Друштво математичара Србије
Адреса: Кнез Михаилова 35/4,
11000 Београд
сајт: www.dms.rs
е-маил: pom@dms.rs

Решење проблема Б за Новембар 2011
Бинарни број

На адресу редакције за први проблем Б овог циклуса стигло је 10ак решења, од којих су скоро сви били прихваћени из првог покушаја. Већина алгоритама су имала сличну идеју, док је пар ученика успело да их имплементира у константном времену. Овде ћемо изложити пар приступа, укључујући и могућу генерализацију проблема. На почетку ћемо описати идеју односно сам алгоритам, а касније продискутовати могуће имплементације истог.

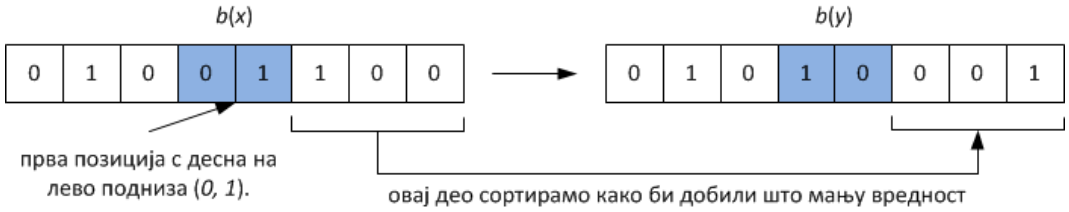
Алгоритам: Означимо са $b(x)$ бинарни запис природног броја x, при чему цифре можемо памтити (приказивати) у облику стринга или низа. Разлика између ова два приступа је минимална, препоручујемо да користите онај у коме се сигурније осећате. У алгоритму који ћемо овде изнети користићемо синтаксу низа. Вештачки ћемо додати још једну цифру 0 у овом бинарном запису, као цифру највеће тежине (разлог за ово ће ускоро бити јасан). Додавање водећих нула не мења вредност самог броја (ово важи не само за бинарне бројеве већ и за бројеве у било којој основи). Примера ради, за $x = 5$ имамо да је $b(5) = (0, 1, 0, 1)$. У даљем тексту када причамо о цифрама броја подразумеваћемо бинарни запис.

Покушајмо прво да ограничимо на неки начин број y тј. дужину низа $b(y)$. Означимо са x' број који се добија пермутацијом цифара броја x на следећи начин: $(1, 0, \ldots, 0, 1, \ldots, 1)$, где смо водећу цифру поставили на 1, затим низ цифара 0 и на крају преосталих $N(x) - 1$ јединица. Овакав број x' задовољава услов да је $N(X) = N(x')$ и има једну бинарну цифру више од број x (то је управо она вештачка нула коју смо додали на почетку). Дакле, тражени број y ће имати највише једну значајну бинарну цифру више (значајне цифре су оне које нису водеће нуле, другим речима биће највише дупло већи од броја x). Како је y најмањи број са горњим условом, имамо да је $y \leq x'$. Конкретније, $b(y)$ представља неку пермутацију низа $b(x)$, тачније "следећу" пермутацију овог низа (гледајући лексикографски поредак).

Проблем смо свели на налажење следеће пермутације бинарног низа. Посматрајмо два природна броја A и B који имају исти број цифара. Уколико важи да је $A > B$, то подразумева да је прва цифра на којој се они разликују, гледано са лева на десно (од цифре највеће тежине ка цифри најмање тежине), једнака 1 код броја A а нула код броја B. Како тражени број y мора да буде најмањи, желимо да тежина цифре на којој се разликују буде што мања. Зато крећемо од цифре најмање тежине и тражимо прву цифру 0 којој претходи цифра 1 тј. подниз $(0, 1)$. За прву цифру на којој се разликују број x и y бирамо управо ову

нулу. Вредност на овој позицији ћемо у број y да променимо у јединицу а јединицу у нулу, чиме добијамо да број y има исти број бинарних јединица као и број x и да је већи од њега. Остаје још да цифре које претходе овим позицијама сортирамо тако да прво иду јединице а затим нуле, како би добили што мању вредност.

За боље разумевање алгоритма погледати пример дат на слици. Препоручујемо читаоцу да сам ручно покуша да симулира алгоритам на пар примера како би боље разумео шта алгоритам заиста ради.



Слика 1. Пример генерисања цифара броја y преко цифара број x.

Сада можемо разумети разлог додавања вештачке нуле у низу $b(x)$. Наиме, једини случај када у бинарном низу $b(x)$ не постоји нула којој претходи јединица јесте ако је он облика $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$. Елегантно решење овог јесте да увек додамо водећу нулу којом добијамо низ $(0, 1, 1, \ldots, 1, 0, 0, \ldots, 0)$. На овај начин сигурно обезбеђујемо постојање траженог подниза. Наравно, проблем се може решити и третирањем овог случаја посебно, али би тиме имплементација алгоритма била мало компликованија.

Алгоритам: Псеудо код генералног алгоритма за проблем Бинарни број

Input: природни број x који испитујемо
Output: тражена вредност y

```
// означимо са b[] бинарни запис броја x ;
if (postoji 0 koja prethodi jedinici u binarnom zapisu broja x) then
    index = најмања позиција нуле која претходи јединици у b;
    b[index] = 1;
    b[index − 1] = 0;
    сортирај подниз b[index − 2], b[index − 3], ..., b[1];
end
else
    m = дужина низа b;
    b[m + 1] = 1;
    b[m] = 0;
    сортирај подниз b[m − 1], b[m − 2], ..., b[1];
end
y = декадна вредност бинарног број записаног у низу b;
return y
```

Имплементација 01: Прва могућност имплементације горе описаног алгоритма је мање више његова симулација. На почетку, иницијализујемо низ $b(x)$ са додавањем вештачке нуле. Иницијализација овог низа има логаритамску сложеност[a]. Након тога, крећемо од цифре

[a]Опис алгоритма за превођење бројева из једне основе у другу (у нашем случају из бинарног записа у декадни и обрнуто) можете наћи у свакој програмерској књизи или збирци.

најмање тежине и тражимо прву позицију подниза $(0, 1)$. Након замене ових вредности, потребно је сортирати обиђени део. Како је ово специјални случај низа, његове вредности су бинарне, сортирање можемо описати на једноставнији начин[b]: померимо све јединице које су биле испре нађење позиције, а остатак попунимо нулама. Укупна сложеност ове имплементације је $O(\log x + \log x + \log x) = O(\log x)$. Имплементацију овог алгоритма можете наћи у пропратном материјалу за овај проблем.

Напомена: У `C++`-у за налажење следеће пермутације можете користити STL функцију $next\_permutation$ која би заправо у овом проблему одрадила већи део посла. Ова функција ради у генералном случају, када низ не мора бити бинаран. Наравно, препоручујемо да исту користите тек када је сами прво имплементирате.

Имплементација 02: Иако на први поглед делује невероватно, овај проблем можемо решити и у константном времену. Као што смо већ напоменули, пар таквих решење смо и примили. Приступ који ћемо сада изнети су послали ученици: Борис Грубић[c] и Лазар Миленковић[d]. Уз помоћ пар елементарних логичких операција (бит операција) тражена вредност се може иницијализовати без експлицитног налажења бинарне репрезентације броја x, као и без експлицитног сортирања датог у генералном алгоритму.

Прва препрека је наћи прво појављивање нуле којој претходи јединица. Разматрамо два случаја:

- број x је паран: У овом случају имамо да је бинарни запис броја x облика $b(x) = (A, 0, 1, \ldots 1, 1, 0, \ldots 0)$, где је A произвољан бинарни низ. Одавде имамо да је $b(x - 1) = (A, 0, 1, \ldots 1, 0, 1, \ldots 1)$. Када применимо логичко OR над ове две вредности добијамо
$$x\ OR\ (x-1) = (A, 0, 1 \ldots 1, 1, 1, \ldots 1)$$
Додавањем јединице горњем резултату добија се број облика $(A, 1, 0, \ldots, 0)$. Позиција прве јединице управо тражена позиција подстринга $(0, 1)$. Можемо приметити да је она већ иницијализована на јединицу. Препоручујемо читаоцу да сам види на конкретном примеру ток описаних операција.

- број x је непаран: Као и у првом случају, применом описане формуле добија се иста форма. Детаљну анализу овог случаја остављамо читаоцу.

Означимо са z број који смо добили горе описаним поступком, тачније

$$z = (x\ OR\ (x-1)) + 1$$

Преостаје да броју z променимо одређени број нула са почетка (тренутно су сви они нуле) у јединице. Ово можемо да урадимо тако што ћемо број z додати вредност 2^k и одузети један, где смо са k означили број јединица који је претходио траженој позицији нуле. Подсетимо се да одузимањем јединице од броја 2^k добијамо $b(2^k - 1) = (1, \ldots, 1)$ ($k - 1$ јединица). Приметимо да додајемо $k - 1$ јединицу, а не k њих, јер смо једну искористили при промени нуле у јединицу у првом делу.

Тражени број 2^k ћемо добити преко прве јединице у броју x. За ово ће нам послужити згодна функција $f(x) = x\ AND(-x)$, која ће имати вредност $f(x) = 2^p$, где је p управо

[b]Ово је бинарна верзија сортирања пребројавањем (енг. *count sort*).
[c]Гимназија "Јован Јовановић Змај", Нови Сад
[d]Прва крагујевачка гимназија, Крагујевац

позиција прве јединице у бинарном запису броја x. Овде морамо имати у виду да се негативна вредност природног броја памти као двоструки комплемент, тачније $NOT(n-1)$. Уколико поделимо вредности $f(z)$ и $f(x)$ добијамо број 2^{k+1}. Дељењем са два, или $shift$-овањем ове вредности за један у десно, добијамо тражени број 2^k. На крају, након одузимања јединице од 2^k, ову вредност треба "убацити" на почетак броја z. Како смо видели да су на почетку број z све нуле, описано убацивање можемо одрадити преко операције OR.

Алгоритам: Псеудо код алгоритма друге имплементације

Input: природни број x који испитујемо
Output: тражена вредност y

1 $z = (x\ OR\ (x-1)) + 1$;
2 $f(z) = z\ AND\ -z$;
3 $f(v) = v\ AND\ -v$;
4 $y = ((f(z)/f(v)) >> 1) - 1$;
5 $y = t\ OR\ y$;
6 **return** y

Имплементација 03: Друга варијанта решења у константом времену, се мање више састоји од једне линије кода. Ово је најкраћа имплементација ове функције (по броју не бланко карактера) која је позната члановима редакције. Функција која за дати параметар враћа први већи број са истим бројем бинарних јединица се може имплементирати као:

```
int getNextNumber(int n)
{
    int k;
    return (n+(k=n&-n)^n)/4/k+n+k;
}
```

Генерализација: На почетку смо поменули генерализацију овог проблема. Овде смо имали специјални случај, када је разматрани број бинарни. Задатак се може генерализовати на следећи начин:

Дат је природни број n и основа b. Наћи најмањи природни број m, који је већи од n и који има исти број цифара $1, 2, \ldots, b$ у својом запису у основи b као и број n.

Заправо ово је модификација познатог проблем да за дату пермутацију нађете следећу лексикографску пермутацију. "Инверзан" проблем овог проблем ја такође интересантан: за дату пермутацију наћи њену лексикографску позицију. Примера ради за низ дужине 3 имамо 6 могућих пермутација

$$(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)$$

које смо управо и сортирали по лексикографском уређењу. Уколико са $indexOfPerm(p)$ означимо позицију пермутације, имали би примера ради да је $indeOfPerm((2,1,3)) = 3$. Препоручујемо читаоцу да реши и ове две варијанте генерализације, јер често можете наићи на проблем где су ово заправо успутне станице ка главном решењу.

Решење задатака припремио:
Андреја Илић,
Природно математички факултет, Ниш