Řešení rep3
Vilém Zouhar
05. 11. 2017

Základní myšlenkou je procházet celé pole blokem (pomocí dvou *jezdců*), u kterého si budeme udržovat aktuální velikost (maximální si vždy poznamenáme) a prvky, které jsou uvnitř bloku (pomocí vhodné datové struktury).

Hodně by nám pomohlo, kdybychom měli hodnoty v rozahu 1..N. Toho můžeme docílit, pokud si zvolíme správnou funkci, která nám nezobrazí dva různé prvky na jeden obraz. Taková funkce je např:

f: x → [x mod N, (x div N) mod N, (x div N^2) mod N]

Tato funkce zobrazí prvek tři na trojici čísel (značme *T*), jejíž každý prvek je v rozsahu *1..N*. Každou "souřadnici" jsme tedy schopni setřídit *bucket sortem* v N čase. Dohromady celou trojici tedy *radix sortem* se stejným časem. Takto setříděné pole nám vytvoří funkci:

g: T → n (n v 1..N)

Složením funkcí dostaneme:

s: x → n (n v 1..N)

Což je přesně funkce, kterou jsme potřebovali. Můžeme si tedy *přejmenovat* náš vstup na čísla v rozsahu 1..N, takže můžeme vytvořit bool pole této délky (značme *V*).

Začneme inicializací obou jezdců (*LJ, PJ*) na 0 a postupně budeme zvyšovat hodnotu *PJ* o 1 než se dostaneme na konec pole (*P*). Budeme si také uchovávat hodnotu nejdelšího dosavadního úseku *M*. V každé takové iteraci provedeme následující:

```
č := P[PJ]
pokud V[s(č)] = false:
      V[s(č)] = true
      pokud PJ - LJ + 1 > M:
            M := PJ - LJ + 1
jinak:
      dokud LJ < PJ a zároveň P[LJ] <> c:
            V[s(č)] = false
            LJ := LJ + 1
      V[s(č)] = true
PJ := PJ + 1
Výsledek: M
```

V první větvi podmínky rozdíl *PJ - LJ* zvyšujeme, tedy pouze zde porovnáváme, zdali to náhodou není nové maximum. V druhé větvi naopak rozdíl snižujeme, tudíž neřešíme, zdali je maximem. Program posouvá dva jezdce směrem doprava, takže v nejhorším případě se jezdci projede vzdálenost *2N*.

Invariant programu je, že pokud je nějaké číslo v *V*, pak se v poli mezi *LJ* a *PJ* vyskytuje právě jednou. (z toho důvodu nám stačí datová struktura *set* a nepotřebujeme např. slovník<*int, int*>). Pokud se dostaneme pomocí *PJ* na číslo, které už *V* obsahuje, začne se posouvat *LJ* dokud se v *P[LJ..PJ]* nevyskytuje dvakrát, čímž se invariant zachrání.

Vynecháné úseky v poli odpovídají těm, které obsahují alespoň dvě stejná čísla a proto jsme je odstranili.