דאסט סיכום טריקים ומסקנות מתרגילים ומבחנים

טיפים כלליים

- **טיפים כלליים למבחנים:**
  - לראות אם סעיפים קודמים(!) של אותה שאלה יכולים לשמש לפתרון / לרמוז מה הוא צריך להיות.
  - גם אם הנתונים של השאלה מאוד מזכירים אלגוריתם מוכר או בעיה מוכרת, צריך להתמקד ולחפש אם יש איזה דקות קטנה שפספסנו.
  - אם מבקשים ליצור מבנה נתונים, שווה לבקש הבהרה לגבי אם הוא יכול להיות מורכב מיותר ממבנה נתונים מוכר אחד.
  - לא להנעל על דרך מסוימת לגשת לפתרון, ולזכור לשאול את עצמנו לאורך הדרך - האם אני לא מוסיפה לעצמי עבודה סתם? האם השלב הזה באמת הכרחי או שיש דרך אחרת לפתור שלא כוללת אותו?

- **איך לחשוב על אלגוריתם:**
  - זמן הריצה שמבקשים מאיתנו / דברים אחרים בנתוני השאלה (כמו דרישה להוכיח באמצעות שמורת לולאה) יכולים מאוד לרמוז על איך הם מצפים שנפתור את זה ואסור להתעלם מזה גם אם אנחנו לא מבינים בהתחלה מה זה רומז.
  - אם יש לנו בעיה רקורסיבית, אפשר לנסות לחשוב להתקדם במורד הרקורסיה גם באמצעות שינוי פרמטר "לא טריויאלי" (כמו שינוי ה-k בשאלה של ה-AVL ב-2020א)
  - כשמדברים על רביעיות, אפשר לחשוב על זה בתור זוגות של זוגות. באותו אופן, שלישיה יכולה להיות צירוף של זוג ויחיד.
  - בשאלות שבהן מבקשים למצוא אם קיימים שני איברים שנסכמים ל-X, יש שתי גישות עיקריות - או לגבב את ההפרש, או ליצור מערך ממוין ולאתחל פוינטרים בקצוות ולקדם אותם אחד לכיוון השני עד שהם נפגשים.
  - באלגוריתמים שבהם מבקשים מאיתנו לחפש ערך k כלשהו, חשוב להוסיף בדיקת שפיות שבודקת אם הערך של k בכלל הגיוני (נגיד אם אנחנו במערך ממוין, אז שהוא לא גדול מהערך הכי גדול במערך).
  - בבעיות שכוללות טווח מסוים, חשוב לשים לב אם הוא כולל את הקצוות או לא ולהתאים את האלגוריתם לכך.
  - אם אומרים לנו למצוא אלגוריתם שעומד בזמן ריצה מסוים, זה לא בהכרח חייב להיות חסום ע"י החסם האסימפטוטי הזה מסיבה אינטואיטיבית (כמו לולאה, רקורסיה ל-$logn$ וכו'), אלא זה יכול להיות כתוצאה מניסוח של נוסחת הנסיגה ושימוש ישיר במשפט האב (גם אם התוצאה לא אינטואיטיבית, לדוגמא אם יוצרים עץ לא מאוזן של $\frac{1}{10}$ ו-$\frac{9}{10}$, שחסומה ע"י $\Theta(n)$).

- **איך להוכיח דברים:**
  - בהוכחות בהם צריך להוכיח ש"כל X הוא בהכרח קטן מ-Y", אפשר לנסות להגדיר חסם עליון של Y וחסם תחתון של X ולהראות את היחס ביניהם. אפשר לחשוב אם אפשר להגדיר את החסמים של X, Y ע"י אותה נוסחא ואז צריך להוכיח אותה רק פעם אחת.
  - אם יש בעיה של "האם קיים X בטווח מסוים", אפשר לבחון אם להשתמש בעקרון שובך היונים.
  - כשמוכיחים נכונות של אלגוריתם באינדוקציה, אם זה אלגוריתם רקורסיבי שמקטין את גודל הבעיה, אז ברגע שמגיעים לשלב באלגוריתם שבו קוראים קריאה רקורסיבית, אפשר להגיד שהיא מחזירה תוצאה נכונה מהנחת האינדוקציה.
  - בשאלות שמכילות אלמנט כלשהו של השוואה (האם קיים אלגוריתם מבוסס השוואות שעושה X), לחשוב ישר על שימוש בעץ החלטות - לחשב כמה פרמוטציות יהיו וככה לחשב את עומק העץ המינימלי. $num\ of\ permutations \leq depth$ ואז עושים לוג לשני הצדדים.
    - לזכור שזה לא עובד עבור מספרים קטנים, אבל שבמקרים כאלו אפשר להשתמש בדיוק באותה שיטה רק הפעם עם מספר קונקרטי של פרמוטציות.
    - עוד אפשרות שלפעמים קלה יותר היא להגיד "נניח בשלילה שקיים אלגוריתם שעושה את זה עם השוואות ולוקח ____ (זמן כלשהו שיותר מהיר מ-$nlogn$), ואז לתאר איך משתמשים בזה כדי למיין מערך ולהראות שזה לוקחת פחות מ-$nlogn$ בסתירה לזה שהוכחנו בהרצאה שזה חסם תחתון.

- **שמורת לולאה**
  - אם מבקשים לכתוב אלגוריתם ואז להוכיח אותו בשמורת לולאה, אפשר לנסות להבין מזה איך האלגוריתם עצמו אמור לעבוד, ע"י להבין מה קורה בסיום האיטרציה האחרונה, ומשם להסיק מה קורה בסיום האיטרציה ה-i ולבנות את האלגוריתם בהתאם!

- חשוב לנסח את הטענה באופן מדויק שמתבסס על איך שכתבנו את האלגוריתם (לא רק את החלק של הלולאה עצמה אלא גם מה שלפניו, אם היא "מתקשרת" איתו).
- לא לשכוח שהוכחה של שמורת לולאה מוכיחה רק את החלק של הלולאה ולא את האלגוריתם כולו, לכן אחרי שמסיימים לכתוב את השמורת לולאה, עדיין צריך להוכיח נכונות לאלגוריתם כולו (שמתבסס על האיטרציה האחרונה בשמורת לולאה).

חסמים אסימפטוטיים

- אם יש לנו נוסחה רקורסיבית שכל פעם גודל הבעיה גדל/קטן בצורה אחרת (כלומר לא מתחלק באופן ברור שאפשר לפרמל), אפשר לתת לו סימון m ולהסביר מה הסימון מייצג, ואז לכתוב $T(m)$ בדיוק כמו שהיינו כותבים $T(\frac{n}{2})$ או כל דבר אחר בסגנון.
- כללי אצבע לסדרי גודל של פונקציות:
  - לכל הלוגים (לא משנה הבסיס) יש אותה סיבוכיות (כי בלוגים מעבר בין בסיסים מתבצע ע"י כפל בקבוע).
  - למרות שלוגים מבסיס שונה הם באותו סדר גודל, אם הם משמשים כמעריך זה לא נכון, והקטן מביניהם יהיה זה עם הבסיס הגדול יותר.
  - אם יש לנו שתי פונקציות f, g שהן Θ אחת של השניה, זה <u>לא</u> אומר שאם נשים אותן בתור מעריך (של 2 לדוגמא), זה עדיין יהיה Θ אחת של השניה.
  - $loglog$ זה יותר איטי מ־log.
- **הוכחת חסם אסימפטוטי:**
  - כדי להוכיח Θ, צריך לחשב את כמות העבודה בכל העץ ואז לחסום מלמעלה ומלמטה.
  - אפשר להשתמש באינדוקציה כדי להוכיח חסמים.
  - גם אם יש משהו שברור לנו מה הסיבוכיות שלו (נגיד $nlogn$), אם אנחנו לא יודעים לתת הסבר מדויק ללמה זה הסיבוכיות, אפשר לעשות הוכחה זריזה של סכימה של כמות העבודה בכל שכבה.
  - בשאלות שמערבות לוגים, לחפש רעיונות בחוקי לוגים בדף נוסחאות!
  - להוסיף כאן על השיטה עם ההצבת S (נגיד במקרה של שורש)
  - **משפט האב:**
    * $\lim_{n\to\infty}\frac{f(n)}{g(n)} = 0 \Longrightarrow g(n) = \Omega(f(n))$. זה שימושי כשרוצים להוכיח ש־g חסומה מלמטה ע"י f (נגיד במשפט האב המורחב).
    * לכל $\varepsilon > 0$ מתקיים $log_2 n = O(n^\varepsilon)$ - שימושי למשפט האב המורחב.
    * אם הפונקציה בסוף היא פשוט $O(1)$ זה אותו דבר כמו n^0.
    * זה בסדר אם זה לא כתוב ישירות בתור $\frac{n}{b}$ אלא $\frac{b_1}{b_2}n$ כי אז $b = \frac{b_2}{b_1}$.
    * אם יש לנו משהו בסגנון $2^{log_5 n}$ אפשר לחשוב להחליף את ה־2 ב־5 (שעוזר לנו לחשב בקלות) ואז זה חוסם את זה מלמעלה.
    * להשתמש בחוק לוגים $a^{logn} = n^{loga}$.
    * לחשוב על שורש בתור חזקה.
    * דוגמאות קלסיות ל**מקרים שלא ניתן להשתמש** במשפט האב המורחב:
      · אם מתקיים המקרה השלישי אבל לא קיים C שמקיים.
      · אם הנוסחא לא מהצורה (נגיד אם הרקורסיה מתקדמת באמצעות חיסור ולא באמצעות חלוקה).
  - **איך מחשבים לפי עץ רקורסיבי:** סוכמים עד גובה העץ, את כמות העבודה ברמה ה־k (כמות העבודה בקודקוד יחיד כפול כמות הקודקודים, לפעמים קל לזהות את הנוסחא ע"י להסתכל על ה־$f(n)$ ולחשוב איך היא תראה ברמה ה־k כתלות בערך של b, ואז את כל זה לכפול בכמות הקודקודים שהיאa^kאם $a > 1$).
  - נוסחה רקורסיבית יכולה להיות מפוצלת למקרים אם צריך (נגיד אם היא מתנהגת באופן רקורסיבי רק החל ממקום מסוים).
  - **טריקים אלגבריים שימושיים** -
    * להפריד סכימה.
    * חוק לוג של מנה.
    * סדרה חשבונית.
    * להוציא מהסיגמה ביטויים שלא תלויים ב־i שהסיגמה רצה עליו.
    * אם יש שורש, לחשוב עליו בתור חזקה.
    * חוקי לוגים ובפרט להעביר את החזקה להיות מקדם.
  - טריק לחסימה מלמטה - לשנות את הסכימה מהאמצע עד לסוף, ואת האיבר שסוכמים עליו לאיבר באמצע.

מיון ופעולות על מערכים

- אפשר לשקול שימוש כלשהו בערמת מינימום אם זה מתאים לנתונים של השאלה (כלומר לא לברוח ישר לאלגוריתמי מיון).
- להוכיח שאלגוריתם הוא יציב: לקחת שני אינדקסים שונים i, j כך ש־$A[i] = A[j]$. להראות מה קורה כשהאלגוריתם מגיע לכל אחד מהאינדקסים כדי להסיק שבסוף הסדר ביניהם נשמר.
- אם המערך מכיל מספרים שלמים ושונים זה מזה, טריק חשוב הוא ש $A[i] < A[i] + 1$.
- **רעיונות מהתרגילים לאלגוריתמים שימושיים**
  - **חיפוש מספר חסר במערך טבעיים ממוין:** אם רק צריך לבדוק אם הוא קיים, נציע אלגוריתם פשוט נורא שבודק אם האיבר האחרון שונה מהאיבר הראשון + $n - 1$. אם צריך להחזיר אותו (אם קיימים כמה, להחזיר אחד מהם שרירותית), נציג אלגוריתם רקורסיבי שמתקדם אחורה עם אריה במדבר (מריץ את האלגוריתם הקודם על שתי מחציות המערך, ומגלה במי מהן יש ערך חסר, וככה יודע באיזה מחצית להמשיך לחפש). רץ ב־$\Theta logn$.
  - **ערכים בטווח:** אלגוריתם ב־$\Theta(n+k)$ שעבור מערך של n מספרים בערכים $[0, k]$ ושני ערכים $a < b$, מחזיר כמה איברים הם בין a, b. הוא פשוט משתמש במערך סופר C כמו ב־$counting\ sort$ (כולל החלק של החיסור של הקודמים), כדי לקבל את האיברים בין a, b צריך לחשב $C[b] - C[a-1]$.
  - **מציאת אחוזון p:** כלומר איבר ש־p אחוז מהאיברים במערך קטנים או שווים לו. בתרגיל 3 הצענו אלגוריתם שעושה $Merge\ Sort$ ואז מחזיר את האיבר ה־$\lfloor n \cdot \frac{p}{100} \rfloor + 1$ במערך הממוין.
  - **קוויק סלקט:** מחזיר ב־$\Theta(n^2)$ את האיבר ה־k בגודלו מתוך מערך. הוא בוחר $pivot$, מפעיל $partition$, ואז ממשיך רקורסיבית לצד שבו האיבר ה־k בגודלו נמצא.
  - **בדיקה האם קיים איבר שחוזר על עצמו:** אם צריך לבדוק אם איבר חוזר על עצמו $\frac{n}{k}$ פעמים עם k כלשהו, אפשר לעשות את זה ב־$\Theta(n)$ ע"י להריץ קוויק סלקט עם $\frac{n}{k}$, ואז לרוץ על המערך ולספור כמה פעמים האיבר מופיע. אם $k = 2$ מספיק לעשות את זה פעם אחת, אחרת צריך לעשות את זה כמו אריה במדבר (כלומר להריץ שוב את קוויק סלקט על חצי המערך המתאים).
  - **איחוד מערכים ממוינים:** הראינו בתרגיל 3 שאפשר לעשות את זה ב־$\Theta(n \cdot klogk)$. האלגוריתם רץ בלולאה חיצונית עד $logk - 1$, ואז בלולאה פנימית עד $\frac{k}{2^i}$, ובתוכה עושה $Merge$ למערכים ה־A_{2j-1}, A_{2j}. כלומר, ממזגים את כל הזוגות הצמודים, אז אחרי כל איטרציה יהיו לנו חצי מהמערכים באיטרציה הקודמת.
  - **מציאת מינימום מקומי:** ב־$logn$, ע"י לבדוק האם האיבר באמצע הוא מינימום, ואם לא, להתקדם רקורסיבית באריה במדבר, כשבוחרים את חצי המערך שנתקדם אליו לפי האיבר שצמוד לאמצע וקטן ממנו.
  - **כשנתון שכל איבר רחוק עד k אינדקסים מהמקום הממוין שלו:** אפשר לעשות את זה ב־$nlogk$ ע"י ליצור ערמת מינימום ולהכניס אליה את k הערכים הראשונים במערך, ואז לעבור מהאיבר ה־$k + 1$ עד הסוף, ובכל פעם להוציא את המינימום מהערמה, לעשות איתו מה שצריך (לשים במערך / להדפיס), ולהכניס לערמה את שני הבנים שלו.
  - **מציאת סכום מינימלי / מקסימלי רצוף במערך:** עושים רקורסיה שמתחלקת לשלוש -
    - שתי קריאות ראשונות: קריאה רקורסיבית (לאותה פונקציה) עם חצי המערך הימני, וחצי המערך השמאלי.
    - קריאה שלישית: קריאה לפונקצית עזר רקורסיבית, שכל פעם מקבלת את הגבולות של המערך (ימין ושמאל) ואת האינדקס אותו רוצים לבדוק (מתחילים עם האמצעי), ואז מתקדמת ממנו ימינה עד שזה מפסיק להגדיל / להקטין (לפי מה שרוצים להשיג), ושמאלה עד שזה מפסיק להגדיל / להקטין, ובסוף לוקחת את המינימום בין הסכום הימני, הסכום השמאלי, והסכום של שניהם.

גיבוב

- בהגדרה של גיבוב אוניברסלי, לא לשכוח לציין שבוחרים את h בהתפלגות אחידה מתוך H.
- אם מדברים על 1 אוניברסלי, הדוגמא שצריכה לקפוץ לראש ישר היא של משפחה של פונקציות **קבועות <u>שונות</u>** (כי קטע חשוב של 1 אוניברסלי זה שאין שתי פונקציות שונות שמחזירות את אותו ערך עבור k כלשהו)
- הוכחנו שאם משפחה היא 2 אוניברסלית אז היא גם משפחה אוניברסלית (לשים לב בהוכחה שלא מספיק להוכיח עבור סתם $i_1 = i_2$, אלא צריך לכפול את זה במספר התאים בטבלה m), לכן אפשר להפריך באותו אופן שמפריכים משפחה אוניברסלית - באמצעות פונקציה קבועה.
- אם משפחה היא k אוניברסלית אז היא גם $k - 1$ אוניברסלית.
- כשאומרים "מצאו אלגוריתם הסתברותי" לחשוב ישר על טבלאות גיבוב.
- טיפ כדי למנוע טעויות חישוב, לכתוב בצורה מסודרת את כל הערכים שמשמשים לחישוב של פונקצית הגיבוב.

- אם צריך להגדיר טבלת גיבוב עם מפתחות שמכילים שני משתנים, אפשר להגדיר איך אנחנו בונים את המפתחות (נגיד $ak+b$).
- **שימושים יצירתיים בטבלאות גיבוב:**
  - **מציאת זוג איברים שהסכום שלהם הוא k:** נגבב את כל האיברים, ואז לכל איבר a נבדוק אם $k-a$ נמצא בטבלה. זה לוקח $O(n)$. אפשר להמיר לשלישיות שנסכמות ל־k ע"י גיבוב של הסכומים של הזוגות ואז חיפוש המשלים, וזה לוקח $O(n^2)$.

ערמות

- העלים בערמה נמצאים בחצי האחרון של האינדקסים.
- האיבר ה־k הכי גדול נמצא ב־k השכבות העליונות בערמה (אפשר להוכיח את זה ע"י להניח שהוא יותר עמוק מ־k ולעלות עד השורש).
- אפשר למצוא את k האיברים הכי גדולים בערמה ב־$klogk$ ע"י אלגוריתם שמאתחל ערמת מקסימום חדשה עם איבר המקסימום, ואז עושה k פעמים: $Extract\ max$ מהערמה החדשה והכנסה שלו למערך הסופי, ואז הכנסה לערמה החדשה של שני הבנים של אותו איבר בערמה המקורית.

עצים

- כמות קודקודים בעץ בינארי בגובה h:
  - מספר הקודקודים <u>בכל העץ</u> הוא <u>לכל היותר</u> $2^h - 1$.
  - מספר הקודקודים <u>הפנימיים</u> הוא <u>לכל היותר</u> $2^{h-1} - 1$.
  - מספר <u>העלים</u> הוא <u>לכל היותר</u> 2^h (באופן כללי, 2^k זו הנוסחא לכמות הקודקודים בשכבה מלאה k).
- כמות קודקודים בעץ AVL בגובה h:
  - מספר הקודקודים <u>בכל העץ</u> הוא <u>לפחות</u> $\left(\frac{3}{2}\right)^h$ (מוכיחים באינדוקציה).
  - מספר הקודקודים ברמה ה־h הוא מספר הקודקודים בשתי הרמות הקודמות לו + 1.
- **דוגמאות לאלגוריתמים על עצי AVL:**
  - **יצירת עץ ממערך ממוין:** אפשר ליצור עץ AVL ממערך <u>ממוין</u> ב־$O(n)$ ע"י ליצור אלגוריתם רקורסיבי שמקבל מערך, בוחר את האמצע שלו, ואז יוצר תת עץ שמאלי מהחצי הראשון של המערך, ותת עץ ימני מהחצי השני של המערך.
  - **איחוד שני עצי AVL:** אפשר לעשות את זה בזמן לינארי (כמות הקודקודים בשני העצים) ע"י ליצור מכל עץ (באמצעות מעבר על העץ ב־$O(n)$) מערך ממוין, לאחד ביניהם עם $Merge$, ואז ליצור עץ עם האלגוריתם בפסקה הקודמת.
- **דרכים לגשת לשאלות עצים:**
  - לחשוב בצורה של תתי בעיות רקורסיביות (ב־AVL כל תת עץ הוא גם AVL וזה יכול לחסוך אינדוקציה). כלומר כדי להוכיח דברים אפשר להשתמש בטריק של במקום "להוריד את השכבה האחרונה", פשוט "להוריד את השכבה הראשונה" ולהסתכל על שני תתי העצים של השורש שהם בגובה $h-1$.
  - לחשוב על לסגור מעגל / להסיר צלע / להסיר צלע ולהעביר לעץ אחר.
  - בהנתן קודקוד, אם אנחנו רוצים לחשב את כמות הקודקודים שקטנים ממנו, אפשר לעשות את זה עי הוספת שדה לכל הקודקודים בעץ שמכיל את מספר הקודקודים בתתי העצים שלהם (כולל את עצמם). אחכ נעלה מהקודקוד שמעניין אותנו עד לשורש, כשעבור כל קודקוד שנפגוש בדרך, אם הוא בן ימני, נקח את כל הקודקודים בתת העץ השמאלי שלו.
  - אם מבקשים מאיתנו למצוא איבר בעץ, צריך לחשוב אם הוא נמצא בכלל. אם הוא לא נמצא ואנחנו רוצים למצוא את העוקב שלו, אפשר להוסיף אותו לעץ, למצוא את העוקב, ואז להסיר אותו מהעץ.
  - כדי למצוא קודם משתמשים באותו אלגוריתם כמו למציאת עוקב, רק שמחליפים בכל מקום שכתוב "ימין" ב־"שמאל", ואת המקסימום במינימום.
  - אפשר להוכיח בקלות את למת הטווח - אם x שורש של תת עץ שהמינימום שלו m והמקסימום שלו M, אז כל קודקוד y נמצא בתת עץ של x אם"ם $y \in [m, M]$.
  - אפשר לחשב **ממוצע** של ערכי הקודקודים בטווח מסוים a, b בעץ ע"י להוסיף שדה של כמות ילדים כולל עצמך, ואז לעשות $a+b$ חלקי כמות הילדים של האב הקדמון המינימלי המשותף.

– אפשר למצוא את **האיבר ה־k בגודלו** בעץ באמצעות הוספת שדה כמות ילדים (כולל עצמך), ואז לרדת מהשורש לפי היחס בין השדה הזה ל־k (כלומר לקרוא רקורסיבית כל פעם לתת העץ הימני או השמאלי, ואם קוראים לשמאלי צריך לעדכן את k ככה שנחסיר ממנו את כמות הקודקודים בעץ הימני + 1).

– שדות שאפשר להוסיף / דברים שאפשר לחשב:

  * אב קדמון משותף מינימלי
  * כמות הילדים השמאליים / ימניים (כולל עצמך)
  * כמות הילדים באופן כללי (כולל עצמך)

גרפים

• **גישה כללית לשאלות:**

– בטענות שקשורות למעגלים שווה לחשוב על הסרה של צלע מהמעגל ובדיקת מצב הקשירות אחרי זה. באותו הקשר, אפשר להשתמש ב־BFS כדי לבדוק קשירות (אם בסיום הריצה, יש מרחק ∞, אז הם לא באותו רכיב קשירות).

– זה שגרף מכוון הוא חסר מעגלים, **לא** אומר שיש רק דרך אחת להגיע מהשורש לכל קודקוד (לדוגמא מעוין שכל החיצים הם כלפי מטה). זה נכון **רק בגרף לא מכוון.**

• **מסקנות חשובות:**

– אם יש מסלול בין שני קודקודים בגרף לא מכוון:

  * זה שיש הרצת DFS שבה ביקרנו בראשון לפני השני **לא** אומר שהשני הוא צאצא של הראשון בעץ DFS.
  * זה **לא אומר** שבכל ריצת DFS נכנס לראשון לפני שנמצא מהשני.
  * דוגמא נגדית לשניהם היא גרף בצורת $\wedge$ כשהחץ הימני הוא רק למטה והחץ השמאלי דו"צ.

– אפשר לזהות מעגלים באמצעות SCC.

• **אלגוריתמים שימושיים כלליים:**

– **לבדוק אם גרף הוא דו צדדי:** כלומר שאפשר לחלק את הקודקודים שלו לשתי קבוצות זרות כך שכל צלע בגרף בהכרח מחברת קודקוד מקבוצה א' וקודקוד מקבוצה ב' (ולא שני קודקודים מאותה קבוצה). נשתמש בזה שגרף הוא דו"צ אם הוא ־2צביע (כלומר שאפשר לצבוע אותו ככה שכל שני קודקודים סמוכים הם בצבע שונה), נוסיף שדה צבע לכל קודקודת נעשה שינוי ב־BFS ככה שאם הגענו לקודקוד שלא ביקרנו בו והוא לא צבוע, נצבע אותו בצבע ההפוך מהאבא שדרכו הגענו אליו. אם אנחנו מגיעים לקודקוד שצבוע באותו צבע שלנו, נחזיר שקר, אם סיימנו לעבור ולצבוע הכל כמו שצריך נחזיר אמת.

• **רכיבי קשירות:**

– מספר רכיבי הקשירות k משפיע על כמות ה־$union$ שנצטרך לעשות בקרוסקל־ מספר האיחודים יהיה $|V|-k$ (אם יש רכיב קשירות אחד, זה פשוט $|V|-1$, אחרת ־ על כל רכיב קשירות שאנחנו מוסיפים, זה חוסך לנו איחוד שהיינו צריכים לעשות).

– בגרף עם k רכיבי קשירות מספר הצלעות הוא בין המינימום $n-k$ (אם נתחיל מ־n רכיבי קשירות וכל פעם נאחד בין שני רכיבי קשירות ע"י הוספת צלע), למקסימום $\frac{(n-(k-1))(n-(k-1)-1)}{2}$. הנוסחא המגעילה הזו נוצרת ממצב בו יש $k-1$ רכיבי קשירות עם קודקוד בודד, ורכיב קשירות אחד מלא עם $n-(k-1)$ קודקודים (צריך להוכיח את זה), כאשר נזהה שבגרף מלא עם m קודקודים יש $\frac{m(m-1)}{2}$ צלעות ואז נציב $n-(k-1)$.

– אפשר לממש מציאת גרף SCCיותר מהר (ב־$|V|$ במקום ה־$|V|log|V|$ שקורה בגלל המיון) ע"י להחליף את המיון ב־$counting\ sort$ (אפשר לעשות את זה כי ערכי ה־$post$ הם בטווח 1 עד $2|V|$ אז ההנחה המקלה מתקיימת).

– גרף ה־SCC הוא DAG (מכוון, קשיר, ללא מעגלים). מוכיחים את זה ע"י להניח בשלילה שיש בו מעגל (אם יש רק רכיב קשירות אחד, אין מעגל. אחרת, אם יש מעגל זה גורם לזה שיש מסלול מכל קודקוד לכל קודקוד ולכן זה בעצם כן רכיב קשירות אחד).

• **גרף מכוון קשיר בלי מעגלים DAG:**

– **ב־DAG, מיון טופולוגי אלטרנטיבי:** אם יש לנו DAG (גרף קשיר <u>חסר מעגלים</u>) אפשר ליצור מיון טופולוגי מערכי pre אם מסדרים אותם בסדר עולה (זה שקול לערכי $post$ בסדר יורד). מוכיחים נכונות באמצעות משפט הסוגריים (בגלל ש־$post(u) < post(v)$ יש שני מקרים ־ או $pre(u) < post(u) < pre(v) < post(v)$ ואז ממשפט הסוגריים אף אחד מהם לא צאצא של השני ביער עצי העומק, או ש־$pre(v) < pre(u) < post(u) < post(v)$ ואז לפי משפט הסוגריים $\underline{u}$ הוא <u>כן</u> צאצא של v ביער עצי העומק, כלומר יש מסלול מ־v ל־u וגם צלע ישירה מ־u ל־v כלומר מעגל בסתירה ל־DAG).

- ב־DAG, **למצוא מסלול הכי ארוך כולל הקודמים במסלול (בגרף לא ממושקל):** מאתחלים את כל הקודקודים עם משקל <u>מינוס אינסוף,</u> עושים מיון טופולוגי, ואז עוברים עליו לפי הסדר ובודקים לכל שכן אם המשקל שלו קטן מהמשקל שלנו + 1 ומעדכנים את המשקל שלו אם כן.
- זה **לא נכון** שבגרף שהוא לא DAG יש בהכרח צלע דוצ.

- **מסלולים:**
  - **לבדוק אם צלע / קודקוד נמצאים בכל / בשני מסלולים קצרים שונים:** להריץ דייקסטרה או BFS (כתלות באם ממושקל או לא), להסיר את הקודקוד / צלע, להריץ דייקסטרה / BFS שוב ולראות אם הערך של קודקוד היעד קטן.
  - **לבדוק אם צלע נמצאת במסלול קצר <u>כלשהו</u>:** נעשה BFS או דייקסטרה (כתלות אם ממושקל) מהשורש עד לתחילת הצלע, ואז מסוף הצלע ליעד. אחרי זה נעשה את זה שוב רק ישירות מהשורש ליעד, ונראה אם זה אותו דבר + 1.
  - **מציאת המסלולים הכי <u>כבדים</u> מהשורש לכל נקודה:** להריץ DFS ולעשות מיון טופולוגי, להגדיר את המשקל של כל קודקוד בתור הסכום של משקלי המסלול אליו ש־DFS מצא. אחרי זה נעשה טרנספוז לגרף, נעבור על המיון הטופולוגי <u>מהסוף להתחלה</u> (כלומר מתחילים מהשורש), ולכל קודקוד נעדכן את המשקל שלו (אם צריך) באמצעות "$Relax$ הפוך".
  - **מסלול קצר משורש בגרף שהוא עץ <u>לא מכוון</u>:** יש רק דרך אחת להגיע ל

- **עצים פורשים מינימליים:**
  - כדי להוכיח עץ פורש, צריך להוכיח ־ קשיר, חסר מעגלים (כמות צלעות $|V|-1$ מה שגורר "פורש"), מינימלי.
  - אם כל משקלי הצלעות שונים וחיוביים, יש עפ"מ יחיד.
  - אם T עץ פורש מינימלי לפי w, אז לכל עץ אחר T', מתקיים $w(T) \leq w(T')$.
  - זה **לא נכון** שהמסלול הכי קצר בין שני קודקודים בהכרח מופיע בעפמ כלשהו (עפמ מתעדף את הסכום הכולל של הצלעות גם במחיר ויתור על מסלולים קצרים בין קודקודים). דוגמא נגדית היא מרובע עם שלוש צלעות במשקל 1, וצלע אחת במשקל 2 (שלא תופיע בשום עפמ, אבל היא המסלול הכי קצר בין הקודקודים שהיא מחברת).
  - אם יש <u>פונקציה w שמחזירה רק k משקלים שונים:</u> אפשר להשתמש בה כדי להחזיר עפמ ב־$O(|E|)$ ע"י להחליף את הערמה ב־$prim$ ב־k רשימות מקושרות דוצ שכל אחת מכילה את כל הקודקודים מאותו משקל, נאתחל אותן ריקות חוץ מהרשימה ה־k שתכיל את קודקוד המקור. נעבור בלולאת $while$ (כל עוד לפחות אחת מהרשימות לא ריקה), בתוך הלולאה נוציא את v האיבר הראשון מהרשימה הראשונה (לפי המספר) שלא ריקה, ולכל אחד מהשכנים שלו u, אם הם עדיין ברשימה כלשהי וגם ניתן לשפר את המשקל שלהם ע"י להחליף אותו במשקל (v,u), נשפר את המשקל שלהם ונכניס אותם לרשימה המתאימה.

- **משקלים:**
  - **השפעות של שינוי של פונקצית משקלים:**
    - <u>העלאה בריבוע:</u> תקין רק אם המספרים חיוביים
    - <u>כפל בסקלר:</u> תקין
    - <u>חיבור:</u> לא תקין (דוגמא נגדית ־ מסלול ישיר כבד יחסית לעומת מסלול ארוך של צלעות קלות). אפילו אם הסקלר שאנחנו מוסיפים זה המשקל של הצלע הכי קטנה בגרף, זה עדיין לא נכון (ובפרט, אי אפשר להשתמש בפונקצית משקל הזאת כדי לעשות דייקסטרה תקין על גרף עם מסלולים שליליים).
  - בשאלות על משקלים שווה לחשוב על **מקרי קיצון**:
    - משקל אפס
    - משקל שלילי (אם מותר בהנחות השאלה)
    - מסלול ארוך של משקלים קטנים (נגיד 1) לעומת מסלול ישיר שהוא הסכום שלהם

- **פלויד ורשל:**
  - אם רוצים להוסיף עוד קודקוד לגרף שיש לנו כבר את הטבלת פלויד ורשל שלו, צריך להוסיף עמודה ושורה לטבלה, ואז למלא אותן ע"י שתי הלולאות הפנימיות של פלויד ורשל, סה"כ n^2.
  - אם כבר יש לנו טבלה קיימת שנוצרה ע"י פלויד ורשל, ואנחנו מקצרים את אחד המסלולים הישירים בין הקודקודים u, v ורוצים לדעת האם צריך לעדכן את הטבלה או לא. כל מה שצריך לבדוק הוא להשוות בין המשקל החדש של הצלע (u,v) לבין הערך שקיים בתא ה־u,v בטבלה. צריך לעדכן את הטבלה רק אם המשקל החדש קטן ממה שיש בתא הזה כרגע.

איחוד קבוצות זרות

- אפשר לעדכן את union find ככה שאפשר יהיה להשתמש בו למציאת מקסימום בקבוצה מסוימת ב־$O(1)$, ע"י לעדכן את האופן שבו מאחדים שתי קבוצות לאיחוד ככה שהנציג החדש יהיה הנציג הגדול מבין שני הנציגים הקיימים.
- אפשר להשתמש ב־union find כדי לבדוק אם יש השמה למשתנים עם אילוץ של שוויונות ואי שוויונות, מאחדים הכל לפי השוויונות, ואז אם יש אי שוויון בודקים אם הם ברכיבים שונים.