Dawid Pawliczek

Lista 1, Zadanie 2

Ułóż algorytm, który dla danych k uporządkowanych niemalejąco list $L_1, \ldots, L_k$ liczb całkowitych znajduje najmniejszą liczbę r taką, że w przedziale $[a, a+r]$ znajduje się co najmniej jedna wartość z każdej z list L_i dla pewnej liczby a. Algorytm nie może modyfikować list L_i i powinien być pamięciowo oszczędny (oraz oczywiście jak najszybszy).

Rozwiązanie:
Mamy k list uporządkowanych niemalejąco

$$\begin{array}{cccc} a, & b, & c, & \ldots \\ x, & y, & z, & \ldots \\ \vdots & \vdots & \vdots & \end{array}$$

Chcemy znaleźć najmniejszy przedział, który pokryje co najmniej po jednej wartości z każdej listy.
Przykład:

$$\begin{array}{ccc} 1, & 2, & 3 \\ 4, & 5, & 6 \\ 2, & 3, & 4 \end{array}$$

Możemy wziąć $a = 1$, $r = 3$: wtedy wybieramy wartości $1, 4, 2$ — każda lista jest pokryta.

Obserwacja: Pokrycie możemy reprezentować k wskaźnikami. Z aktualnych wartości wyznaczamy minimalny i maksymalny element — to nasz bieżący przedział. Aby go zmniejszyć, zawsze przesuwamy wskaźnik wskazujący na *najmniejszy* element; przesunięcie największego tylko wydłużyłoby przedział.

Algorytm (wersja uproszczona):

Algorithm 1 getMinRange

Require: listy $L_1, \ldots, L_k$

```
for i ← 1 to k do pointers[i] ← 0
range ← max(val(pointers)) − min(val(pointers))
while prawda do
    p ← argmin(val(pointers))
    pointers[p] ← pointers[p] + 1
    if pointers[p] poza zakresem then break
        newRange ← max(val(pointers)) − min(val(pointers))
        if newRange < range then range ← newRange

            return range
```

Trzymanie k wskaźników i wyszukiwanie minimum w każdej iteracji daje złożoność $O(k \cdot n)$, gdzie n to liczba wszystkich elementów. Można to poprawić, przechowując wskaźniki w kopcu, redukując czas do $O(n \log k)$.

Algorytm (kopiec):

Algorithm 2 GETMINRANGEHEAP

Require: listy $L_1, \ldots, L_k$

$H \leftarrow$ nowy *min-heap*
$left \leftarrow -\infty$, $right \leftarrow \infty$
for $i \leftarrow 1$ **to** k **do**
    $left \leftarrow \min(left, L_i[0])$
    $right \leftarrow \max(right, L_i[0])$
    H.PUSH($L_i[0]$, i, 0)
end for
while prawda **do**
    $(val, idx\_list, idx\_val) \leftarrow H$.POPMIN()
    $idx\_val \leftarrow idx\_val + 1$
    if idx_val poza zakresem **then break**
        $next \leftarrow L_{idx\_list}[idx\_val]$
        H.PUSH($next$, idx_list, idx_val)
        $right \leftarrow \max(right, next)$
        $left \leftarrow H$.MINVALUE()
        if $right - left <$ bestRange **then**
            aktualizuj najlepszy przedział
        end if

        return $[left, right]$