P-I-3 Nevhodný dárek

Thursday, 10 November 2022 14:25

```
Int c, l, r;
Int result = 0;
Input(c, l, r);
Int n = log₂(c)+ 1
Bool x[n] = c.toBinary()
For(int y = 1, i = 0; i<n && r >= y; i++, y*= 2){
        Result += x[i]*((r+y)/2*y)
}
For(int y = 1, i = 0; i<n && l-1 >= y;i++, y*= 2){
        Result -= x[i]*((l-1+y)/2*y)
}
Print(result)
```

Čísla se v paměti počítačů ukládají v binárních hodnotách, tudíž funkce běží v konstantním čase - O(1)

Běží v čase O(n) nebo v čase $O(\mathrm{Log}_2(r))$ - podle toho co je menší

Běží v čase O(n) nebo v čase $O(\mathrm{Log}_2(l))$ - podle toho co je menší

Aritmetické operace umíme ve složitosti O(1)

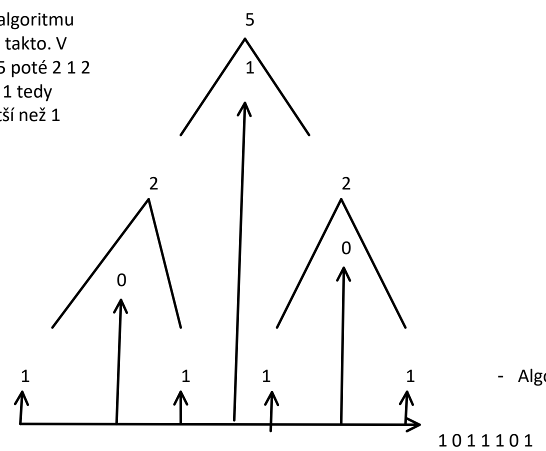
Tudíž **$O(\mathrm{Log}_2(\min(c,r))$**

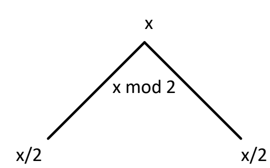
Důkaz: Algoritmus popsaný v zadání vždy rozdělí číslo na **dva celé stejné díly** a na **zbytek**. Tudíž se nám nemůže ani ztratit ani přibýt jediná jednička. To znamená, že ve výsledné posloupnosti vždy bude c jedniček.

Algoritmus si můžeme rozepsat na tento strom:

K tomuto diagramu jsme přišli jednoduše tím, že dělíme na dva díly a zbytek je ve prostřed:

- vzhledem k tomu, že na pořadí algoritmu nezáleží, můžeme si to rozepsat takto. V algoritmu bychom nejdřív měli 5 poté 2 1 2 poté 1 0 1 1 2 a poté 1 0 1 1 1 0 1 tedy rozkládali bychom vždy číslo větší než 1 které je co nejvíce vlevo.

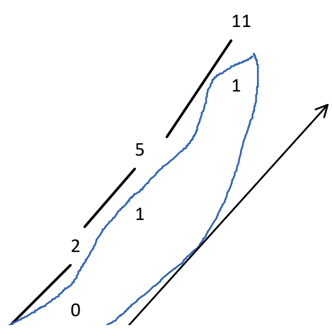


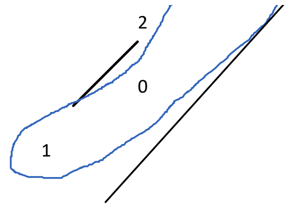


- Algoritmus se zastaví u jedničky

Z tohoto diagramu se dá výsledná posloupnost vypsat pohledem ze zdola zleva doprava - stejně jako u binárních stromů - viz. průvodce labyritnem algoritmů - Martin Mareš, Tomáš Valla - cz.nic vydání str. 178 bvsShow

Pozorování: Při rozdělování čísla na dvě stejné části je zbytečné si držet čísla obě, ale stačí sledovat 1 z rozdělených částí a zbytky:





V tomto je však vidět, že se vlastně jedná o převod do dvojkové soustavy že? Provádíme kompletně stejné operace jako u převodu do binární soustavy dělíme dvěmi a píšeme si zbytky. Jenom skončíme o jeden krok dříve - takže do výsledného binárního přepočtu zde musíme vzít i poslední číslo. Příklad stejného převodu:

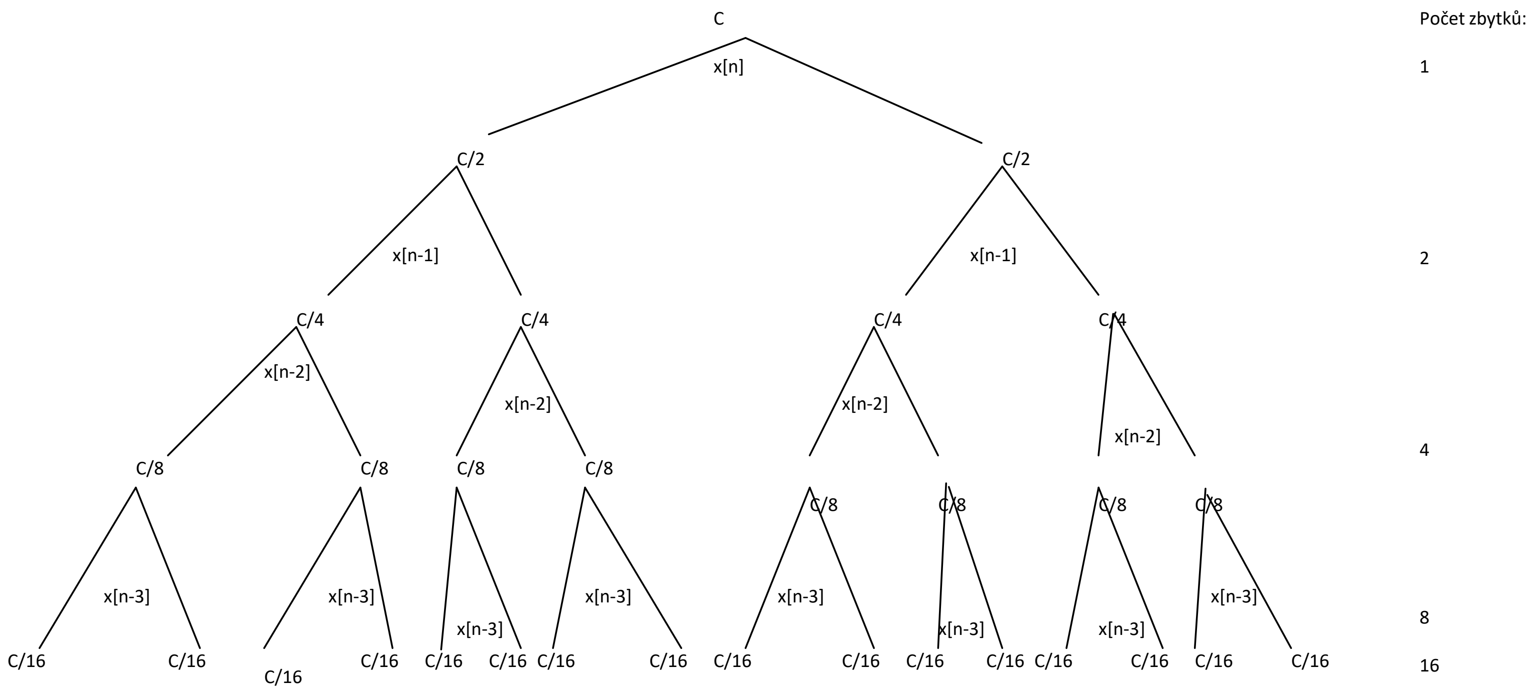
11 : 2 = 5 zb. 1
5 : 2 = 2 zb. 1
2 : 2 = 1 zb. 0
1 : 2 = 0 zb. 1

$11_{10} = 1011_2$

Tuto binární posloupnost nyní pojmenujeme x. V tomto případě x = {1,0,1,1}

Převod z binární soustavy x do výsledné binární posloupnosti

Vrátíme se zpátky ke stromu, ukážeme si to na příkladu n = 5 (z ukázky kódu - n = log2(c) +1 - bez decimálních čísel)



Protože vždy dělíme číslo na dvě stejné části strom je kompletně symetrický.
Ve finální binární posloupnosti by to mohlo vypadat takto

C

C/2 x[n] C/2

C/4 x[n-1] C/4 x[n] C/4 x[n-1] C/4

C/8 x[n-2] C/8 x[n-1] C/8 x[n-2] C/8 x[n] C/8 x[n-2] C/8 x[n-1] C/8 x[n-2] C/8

C/16 x[n-3] C/16 x[n-2] C/16 x[n-3] C/16 x[n-1] C/16 x[n-3] C/16 x[n-2] C/16 x[n-3] C/16 x[n] C/16 x[n-3] C/16 x[n-2] C/16 x[n-3] C/16 x[n-1] C/16 x[n-3] C/16 x[n-2] C/16 x[n-3] C/16

x[n-4] x[n-3] x[n-4] x[n-2] x[n-4] x[n-3] x[n-4] x[n-1] x[n-4] x[n-3] x[n-4] x[n-2] x[n-4] x[n-3] x[n-4] x[n] x[n-4] x[n-3] x[n-4] x[n-2] x[n-4] x[n-3] x[n-4] x[n-1] x[n-4] x[n-3] x[n-4] x[n-2] x[n-4] x[n-3] x[n-4]

Z tohoto je jasně vidět, že nejnovější zbytek bude vždy každý druhý v posloupnosti. Je to proto, že původní číslo C je vždy z obou stran původního zbytku. Díky tomu také můžeme říci, že začíná na indexu 1.
Co se týče zbytků výše ve stromu (v tomto případě např. x[n-3]) pokaždé, co jdeme o jeden zbytek výše ve stromu tak máme 2x méně zbytků ve finální posloupnosti a tudíž budou 2x delší intervaly mezi těmito zbytky. Jak od začátku (modrá barva na grafu
Příklad:
Zbytků x[n-4] je 16, jsou poslední, tudíž se vyskytují na každém 2. místě a začínají na indexu 1
Zbytků x[n-3] je 8,vyskytují se 2x méně, tudíž se vyskytují na každém 4. místě (hlavně proto že strom je symetrický) a začínají na indexu 2
...

Zapíšeme si do tabulky

i	1	2	3	4	5
počet	16x	8x	4x	2x	1x
x:	x[1]	x[2]	x[3]	x[4]	x[5]
Výskyt y	2.	4.	8.	16.	32.
První index y/2	1.	2.	4.	8.	16.

Nyní, pokud potřebujeme zjistit kolik se vyskytuje daných zbytků x[i] v intervalu stačí si vzít první výskyt, přičíst k němu pravé uzavření daného intervalu (levé uzavření je vždy 0) a vydělit to výskytem y. Nyní je jasné, že pokud interval bude dosahovat na první výskyt daného zbytku, vyrovná se dvojnásobku prvního výskytu (protože pravé uzavření intervalu musí být rovno nebo více prvnímu výskytu), aby po dělení vyšlo číslo o jedno větší je potřeba přidat znova další výskyt y, aby výsledek po dělení byl o jedno větší. Tudíž si dovoluji tvrdit, že vám to vyhodí počet daných zbytků v daném intervalu. Zapsané to vypadá nějak takto:

$$\frac{r + y_i/2}{y_i}$$

Pro finální součet v intervalu od 0 do r stačí pro každý prvek použít tuto rovnici a vynásobit jí x[i], abychom nepočítali i nuly

$$x_1 \times \frac{r + \frac{y_1}{2}}{y_1} + x_2 \times \frac{r + \frac{y_2}{2}}{y_2} + x_3 \times \frac{r + \frac{y_3}{2}}{y_3} + x_4 \times \frac{r + \frac{y_4}{2}}{y_4} + \cdots + x_n \times \frac{r + \frac{y_n}{2}}{y_n}$$

Sumou se toto dá zapsat následovně

$$\sum_{i=1}^{n} x_i \times \frac{r + \frac{y_i}{2}}{y_i}$$

Avšak stále nejsme hotoví, protože interval musí být uzavřen i zleva. Pro to nám stačí odečíst počet jedniček v intervalu 0 - l-1 (protože v zadání je včetně l). Tedy naprosto stejný případ

$$x_1 \times \frac{l - 1 + \frac{y_1}{2}}{y_1} + x_2 \times \frac{l - 1 + \frac{y_2}{2}}{y_2} + x_3 \times \frac{l - 1 + \frac{y_3}{2}}{y_3} + x_4 \times \frac{l - 1 + \frac{y_4}{2}}{y_4} + \cdots + x_n \times \frac{l - 1 + \frac{y_n}{2}}{y_n}$$

V sumě:

$$\sum_{i=1}^{n} x_i \times \frac{l + \frac{y_i}{2}}{y_i}$$

Výsledný součet tedy získáme následovně:

$$\sum_{i=1}^{n} x_i \times \frac{r + \frac{y_i}{2}}{y_i} - \sum_{i=1}^{n} x_i \times \frac{l - 1 + \frac{y_i}{2}}{y_i}$$

Po úpravě:

$$\sum_{i=0}^{n} x_i \times \frac{r + 2^i}{2^{i+1}} - \sum_{i=0}^{n} x_i \times \frac{l - 1 + 2^i}{2^{i+1}}$$

V algoritmu máme také urychlovací podmínku. Ta vznikla tak, že kontrolujeme jestli r + y >= 2y (případně l-1 + y >= 2y). Je to proto, že pokud r+y bude menší tak bude vycházet nula a vzhledem k tomu, že se každou iteraci zvětšuje pouze y tak už může loop skončit, protože jsme mimo interval. Po úpravě /-y máme r >= y a l-1 >= y.