Řešení voleb s d výherci v O(dn) čase, O(d) prostoru

Vilém Zouhar

2. 12. 2017

Úvod

Stačí jen zobecnit Petrův algoritmus z hodiny. Pro $d = 1$ budeme postupně procházet pole a budeme se řídit následujícími pravidly:

1. Pokud je prvek $a_i = v$:
   $c := c + 1$
2. Pokud je prvek $a_i \neq v$:
   $c := c - 1$
3. Pokud je $c = 0$:
   $v := a_i$
   $c := 1$

Ve v bude tedy nejvíce možný kandidát. To evidentně funguje, neboť pokud nějaký vítěz existuje, pak "převáží" (podrobněji na další straně) ostatní tím, že musí mít alespoň $\frac{n}{2}$ hlasů. Jestli je opravdu výherce, tak se čítadlo vícekrát inkrementuje než dekrementuje. Na konci jen stačí ověřit, že vítěz "prvního kola" je opravdu vítězem. To lze přirozeně v $O(n)$ čase.

Pro 2 vítěze algoritmus vypadá jen o trochu složitěji:

Pro průchod polem se řiďme pravidly:

1. Pokud je prvek $a_i = v_1$:
   $c_1 := c_1 + 1$
2. Jinak pokud $c_1 = 0$:
   $c_1 := 1$
   $v_1 := a_i$
3. Jinak pokud je prvek $a_i = v_2$:
   $c_2 := c_2 + 1$
4. Jinak pokud $c_2 = 0$:
   $c_2 := 1$
   $v_2 := a_i$
5. Pokud nic z toho:
   $c_1 := c_1 - 1$
   $c_2 := c_2 - 1$

Což také funguje, neboť pokud přistupujeme k nějakému prvku, které nemáme mezi kandidáty na vítězy prvního kola $\{v_1, v_2\}$, pak čítače obou dvou dekrementujeme. Pokud najdeme volné místo (z těch dvou kandidátů na vítěze prvního kola), tedy jeho čítač je 0, tak ho nahradíme. Pokud nějaký vítěz existuje, pak "převáží" ostatní tím, že má alespoň $\frac{n}{3}$ hlasů. Zachrání jej totiž to, že když se přistupuje k prvku nějakého jiného kandidáta na vítěze prvního kola, tak se nedekrementuje v_1, ale jen se inkrementuje v_2. Samozřejmě je nutné na konci ověřit, zdali vítězové prvního kola jsou opravdu vítězové celých voleb.

Nyní je zřejmé jak rozšířit program (tedy systém podmínek) na d vítězů. Stačí nám k tomu smyčka přes d pozic kandidátů na vítěze prvního kola. Průchod polem se bude řídit tedy následujícími podmínkami:

1. Pokud je nějaký prvek $v_j = a_i$:
   $c_j := c_j + 1$
2. Pokud žádný nebyl, tak pokud je nějaký prvek $c_j = 0$:
   $c_j := 1$
   $v_j := a_1$
3. Pokud také žádný nebyl, tak pro všechna c_j:
   $c_j := c_j - 1$

Kdykoliv přistupujeme na prvky, které nemáme uložené mezi kandidáty, tak se jejich čítače snižují. Pokud nějaký vítěz existuje a má aspoň $\frac{n}{d+1}$ hlasů, pak musí existovat úsek, ve kterém je mezi d nejvíce zastoupenými, tedy ho přidáme do seznamu. Byl by z tohoto seznamu vytlačen, pokud by následovalo aspoň $d\frac{n}{d+1} + 1$ různých dalších kandidátů, což ale nastat nemůže (pro $0 < d < n$).

$$\text{pro jednoduchost předpokládejme } (d+1)|n \tag{1}$$

$$n - \frac{n}{d+1} \geq \frac{dn}{d+1} + 1 \tag{2}$$

$$dn + n - n \geq dn + d + 1 \tag{3}$$

$$\text{nemá řešení pro } \; 0 < d < n \tag{4}$$

Tedy pokud existuje, tak ho tímto způsobem dostaneme do seznamu. Na konci stačí jen ověřit, zdali všichni vítězi prvního kola jsou opravdovými vítězi (tedy v_j je vítěz $\Leftrightarrow$ $c_j > \frac{n}{d+1}$). Časová náročnost tohoto algoritmu je $O(d \cdot n)$, neboť v prvním průchodu polem jsme nuceni procyklit v nejhorším případě všechny kandidáty na vítěze prvního kola a na konci musíme ověřit, zdali jsou opravdu vítězi. Paměťová náročnost je $O(d)$, neboť si musíme udržovat pole délky d na identifikace kandidátů a na jejich počítadla.

Proof of concept

Např. v Pythonu 3 implementace vypadá následovně:

```
#!/usr/bin/python3
from math import floor

n = int(input('n: ')) # da se v O(n) zjistit, ale pro komfort..
d = int(input('d: '))

citac     = [0]*d # O(d)
kandidati = [0]*d # O(d)
pole      = [int(i) for i in input('arr: ').split(' ')] # oddeleno mezerou

for i in pole: # O(n)
    for j in range(d): # O(d)
        tick = False
        if kandidati[j] == i:
            citac[j] += 1
            tick = True
            break
        elif citac[j] == 0:
            citac[j] = 1
            kandidati[j] = i
            tick = True
            break

    if not tick:
        for j in range(d): # O(d)
            citac[j] -= 1

citac = [0]*d # O(d)
for i in pole: # O(n)
    for j in range(len(kandidati)): # O(d)
        if i == kandidati[j]:
            citac[j] += 1

hranice = floor(n/(d+1))
for i in range(len(kandidati)): # O(d)
    if citac[i] >= hranice:
        print(kandidati[i])
```