

Lecture 19

COL 351: Analysis and Design of Algorithms

Lecture 19

Computing Majority Element

2-Majority Problem

Definition:

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

$> \frac{n}{2}$

(List/Set with repeated elements)

Problem:

Computing the majority element in a multiset S .

Example:

$S = \{a, b, a, c, c, a, b, a, a\}$
 multiset

$|S| = 9$

count(a) = 5

Majority(S) = "a"

Naive Algorithm

- Count frequency of each element in multiset $S = \{a_1, a_2, \dots, a_n\}$.
- 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 $S = \{a_1, a_2, \dots, a_n\}$ it takes $O(n)$ time.

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

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

$$S = \{ \begin{matrix} a_1 & b_1 & & a_2 & b_2 & & \dots & a_n & b_n \end{matrix} \}$$

$$\begin{matrix} a_1 = b_1 & a_2 = b_2 & & a_n = b_n \end{matrix}$$

$$S_0 = \{ a_1, a_2, \dots, a_n \}$$

$$\rightarrow x \text{ is majority of } S \iff x \text{ is majority of } S_0$$

Cancellation Lemma

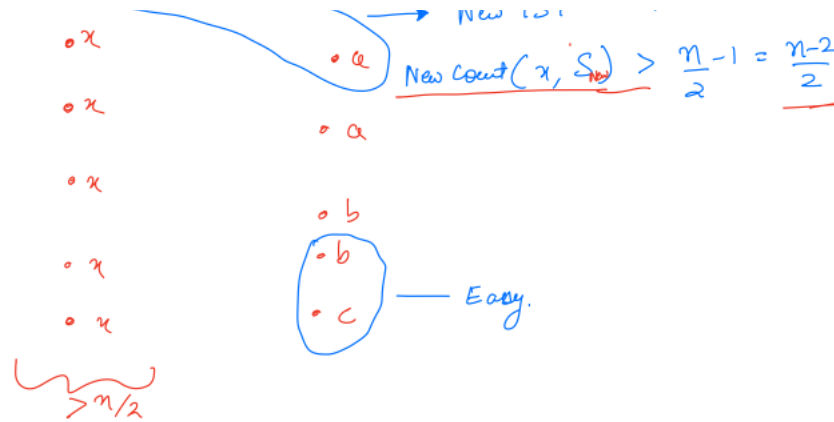
Lemma: Let $x = \text{majority}(S)$ and, $a, b \in S$ be two distinct elements, then $x = \text{majority}(S - \{a, b\})$.

removing one copy.

Proof:

$$x$$

$$n - 1 - 1 = n - 2$$



Reducing problem of size $\frac{2n}{5}$ to problem of size at most $\frac{n}{5}$

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 .

$$S = \{ \underline{aa}, \underline{cb}, \underline{ab}, \underline{bb}, \underline{aa}, \underline{aa}, \underline{aa} \}$$

① $S' = \{ \underline{aa}, \underline{bb}, \underline{aa}, \underline{aa}, \underline{aa} \}$ If $x = \text{maj}(S) \Rightarrow x = \text{maj}(S')$

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

Algorithm

Copy elements of S to A .

While($|A| > 1$):

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

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

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 ~~A~~
 A

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

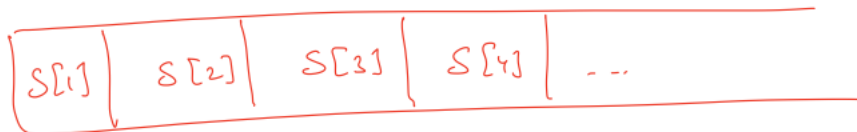
$$= O(n)$$

$$\text{Space used} = O(n)$$

Reduce to $O(1)$?

"Online Algorithms"

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



Find S has majority.

Improved Algorithm

$$S = \{ \cancel{x} \mid \cancel{x} \mid \cancel{x} \mid \cancel{x} \mid c \mid c \mid c \}$$

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

count = 0;

For ($i = 1$ to n):

if (count = 0):

```

if (count = 0):
    x = S[i];
    count = 1;
else:
    if (x ≠ 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
canc - no of times

Working with Seq.

$x^{count}, S[i] \dots S[n]$ ← New Multiset
cancellations uns cancell list

Improved Algorithm

```

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';

①
S has NO MAJORITY

Algorithm is correct

②
S has a MAJORITY

Let "a" be
the
majority of S

Need to prove that
a is output

Improved Algorithm

ASSUMPTION: a is MAJ of $(S[i], \dots, S[n])$
BASE CASE: a is MAJ of $(x^0, S[i], \dots, S[n])$

```

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

```

Claim:

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

Case 1: $\text{count} = 0$, $\text{count-new} = 1$

$(S[i], \dots, S[n]) = (x = S[i], S[i+1], \dots, S[n])$

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

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

Case 3: $\text{count} > 0$, $x = S[i]$

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

Improved Algorithm

```

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

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

Claim:

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

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

Assumed
a is majority of $(x^{\text{count}}, S[i], \dots, S[n])$

a will be majority of $(x^{\text{count}}, S[i], \dots, S[n])$