

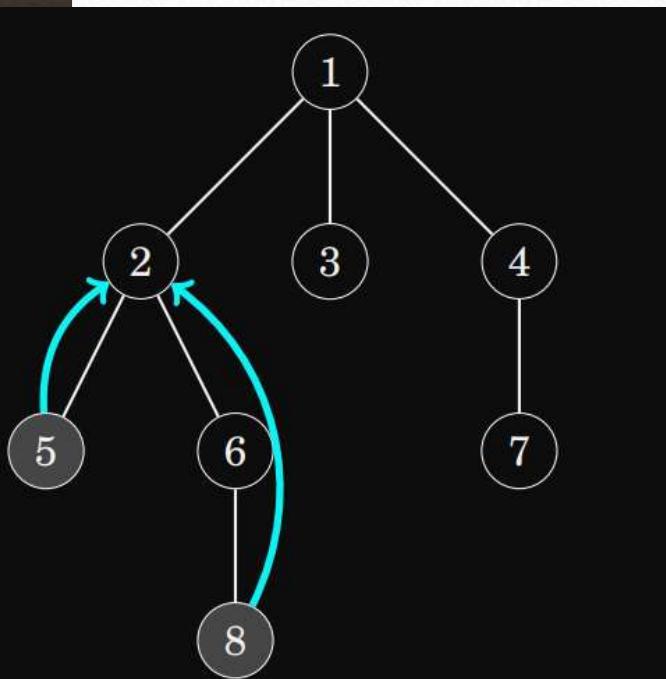
# **נושא מוסטרי וחדש 3**

---

**וואו, נושא חדש תמיד מרים לי את מצב הרוח**

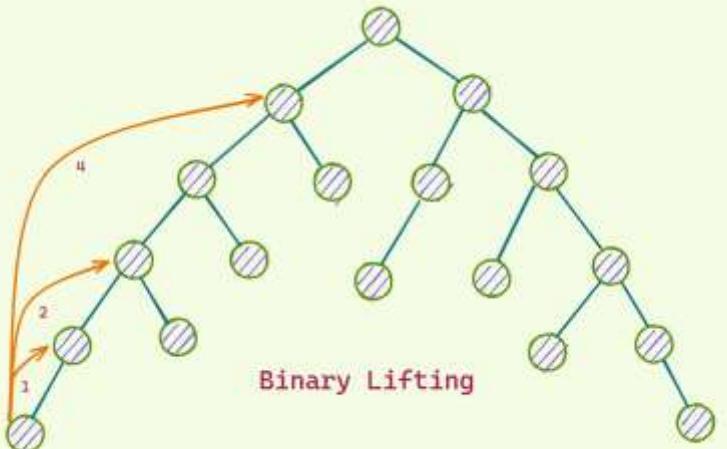
# LCA

- במבנה 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

- נשותמש בטכנית בשם binary lifting.
- הרעיון: כל אחד יזכור את כל ההורים מדרגות שהן חזקות של 2 שלו.
- ככלומר אני זוכר את אבא שלי (דרגה אחת), את אבא שלו (דרגה 2), וככל דרגה שהוא חזקה של 2.



# בנייה

- 
- הדרך לעשות את הבניה 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 ( $n \log n$  אבות).
  - איך נבנה את זה בזמן ( $O(n \log(n))$ ?

# מיושן בנייה

- כאשר  $\text{maxn}$ ,  $\text{logn}$  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

- במקומות לעבור צומת צומת ולגלוות את האבות מפרק שהוא חזקה של 2, אנחנו נגלה עברו כל הצלמים את האבות מגובה 1, וזו עברו כולם את גובה 2, וזה עברו כולם את גובה 4...>.
- רעיון- אם יש לנו את האב מגובה  $1^{-i}2$ , נוכל לגלוות את האב מגובה  $i2$  בזמן  $(1)O$ , בכך שנלך לאב מגובה  $1^{-i}2$ , וזו נלך לאב מגובה  $1^{-i}2$  שלו, וקיבלנו את האב מגובה  $i2$  שלנו.
- נעשה את זה עברו כל גובה שהוא חזקה של 2 וכל צומת, נגלה כל אחד מהם בזמן  $(n)O$  כל כל מציאות אב היא  $(1)O$ , אז סך הכל  $((n\log(n))O)$  כי יש  $\log(n)$  גבהים.

# שימוש ישיר של binary lifting

- בעצם בעזרת זה אנחנו יכולים למצוא מעורר  $\lceil \log_2 k \rceil$  מסויים, את האב ה- $k$  שלו, בזמן  $\text{logn}$ . (נקפוץ את חזקה של 2 הכי גדולה שקטנה מ- $k$ , ונוריד את הערך שלה מ- $k$ , עד ש- $0 = k$ . סך הכל זה מקסימום  $\text{logn}$  קפיצות).
- בנוסף לכך נשמר לכל צומת את  $\text{depth}$  שלו (מרחק מהשורש, קל למצוא עם bfs).
- איך נוכל להשתמש בשתי הפעולות האלה כדי לעשות LCA בזמן  $O(\log(n))$ ?

# פתרון - LCA

- נסתכל על הツומת מהdepth הנמוך יותר. עכשו נעלם ממנה לאב ה-k כך שני הツומתים שאנו חנו מסתכלים עליהם הם באותו גובה.
- עכשו עבורם כל פעם נבחר ללקת לחזקה של 2 הכיל גדולה שאפשר כך שהם לא מגיעים באותו צומת.
- זה כמו לבחור את האב ה-k, לאחר  $O(\log(n))$  תזוזות משליהם 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];
}
```

# ΟΙΚום

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