Dawid Pawliczek
Lista 3, Zadanie 2

Treść

Tablica $A[1 \dots n]$ ma *element większościowy*, jeśli ponad połowa jej elementów ma tę samą wartość. Zaprojektuj algorytm, który:

1. sprawdza, czy element większościowy istnieje,
2. jeśli tak – zwraca jego wartość,

dysponując wyłącznie testem równości ? $A[i] = A[j]$ (o koszcie stałym).

Poniżej dwie strategie **Dziel i Zwyciężaj**:

(a) $\underbrace{\text{„Podziel tablicę na dwie połowy \dots”}}_{\text{rekursja klasyczna}}$ (b) $\underbrace{\text{„Pogrupuj elementy w pary \dots ”}}_{\text{eliminacja parami}}$

1 Strategia (a) — podział na połowy

Intuicja

Element większościowy musi wystąpić w co najmniej jednej z połówek podtablicy. Jeśli każda połowa wskaże swojego kandydata, sprawdzamy, który (jeśli którykolwiek) dominuje w całości.

Algorytm

Algorithm 1 MAJORITYDIVIDE($A[l\,..\,r]$)

```
if l = r then return A[l]
end if
m ← ⌊(l + r)/2⌋
x ← MajorityDivide(A[l .. m])
y ← MajorityDivide(A[m + 1 .. r])
if x ≠ nil and x występuje > (r−l+1)/2 razy then return x
else if y ≠ nil and y występuje > (r−l+1)/2 razy then return y
elsereturn nil
end if
```

Poprawność

- Jeśli element większościowy istnieje w $A[l\,..\,r]$, musi wystąpić również w którejś połowie, więc co najmniej jedna rekurencja zwróci jego wartość.
- Po uzyskaniu kandydatów x, y liczymy ich *faktyczne* wystąpienia w całości (używając testów równości); zwracamy ten, który spełnia definicję większości. Brak takiego $\Rightarrow$ tablica nie ma elementu większościowego.

Złożoność

Rekurencja daje równanie $T(n) = 2T(n/2) + O(n)$, którego rozwiązaniem (tw. Master) jest

$$T(n) = \Theta(n \log n).$$

2 Strategia (b) — eliminacja parami (Boyer–Moore)

Intuicja

Sparuj elementy kolejno. Różne w parze *neutralizują się* – żaden z nich nie może być większością. Jeśli para zawiera dwa identyczne elementy, *zastępujemy* ją pojedynczym przedstawicielem. Po jednym przebiegu rozmiar wejścia maleje o $\approx \frac{1}{2}$, a element większościowy (jeśli istniał) *przetrwa*. Powtarzamy aż zostanie ≤ 1 kandydat.

Algorytm

Algorithm 2 MajorityPair($A[1..n]$)

while $n > 1$ **do**
    $B \leftarrow [\,]$
    for $i \leftarrow 1$ **to** $\lfloor n/2 \rfloor$ **do**
        if $A[2i-1] = A[2i]$ **then**
            dołącz $A[2i]$ do B
        end if
    end for
    if n nieparzyste **then** dołącz $A[n]$ do B
        $A \leftarrow B,\ n \leftarrow |B|$

        if $n = 1$ **and** kandydat występuje $> \frac{|A_0|}{2}$ razy **then return** kandydat
        elsereturn `nil`
        end if

(Łatwiej: implementacja w jednym przebiegu z licznikiem *Boyer–Moore Majority Vote*, ale powyższy wariant używa wyłącznie „par+kasuj".)

Poprawność

- **Zachowanie większości.** Para z różnych elementów usuwa po 1 z liczebności każdego rodzaju; równoliczność większości pozostaje $> \frac{1}{2}$.
- **Zbieżność.** Po każdej iteracji tabl. skraca się co najmniej o połowę, więc pętla wykona $O(\log n)$ rund.
- **Weryfikacja.** Ostateczny kandydat (jeśli istnieje) jest sprawdzany na oryginalnym ciągu, unikając fałszywie pozytywnego wyniku.

Złożoność

Każdy element bierze udział w ≤ 1 porównaniu na rundę, a rund jest $O(\log n)$. Łącznie:

$$\Theta(n) \text{ porównań} \quad + \quad \Theta(n) \text{ pamięci pomocniczej (można } O(1) \text{ przy wersji Boyer–Moore).}$$

Porównanie

Metoda	Czas	Pamięć
(a) połówki	$\Theta(n \log n)$	$\Theta(\log n)$ rek.
(b) pary	$\Theta(n)$	$O(1)$ (Boyer–Moore)

Metoda (b) jest więc asymptotycznie lepsza przy ograniczeniu do testów równości.