פרק 5
תרגילים מהספר

שאלה 5.1

You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values—so there are $2n$ values total—and you may assume that no two values are the same. You'd like to determine the median of this set of $2n$ values, which we will define here to be the n^{th} smallest value.

However, the only way you can access these values is through *queries* to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most $O(\log n)$ queries.

תשובה 5.1

```
median(n, a, b)
    if n=1 then return min(A(a+k), B(b+k)) // base case
    k=⌈½n⌉
    if A(a+k)<B(b+k)
      then return median (k, a+⌊½n⌋ ,b)
      else return median (k, a, b+⌊½n⌋)
```

שאלה 5.2

Recall the problem of finding the number of inversions. As in the text, we are given a sequence of n numbers $a_1, \ldots, a_n$, which we assume are all distinct, and we define an inversion to be a pair $i < j$ such that $a_i > a_j$.

We motivated the problem of counting inversions as a good measure of how different two orderings are. However, one might feel that this measure is too sensitive. Let's call a pair a *significant inversion* if $i < j$ and $a_i > 2a_j$. Give an $O(n \log n)$ algorithm to count the number of significant inversions between two orderings.

תשובה 5.2

האלגוריתם המקורי לספירת היפוכים.

- let $k = \lfloor n/2 \rfloor$.
- call $\mathsf{ALG}(a'_1, \ldots, a'_k)$. Say it returns N_1 and $b_1, \ldots, b_k$.
- call $\mathsf{ALG}(a'_{k+1}, \ldots, a'_n)$. Say it returns N_2 and $b_{k+1}, \ldots, b_n$.
- compute the number N_3 of significant inversions (a_i, a_j) where $i \leq k < j$.
- return $N - N_1 + N_2 + N_3$ and $a'_1, \ldots, a'_n - \mathsf{MERGE}(b_1, \ldots, b_k,\ b_{k+1}, \ldots, b_n)$

האלגוריתם המעודכן לספירת היפוכים משמעותיים.

- Initialize counters: $i \leftarrow k,\ j \leftarrow n,\ N_3 \leftarrow 0$.
- If $b_i \leq 2b_j$ then
  - if $j > k+1$ decrease j by 1.
  - if $j = k+1$ return N_3.
- If $b_i > 2b_j$ then increase N_3 by $j - k$. Then
  - if $i > 1$ decrease i by 1.
  - if $i = 1$ return N_3.

שאלה 5.3

Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are *equivalent* if they correspond to the same account.

It's very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

תשובה 5.3

בהינתן קבוצת כרטיסים S נחלק אותה לשתי קבוצות זהות בגודלן : A ו-B.

תובנה בסיסית : בקבוצה S תיתכן קבוצת שקילות רק אם באחת משתי הקבוצות A ו/או B יש קבוצת השקילות. ההיפך אינו נכון : ייתכן שב- Aו/או ב-B תהיה קבוצת שקילות, אך אף אחד מהכרטיסים (שנניח שהם שונים זה מזה) המרכיבים את שתי קבוצות השקילות אינו מהווה קבוצת שקילות בקבוצה S.

נפעיל ברקורסיה את האלגוריתם על A ונמצא האם יש קבוצת שקילות בגודל של יותר מ- |A|/2. אם כן, הרקורסיה מחזירה לנו כרטיס השייך לקבוצה זו - נקרא לו C_A. נעשה כך גם לגבי B וייתכן שנקבל ממנה C_B.

התשובה שנחזיר מרמת הרקורסיה הנוכחית היא או C_A או C_B.

נספור כמה כרטיסים שקולים ל-C_A או ל-C_B יש בקבוצה S.

אם אחד המספרים גדול מ- |S|/2 - נחזיר את הכרטיס ואת מספר הכרטיסים השקולים לו בקבוצה S. אחרת נחזיר שאין לנו מה להציע מהקבוצה S.

תנאי העצירה של הרקורסיה – כאשר יש בקבוצה כרטיס אחד – נחזיר אותו. אם יש בקבוצה שני כרטיסים – אם הם שקולים נחזיר אחד מהם עם הספירה 2, אחרת נתעלם משניהם.

האפשרות להתעלם מהם נובעת מהתובנה שמחיקת שני כרטיסים שלכל היותר אחד מהם הוא חלק מקבוצת השקילות שאנו מחפשים, לא תשפיע על קיומה או העדרה של קבוצת שקילות.

נוסחת הנסיגה :

$$T(n) = 2T(n/2) + cn$$

כלומר

$$O(n\log n)$$

If $|S| = 1$ return the one card
If $|S| = 2$
    test if the two cards are equivalent
    return either card if they are equivalent
Let S_1 be the set of the first $\lfloor n/2 \rfloor$ cards
Let S_2 be the set of the remaining cards
Call the algorithm recursively for S_1.
If a card is returned
    then test this against all other cards
If no card with majority equivalence has yet been found
    then call the algorithm recursively for S_2.
    If a card is returned
        then test this against all other cards
Return a card from the majority equivalence class if one is found

שאלה 5.6

Consider an n-node complete binary tree T, where $n = 2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a *local minimum* if the label x_v is less than the label x_w for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labeling is only specified in the following *implicit* way: for each node v, you can determine the value x_v by *probing* the node v. Show how to find a local minimum of T using only $O(\log n)$ *probes* to the nodes of T.

נתחיל בשורש r.
אם השורש r קטן משני צאצאיו.
אם כן – הוא המינימום המקומי שאנו מבקשים.
אם לא – נעבור לאחד מבניו שהוא קטן מאביו, ונמשיך ברקורסיה.
האלגוריתם עוצר כיון ש :
או שעצרנו כשהגענו למינימום מקומי,
או שהגענו לעלה שאין לו בנים ושאביו גדול ממנו (אחרת לא היינו מגיעים מהאב לעלה), ולכן אפשר להחזיר אותו כמינימום מקומי.