלומר הגענו לU תוך כדי בדיקה של V, אז הצלע הזאת לוקחת אותנו אחורה במסלול. זה גם אומר שהגענו מ VלU ושוב לV, כלומר מעגל.
אם V שחור: זה אומר שכבר סיימנו את V, אז יש שתי אפשרויות:
- אם גילינו את U לפני V, זה אומר שV צאצא של U, פשוט גילנו את V בדרך עוקפת. זה צלע **קדימה** (forward edge) – מתארת "קיצור דרך" במסלול.
  בגרף לא מכוון זה לא יקרה כי כשבדקנו את V היינו חוזרים לU אז זאת הייתה צלע אחורה.
- אם גילינו את V לפני U, זה אומר שהתחלנו וסיימנו את V לפני שהתחלנו את U, כלומר אף אחד לא צאצא של השני בעץ. יש צלע מU לV אבל כבר סיימנו את V. זה צלע **חוצה.** (cross edge).
  גם לא יקרה בגרף לא מכוון, כי אם יש צלע אז היינו מגלים את U בזמן בדיקת V.

סיכום סוגי צלעות:
עץ – מתאר את המסלול
אחורה – רק אם יש מעגל
קדימה או חוצה – רק בגרף מכוון

אם יש צלע אחורה זה אומר שיש מעגל.
ואם יש מעגל אז יש צלעות אחורה.
נב"ש שיש מעגל בלי צלעות אחורה: נסתכל על הקודקוד הראשון שמגלים (נקרא לו U) הוא ראשון שמגלים במעל אז באותו רגע כל המעגל לבן, אז לפי משפט המסלול הלבן כל המעגל יהיו צאצאים שלו, וכשנגיע לV, הקודקוד שקודם לU במעגל, נבדוק את הצלע (V,U) ולפי הגדרה זה צלע אחורה.
מסקנה, יש צלע אחורה אם ורק אם יש מעגל.

עוד מסקנה – בעץ לא מכוון, יהיו רק צלעות עץ.

**גרף מכוון חסר מעגלים – DAG**
אפשר לעשות לו **מיון טופולוגי** – מיון לפי כיוון הצלעות. סידור של הקודקודים ככה שכל הצלעות הולכות באותו כיוון.
השיטה: נעשה DFS וכל קודקוד שנסיים, נכניס לראש הרשימה.
הוכחה שזה נותן מיון טופולוגי: נוכיח שאם יש צלע מU לV, נסיים את V לפני U:
נבדוק את הצבע של V בזמן בדיקת הצלע ביניהם:
- אם V לבן, V צאצא של U בעץ אז לפי משפט הסוגריים נסיים את V לפני U.
- V לא יכול להיות אפור כי זה היה מגדיר צלע אחורה, שזה סתירה לכך שאנחנו בDAG.
- אם V שחור, זה אומר שכבר סיימנו את V, כנדרש.

**רכיבי קשירות חזקה:** קבוצת קודקודים שאפשר להגיע מכל אחד לכל אחד, ואין עוד קודקודים שאפשר להוסיף לקבוצה הזאת בלי להרוס את התכונה.

אלגוריתם למציאת **גרף רכיבים** – נעשה מיון טופולוגי כדי לדעת באיזה סדר לרוץ, ואז נעשה DFS על הגרף הtranspose, לפי הסדר של המיון. כל עץ ביער שנקבל זה רכיב נפרד.
זה עובד כי אם אנחנו בודקים לפי המיון הטופולוגי, ברגע שנתחיל מקודקוד אנחנו נגלה את כל מה שאפשר להגיע אליו ממנו, ואין קודקוד אחר שאפשר להגיע ממנו אליי. ואחרי שנגלה את כל מה שאפשר, נתחיל עץ חדש וזה הרכיב הבא.
הוכחה פורמלית:
נוכיח טענת עזר: נסתכל על שני רכיבי קשירות B,A ונניח שיש מסלול מA לB. אז נסיים את B לפני A.
הוכחה:
אם הקודקוד הראשון בA התגלה לפני B, אז כל A צאצאים שלו (ממשפט המסלול הלבן) ובגלל שיש מסלול מA לB אז גם B צאצאים שלו. אז ממשפט הסוגריים, נסיים את B לפני A.
אם הקודקוד הראשון בB התגלה לפני A, ואין מסלול מB לA (כי יש מסלול מA לB, אז אם היה מסלול גם הפוך זה היה מעגל ואז הם לא היו רכיבי קשירות שונים), אז נסיים את כל B לפני שנעבור לA.

לכן, כשנעשה DFS לפי סדר זמני סיום, נבדוק קודם את A. ובגלל שאין מסלול מB לA בגרף המקורי, אין מסלול מA לB בtranspose, אז B יהיה עץ נפרד.

**בדיקה אם גרף הוא דו"צ:**
נרצה לבדוק אם אפשר לחלק את הקודקודים לשתי קבוצות כך שאין צלעות בתוך קבוצה.
נעשה BFS, אבל כשנמצאים בקודקוד ובודקים את השכנים, במקום לצבוע קודקוד באפור, נצבע אותו בכחול או אדום, לסירוגין.
ואם הקודקוד כבר צבוע באותו צבע של הקודקוד שאנחנו נמצאים בו, נחזיר FALSE.
הוכחת נכונות: אם חזר TRUE, זה אומר שכל קודקוד צבוע בכחול או אדום, ואין צלע בין שני קודקודים באותו צבע. אז הצבעים מתארים את הקבוצות.
אם חזר FALSE, זה אומר שהגענו לקודקוד שכבר ראינו. זה מעגל. ובגלל שהצבע מתחלף בכל צעד, זה אומר שהמעגל אי זוגי.
אם יש בגרף מעגל אי זוגי, הוא לא 2 צביע: כי כשנלך על המעגל, כל קודקוד חייב להיות בצבע שונה, ונגיע חזרה לאותו צבע.
אם גרף לא 2 צביע, הוא לא דו צדדי. (טענה ממבנים דיסקרטיים, גרף דו"צ אמ"מ הוא 2-צביע)

**כל שורש של גרף יהיה בעץ האחרון של היער**. כי ברגע שהגענו אליו, נגיע ממנו לכל שאר הקודקודים. פורמלית: נב"ש שיש שורש שנמצא בעץ אחר (נקרא לו R). כשמתחילים את הDFS, יש מסלול ממנו אל השורש של העץ האחרון (נקרא לעץ האחרון Z). חלק מקודקודים במסלול הזה נמצא בZ (יכול להיות שזה רק השורש של העץ). ניקח את הקודקוד הכי מוקדם במסלול שנמצא בZ (נקרא לו V). נסתכל על הקודקוד שקודם לו במסלול (נקרא לו U). הקודקוד הזה נמצא בעץ עם R, לפי משפט המסלול הלבן. ובגלל שיש צלע בין U לV, הם צריכים להיות באותו עץ.
אם יש שורשים, אז השורש של העץ האחרון הוא שורש. כי כל השורשים מוכלים בעץ האחרון, ומהשורש של העץ אפשר להגיע לכל קודקוד בעץ.

**גרף הוא גרף מעקפים אם ורק אם יש בו יעד גלובלי:**
כיוון ראשון, אם יש יעד גלובלי הוא מקיים את התנאי לגרף מעקפים לכל שני קודקודים.
כיוון שני, נניח שזה גרף מעקפים אבל אין יעד גלובלי. אז לכל קודקוד, מספר הקודקודים האחרים שאפשר להגיע מהם אליו (נקרא לזה ה-P של הקודקוד) קטן ממש מN. ניקח את הקודקוד עם הP המקסימלי (נקרא לו V). וניקח קודקוד שאי אפשר להגיע ממנו לV – בהכרח קיים כזה, כי אין יעד גלובלי. נקרא לו Z. אם יש מסלול מV לZ, אז אפשר להגיע לZ מכל קודקוד שאפשר להגיע ממנו לV, וגם מV. אז P של Z גדול מP של V. סתירה. ואם אין מסלול מV לZ, אז בגלל שזה גרף מעקפים, קיים קודקוד שאפשר להגיע אליו גם מV וגם מZ. אז הP שלו גדול מהP של V. סתירה.

יעד גלובלי הוא שור של הגרף הtranspose.

כדי לבדוק אם גרף הוא גרף מעקפים, נמצא שורש פוטנציאלי על הtranspose. (כדי למצוא שורש, נעשה DFS ואם מקבלים יותר מעץ אחד נעשה DFS מהשורש של העץ).

נתונה תת-קבוצה של קודקודים בגרף, נקרא לה W. רוצים למצוא את המסלול הכי קצר מאחד מקודקודי W (לא משנה איזה, כל אחד מהם), לקודקוד T. אפשר לעשות BFS מT על הגרף הtranspose, ולבדוק לאיזה קודקוד בW יש דלתא מינימלי. או, להוסיף קודקוד S שמחובר לכל הקודקודים בW, ולעשות BFS ממנו.

**בניתוח לשיעורין,** רוצים לחסום את הסיבוכיות לפעולה ממוצעת. יש כמה דרכים:
דרך אחת היא למצוא חסם עליון לN פעולות. (או חסם מאוחד לM פעולות מסוג אחד, ועוד K פעולות מסוג אחר, וכו...).
דרך שנייה היא להראות שכדי להגיע לפעולה שתיקח זמן N, צריך לעשות לפחות K פעולות שכל אחת היא סיבוכיות T, בדרך כלל O(1). ואז הסיבוכיות הממוצעת היא N כפול T, חלקי K.

**במבנה UNION-FIND,**
כשנאחד שורשים נכניס את השורש עם דרגה נמוכה מתחת לגדול, וזה גורם לכך שהדרך היחידה להגדיל דרגה היא לאחד שני קודקודים מאותה דרגה, ולכן בעץ עם דרגה K יש לפחות 2 בחזקת K קודקודים.

ובגלל שכדי להגדיל דרגה צריך שני קודקודים עם אותה דרגה, סכום הדרגות חסום בN.

ואם נעשה דחיסת מסלול, זה גורם לקודקודים להתקרב לשורשים.
נגיד שיש עץ שהוא רק ענף אחד באורך N.
אם נחפש קודם את האחרון – הזמן לזה יהיה N, והזמן לכל החיפושים אחרי יהיה 1.
אם נחפש את הראשון, ואז השני, וכו', הזמן לכל אחד יהיה 1.
אם נתחיל מהאמצעי, הזמן שלו יהיה חצי N, ואחרי זה לכל מה שהיה מעליו יהיה 1, ולמקסימלי ממה שהיה מתחתיו חצי N...
בסה"כ רואים שדחיסת מסלול גורמת לזמן חיפוש ממוצע להיות 1. אבל יכול להיות שהדחיסה עצמה לוקחת הרבה זמן.
אז רק צריך לחסום את מספר הצעדים שעושים בהרבה דחיסות מסלול ביחד, ולחסום את זה.
כדי לעשות את זה, נגדיר יחס בין הדרגה של קודקוד לדרגה של אבא שלו.
נראה שיש לו חסם עליון שמוגדר על ידי נוסחה רקורסיבית (כאן האקרמן נכנס – אקרמן היא פונקציה רקורסיבית).
בכל צעד בדחיסת מסלול, היחס הזה משתנה. (קוראים לזה "קידום" של הקודקוד).
בכל דחיסת מסלול יש חסם תחתון למספר הקידומים, שמגדיר לנו חסם תחתון למספר הצעדים בכל דחיסות המסלול.
אבל יש גם חסם עליון למספר הפעמים שקודקוד יכול להיות מקודם, בגלל החסם העליון ליחס של הדרגות.
החסם הזה מוגדר על ידי הפונקציה ההפוכה של אקרמן (כי זה ההפוך של מה שמגדיר את היחס של הדרגות).
אז לכל דחיסת מסלול צריך לפחות X קידומים, אבל כל קודקוד עם דרגה Y יכול להיות מקודם עד Y כפול אלפא פעמים.
וסכום כל הדרגות חסום בN. זה מגביל לנו את מספר הצעדים הכולל לבערך N כפול אלפא, עבור N קודקודים.
אז הזמן הממוצע לכל קודקוד הוא אלפא.

**עץ פורש מינימלי – MST:**
אם יש קבוצה שבוודאות מוכלת בעץ פורש מינימלי, ומוסיפים לה צלע שבוודאות חלק מעץ, היא עדיין מוכלת בMST כלשהו.
ואם נעשה את זה N-1 פעמים, נקבל MST.

צלע כזאת, שבוודאות חלק מMST כלשהו – נקראת צלע בטוחה.

אם יש קבוצה שממוכלת בMST, ויש חתך שמכבד את הקבוצה הזאת, אז הצלע הכי קלה שחוצה את החתך הזה, היא בטוחה.
הוכחה: יהי MST כלשהו, נקרא לו T. אם נניח שהצלע (נקרא לה E) לא חלק מ-T, אז יש עוד צלע ב-T (נקרא לה F) שחוצה את החתך (כי חייב להיות חיבור בין הקבוצות כדי שזה יהיה עץ פורש). F כבדה יותר מE (כי הנחנו שE הכי קלה), אז נוכל לקחת את E במקום F ולקבל עץ קל יותר. סתירה.

**האלגוריתם של קרוזקל:** משתמש בUNION-FIND:
נמיין את הצלעות לפי משקל.
כל קודקוד מתחיל לבד.
נבדוק כל צלע – אם הקודקודים שלה נמצאים באותה קבוצה, היא לא חוצה את החתך.
אם היא כן חוצה את החתך, בגלל שבודקים לפי הסדר היא הצלע המינימלית שחוצה את החתך, אז היא חלק מהעץ. נאחד את הקודקודים שלה ונוסיף את הצלע לMST.

**האלגוריתם של פרים:**
נעשה ערמת מינימום של קודקודים – המפתח של כל קודקוד הוא המשקל של הצלע הכי קלה שמחברת אותו לעץ, וכל קודקוד מחזיק שדה P שזה אבא שלו בMST.
ככה תמיד ניקח את הצלע הבטוחה.
נוציא את הקודקוד המינימלי – V. הוא חלק מהעץ. (בקודקוד הראשון זה לא משנה איזה).
נעבור על השכנים שלו שעדיין נמצאים בערמה.
לכל אחד (נקרא לו U), אם הצלע בין V ל-U קלה יותר מהמפתח הנוכחי של U, נשנה את המפתח ונעדכן שV אבא של U בעץ.
בסוף יש לנו PARENT POINTER TREE – זה הMST.

אם יש צלע קלה ביותר, היא תהייה חלק מהMST. הוכחה – נב"ש שהיא לא בMST, אז היא סוגרת מעגל בMST, אז נוכל לקחת את הצלע הזאת במקום צלע אחרת במעגל ולקבל MST קל יותר.

נתון גרף לא מכוון עם משקלים על הקודקודים. נחשב משקל של עץ בצורה הבאה: לכל קודקוד, המשקל שלו כפול הדרגה שלו. נרצה למצוא MST כזה: נגדיר משקלים על הצלעות: כל צלע מקבלת משקל שהוא סכום הקודקודים, ונמצא MST רגיל. נוכיח שהMST הזה הוא מה שאנחנו רוצים:
הMST הזה ממזער את הסכום של: לכל צלע בעץ, סכום משקלי הקודקודים שלו. שזה אומר, נספור את המשקל של כל קודקוד פעם אחת עבור כל צלע שמחוברת אליו שנמצאת בעץ. שזה בדיוק הסכום שאנחנו רוצים למזער.

בהינתן גרף לא מכוון וקשיר, נרצה למצוא לפחות צלע אחת מכל מעגל, עם משקל מינימלי : נמצא עץ פורש **מקסימלי** T, (באלגוריתם של פרים, למיין מהגדול לקטן. או באלגוריתם של קרוזקל, להשתמש בערמת מקסימום). ונחזיר את כל הצלעות שלא בעץ הזה. (נקרא לקבוצה הזאת 'E)
הוכחה שזה עובד: בגלל שזה מחזיר עץ, לכל מעגל יש לפחות צלע אחת שלא בעץ.
נוכיח שהצלעות מינימליות – נב"ש שיש קבוצה יותר מינימלית, *E, שאפשר להחזיר. אם נסתכל על הצלעות שקיימות בE ולא ב*E, זה יער שהוא תת קבוצה של עץ פורש, נקרא לו 'T. המשקל של 'T ועוד *E הוא לפחות המשקל של E, ששווה למשקל של T ועוד 'E. ובגלל שT עץ פורש מקסימלי, המשקל שלו הוא לפחות המשקל של 'T. אז המשקל של *E צריך להיות יותר מ- 'E. סתירה.

לגרף לא מכוון וקשיר שבו אין 2 צלעות עם אותו משקל, יש MST אחד. הוכחה: נב"ש שיש 2 שונים, אז קיימת לפחות צלע אחת שנמצאת באחד ולא בשני. ניקח את הקלה מהן. היא סוגרת מעגל בעץ השני. במעגל הזה היא הכי קלה כי אחרת היינו לוקחים צלע אחרת בעץ הראשון. אז נוסיף אותה לעץ השני ונוריד צלע אחרת כבדה יותר מהמעגל. וקיבלנו עץ קל יותר.

**מסלולים קצרים ממקור יחיד:**
אם הקודקוד הקודם ל-V במסלול הקצר (נקרא לו U) כבר התכנס, ועושים RELAX לצלע UV, אז גם V מתכנס. הוכחה: נב"ש שלא. זה אומר שגם אחרי שקבענו ש-D של V הוא D של U ועוד הצלע ביניהם, יש דרך קצרה יותר ל-V. אז זה אומר ש-U הוא לא הקודקוד הקודם ל-V במסלול הקצר, סתירה.

**אלגוריתם דיקסטרה:** תור קדימויות של הקודקודים, לכל קודקוד שיוצא נעשה RELAX על הצלעות שלו. זה מעדכן את ה-P של הקודקודים וגם מעלה אותה למעלה בערמה אם צריך.
האלגוריתם עובד כי כל קודקוד שיצא מהערמה כבר התכנס: הראשון שיוצא זה המקור S, הוא מכונס בהגדרה. נב"ש שיש קודקודים שיצאו לפני שהתכנסו, ניקח את U, זה שיצא ראשון מביניהם, ונקרא P למסלול הקצר מ-S ל-U. בזמן ש-U יצא מהערמה, היה לפחות קודקוד אחד של P שנמצא בערמה. ניקח את ההוא שנמצא הכי מוקדם ב-P, ונקרא לו Y.
(יכול להיות ש- U=Y). נסתכל על אבא של Y במסלול – נקרא לו X. בגלל ש-U הוא הראשון שלא התכנס, ו-Y הוא הכי מוקדם במסלול שלא יצא מהתור, X בהכרח התכנס ויצא מהתור. כש-X יצא, עשינו RELAX על הצלע XY, אז Y התכנס. כלומר Y הוא לא U. (אם המסלול היה באורך שמחייב ש Y=U, הגענו לסתירה. נניח שהמסלול מספיק ארוך). אז מתקיים:

1. D של Y שווה דלתא של Y (כי Y התכנס).
2. דלתא של Y קטן או שווה לדלתא של U (כי Y קודם לU במסלול)[1]
3. דלתא של U קטן ממש מ-D של U (כי U לא התכנס)

בסה"כ, ברגע שמוציאים את U, ה-D של Y קטן מה-D של U. אבל לפי האלגוריתם, מוציאים קודקודים לפי סדר ה-D שלהם. אז היינו אמורים להוציא את Y קודם. סתירה.
הסיבוכיות של דיקסטרה: אם משתמשים בערמת פיבונאצ'י, עושים M פעמים DECREASE, שכל אחד הוא זמן קבוע, ועושים N פעמים EXTRACT, שכל אחד הוא LOGN. סה"כ M+NLOGN.

**אלגוריתם BELLMAN FORD**
נרוץ N-1 פעמים, נעשה RELAX על כל הצלעות. הוכחה שזה עובד: כבר הוכחנו שאם U, הקודקוד הקודם ל-V במסלול הקצר, התכנס, אז RELAX(UV) מכנס את V. בכל מסלול יש עד N-1 צלעות. ובכל פעם נעשה RELAX על כל הקודקודים. אז בכל פעם, לפחות עוד קודקוד אחד בכל מסלול יתכנס.
בפעם הN, אם יש עוד צלע שיכולה להתכנס, זה אומר שיש מעגל שלילי (ושהצלע הזאת מחוברת אליו).

נתון גרף לא מכוון, ותת קבוצה של הקודקודים, W. רוצים מסלול קצר מ-S ל-T שעובר דרך כמה שפחות קודקודים מW. ניתן לכל צלע משקל: על כל קודקוד מ-W שמחובר לצלע, נוסיף 1. (המשקל של כל צלע יהיה 0, 1, או 2). נמצא מסלול קצר עם

[1] זו הסיבה שהאלגוריתם לא עובד עם משקלים שליליים – הטענה לא מתקיימת אם המשקל של המסלול לא מונוטוני עולה

דיקסטרה. כל קודקוד ששייך ל-W שנעבור בו יוסיף 2 למשקל המסלול, אז אם נמזער את משקל המסלול נקבל מסלול עם כמה שפחות קודקודים מ-W.

נתון: גרף לא מכוון, עם משקלים אי-שליליים. נרצה למצוא את המסלול **באורך זוגי** הקצר ביותר מ-S ל-T. נייצר גרף חדש: נייצר שתי עותקים של כל קודקוד, A ו-B. נייצר צלעות לפי הגרף המקורי: כל צלע מחברת בין קודקוד מ-A וקודקוד מ-B. נמצא מסלול בין S של A ל-T של A. בגלל שאין צלע ישירה מ-A ל-A, המסלול שנמצא חייב להיות זוגי. ובגלל שבנינו צלעות רק איפה שכבר היו, המסלול הזה קיים בגרף המקורי.

צלע נקרא **צלע טובה** היא הצלע האחרונה במסלול קצר כלשהו. מסלול קצר אמ"מ מורכב כולו מצלעות טובות. הוכחה: כיוון ראשון: אם מסלול הוא קצר, לפי תכונת תת מסלו קצר כל הצלעות הן טובות. כיוון שני: אם יש מסלול שכולו צלעות טובות, נוכיח באינדוקציה שהוא קצר: אם הוא באורך 1, הצלע האחרונה זה כל המסלול. נניח שמתקיים ונוכיח עבור N: אם המסלול כולו צלעות טובות, בפרט האחרונה טובה אז יש מסלול קצר שמסתיים בה. ניקח את המסלול בלי הצלע, ע"פ הנ"א זה מסלול קצר. נוסיף חזרה את הצלע האחרונה, אם זה לא מסלול קצר זה סתירה לכך שהצלע טובה.

כדי למצוא את כל הצלעות הטובות בגרף, נריץ דיקסטרה ונבדוק לכל צלע אם הדלתא של V שווה הדלתא של U ועוד המשקל של הצלע UV.

כדי לבדוק אם צלע מסוימת נמצאת על **כל** מסלול קצר מ-S ל-T, קודם נבדוק אם היא צלע טובה. אם היא כן, נייצר גרף שהוא רק הצלעות הטובות בלי הצלע הזאת, ונראה אם עדיין קיים מסלול מ-S ל-T. אם קיים זה אומר שיש מסלול שכולו צלעות טובות (כלומר קצר) שלא משתמש בצלע הזאת.

נתון שאין מעגלים שליליים: כדי לבדוק אם יש מעגל במשקל 0: נוסיף קודקוד S שמחובר לכל הקודקודים עם צלע במשקל 0. נריץ BF ונייצר גרף חדש שהוא רק הצלעות הטובות. אם יש מעגל, הוא במשקל 0. (כי אין מעגלים שליליים, אז אם יש מעגל שכולו צלעות טובות זה מסלול קצר, אז הוא חייב להיות 0).

אפשר להריץ דיקסטרה יעיל יותר על DAG: בלי ערמה, פשוט לעשות מיון טופולוגי ולעשות RELAX לפי הסדר הזה. כי כל המסלולים חייבים להיות גם לפי הסדר הזה, ואם עושים RELAX לפי הסדר של המסלול, כל קודקוד מתכנס בתורו).

**ערמת פיבונאצ'י:**
כשמוציאים את המינימלי, עושים CONSOLIDATE – זה מה שנותן דרגה מינימלית לכל קודקוד, ונותן לנו לחסום את מספר הפעולות הכולל עבור מספר הכנסות, הוצאות, והקטנות.
נייצר רשימה של פעולות שכל אחת היא זמן קבוע, ונראה כמה פעמים כל אחת תקרה.
הטענה העיקרית היא שבגלל הדרך שבה ממזגים קודקודים, לכל קודקוד עם דרגה K יש לפחות F(K+2) צאצאים. (המספר פיבונאצ'י ה- K+2). ויש חסם תחתון למספר הילדים של כל קודקוד, אז יש חסם עליון למספר הילדים של שורש. זה נותן לנו חסם עליון למספר הפעמים שנהפוך קודקוד לבן של קודקוד אחר, שזו הפעולה שתקרה הכי הרבה פעמים (אסימפטוטית).

**מסלולים קצרים בין כל הזוגות:**
נותן שתי מטריצות: אחת של המרחקים, אחת של מצביעים לקודקוד הקודם במסלול: כדי להגיע מ-I ל-J, צריך להגיע לקודקוד שנמצא שם, נגיד K. אז נבדוק איך להגיע מ-I ל-K... נדפיס את זה ברקורסיה, תנאי הבסיס הם אם אין דרך (במקרה הזה המצביע יהיה NULL) או אם הגענו ל-I.

פתרונות לכל המסלולים:
1) להשתמש באלגוריתם SINGLE SOURCE, N פעמים. הסיבוכיות זה פשוט הסיבוכיות המקורית כפול N.
2) תכנות דינאמי על מטריצת שכנויות: נרוץ על אותה מטריצה, בכל פעם נוסיף 1 לאורך של המסלול שמותר. הדרך לעשות את זה: בשביל המיקום ה-J I, נרוץ על כל השורה ה-I ובכל מקום נבדוק מה יקרה אם נגיע לשם ונוסיף את הצלע האחרונה במסלול. אם הצלע לא קיימת, זה נותן מרחק אינסוף. ניקח את המינימום מכל האופציות האלה. כל בדיקה כזאת היא N. זה קורה בכל אחד מ- N^2 תאים, N פעמים. סה"כ N^4.
3) גיאומטריה טרופית: במקום למצוא מינימום מבין N אופציות, נשתמש בכפל מטריצות שמיישם את זה: חיבור במקום כפל ומינימום במקום חיבור. זה אסוציאטיבי, ואם נכפיל ככה מטריצות, המטריצה "בחזקת N" מייצגת את המטריצה עבור מסלולים עד אורך N. כל שלב הוא עדיין N^3, אבל אם מכפילים בחזקות של 2 צריך רק LOGN כאלה. אז N^3LOGN.
4) אלגוריתם FLOYD WARSHALL – קודקודי ביניים. נרוץ על אותה מטריצה N פעמים. בכל פעם, "מרשים" להשתמש בעוד קודקוד K בתור קודקוד ביניים. הבדיקה היא רק להשוות בין המרחק שהיה לפני, למרחק שיהיה אם נלך ל-K ואז ל-J. זה N^3, חסם הדוק.

נשים לב שאם הגרף יחסית ריק – אם M קרוב ל-N – אז יותר מהיר לעשות דיקסטרה N פעמים. (זה N^2).
אם יש משקלים שליליים אי אפשר, אבל אפשר לעשות REWEIGHTING. המשפט אומר שאם מגדירים משקל לכל קודקוד, ולכל צלע UV נוסיף את U ונחסיר את V, זה ישמר את המשקל היחסי של המסלולים (כי כל המשקלים חוץ מהראשון והאחרון יצטמצמו, אז השינוי במשקל של כל מסלול מ-S ל-T יהיה ועוד S פחות T. אם כל המסלולים מ-S ל-T השתנו באותו מספר, המסלול הקצר שבחרנו ככה יהיה הקצר ביותר גם בגרף המקורי).
הדרך לבחור משקלים – אלגוריתם ג'ונסון: נוסיף קודקוד S שמחובר לכל הקודקודים עם צלע במשקל 0. נריץ בלמן פורד אחד. הערך שנקבע להוסיף לכל קודקוד זה הדלתא שלו. אם יש מסלול במשקל שלילי מקודקוד לקודקוד כלשהם, האלגוריתם הזה ימצא אותו.
(פורמלית, אי שוויון המשולש מראה שהערך שנוסיף לV קטן או שווה לערך שנוסיף לU ועוד המשקל של UV, ואם נעביר אגף נקבל את המשקל החדש של הצלע שגדול או שווה 0).
עכשיו הגרף אי שלילי, נעשה דיקסטרה N פעמים. בכל אחד, אחרי שמצאנו את המרחקים, נחסיר ונוסיף את מה ששינינו כדי לקבל את המרחק האמיתי.

כדי למצוא את המסלול הקל ביותר עם בדיוק K צלעות: תכנות דינאמי, טבלה בגודל NK שבה התא ה-J I הוא המסלול הקל מ-S ל-I באורך J. בכל תא ניקח את המינימלי מבין השורות הקודמות.

מציאת מסלול שעובר דרך לפחות קודקוד אחד בקבוצה: נבנה 3 עותקים של הגרף ונחבר ביניהם רק עם צלעות שמגיעות/יוצאות מהקבוצה הזאת.

**רשת זרימה:**
הזרימה של כל חתך שווה למה שיוצא מ-S: כי בכל פעם שנוסיף קודקוד לחתך, הצלעות קדימה/אחורה משתנות וזה מצטמצם. כי מה שנכנס שווה מה שיוצא.
ערך F חסום בקיבולת החתך. (כי זה המקסימום שיכול לעבור קדימה).
כדי להגדיל פונקציית זרימה, נמצא מסלול מגדיל ונגדיל לפי הקיבולת השיורית. זה יגדיל את הזרימה כי לכל צלע קדימה מוסיפים את הקיבולת השיורית, ולכל צלע אחורה מורידים אותה. הצלע היחידה במסלול שמחוברת ל-S יוצאת מ-S, אז מגדילים את הזרימה של החתך שהוא רק S. והראנו שהזרימה בכל חתך שווה לזה.

פונקציית זרימה מקסימלית אמ"מ אין מסלולים מגדילים, והזרימה המקסימלית שווה החתך המינימלי:
הוכחה ע"י זה ששלושת ההיגדים שקולים:
1) F מקסימלית
2) אין מסלול מגדיל
3) קיים חתך שבו ערך F שווה קיבולת החתך

1 גורר 2 כי אם היה מסלול מגדיל היא לא הייתה מקסימלית.
3 גורר 1 כי הראנו שקיבולת חתך תמיד גדולה שווה לערך F.
2 גורר 3: נניח שאין מסלול מגדיל – כלומר אין מסלול מ-S ל-T ברשת השיורית. נסתכל על הרשת השיורית, בוודאות יש קודקודים שאי אפשר להגיע אליהם מ-S (אולי רק T, אולי יותר). אם ניקח חתך שבו הקבוצה של S זה כל הקודקודים שאפשר להגיע אליהם מ-S, אז בגלל שברשת השיורית אין צלעות שחוצות את החתך קדימה, בגרף המקורי לכל הצלעות שחוצות את החתך קדימה יש זרימה מקסימלית. ובגלל שאין צלעות שחוצות את החתך אחורה, לכל הצלעות אחורה יש זרימה 0. אז ערך הזרימה שווה לקיבולת החתך.

**שיטת FORD FULKERSON**
בכל פעם לוקחים מסלול מגדיל ומגדילים לפי הקיבולת השיורית.

**אלגוריתם EDMONDS KARP**
כדי להחליט מה המסלול המגדיל לוקחים את המסלול הקצר (לפי מספר צלעות, BFS).
בכל פעם שלוקחים את המסלול הזה, לפחות צלע אחת נעלמת (הצלע הקריטית) וזה מגדיל את מספר הצלעות במסלול המגדיל. המסלול לא יכול להיות יותר מאורך N כי זה מסלול פשוט, אז זה חוסם את מספר הפעמים שכל צלע יכולה להיות קריטית – חצי N. כפול מספר צלעות – סה"כ NM חלקי 2. כפול BFS שהוא M כי הגרף קשיר, אז O(NM^2)