|  |  |  |
| --- | --- | --- |
| **סוג מבנה** | **פעולה** | **זמן ריצה** |
| **רשימה מקושרת** | הכנסה (לסוף עם זנב / להתחלה) – insert() | O(1) |
| הכנסה (לסוף ללא זנב) – insert() | O(n) |
| מחיקה / חיפוש – delete()/ find() | O(n) |
| **רשימה מקושרת ממוינת** | הכנסה - insert() | O(n) |
| מחיקה ערך מינמלי deleteMin() | O(1) |
| מחיקה / חיפוש - delete()/ find() | O(n) |
| **עץ בינארי**  עץ שלכל צומת 0-2 בנים | הכנסה - insert() | O(h) במקרה הטוב o(log(n)) במקרה הגרוע o(n) |
| מחיקה / חיפוש delete()/ find() | O(n) |
| **עץ חיפוש בינארי**  **עץ בינרי** שכל צומת גדולה מכל תת העץ השמאלי וקטנה מתת העץ הימני. | הכנסה - insert() | O(h) במקרה הגרוע o(n) |
| מחיקה / חיפוש - delete()/ find() | O(h) במקרה הגרוע o(n) |
| **עץ AVL**  **עץ חיפוש בינרי** מאוזן – עבור כל צומת בעץ הפרש הגבהים בין תת העץ הימני לשמאלי הוא לא יותר מ-|1| | הכנסה - insert() | O(Log(n)) |
| מחיקה / חיפוש delete()/ find() | O(Log(n)) |
| גלגול- רוטציה | O(1) |
| **תור**  נכנס ראשון יוצא ראשון | הכנסה- enqueue()  והוצאה- dequeue() | O(1) |
| **מחסנית**  נכנס אחרון יוצא ראשון | הכנסה- push() הוצאה- Pop() הצצה- top() | O(1) |
| **תור עדיפויות - ערימה בינארית**  עץ בינרי כמעט שלם שמיוצג במערך שבו כל צומת גדולה מכל צאצאיה/ קטנה מכל צאצאיה.  (לפי רמת העדיפות) | חיפוש | O(n) |
| הכנסה - insert()  (הכנסה כעלה-O(1) וגלגול כלפי מעלה (siftUp -O(log(n)) | O(log(n)) |
| מציאת מקסימום/מינימום | O(1) |
| מחיקת מקסימום/מינימום- Extract Max  (שמים את העלה האחרון בראש ועושים גלגול כלפי מטה - siftDoun O(log(n))) | O(log(n)) |
| מחיקת צומת i – delete()  )שמים במקום הצומת ∞, מגלגלים כלפי מעלה, ואז עושים מחיקת מקסימום) | O(log(n)) |
| בנייה – buildHeap() | O(n) |
| מיון – sortHeap()  (עושים extractMax() ושמים את המקסימום בסוף המערך) | O(n log(n)) |
| **טבלת גיבוב** | הוספה , חיפוש ומחיקה | בממוצע O(1) |

|  |  |  |
| --- | --- | --- |
| **שם העץ** | **תיאור** | **גובה-** O(h) |
| מלא | לכל צומת 0 או 2 בנים | במקרה הטוב o(log(n)) במקרה הגרוע o(n) |
| כמעט שלם | כל הרמות מלאות, הרמה האחרונה משמאל לימין | O(log(n)) |
| שלם | כל הרמות מלאות (כולל האחרונה), מס צמתים 2h-1 | O(log(n)) |
| מאוזן - AVL | **עץ חיפוש בינארי** שבו עבור כל צומת, ההבדל בין גובה התת עץ הימני לשמאלי הוא מקסימום אחד (בערך מוחלט) | O(log(n)) |

**ערימה (בד"כ מקסימום)**:

ערימה מקסימלית היא ערימה שבה כל צומת גדולה מצאצאיה.

ערימה מינימלית היא ערימה שבה כל צומת קטנה מצאצאיה

כדי להקצות מערך המייצג ערימה בינארית צריך להקצות מערך בגודל arr[2h-1] ויתכנו תאים ריקים.

המערך לא ממוין!!

Size- גודל הערימה

maxSize- גודל המערך

הדרך להגיע מצומת מסוימת לבנים /אבא בערימה בינארית :

|  |  |  |
| --- | --- | --- |
|  | כאשר המערך מתחיל מ-0 | כאשר המערך מתחיל מ-1 |
| בן שמאלי | 2\*i+1 | 2\*i |
| בן ימני | 2\*i+2 | 2\*i+1 |
| מבן שמאלי / ימני לאבא | i-1)/2) | i /2 |

**עצים**

סריקה תחילית- שורש, שמאל, ימין

סריקה תוכית- שמאל, שורש, ימין

סריקה סופית – שמאל, ימין, שורש

סריקה רוחבית- סריקה לפי רמות ממומש ע"י תור.

עץ חיפוש בינרי

סריקה תוכית תביא לנו את הערכים ממוינים מקטן לגדול,

החיפוש הוא בזמן גובה העץ O(h), בעץ מאוזן O(log(n))

**טבלת גיבוב (hash\_table)**

שמירת נתונים בטבלת גיבוב מתבצעת ע"י הפעלת פונקציית גיבוב יעילה שמגיעה למספר בטווח בין 0 ל- m (גודל המערך בין n ל- 2n) בהתאם למפתח שמקבלת.

|  |  |  |
| --- | --- | --- |
|  | **שיטת השרשור** | **שיטת המיעון הפתוח** |
| הקצאת המערך | מקצים מערך בגודל מספר ראשוני בין n ל- 2n  כל איברי המערך הם מטיפוס Node | מקצים מערך בגודל מספר ראשוני בין n ל- 2n |
| הוספה | נפעיל פונקצית גיבוב יעילה ונוסיף אותו למערך במיקום המתאים  אם המקום תפוס נשרשר אותו לסוף הרשימה המקושרת | נפעיל פונקצית גיבוב יעילה ונוסיף אותו למערך במיקום המתאים  אם המקום תפוס מחפשים את התא הפנוי הבא לפי אחד מהגיבובים המופיעים למטה. |
| חיפוש | נפעיל את פונקצית הגיבוב אם המפתח לא נמצא במקום יש לעבור על כל הרשימה המקושרת שהמקום מצביע אליה | נפעיל את פונקצית הגיבוב אם המפתח לא נמצא במקום יש לחפש את האיבר כל עוד לא הגענו למקום ריק לפי פונקצית הגיבוב שהפעלנו בהכנסה |
| מחיקה | נפעיל את פונקצית הגיבוב אם המפתח לא נמצא במקום יש לעבור על כל הרשימה המקושרת שהמקום מצביע אליה ולמחוק את האיבר מהרשימה | נפעיל את פונקצית הגיבוב אם המפתח לא נמצא במקום יש לחפש את האיבר כל עוד לא הגענו למקום ריק לפי פונקצית הגיבוב שהפעלנו בהכנסה  כשהמיקום נמצא נסמן את המיקום כפנוי ולא כריק |
| יתרונות | הכנסה O(1) | לא תופס מקום נוסף |
| חסרונות | תופס יותר מקום בזיכרון | עלול לעבוד יותר זמן (תלוי בגיבוב) או יותר פונקציות גיבוב (יותר מורכב) |

סוגי גיבובים בשיטת המיעון הפתוח

גיבוב לינארי- בקפיצות של 1 בממוצע- O(n)

גיבוב ריבועי- קפיצות של i+i2 בממוצע- O(n)

גיבוב מוגבר- הפעלות פונקציות גיבוב נשנות בהתאם ל i-. בממוצע- O(1)

גיבוב כפול (גיבוב משני)- פונקצית גיבוב שנותנת את הפרש הקפיצות (בהתאם ל- i). בממוצע- O(1)