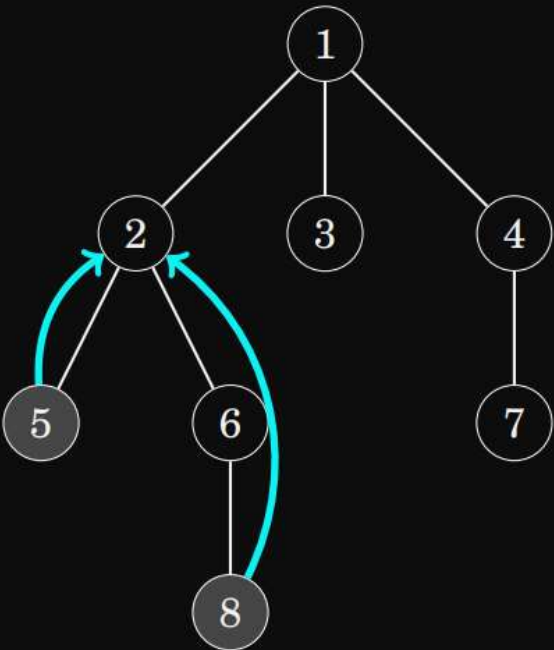


נושא מסתורי וחדש 3

וואו, נושא חדש תמיד מרים לי את מצב הרוח

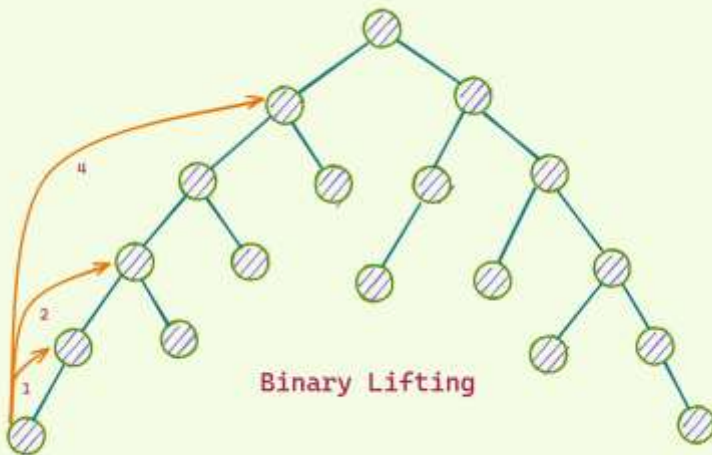
LCA

- במבני נתונים למדתם lca, שזאת טכניקה יעילה בשאלות של עצים.
- אנחנו רוצים לעשות את זה בסיבוכיות זמן $\log(n)$.
- איך נוכל בהינתן שני צמתים, לדעת את ה-LCA ב- $O(\log(n))$?



Binary lifting

- נשתמש בטכניקה בשם binary lifting.
- הרעיון: כל אחד יזכור את כל ההורים מדרגות שהן חזקות של 2 שלו.
- כלומר אני זוכר את אבא שלי (דרגה אחת), את אבא שלו (דרגה 2), וכל דרגה שהיא חזקה של 2.



בנייה

- הדרך לעשות את הבנייה של binary lifting לוקחת $O(n \log n)$ זמן, וברור שאי אפשר פחות כי צריך לזכור $O(n \log n)$ מקום. (לכל אחד מקסימום של $\log(n)$ אבות).
- איך נבנה את זה בזמן $O(n \log n)$?

מימוש בנייה

- כאשר $\log n$, $\max n$ מוגדרים מראש, ו $\text{parent}[i][j]$ מאותחל ל-2-.

```
void binary_lifting(){
    for(int i = 1; i < logn; i++){
        for(int j = 1; j < maxn; j++){
            if(parents[j][i-1] != -2)
                parents[j][i] = parents[parents[j][i-1]][i-1];
        }
    }
}
```

בנייה- פתרון

- במקום לעבור צומת צומת ולגלות את האבות ממרחק שהוא חזקה של 2, אנחנו נגלה עבור כל הצמתים את האבות מגובה 1, ואז עבור כולם את גובה 2, ואז עבור כולם את גובה 4...
- רעיון- אם יש לנו את האב מגובה 2^{i-1} , נוכל לגלות את האב מגובה 2^i בזמן $O(1)$, בכך שנלך לאב מגובה 2^{i-1} , ואז נלך לאב מגובה 2^{i-1} שלו, וקיבלנו את האב מגובה 2^i שלנו.
- נעשה את זה עבור כל גובה שהוא חזקה של 2 וכל צומת, נגלה כל אחד מהם בזמן $O(n)$ כל כל מציאת אב היא $O(1)$, אז סך הכול $O(n \log(n))$ כי יש $\log n$ גבהים.

שימוש ישיר של binary lifting

- בעצם בעזרת זה אנחנו יכולים למצוא עבור v מסוים, את האב k שלו, בזמן $\log n$. (נקפוץ את החזקה של 2 הכי גדולה שקטנה מ- k , ונוריד את הערך שלה מ- k , עד ש- $k=0$. סך הכול זה מקסימום $\log n$ קפיצות).
- בנוסף לכך נשמור לכל צומת את ה- $depth$ שלו (מרחק מהשורש, קל למצוא עם bfs).
- איך נוכל להשתמש בשתי הפעולות האלה כדי לעשות LCA בזמן $O(\log(n))$?

פתרון- LCA

- נסתכל על הצומת מהdepth הנמוך יותר. עכשיו נעלה ממנה לאב ה-k כך ששני הצמתים שאנחנו מסתכלים עליהם הם באותו גובה.
- עכשיו עבורם כל פעם נבחר ללכת לחזקה של 2 הכי גדולה שאפשר כך שהם לא מגיעים לאותו צומת.
- זה כמו לבחור את האב ה-k, לאחר $O(\log(n))$ תזוזות משניהם נגיע לצומת שהוא אחד לפני ה-LCA שלהם.

קוד עבור LCA

```
int lca(int a, int b) {
    if (depth[a] < depth[b]) {
        swap(a, b);
    }
    int depth_difference = depth[a] - depth[b];
    for (int j = 19; j >= 0; --j) {
        if ((1 << j) & depth_difference) {
            a = jmp[a][j];
        }
    }
    if (a == b) {
        return a;
    } else {
        for (int j = 19; j >= 0; --j) {
            if (jmp[a][j] != jmp[b][j]) {
                a = jmp[a][j];
                b = jmp[b][j];
            }
        }
        return jmp[a][0];
    }
}
```

סיכום

- Binary lifting זאת טכניקה שנותנת לנו לגלות את ה-LCA, ואת האב ה- k של כל node בזמן $O(\log(n))$, עם זמן בנייה של $O(n \log(n))$.