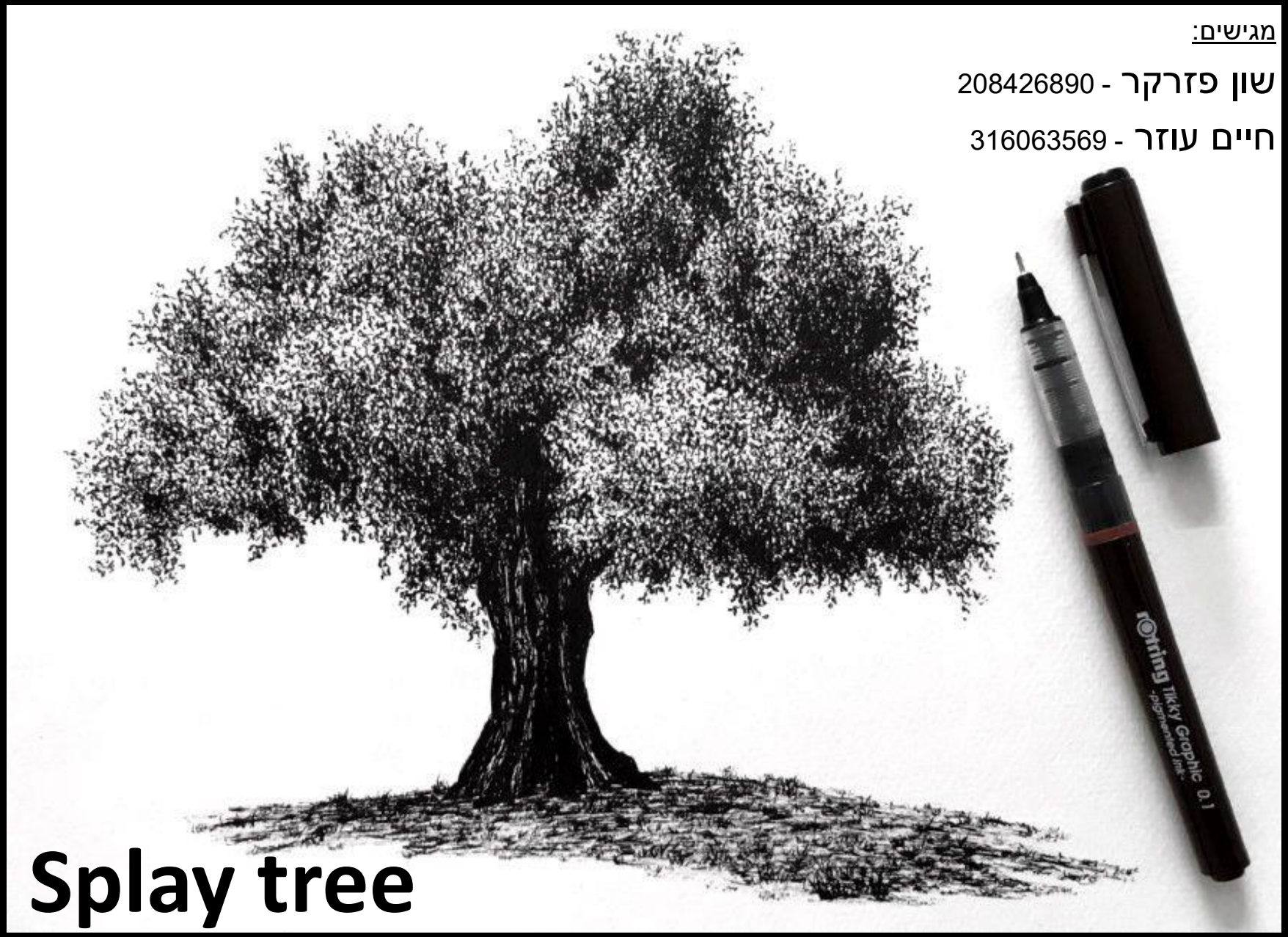
מעבדה מתקדמת לתכנות - ולדי בודניצקי

מגישים:

שון פזרקר - 208426890

חיים עוזר - 316063569



Splay tree

Topic Theory

עץ ספליי הוא עץ חיפוש בינארי המאזן את עצמו לפי הרשומה הכי שכיחה.

הוא מבצע פעולות בסיסיות כמו הכנסה, חיפוש והסרה בזמן מופחת O(log_n).

הרעיון הבסיסי מאחורי עצי ספליי הוא להביא את האלמנט שאליו ניגשים לשורש העץ, כך שגישה עתידית לאלמנט זה יכולה להיות מהירה יותר.

מטרה זו מושגת על ידי ביצוע סיבובי עצים בדרך מהאלמנט לשורש.

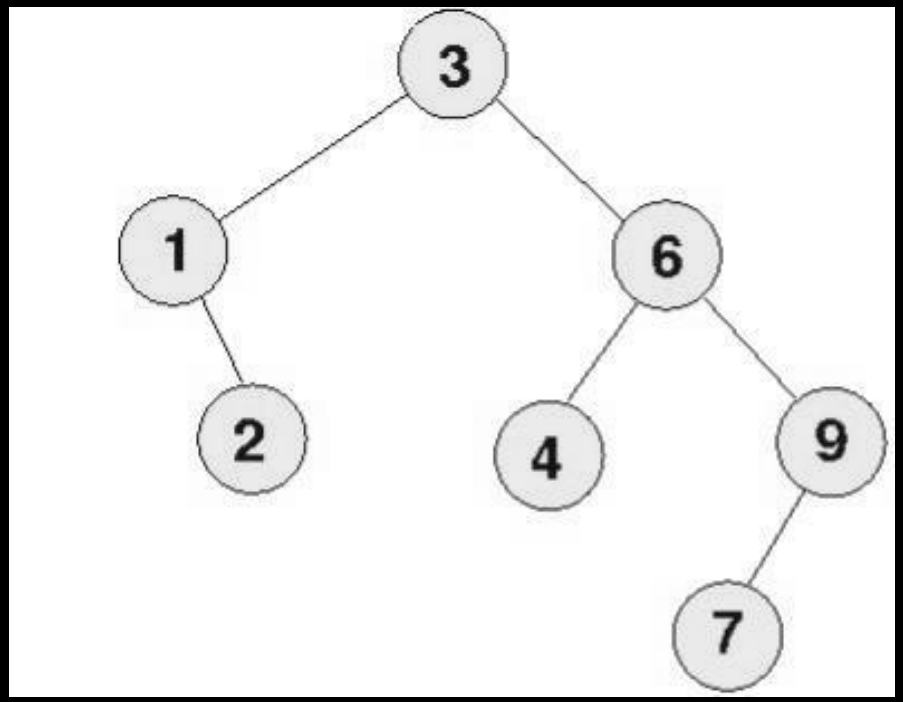
מאפייניו המרכזיים של העץ (splay tree) :

- זהו עץ חיפוש בינארי, כך שהוא עוקב אחר אותם כללים כמו עצי חיפוש בינאריים אחרים מבחינת האופן שבו אלמנטים מאורגנים וכיצד ניתן לחפש אותם.
- הוא באיזון עצמי, כלומר העץ מתאים את המבנה שלו כדי להבטיח שגובה העץ יהיה תמיד לוגריתמי ביחס למספר האלמנטים בעץ. זה מבטיח שהפעולות הבסיסיות יהיו יעילות.
- הוא מביא אלמנטים שניגשו לאחרונה לשורש העץ, כך שניתן יהיה לגשת אליהם שוב במהירות בעתיד.
- הוא יעיל בטיפול בדפוסי גישה עוקבים ואקראיים כאחד, מכיוון שהוא יכול להסתגל לדפוסי הגישה של האלמנטים המאוחסנים.

כיצד בנוי עץ Splay

עץ ספליי הוא עץ חיפוש בינארי המאזן את עצמו, מה שאומר שיש לו את המאפיינים הבאים:

1. זהו עץ בינארי, כלומר לכל צומת יש לכל היותר שני ילדים.
2. הוא עוקב אחר מאפיין עץ החיפוש הבינארי, הקובע שהערך של הילד השמאלי של הצומת קטן מערכו של הצומת, והערך של הילד הימני של הצומת גדול מערכו של הצומת.

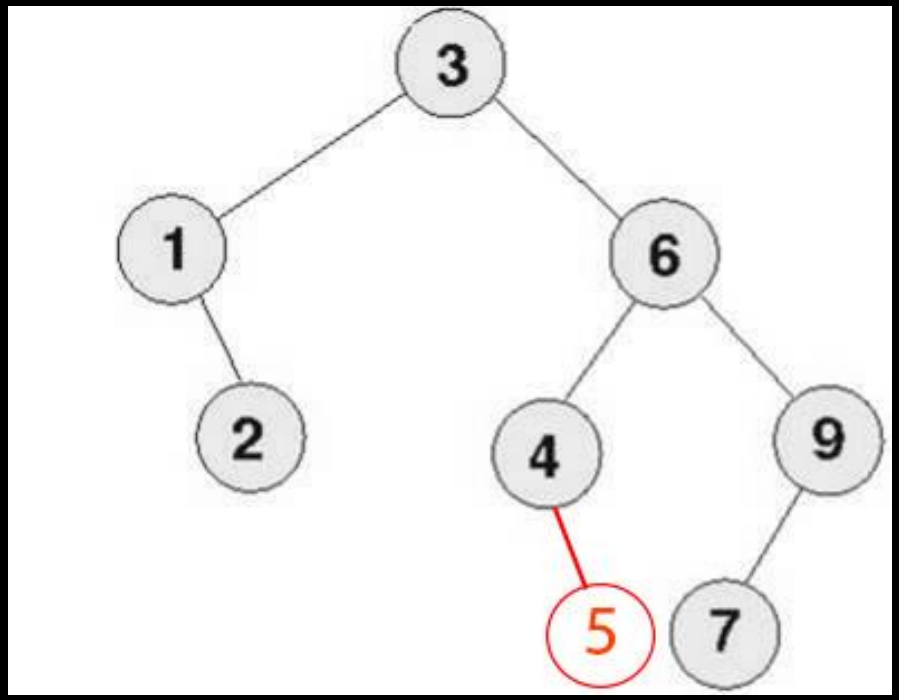


כיצד להוסיף אלמנט לתוך עץ Splay

על מנת להוסיף אלמנט לעץ יש לבצע את השלבים הבאים:

1. משורש העץ נבצע חיפוש בינארי סטנדרטי כדי למצוא את המיקום הנכון עבור האלמנט החדש.
2. לאחר שנמצא את המיקום המתאים, נכניס את האלמנט החדש לעץ כצומת עלים.
3. נבצע פעולות "פיזור" על השביל מצומת העלה החדש לשורש העץ.

פעולות ה"פיזור" הללו הן סיבובי עצים שמקרבים את האלמנט החדש לשורש העץ.

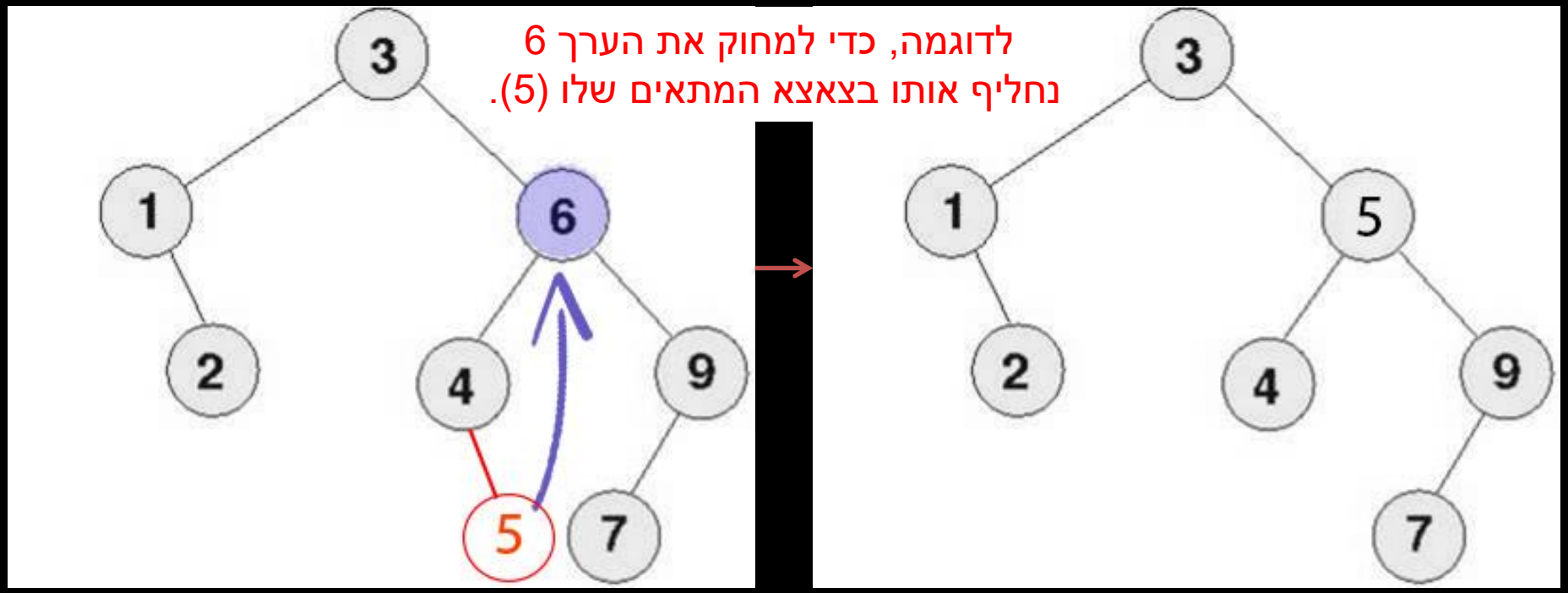


לדוגמה, כדי להכניס את הערך 5 לעץ, נבצע חיפוש בינארי ונגלה שיש להכניס אותו בתור הבן הנכון של צומת 4.

כיצד להוריד אלמנט מתוך עץ Splay

על מנת למחוק אלמנט מעץ ספליי יש לבצע את השלבים הבאים:

1. משורש העץ נבצע חיפוש בינארי רגיל כדי למצוא את האלמנט שנרצה למחוק.
2. לאחר שהאלמנט נמצא, נמחק אותו מהעץ באמצעות כללי מחיקת עץ החיפוש הבינארי הסטנדרטי.
3. נבצע פעולות חלוקה על הנתיב מהשורש החדש של העץ (שהיה הצומת שהחליף את הצומת שנמחק) לשורש הישן של העץ.

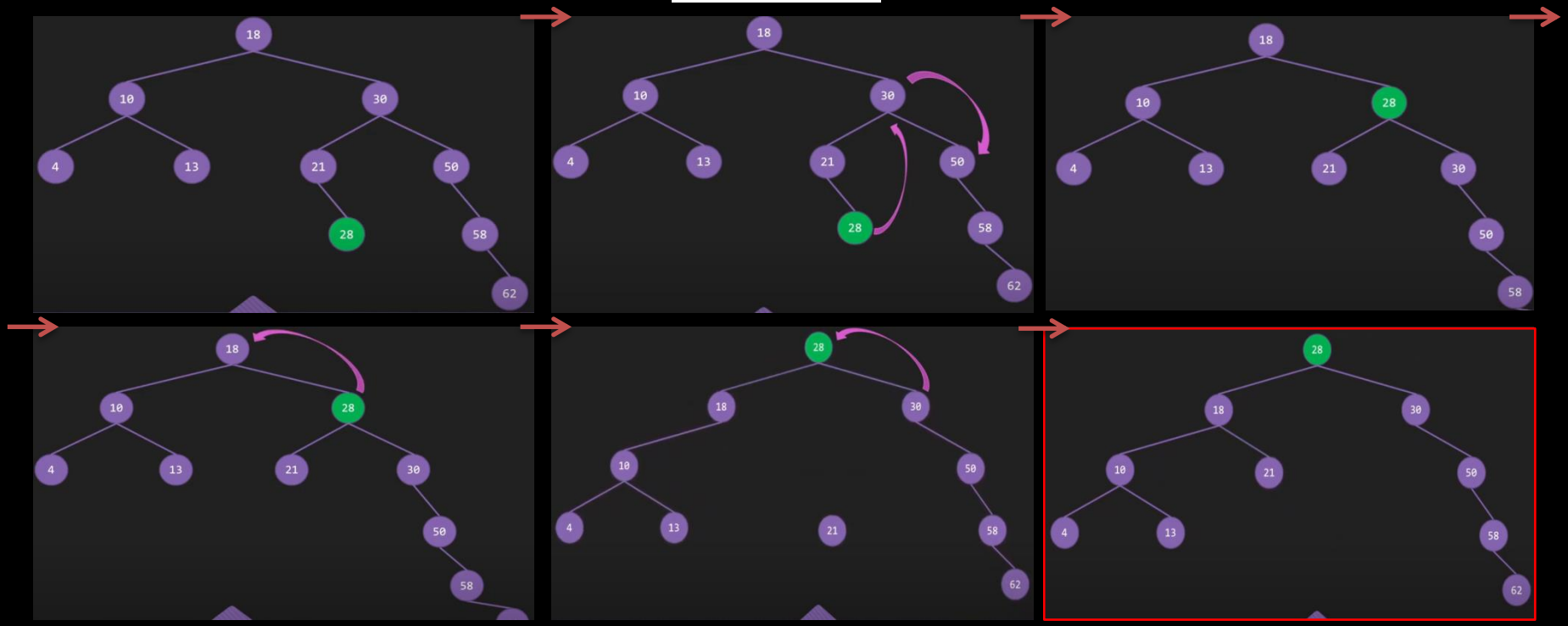


פעולות פיזור אלמנטים בעץ Splay

פעולות ההפצה הנפוצות בהן נעשה שימוש בעץ :

- פעולת פיזור נפוצה אחת היא סיבוב "זיג-זיג", הכולל סיבוב של צומת והאב שלו באותו כיוון.
- פעולת פיזור נפוצה נוספת היא סיבוב "זיג-זג", הכולל סיבוב של צומת בכיוון אחד והאב שלו בכיוון ההפוך.

<u>לדוגמא זיג זג:</u>



סיבובים אלה משמשים כדי לקרב את האלמנט הנגיש לשורש העץ.

מורכבות הזמן של פעולות נפוצות

מורכבות הזמן של פעולות נפוצות בעץ ספליי תלויה בגובה העץ, שהוא לוגריתמי ביחס למספר האלמנטים בעץ.

המשמעות היא שמורכבות הזמן של פעולות אלו היא בדרך כלל O(log_n), כאשר n הוא מספר האלמנטים בעץ.

להלן סיכום של מורכבות הזמן של פעולות נפוצות בעץ ספליי:

1. הוספה: O(log_n)
2. חיפוש: O(log_n)
3. הסרה: O(log_n)

יש לשים לב כי מורכבות זמן אלו מופחתת, כלומר לוקחות בחשבון את עלות פעולות החלוקה המתבצעות במהלך הפעולה.

במקרה הגרוע, עץ ספליי יכול להתדרדר למורכבות זמן ליניארית, אך לא סביר שזה יקרה בפועל כל עוד העץ מאוזן היטב.

השוואה בין BST, AVL ו-Splay Tree

BST ,AVL, Splay הם כולם עצי חיפוש בינאריים המאזנים את עצמם, מה שאומר שהם מתוכננים לשמור על מאפיין עץ החיפוש הבינארי *(כלומר, הערך של הצאצא השמאלי של הצומת קטן מהערך של הצומת, והערך של הילד הימני של הצומת גדול מהערך של הצומת)* תוך שמירה על גובה העץ לוגריתמי ביחס למספר האלמנטים בעץ.

אפיון זה מאפשר להם לשמור על מורכבות זמן טובה עבור פעולות בסיסיות כגון הכנסה, חיפוש והסרה.

<u>להלן כמה הבדלים עיקריים בין שלושת סוגי העצים הללו:</u>

1. <u>BSTs</u> - הוא הסוג הפשוט והבסיסי ביותר של עץ חיפוש בינארי אשר מאזן את עצמו.

הם שומרים על מאפיין עץ החיפוש הבינארי, אך הם אינם מכוונים באופן אוטומטי את המבנה שלהם כדי לאזן את העץ.

כתוצאה מכך, הם עלולים להיות לא מאוזנים אם אלמנטים מוכנסים בסדר שאינו מאוזן.

2. <u>AVL</u> - הוא סוג של עץ חיפוש בינארי באיזון עצמי המשתמש בסיבובים כדי לשמור על מצב האיזון שבו הגובה של תתי העצים השמאלי והימני של כל צומת שונה ב-1 לכל היותר.

זה מבטיח שגובה העץ הוא תמיד לוגריתמי ביחס למספר האלמנטים בעץ.

עצי AVL מאוזנים בקפדנות יותר מאשר עצי BSTs אבל הם יכולים להיות איטיים יותר בפועל, כי הם משתמשים ביותר סיבובים.

3. <u>Splay</u> – הוא סוג של עץ חיפוש בינארי באיזון עצמי המשתמש בסיבובים כדי לקרב אלמנטים שכיחים לשורש העץ.

פעולה זו הופכת את הגישה לאלמנטים אלה מהירה יותר ולכן עצי ספליי הם בדרך כלל מהירים יותר מעצי AVL ועצי red-black מכיוון שהם משתמשים בפחות סיבובים ויש להם גורם קבוע קטן יותר.

עם זאת, הם אינם מאוזנים בקפדנות כמו העצים האחרים הללו, מה שיכול להפוך את הביצועים שלהם לפחות צפויים במקרים מסוימים.

לסיכום, BSTs הם הסוג הפשוט והבסיסי ביותר של עץ חיפוש בינארי המאזן עצמי, בעוד שעצי AVL ועצי ספליי מציעים ביצועים ואיזון משופרים במחיר של מורכבות זמן גבוה יותר.

הבחירה באיזה סוג עץ להשתמש תלויה בצרכים הספציפיים של היישום הנדרש.

דוגמה ליישומים שבהם Splay יועדף על AVL

להלן כמה דוגמאות ליישומים שבהם עץ Splay עשוי להיות מועדף על פני עץ AVL:

1. יישומים הדורשים גישה מהירה לאלמנטים שניגשו לאחרונה:

עצי Splay נועדו לקרב אלמנטים שניגשו לאחרונה לשורש העץ, מה שמאפשר גישה מהירה יותר לאלמנטים אלה בעתיד.

זה יכול להיות שימושי ביישומים שבהם יש רמה גבוהה של מקומיות בדפוסי הגישה של האלמנטים המאוחסנים.

2. יישומים עם דפוסי גישה בלתי צפויים:

עצי ספליי יכולים להתאים את עצמם לדפוסי הגישה של האלמנטים המאוחסנים, מה שאומר שהם יעילים בטיפול בדפוסי גישה עוקבים ואקראיים כאחד.

זה יכול להיות שימושי ביישומים שבהם קשה לחזות את דפוסי הגישה או שהם משתנים כל הזמן.

3. יישומים עם זיכרון או משאבי חישוב מוגבלים:

עצי ספליי הם בדרך כלל מהירים יותר בפועל מעצי AVL מכיוון שהם משתמשים בפחות סיבובים.

זה יכול להפוך אותם לבחירה טובה ביישומים שבהם הזיכרון או משאבי החישוב מוגבלים.

דוגמה ליישומים שבהם AVL יועדף על Splay

דוגמאות ליישומים שבהם עץ AVL עשוי להיות מועדף על פני עץ Splay :

1. יישומים הדורשים איזון קפדני:

עצי AVL מאוזנים בקפדנות רבה יותר מעצי ספליי, מה שאומר שסביר יותר שהם לא יהיו מאוזנים במקרים בהם האלמנטים המוכנסים אינם מאוזנים.

זה יכול להיות שימושי ביישומים שבהם האלמנטים המוכנסים אינם מפוזרים היטב או שבהם דפוסי הגישה אינם ידועים.

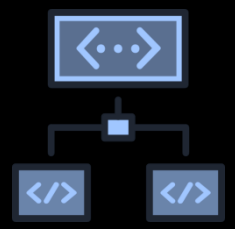
2. יישומים עם דפוסי גישה ניתנים לחיזוי:

עצי AVL הם בדרך כלל צפויים יותר מבחינת הביצועים שלהם מאשר עצי ספליי, מה שיכול להפוך אותם לבחירה טובה ביישומים שבהם דפוסי הגישה ידועים ועקביים.

3. יישומים הדורשים מידה גבוהה של הוגנות:

מכיוון שעצי ספליי נועדו לקרב אלמנטים שניגשים אליהם לאחרונה לשורש העץ, ייתכן שהם לא יהיו הוגנים כמו עצי חיפוש בינאריים אחרים המאזנים את עצמם מבחינת כמות העבודה נדרש לגשת לאלמנטים שונים.

זה יכול להוות בעיה ביישומים שבהם חשוב להבטיח שכל המרכיבים יטופלו באופן שווה.



היתרונות והשימושים של עצי Splay בניתוב רשת

השימוש בעצי Splay בניתוב רשת הוא יישום טבלאות ניתוב המאפשרות ניתוב כתובות דרך הרשת על סמך כתובת היעד שלהן.
כלומר, אלמנטי השורש בעץ Splay הם הכתובת הנוכחית והעץ משמש כדי לחפש במהירות את הקפיצה הבאה עבור כתובת היעד הרצויה.

<u>היתרונות של שימוש בעצי ספליי לצורך ניתוב רשתי כוללים:</u>

- <u>חיפוש מהיר:</u>
  עצי ספליי נועדו לקרב אלמנטים שניגשים אליהם הרבה לשורש העץ, מה שהופך את העץ למהיר יותר לגשת לאלמנטים אלו בעתיד.
  תכונה זו יכולה להיות שימושית בניתוב רשת, שם חשוב לחפש במהירות את הקפיצה הבאה עבור כתובת יעד רצויה.
- <u>יכולת הסתגלות:</u>
  עצי ספליי יכולים להסתגל לדפוסי הגישה של האלמנטים המאוחסנים, מה שאומר שהם יעילים בטיפול בדפוסי גישה עוקבים ואקראיים כאחד.
  תכונה זו יכולה להיות שימושית בניתוב רשת, שם דפוסי הגישה של כתובות יעד עשויים להיות קשים לניבוי.
- <u>פשטות:</u>
  עצי Splay הם פשוטים יחסית ליישום ותחזוקה, לכן העץ שימושי בניתוב רשת בו נדרש לעדכן את טבלאות הניתוב בתדירות גבוהה וביעילות.
- <u>Scalability:</u> *"מדרגיות - היכולת של מערכת להתמודד בצורה טובה עם כמות הולכת וגדלה של עבודה"*
  לעצי ספליי יש מורכבות זמן לוגריתמית ביחס למספר האלמנטים בעץ, מה שאומר שהם יכולים לשנות את קנה המידה כדי לטפל במספרים גדולים של כתובות יעד מבלי להיות איטיים מדי.

לסיכום, עצי ספליי יכולים להיות מבנה נתונים שימושי ליישום טבלאות ניתוב יעילות ביישומי ניתוב רשת.
הם מציעים איזון של יעילות, הסתגלות ופשטות שהופך אותם למתאימים היטב למשימה זו.

מהו עץ קישור/חיתוך LCT הקשר שלו לעץ Splay

<u>Link/Cut Tree (LCT):</u>

הוא מבנה נתונים המבוסס על עצי Splay ומשמש ליישום יעיל של מבני נתוני עצים דינמיים.

כמו עצים מפוזרים, LCT משתמשים בסיבובי עצים כדי לקרב אלמנטים שכיחים לשורש העץ, מה שהופך אותם ליעילים בטיפול בדפוסי גישה עוקבים ואקראיים כאחד.

ההבדל העיקרי בין LCT ועצי splay הוא ש- LCT מתוכננים לתמוך בפעולות עצים דינמיות, כגון קישור וחיתוך, המאפשרות לקשר צמתים זה לזה ולהפריד אותם בזמן לוגריתמי.

פעולות אלו שימושיות במגוון יישומים, כגון אלגוריתמי גרפים, קישוריות דינמית ומיון טופולוגי.

באופן כללי, LCT משמשים להטמעת מבני נתוני עצים דינמיים, בעוד עצי splay משמשים ליישום עצי חיפוש בינאריים באיזון עצמי.

עם זאת, LCT מבוססים על עצי splay ומשתמשים לרוב באותן הטכניקות כמו סיבובי עצים, כדי לשמור על המבנה שלהם.


THANK YOU!