שאלה 3

<u>סעיף א</u>

מבנה הנתונים

נשתמש במימוש של Union Find מההרצאה על ידי עצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים – וספציפית במימוש שקול על ידי מערכים. מימוש זה מבטיח לנו את הסיבוכיות של MergeParties ו-IsSameParty. בנוסף, נשמור משתנה בשם count המחזיק את מספר המפלגות הקיים בכל רגע נתון.

מימוש

$$\text{Init}(n)$$

כפי שראינו, אתחול של מבנה Union Find במימוש כנ"ל נעשה ב-$O(1)$ (אתחול מערך ריק בעל ערכים דיפולטיביים), וכמו כן אתחול של count למספר המפלגות ההתחלתי, כלומר n.

$$\text{MergeParties}(p_1, p_2)$$

תחילה נמצא את שורשי שתי הקבוצות ונאחד אותן, דבר שלפי האלגוריתם הנתון מההרצאה מתקבל ב-$O(\log^* n)$ משוערך יחד עם IsSameParty (שני find ו-merge יחיד). לאחר מכן נפחית את ה-count ב-1, דבר המתבצע ב-$O(1)$. בסך הכל סיבוכיות של $O(\log^* n)$ משוערך יחד עם IsSameParty.

$$\text{IsSameParty}(p_1, p_2)$$

שוב, לפי האלגוריתם מההרצאה נקבל סיבוכיות של $O(\log^* n)$ משוערך יחד עם MergeParties.

$$\text{AveragePartySize}()$$

נחזיר את החישוב הפשוט של הממוצע - $\frac{n}{\text{count}}$, מתבצע ב-$O(1)$.

סעיף ב

מבנה הנתונים

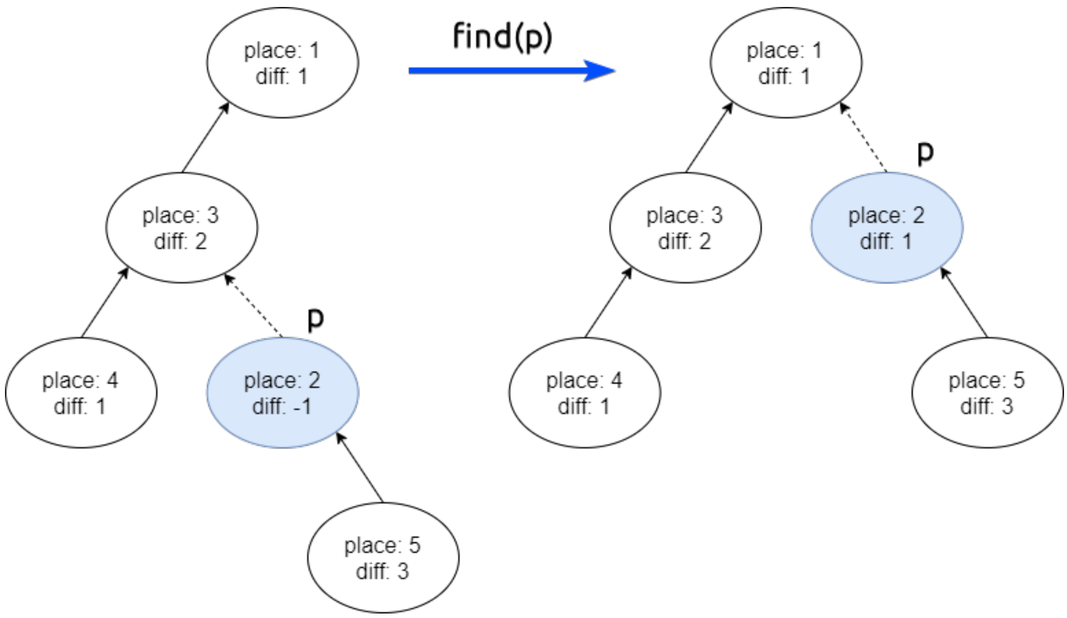
נשתמש מבנה נתונים זהה לזה שבסעיף הקודם, כאשר בנוסף לגודל הקבוצה (שנשמר כחלק ממימוש Union Find בו אנחנו משתמשים) נשמור גם את ההפרש בין מיקום המועמד בצומת הנוכחית למיקום הצומת עליו הוא מצביע – $\mathrm{diff}(p)$, כאשר בשורשי העץ פשוט נשמור את מיקום המועמד במפלגה.

מימוש

על מנת לשמור על המימוש שלנו תקין, נצטרך לעדכן את פעולות ה-find וה-union הנתונות לנו מהיישום המקורי של המבנה:

* find: כאשר אנחנו מוצאים צומת p, נסכום את ההפרשים של כל הצמתים במסלול (מלבד השורש) ונוסיף אותם ל-$\mathrm{diff}(p)$. באופן זה, מובטח לנו שההפרש של p מהצומת עליה הוא מצביע – השורש, לאחר העדכון – יישמר (מצורפת דיאגרמה המדגימה את הפעולה).
* union: לפני איחוד בין שתי קבוצות p_1 ו-p_2 נוסיף ל-$\mathrm{diff}(p_2)$ את $\mathrm{diff}(p_1)$. מכיוון שמדובר בהפרשים, מובטח לנו שכל הצמתים ב-p_2 "יזוזו" למקומם לאחר האיחוד באופן טבעי.

כמו כן ב-Init נאתחל גם את $\mathrm{diff}(p)$ ל-1 בתור ערך ברירת מחדל.



$$\mathrm{PlaceInParty}(p)$$

נבצע פעולה של find כדי לאתר את השורש. מכיוון שאנחנו משתמשים ביישום הכולל כיווץ מסלולים, מובטח לנו ש-p מצביע לשורש המפלגה, או שהוא השורש בעצמו. לפיכך, ניתן לחשב את מקומו במפלגה באופן מיידי, אם הוא השורש נחזיר את $\mathrm{diff}(p)$, אחרת אם הוא מצביע לשורש r נחזיר את $\mathrm{diff}(r)+\mathrm{diff}(p)$. אנחנו משתמשים בפעולת find בודדת, ולאחריה בחישוב יחיד, לפיכך הסיבוכיות שלנו היא $O(\log^* n)$ משוערך יחד עם IsSameParty ו-MergeParties.

שאלה 4

נעזר בתשובתנו בכך שנוכל להשתמש בכל זוג ערכים שיניב סיומת תקינה בשחזור העץ.

האלגוריתם שנציע ייצטרך לשחזר את העץ המקורי מאורכי הסיומות בכל אחד מהמסלולים בעץ. סימונים:

$l_i \coloneqq$ מספר התווים בקשת ה-i

$p_i \coloneqq$ המצביע לתחילת המחרוזת ה-i בעץ המקורי

$e_i \coloneqq$ המצביע לסיום המחרוזת ה-i בעץ המקורי

כלומר אם בעץ ה"מקולקל" יופיע l_i, בעץ המשוחזר יופיע הצמד (p_i, e_i). כעת נשים לב לתכונות הבאות:

* עבור קשת המחברת עלה, בהכרח מתקיים $e_i = |s|$, שכן סוף הסיומת תמיד יהיה סוף המחרוזת המקורית.
* עבור קשת המחברת צמתים שאינן עלים מתקיים $e_i = p_k - 1$ כאשר p_k הוא אחד המצביעים מבין שתי הקשתות המחברות את הצומת הרחוקה מהשורש.
* מתקיים: $l_i = e_i - p_i + 1 \iff p_i = e_i - l_i + 1$.

האלגוריתם שלנו יכלול סיור postorder בעץ: כשנגיע לעלה, נחזיר את הצמד $(|s| - l_i + 1, |s|)$, ובכל מקרה אחר נחזיר את הצמד $(p_k - l_i, p_k - 1)$ כאשר p_k הוא המצביע ההתחלתי של הבן השמאלי (בחירה שרירותית).

כפי שראינו בהרצאה גודל העץ הוא $O(|s|)$, ולכן הסיור מתרחש בסיבוכיות $O(|s|)$ כנדרש.