Introduction to Algorithms

Exercise Class

Homework 1

2021-03-12

Homework explanation

1. Exercise 2.2-2

Consider sorting n numbers stored in array A by first finding the smallest element of A, and exchanging it with the element in $A[1]$. Then find the second smallest element of A, and exchange it with $A[2]$. Continue in this manner for the first $n - 1$ elements of A. Write pseudocode for this algorithm, which is known as ***selection sort***. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n - 1$ elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in Θ-notation.

SELECTION-SORT (A)

```
n = A.length
for j = 1 to n − 1
    smallest = j
    for i = j + 1 to n
        if A[i] < A[smallest]
            smallest = i
    exchange A[j] with A[smallest]
```

The algorithm maintains the loop invariant that at the start of each iteration of the outer **for** loop, the subarray $A[1 \ldots j-1]$ consists of the $j-1$ smallest elements in the array $A[1 \ldots n]$, and this subarray is in sorted order.

1. Exercise 2.2-2

After the first $n-1$ elements, the subarray $A[1 \ldots n-1]$ contains the smallest $n-1$ elements, sorted, and therefore element $A[n]$ must be the largest element.

The running time of the algorithm is $\Theta\left(n^2\right)$ for all cases.

$$\sum_{i=1}^{n-1} n - i = n(n-1) - \sum_{i=1}^{n-1} i = n^2 - n - \frac{n^2-n}{2} = \frac{n^2-n}{2} = \Theta\left(n^2\right)$$

Design an algorithm to compute Fibonacci numbers in pseudo code. Give an analysis for the time complexity of your algorithm.

Solution 1: Recursion

Fibonacci (n)
if $n \leq 1$
    return n
return Fibonacci ($n - 1$) + Fibonacci ($n - 2$)

Solution 1: Recursion

The formula for n-th term in the Fibonacci numbers:

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$$

Time Complexity: $O\left(2^n\right)$ or $\Theta\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$

Solution 2: Iteration

Fibonacci (n)
if $n \leq 1$
    return n
$a = 0$
$b = 1$
for $i = 2$ **to** n
    $c = a + b$
    $a = b$
    $b = c$
return b

Time Complexity: $\Theta(n)$

3. Exercise 2.3-7

Describe a $\Theta(n \lg n)$-time algorithm that, given a set S of n integers and another integer x, determines whether or not there exist two elements in S whose sum is exactly x.

Solution 1

Algorithm_2.3-7 (S, x)

Use Merge Sort to sort the array A in time $\Theta(n \lg n)$

$i = 1$

$j = n$

while $i < j$

    if $A[i] + A[j] == S$

        return **true**

    if $A[i] + A[j] < S$

        $i = i + 1$

    if $A[i] + A[j] > S$

        $j = j - 1$

return **false**

Solution 2

1. Sort the elements in S.
2. Form the set $S' = \{z : z = x - y \quad \textit{for some } y \in S\}$.
3. Sort the elements in S'.
4. If any value in S appears more than once, remove all but one instance. Do the same for S'.
5. Merge the two sorted sets S and S'.
6. There exist two elements in S whose sum is exactly x if and only if the same value appears in consecutive positions in the merged output.

4. Problem

Answer the following problem and justify your answer.

Input: integer n
Output: number of line 5 that is executed

```
1) count=0
2) for i=1 to n
3)     m=[n/i]
4)     for j=1 to m
5)         count=count+1
6)     end for
7) end for
8) return count
```

4. Problem

Answer: $\sum_{i=1}^{n} \left[\frac{n}{i}\right] \to \sum_{i=1}^{n} \left\{\left(\frac{n}{i}\right) - 1\right\} < \sum_{i=1}^{n} \left[\frac{n}{i}\right] \leq \sum_{i=1}^{n} n/i,$

Upper bound: $\sum_{i=1}^{n} n/i = n \sum_{i=1}^{n} \frac{1}{i} = n \log n,$

Lower bound: $\sum_{i=1}^{n} \left(\frac{n}{i} - 1\right) = n \log n - n,$

Thus, the answer is $\theta\,(n \log n)$.

Exercise

Problem 2-4 Inversions

Let $A[1 \ldots n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an **inversion** of A.

a. List the five inversions of the array $< 2, 3, 8, 6, 1 >$.
b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions?
c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

Problem 2-4 Inversions

a. List the five inversions of the array $< 2, 3, 8, 6, 1 >$.

The inversions are $(1, 5), (2, 5), (3, 4), (3, 5), (4, 5)$. (Remember that inversions are specified by indices rather than by the values in the array.)

b. What array with elements from the set $\{1, 2, \ldots, n\}$ has the most inversions?

The array with elements from $1, 2, \ldots, n$ with the most inversions is $< n, n-1, n-2, \ldots, 2, 1 >$. For all $1 \leq i < j \leq n$, there is an inversion (i, j). The number of such inversions is $\binom{n}{2} = n(n-1)/2$.

c. What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

Suppose that the array A starts out with an inversion (k, j). Then $k < j$ and $A[k] > A[j]$. At the time that the outer **for** loop of lines 1–8 sets $key = A[j]$, the value that started in $A[k]$ is still somewhere to the left of $A[j]$. That is, it's in $A[i]$, where $1 \le i < j$, and so the inversion has become (i, j). Some iteration of the **while** loop of lines 5–7 moves $A[i]$ one position to the right. Line 8 will eventually drop *key* to the left of this element, thus eliminating the inversion. Because line 5 moves only elements that are less than *key*, it moves only elements that correspond to inversions. In other words, each iteration of the **while** loop of lines 5–7 corresponds to the elimination of one inversion.

Thank you for listing!

Introduction to Algorithms
TA: Tianlu Fei
2021-03-12