Jméno: Petr Valenta **UČO:** 487561

1. [20 bodů] Vaším úkolem je analyzovat data ze dvou databází. Každá databáze obsahuje n čísel. Potřebujete zjistit, jaké je n-té nejmenší číslo ze všech $2n$ hodnot. Jediná možnost, jak přistoupit k hodnotám v databázích je prostřednictvím funkcí $\textsc{Get1}(k)$ a $\textsc{Get2}(k)$, kde k je parametr, $1 \leq k \leq n$. Výsledkem volání $\textsc{Get1}(k)$ je k-té nejmenší číslo v první databázi; analogicky pro $\textsc{Get2}$. Navrhněte algoritmus, který najde n-té nejmenší číslo ze všech čísel uložených v databázích použitím $\mathcal{O}(\log n)$ volání funkcí $\textsc{Get1}$ a $\textsc{Get2}$.
Můžete předpokládat, že žádná dvě z $2n$ čísel nejsou stejná a že n je mocninou 2.

Hlavní myšlenka
Upravil jsem algoritmus binárního vyhledávání.

Algoritmus

```
left1, left2 ← 1
right1, right2 ← n
function findN(left1, right1, left2, right2)
    if left1 = right1 then
        return minimum(Get1(left1), Get2(left2))
    index1 ← left1 + floor((right1 − left1)/2)
    index2 ← left2 + floor((right2 − left2)/2)
    value1 ← Get1(index1)
    value2 ← Get2(index2)
    if value1 < value2 then
        left1 ← index1 + 1
        right2 ← index2
    else
        left2 ← index2 + 1
        right1 ← index1
    return findN(left1, right1, left2, right2)
```

Zdůvodnění korektnosti
Důkaz provedu indukcí nad x. Velikost databáze je $n = 2^x$. Nechť $P(x)$ je předpoklad, že algoritmus `findN` je korektní pro všechny vstupy, kde $right1 - left1 + 1 = right2 - left2 + 1 = 2^x$. Pokud dokážeme, že $P(x)$ je pravdivý pro všechna $x \epsilon \mathbb{N}_0$, dokážeme, že algoritmus `findN` je korektní pro všechny možné vstupy.

Základní krok $P(0)$: $x = 0$, $n = right1 - left1 + 1 = right2 - left2 + 1 = 1$

Úloha je zpracována nerekurzivně. Velikost rozsahu, ve kterém hledáme je 1 v obou databázích. Hledáme n-tý nejmenší prvek, kde n je velikost rozsahu, tedy úplně nejmenší prvek. Dojde k volání $\textsc{Get1}(1)$ a $\textsc{Get2}(1)$ a výsledkem je menší z vrácených prvků.

Indukční krok $P(k)$: $x = k$, $n = right1 - left1 + 1 = right2 - left2 + 1 = 2^k$

Za předpokladu, že $P(x)$ platí pro všechna $0 \leq x \leq k - 1$, dokážeme, že platí $P(k)$.

Velikost rozsahu, ve kterém hledáme je 2^k v obou databázích. Hledáme 2^k-tý nejmenší prvek. Pomocí operací $\textsc{Get1}(index1)$ a $\textsc{Get2}(index2)$, zjistíme a porovnáme hodnoty prvků ve středech rozsahů. Výpočet středu rozsahu viz pseudokód.

Můžou nastat dvě možnosti:

$value1 < value2$:

V spodních polovinách rozsahů je dohromady právě n prvků. Prvek na pozici $index1$ je tudíž nejvýše $(n-1)$-ní a prvek na pozici $index2$ je nejméně n-tý. Hledaný prvek se musí nacházet v horní polovině prvního rozsahu, nebo ve spodní polovině druhého rozsahu. Získáme tak nové rozsahy, jejichž velikost je poloviční $n/2 = 2^{k-1}$. Následuje rekurzivní volání s novými rozsahy. $P(k-1)$ je dle našeho předpokladu korektní, $P(k)$ je tedy také korektní.

$value1 \geq value2$:

Prvek na pozici $index1$ je nejméně n-tý a prvek na pozici $index2$ je nejvýše $(n-1)$-ní. Hledaný prvek se musí nacházet ve spodní polovině prvního rozsahu, nebo v horní polovině druhého rozsahu. Získáme tak nové rozsahy, jejichž velikost je poloviční $n/2 = 2^{k-1}$. Následuje rekurzivní volání s novými rozsahy. $P(k-1)$ je dle našeho předpokladu korektní, $P(k)$ je tedy také korektní.

Analýza složitosti

Nechť $G(n)$ je složitost operací GET. Rekurentní rovnice:

$$T(n) \begin{cases} 2 \times G(n) + 2, & \text{if } x = 0 \\ T(n/2) + 2 \times G(n) + 12, & \text{otherwise} \end{cases}$$

Předpokládejme, že algoritmus použije $\mathcal{O}(2 \log n)$ volání funkcí GET. Použijeme kuchařkovou větu (alternativní variantu) [1]:

Nechť $a \geq 1$, $b \geq 1$ a $c \geq 0$ jsou konstanty a nechť T(n) je definována na nezáporných číslech rekurentní rovnicí

$$T(n) = aT(n/b) + \Theta(n^c)$$

Potom platí

$$T(n) \begin{cases} \Theta(n^c), & \text{když } a < b^c & (1) \\ \Theta(n^c \log n), & \text{když } a = b^c & (2) \\ \Theta(n^{\log_b a}) & \text{když } a > b^c & (3) \end{cases}$$

Z naší rekurentní rovnice vyplývá $a = 1$, $b = 2$ a $c = 0$. Algoritmus tedy spadá pod druhý případ a jeho složitost je $\Theta(G(n) \log n)$.

[1] Převzato z IB002-slajdy.pdf, str. 64)