nedelja informatike 3

Veeeeeliki brojevi

Aleksandar Ivanović

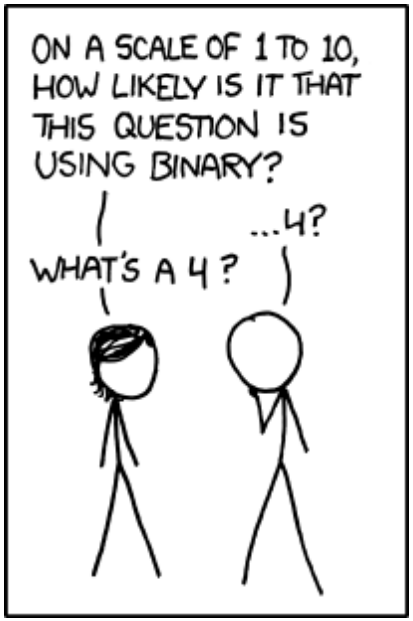
Matematička gimnazija
Nedelja informatike 3

12. decembar 2016.

Uvod

- Postoji 10 tipova ljudi na svetu, oni koji razumeju binarni sistem, oni koji ne razumeju binarni sistem i oni koji nisu očekivali šalu u ternarnom sistemu.

Slika: http://xkcd.com/953/

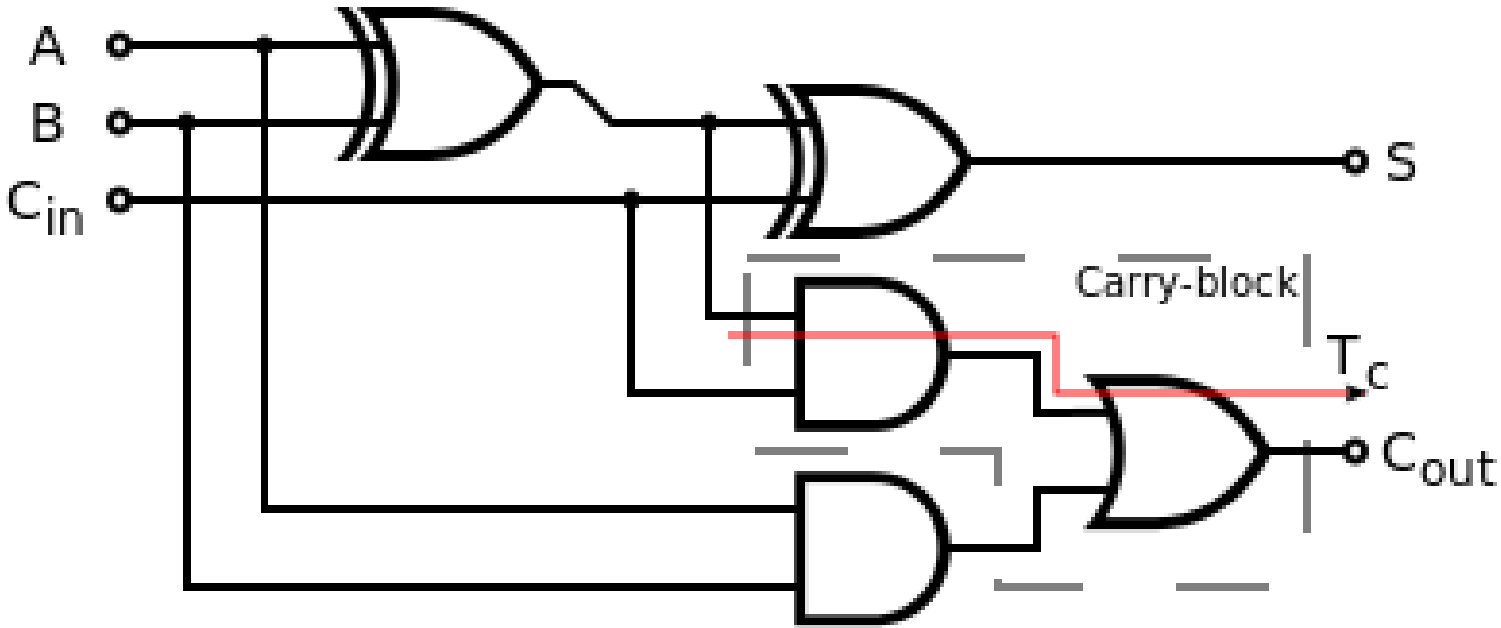


Uvod

- Kako računari sabiraju brojeve?
- ... i kako ih oduzimaju?

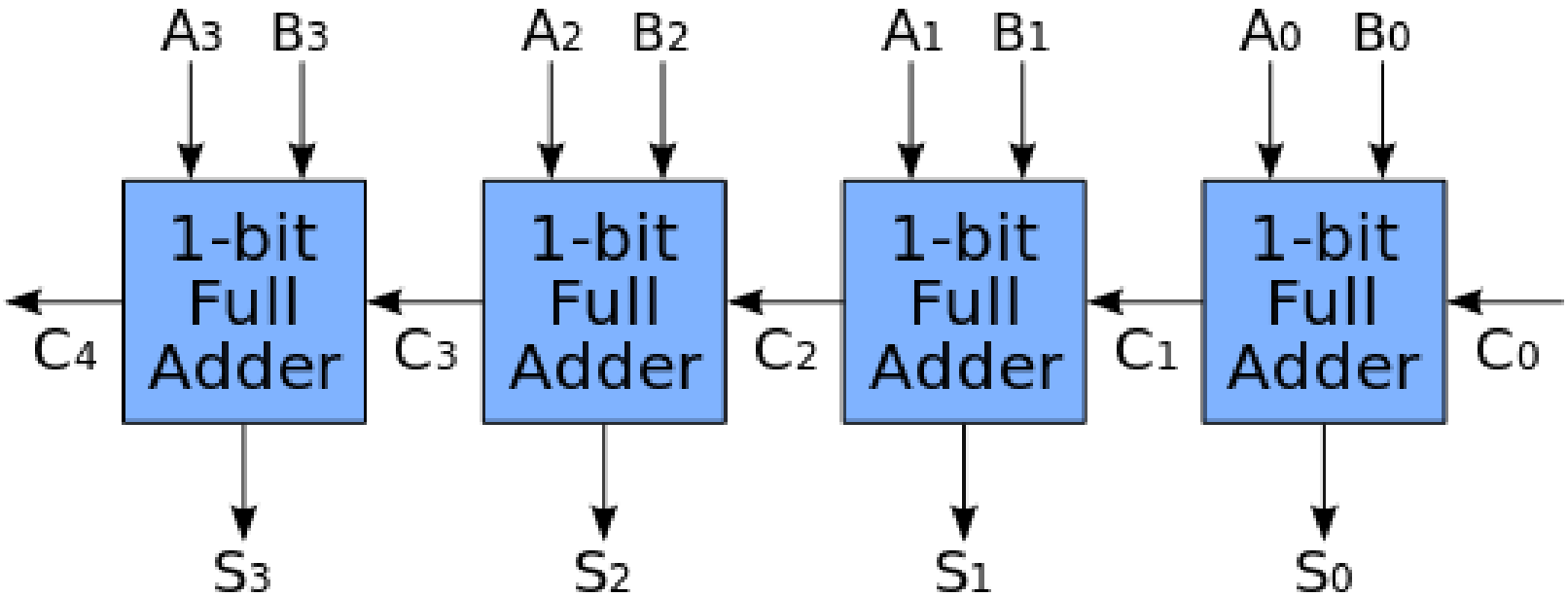
Sabiranje

Slika: 1-bitni kompletni sabirač



Sabiranje

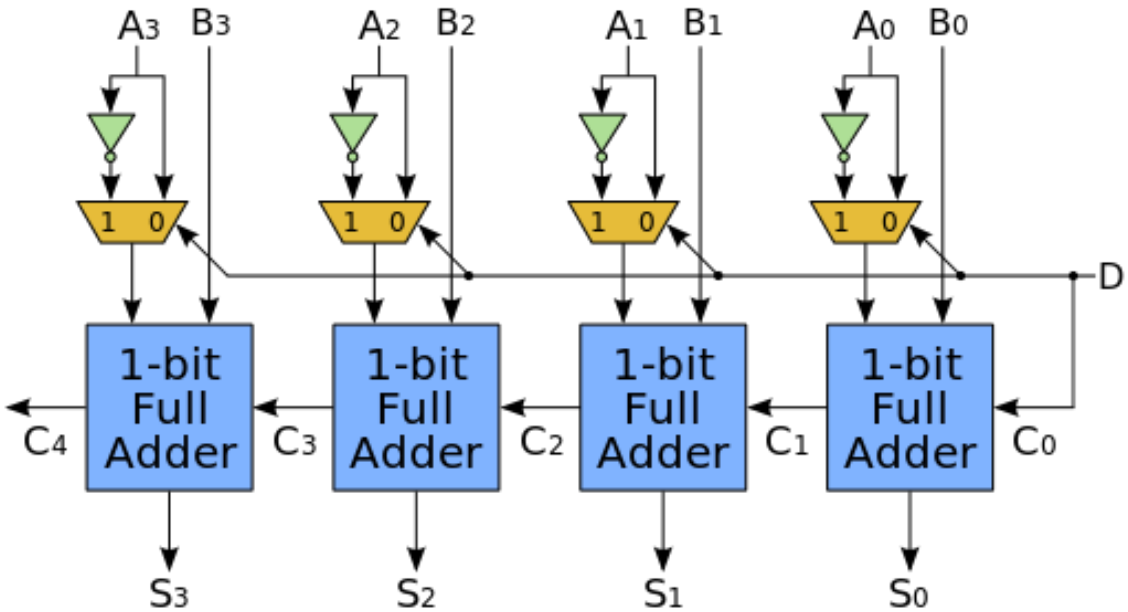
Slika: 4-bitni sabirač



Oduzimanje

- Celi brojevi se u računaru čuvaju zapisani u komplementu dvojke
- Ako je $D = 0$, onda je $S = A + B$, a ako je $D = 1$, onda je $S = B - A = B + \overline{A} + 1$

Slika: 4-bitni sabirač/oduzimač



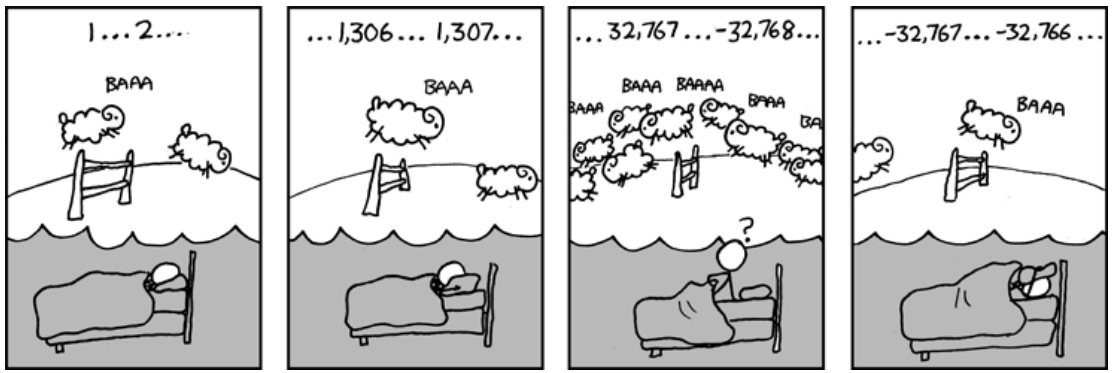
Zašto veeeeeliki brojevi?

- Kriptografija
- Visoka preciznost
  - Racionalne brojeve možemo implementirati kao razlomke čiji su brojilac i imenilac veeeeeliki brojevi

```
for (double i = 0.1; i != 1.0; i += 0.1) {
  cout << i << endl;
}
```

- Overflow

Slika: https://xkcd.com/571/



Write in C

Slika: Sabiranje velikih brojeva

```
BigNumber operator+(BigNumber a, BigNumber b) {
  Clean(a);
  Clean(b);
  BigNumber result;
  int carry = 0;
  for (int i = 0; i < max(a.size(), b.size()); i++) {
    if (i < a.size()) {
      carry += a[i];
    }
    if (i < b.size()) {
      carry += b[i];
    }
    result.push_back(carry % BASE);
    carry /= BASE;
  }
  if (carry > 0) {
    result.push_back(carry);
  }
  return result;
}
```

Russian Peasant Multiplication

Slika: Množenje velikih brojeva

```
BigNumber SlowMultiply(BigNumber a, BigNumber b) {
  Clean(a);
  Clean(b);
  BigNumber result;
  result.assign(a.size() + b.size(), 0);
  for (int i = 0; i < a.size(); i++) {
    for (int j = 0; j < b.size(); j++) {
      result[i + j] += a[i] * b[j];
    }
  }
  int carry = 0;
  for (int i = 0; i < result.size(); i++) {
    int digit = result[i] + carry;
    result[i] = digit % BASE;
    carry = digit / BASE;
  }
  Clean(result);
  return result;
}
```

Problem?

- Složenost ovog jako jednostavnog pristupa je $\mathcal{O}(n^2)$, što je potpuno neprihvaljtivo za veeeeeelike brojeve
- Da li možemo bolje?
- Naravno! ☺

Polinomi

- Svaki broj se može predstaviti kao polinom tako što će baza brojnog sistema biti nezavisna promenljiva tog polinoma, a cifre broja koeficijenti tog polinoma.
- $123 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 = A(10)$ za $A(x) = 1 \cdot x^2 + 2 \cdot x^1 + 3 \cdot x^0$
- $456 = 4 \cdot 10^2 + 5 \cdot 10^1 + 6 \cdot 10^0 = B(10)$ za $B(x) = 4 \cdot x^2 + 5 \cdot x^1 + 6 \cdot x^0$
- $A(x)B(x) = 4 \cdot x^4 + 13 \cdot x^3 + 27 \cdot x^1 + 18 \cdot x^0$
- $A(10)B(10) = 4 \cdot 10^4 + 13 \cdot 10^3 + 27 \cdot 10^1 + 18 \cdot 10^0 = 56088 = 123 \cdot 456$

Diskretna Furijeova transformacija

- Diskretna Furijeova transformacija je zapravo preslikavanje između vektora

$$\boldsymbol{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} \longmapsto \hat{\boldsymbol{a}} = \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}$$

$$\hat{a}_j = \sum_{k=0}^{n-1} a_k \omega_n^{jk}, j = 0, \ldots, n-1$$

$$\omega_n = e^{i\frac{2\pi}{n}}$$

$$i^2 = -1$$

Diskretna Furijeova transformacija

- Diskretnu Furijeovu transformaciju možemo zapisati i u matričnom obliku

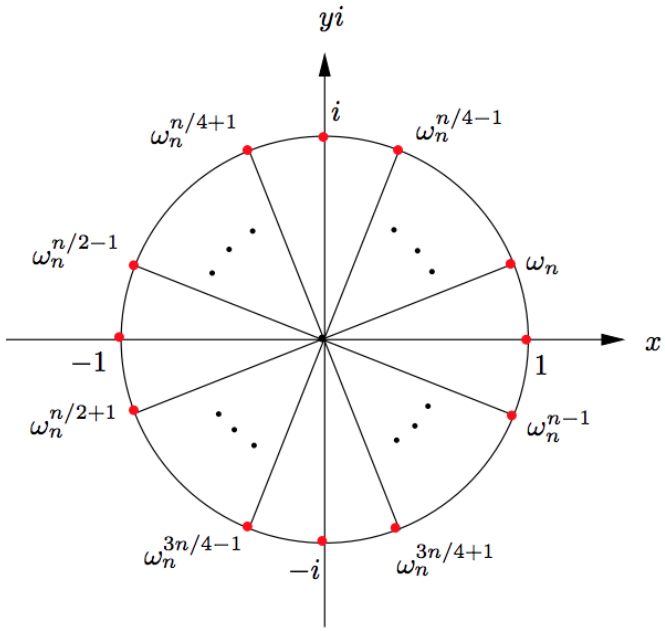
$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)^2} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \\ \vdots \\ \hat{a}_{n-1} \end{pmatrix}$$

- Ova matrica se naziva Vandermondeova matrica i obeležava se kao V_n

Diskretna Furijeova transformacija - uprošćeno

- Algoritam diskretne Furijeove transformacije se zasniva na evaluiranju polinoma $p(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1}$ u n-tim kompleksnim korenima jedinice $\omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1}$
- $\hat{a}_k = p(\omega_n^k), k = 0, 1, \ldots, n-1$
- Nadalje možemo predpostaviti da je n stepen dvojke jer uvek možemo polinom dopuniti vodećim koeficijentima koji su nule

n-ti kompleksni koreni jedinice



- $\omega_n^n = 1, \omega_n^{n+k} = \omega^k$
- $\omega_n^{\frac{n}{2}} = -1, \omega_n^{\frac{n}{2}+k} = -\omega_n^k$

DFT - algoritam

- *Divide&Conquer* strategija
- Podelimo $p(x)$ na dva polinoma $p_0(x)$ i $p_1(x)$
- $p_0(x) = a_0 + a_2 x + \cdots + a_{n-2} x^{\frac{n}{2}-1}$
- $p_1(x) = a_1 + a_3 x + \cdots + a_{n-1} x^{\frac{n}{2}-1}$
- $p(x) = p_0(x^2) + x p_1(x^2)$
- ???
- Profit!

DFT - algoritam

```
RECURSIVE-DFT(a, n)
1   if n = 1
2       then return a
3   w_n ← e^{i 2π/n}
4   w ← 1
5   a^[0] ← (a_0, a_2, ..., a_{n-2})
6   a^[1] ← (a_1, a_3, ..., a_{n-1})
7   â^[0] ← RECURSIVE-DFT(a^[0], n/2)
8   â^[1] ← RECURSIVE-DFT(a^[1], n/2)
9   for k = 0 to n/2 − 1 do
10      â_k ← â_k^[0] + ω â_k^[1]
11      â_{k+n/2} ← â_k^[0] − ω â_k^[1]
12      ω ← ω ω_n
13  return (â_0, â_1, ..., â_{n-1})
```

DFT - algoritam

- $\hat{a}_{k+\frac{n}{2}} = \hat{a}_k^{[0]} - \omega_n^k \hat{a}_k^{[1]} = \hat{a}_k^{[0]} + \omega_n^{k+\frac{n}{2}} \hat{a}_k^{[1]}$
- $\hat{a}_{k+\frac{n}{2}} = p_0(\omega_n^{2k}) + \omega_n^{k+\frac{n}{2}} p_1(\omega_n^{2k})$
- $\hat{a}_{k+\frac{n}{2}} = p_o(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} p_1(\omega_n 2k + n)$
- $\hat{a}_{k+\frac{n}{2}} = p(\omega_n^{k+\frac{n}{2}})$
- Neka je $T(n)$ složenost `Recursive-DFT` algoritma
- $T(n) = 2T(\frac{n}{2}) + \Theta(n)$
- $T(n) = \Theta(nlog_2(n))$

Inverzna diskretna Furijeova transformacija

- Da li je moguće odrediti koeficijente polinoma $p(x) = a_0 + \cdots + a_{n-1}x^{n-1}$ ukoliko znamo vrednosti tog polinoma u tačkama $\omega_n^0, \ldots, \omega_n^{n-1}$?
- $a = V_n^{-1}\hat{a}$

$$V_n^{-1} = \frac{1}{n}\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega_n^{-1} & \omega_n^{-2} & \cdots & \omega_n^{-(n-1)} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \omega_n^{-(n-1)} & \omega_n^{-2(n-1)} & \cdots & \omega_n^{-(n-1)^2} \end{pmatrix}$$

Inverzna diskretna Furijeova transformacija

- Dakle, možemo primeniti `Recursive-DFT` algoritam tako što ćemo a zameniti sa $\hat{a}$, $\hat{a}$ sa a, ω_n sa $\omega_n^{-1} = \omega_n^{n-1}$ i pomnožiti dobijeni vektor sa $\frac{1}{n}$
- (Teorema o konvoluciji): Neka su a i b vektori dužine n, gde je n stepen dvojke.
- $a \otimes b = DFT_{2n}^{-1}(DFT_{2n}(a) \cdot DFT_{2n}(b))$, gde su vektorima a i b dopisane nule sa leve strane do dužine $2n$, a $\cdot$ predstavlja skalarni proizvod vektora.

That's all folks!

- Hvala na pažnji!