

CS 410/586: Quiz 1, 30 March 2009

Name: KEY

No books or notes. Work individually.

These questions are about the Freq algorithm.

Question 1: What is this general style of algorithm called?

Divide and Conquer

Question 2: What happens if Freq is called on collection S where two different elements both have the maximum frequency?

It is possible that either one can be returned as the most frequent element. Suppose elements e_1 and e_2 both have maximum frequency f . If e_1 is chosen as the initial partition element, then Freq will return e_2 and vice versa.

Question 3: Suggest a method to perform Steps 2 and 3 of Freq with one pass through S (not counting the recursive calls). What data structure for S would support that method?

Make a pass through S , adding elements to lists S_{low} and S_{high} as appropriate. Elements that equal e are not added to either list should be counted to get f .

A linked list will support this operation, though it is possible to do it in place in an array.

Question 4: Suppose $C(n)$ is the number of comparisons Freq makes on an input of size n . Assume we are “lucky”, and the choice of e splits S into subsets of approximately the same size. Explain why

$$C(n) \approx n + 2 \cdot C(n/2)$$

There are $O(n)$ comparisons required on the pass through S to decide what should happen with each element. Then we make two calls on problems of approximately half the size.

Question 5: Assume we are always lucky as described in Question 4 on all recursive calls. What is the maximum depth of recursion Freq can have on an input of size n ?

If we are lucky, each time Freq makes a recursive call to itself, it is on an input of half the size. That is, if the initial input has n elements, then the recursive call from that has an input of size $n/2$; the recursive call from that one has input size $n/4$; the next is $n/8$, and so forth. So at depth k , the input has size $n/2^k$. The recursive calls must bottom out when $n/2^k = 1$, which is $k = \log_2 n$.

The following routine finds the most frequent element in a collection S of integers, plus the frequency (number of occurrences) of that element (without having to sort S). Note that S can have duplicates.

$|S|$ means the number of elements in S . For example $|\{3, 9, 8, 3\}| = 4$.

Freq(S)

0. If $|S| = 0$, return (null, 0)

1. Pick some element $e \in S$.

2. Split S into two subsets:

$$S_{low} \leftarrow \{d \in S \mid d < e\}$$

$$S_{high} \leftarrow \{d \in S \mid d > e\}$$

3. Set $f \leftarrow |S| - |S_{low}| - |S_{high}|$

Set $(e_{low}, f_{low}) \leftarrow \text{Freq}(S_{low})$

Set $(e_{high}, f_{high}) \leftarrow \text{Freq}(S_{high})$

4. If $f > f_{low}$ **and** $f > f_{high}$, **return** (e, f)

else if $f_{low} > f_{high}$, **return** (e_{low}, f_{low})

else return (e_{high}, f_{high})

Show a sequence of recursive calls for

Freq($\{4, 2, 1, 6, 6, 1, 2, 2, 8, 4\}$)