Merge Sort

~ עמית מאיר שובל

מה זה?

אלגוריתם המשמש לסידור מערך בצורת "הפרד ומשול".

בפועל:

* נעביר מערך כארגומנט לפונקציית ה-Merge Sort והיא תחלק את המערך ל-2. חצאי המערך יהיו "תתי-מערכים" והם יחלקו ביניהם את הערכים של המערך המקורי.

A screenshot of a computer

Description automatically generated

* Merge Sort היא פונקציה שהיא "רקורסיבית" אז בכל פעם שהפונקציה תסיים את הפעולה שלה, נקרא לאותה הפונקציה שוב. כלומר היא עושה את מה שהיא צריכה ואיך שהיא באה לסיים היא עושה את אותו הדבר שוב

A screenshot of a computer game

Description automatically generated

* {חילקה את המערך ל-2, מחלקת כל חצי ל-2 וכן הלאה עד שאי אפשר לחלק עוד.

A screenshot of a computer

Description automatically generated

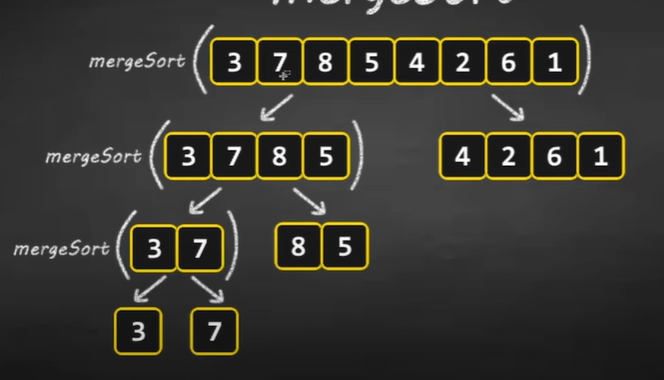
עכשיו אחרי שסיימנו לחלק את המערך לתתי מערכים נכנס הסידור שלו.

* ניצור פונקציה חדשה שתבצע איחוד של תתי-המערכים למערך מסודר, פונקציית Merge.   
  אותה פונקציה תקבל 3 ארגומנטים{ תת-מערך ימין, תת-מערך שמאל והמערך המקורי שהכיל את שניהם} ויבצע בדיקה ביניהם (X<Y) ויסדר אותם חזרה במערך המקורי לפי הסדר הנכון.
* דבר זה יקרה עד שכל המערך המקורי יהיה מסודר.

A screenshot of a computer game

Description automatically generated

* בפועל האלגוריתם לא באמת מבצע את הבדיקה במקביל על כל הענפים במקביל אלא הוא בודק ענף אחד בכל פעם. הסידור יראה כך, ייקח את הענף השמאלי ביותר וימשיך עד שכל המערך יהיה מסודר.



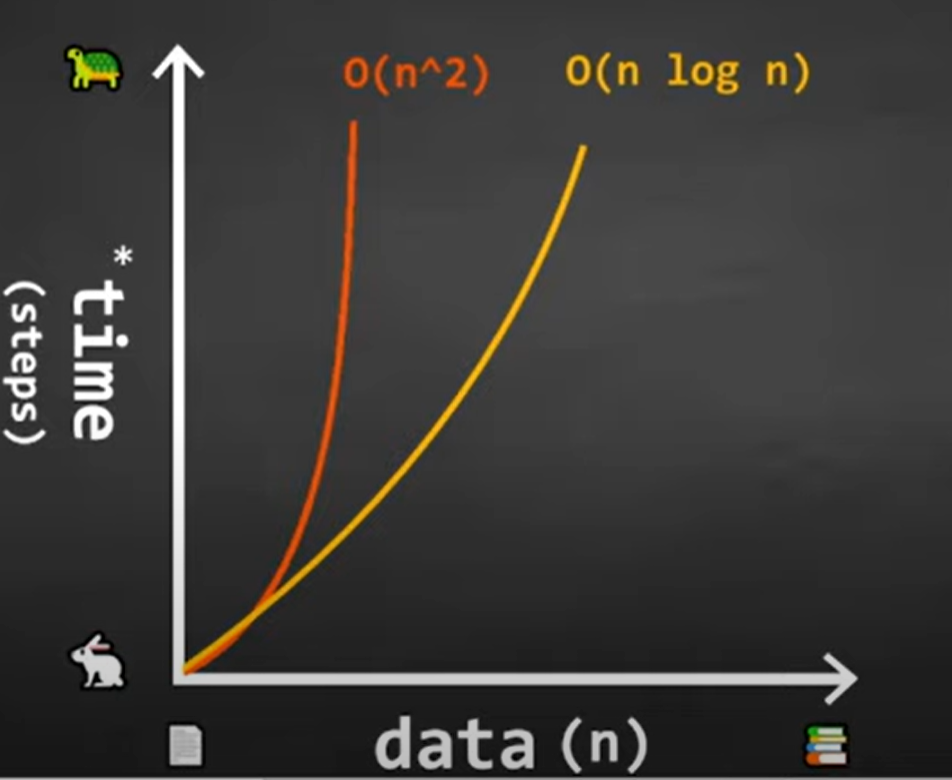
|  |  |
| --- | --- |
| שלב 1 |  |
| שלב 2 |  |
| שלב 3 |  |
| שלב 4 |  |
| עד כאן פיצלנו | |
| שלב 5 |  |
| שלב 6 |  |

סיבוכיות

O(n log n) = quasilinear time

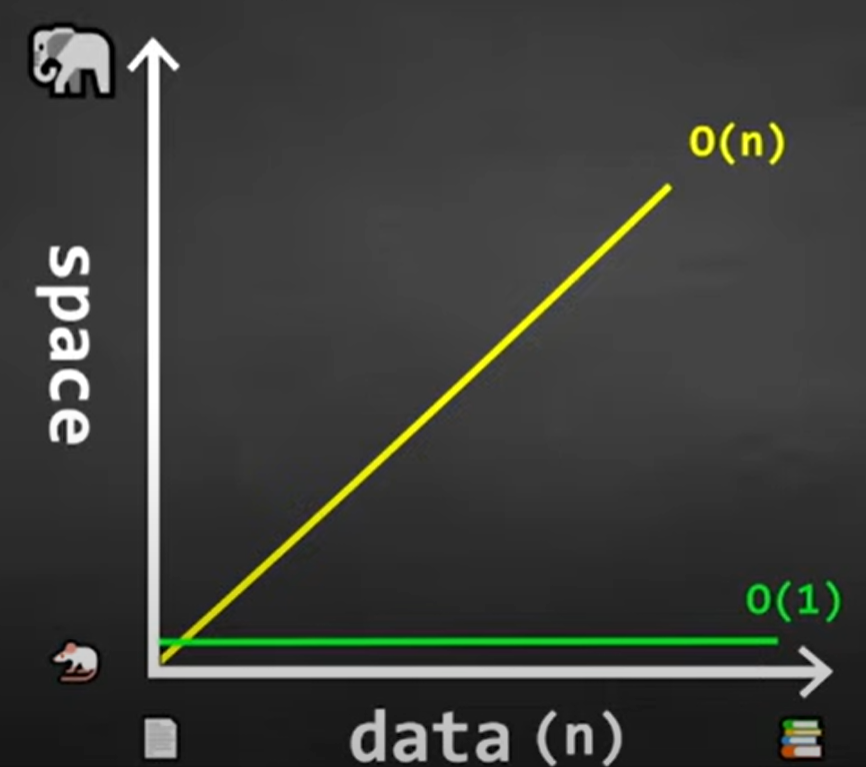
יתרון:

האלגוריתם עובד באופן מהיר יותר מ-Bubble, Selection, insertion כשמדובר בכמויות גדולות של מידע.



חסרון:

משתמש ביותר מקום בזיכרון מ-Bubble, Selection, insertion, וזאת בגלל שהוא צורך תתי-מערכים.



Code

#define arr\_length 8 // מגדיר גודל למערך

int arr[arr\_length];// יוצר מערך

void main()

{

    int i;// יצירת משתנה

    int j;// יצירת משתנה

    printf("Enter and print unsorted element");

    for( i=0 ; i<arr\_length-1 ; i++)// מעבר על כל איברי המערך

    {

        scanf("%d",%arr[i]);// קלט מהמשתמש לתוך המערך

        printf("%d", arr[i]);// בדפסה של הקלט הקודם

    }

    merge\_sort(0,arr\_length-1);// יצירת פונקציה לסידור המערך המקבלת מאיפה להתחיל ועד לאן

    printf("sorted array: \n")

    for(i=0 ; i<arr\_length-1 ; i++)

    {

        printf("%d \t",arr[i]);// הדפסה של המערך המסודר

    }

}

merge\_sort(int low, int high)

// low = 0 , high = arr\_length = 7

{

    if(low != high)

    // כל עוד נמוך שונה מגבוה

    // אם גבוהה שווה לנמוך אז הגענו לקצה ולא ניתן עוד לחלק את המערך

    {

        int mid;//יצירת משתנה ביניים

        mid = (low+high)/2;// משתנה הביניים יהיה הממוצע בין המספר הגבוהה לנמוך

        merge\_sort(low,mid);// קריאה חוזרת לפונקציה כדי לפצל את המערך פעם נוספת

        merge\_sort(mid+1,high);// מעבר על החצי השני של המערך, על הענף השני

        merge(low,mid,high);// קריאה לפונקציית האיחוד של המערך, סידור שלו

    }

}

merge(int low, int mid, int high)

// low = 0 , high = arr\_length = 7, mid = (low+high)/2

{

    int temp[arr\_length];// יצירת מערך זמני

    int i = low;// הקצאה של המספר הנמוך

    int k = low;// הקצאה של המספר הנמוך לבקרה

    int j = mid + 1;

    while((i <= mid) && (j <= high)){

        if(arr[i] <= arr[j])

        {

            temp[k++] = arr[i++];// הצבה לתוך המערך בסדר הנכון

        }

        else

        {

            temp[k++] = arr[j++];// הצבה לתוך המערך בסדר הנכון

        }

    }

    while(i <= mid)

    {

        temp[k++] = arr[i++];

    }

    while(j <= high)

    {

        temp[k++] = arr[j++];

    }

    for(i = low; i <= high ; i++)

    {

        arr[i] = temp[i];// מעבר על כל המערך והצבה מהמערך שסידרנו חזרה למערך המקורי

    }

}