Lecture 19

COL 351: Analysis and Design of Algorithms

Lecture 19

Computing Majority Element

2-Majority Problem

Definition:

$> \frac{n}{2}$

Let $S = \{a_1, a_2, \ldots, a_n\}$ be a multiset. An element $x \in S$ is said to be majority element if it appears more than $n/2$ times in S.

(List/ Set with repeated elements)

Problem:

Computing the majority element in a multiset S.

Example:

$$\underbrace{S}_{\text{multiset}} = \{\underset{1}{a}, b, \underset{2}{a}, c, c, \underset{3}{a}, b, \underset{4}{a}, \underset{5}{a}\}$$

$|S| = 9$

Count $(a) = 5$

Majority $(S) =$ "a"

Naive Algorithm

- Count frequency of each element in multiset $S = \{a_1, a_2, \ldots, a_n\}$.
- Report the element with frequency larger than $n/2$ if exists.

\# of elements − n

Time per element − n

Time Complexity $O(n^2)$

$O(n^2)$

How much time does it take to verify majority?

To verify if an element x is majority of $S = \{a_1, a_2, \ldots, a_n\}$ it takes $O(n)$ time.

$\text{freq}(x, S) > \frac{n}{2} \rightarrow x$ is majority.

Simple Observation

Lemma: Suppose S contains n pairs each having identical elements, and S_0 is obtained from S by keeping only one element per pair.

Then a majority of S is also a majority of S_0

$$S \; \{ a_1 \; b_1 \quad a_2 \; b_2 \quad \ldots \quad a_n \; b_n \}$$

$$a_1 = b_1 \quad a_2 = b_2 \quad \ldots \quad a_n = b_n$$

$$S_0 = \{ a_1 \; a_2 \; \ldots \; a_n \}$$

$\rightarrow$ x is majority of S iff x is majority of S_0

Cancellation Lemma

Lemma: Let $x = majority(S)$ and, $a, b \in S$ be two distinct elements, then $x = majority(S - \{a, b\})$.

remove one copy.

Proof:

$\cdot x$ $\quad |S| = n-2$

x x x x x $\underbrace{}_{> n/2}$

a a b b c — Easy.

$\text{New Count}(x, S_{new}) > \frac{n}{2} - 1 = \frac{n-2}{2}$

Reducing problem of size $\underset{S}{\underline{2n}}$ to problem of size $\underset{S_0}{\underline{\text{at most } n}}$

Lemma: Let S be multiset of even size. Obtain S_0 from S by

- removing those pairs which have non-identical elements, and
- keeping only one element per identical pair.

Then a majority of S is also a majority of S_0.

$S = \{ \underline{a\,a} \quad \underbrace{c\,b}_{\times} \quad \underbrace{a\,b}_{\times} \quad \underline{b\,b} \quad \underline{a\,a} \quad \underline{a\,a} \quad \underline{a\,a} \}$

① $S' = \{ \underline{a\,a} \quad \underline{b\,b} \quad \underline{a\,a} \quad \underline{a\,a} \quad \underline{a\,a} \}$ If $x = maj(S) \Rightarrow x = maj(S')$

② $\underline{S_0} = \{ a \quad b \quad a \quad a \quad a \}$ $maj(S') = maj(S_0)$ Time $= O(n)$

Algorithm

Copy elements of S to $\underline{A}$.

While($|A| > 1$):

$T(n) = T(\frac{n}{2}) + O(n)$

$= O(n + \frac{n}{2} + \frac{n}{4} + \dots)$

If ($|A|$ is even):

1. Pair up the elements;
2. Eliminate all pairs of distinct elements;
3. Keep one element per pair of identical elements.

} Last lemma

Else:

Remove last element of A and check whether it is majority of ~~S~~ A

If (Only element left in A is a majority element of S): Return element in A;
Else : Return 'No majority';

$= O(n)$

Space used $= O(n)$

reduce to $O(1)$?

"Online Algorithms"

Scanning the list 'S' only ~~once~~ twice

| S[1] | S[2] | S[3] | S[4] | . . . |

Find S has majority.

Improved Algorithm

$S = \{ \cancel{a} \mid \cancel{a} \mid \cancel{b} \mid \cancel{c} \mid c \mid c \mid c \}$

$(a, 1) \mid (a, 2) \mid (a, 1) \mid (a, 0) \mid (c, 1) \mid (c, 2) \mid (c, 3)$

count = 0;
For (i = 1 to n):
if (count = 0):

if (count = 0):
$x = S[i]$;
count = 1;
else:
if ($x \neq S[i]$) : count = count -1;
else : count = count +1;

If (x is a majority element of S): Return x;
Else : Return 'No majority';

~~S[i]~~

x ... X
count - no of ties

Working with seq.

$$\boxed{\underbrace{x^{count},}_{\text{cancellations}} \underbrace{S[i] \ldots S[n]}_{\text{unscanned list}}} \leftarrow \text{New Multiset}$$

Improved Algorithm

count = 0;
For ($i = 1$ to n):
if (count = 0):
$x = S[i]$;
count = 1;
else:
if ($x \neq S[i]$) : count = count -1;
else : count = count +1;

If (x is a majority element of S): Return x;
Else : Return 'No majority';

1	2
S has NO MAJORITY	S has a MAJORITY
Algorithm is correct	Let "a" be the majority of S Need to prove that a is output.

Improved Algorithm

ASSUMPTION : a is MAJ of (S[1], .. S[n])
BASE CASE : a is MAJ of (x^0, S[1], .., S[n])

count = 0;
For ($i = 1$ to n):

Claim:
If a is majority of (x^{count}, $S[i], \ldots, S[n]$) before i^{th} round,

```
        if ( count = 0 ):
            x = S[i];
            count = 1;
        else:
            if ( x ≠ S[i] ) : count = count -1;
            else :  count = count +1;

If (x is a majority element of S): Return x;
Else : Return 'No majority';
```

then,
a is majority of ($x^{count-new}$, $S[i+1], \ldots, S[n]$) after i^{th} round.

Case 1: count = 0, count-new = 1

$(S[i], .., S[n]) = (\underline{x = S[i]}, \underline{S[i+1], .., S[n]})$

Case 2: count > 0, $x \neq S[i]$

$(x^{count}, S[i], .., S[n]) \rightarrow (x^{count-1}, S[i+1], .., S[n])$

Case 3: count > 0, $x = S[i]$

$(x^{count}, S[i], .., S[n]) \rightarrow (x^{count+1}, S[i+1], .., S[n])$

Improved Algorithm

H.W. 3-majority problem.
An element of freq $> \frac{n}{3}$

```
count = 0;
For (i = 1 to n):
        if ( count = 0 ):
            x = S[i];
            count = 1;
        else:
            if ( x ≠ S[i] ) : count = count -1;
            else :  count = count +1;

If (x is a majority element of S): Return x;
Else : Return 'No majority';
```

O(1) space
O(n) time

Claim:
If a is majority of (x^{count}, $S[i], \ldots, S[n]$) before i^{th} round,
then,
a is majority of ($x^{count-new}$, $S[i+1], \ldots, S[n]$) after i^{th} round.

Assumed
a is majority of $(\underline{x^{0}}\ S[i] \ldots S[n])$

---

a will be majority of $(\underline{x^{count}})$