Potence z racionalnimi eksponenti — 30. domača naloga

Matematika, Gimnazija Bežigrad

Profesor: prof. Vilko Domajnko
Avtor: Anton Luka Šijanec, 2. a

12. april 2021

Povzetek

Ta dokument vsebuje navodila in rešitve domače naloge snovi *Potence z racionalnimi eksponenti* pri matematiki, ki sem jih spisal sam. Kjer je bilo potrebno izbrati neke poljubne naloge, sem jih vedno izbiral naključno.

Navodilo naloge listi: *Potence z racionalnimi eksponenti* / 1, ... 6 (povsod po en primer)

1. Zapiši v obliki potence z racionalnim eksponentom:

č) $\sqrt{a^3} \cdot \sqrt{a^5} = a^4$

2. Zapiši v obliki potence z racionalnim eksponentom:

a) $\sqrt{a\sqrt{a}} = \sqrt{a \cdot a^{\frac{1}{2}}} = \sqrt{a^{\frac{3}{2}}} = \left(a^{\frac{3}{2}}\right)^{\frac{1}{2}} = a^{\frac{3}{4}}$

3. Izračunaj:

e)

$$\left(\sqrt[6]{9-4\sqrt{5}+\sqrt[3]{2-\sqrt{5}}}\right)\sqrt[3]{2+\sqrt{5}} = \left(\sqrt[6]{\left(2-\sqrt{5}\right)^2}+\sqrt[3]{2+\sqrt{5}}\right)\sqrt[3]{2+\sqrt{5}} =$$

$$= \left(\sqrt[3]{\sqrt{5}-2}+\sqrt[3]{2-\sqrt{5}}\right)\sqrt[3]{2+\sqrt{5}} = \sqrt[3]{\left(\sqrt{5}-2\right)\cdot\left(2+\sqrt{5}\right)}+\sqrt[3]{\left(2-\sqrt{5}\right)\cdot\left(2+\sqrt{5}\right)} =$$

$$= \sqrt[3]{\left(\sqrt{5}-2\right)\cdot\left(\sqrt{5}+2\right)}+\sqrt[3]{4-5} = \sqrt[3]{5-4}+\sqrt[3]{-1} = 1-1 = 0$$

4. Poenostavi in zapiši v obliki ulomka (brez negativnih eksponentov):

b) $\left(2x^{\frac{1}{5}}y^{\frac{4}{5}}\right)^{-5} = \dfrac{1}{\left(2x^{\frac{1}{5}}y^{\frac{4}{5}}\right)^5} = \dfrac{1}{32xy^4}$

5. Poenostavi:

b) $\sqrt{a^3 \cdot \sqrt[3]{\dfrac{1}{a^2}\cdot\sqrt{a}}} = \sqrt{a^3 \cdot a^{\frac{-1}{2}}} = \sqrt{a^{\frac{5}{2}}} = \left(a^{\frac{5}{2}}\right)^{\frac{1}{2}} = a^{\frac{5}{4}} = \sqrt[4]{a^5} = a\sqrt[4]{a}$

6. Poenostavi:

b) Vau, štiri korene smo spremenili v enega! Spodaj je praktična zanimivost.

$$\sqrt[3]{\frac{x^2}{x-1}}\cdot\sqrt{\frac{(x-1)^2}{x^5}}\cdot\sqrt[6]{\frac{x^{20}}{(x+1)^3}}\cdot\sqrt{\frac{x+1}{x^3}} = \sqrt[6]{\left(\frac{x^2}{x-1}\right)^2\cdot\left(\frac{(x-1)^2}{x^5}\right)^3\cdot\frac{x^{20}}{(x+1)^3\cdot\left(\frac{x+1}{x^3}\right)^3}} =$$

$$= \sqrt[6]{\left(\frac{x^2}{x-1}\right)^2\cdot\left(\frac{(x-1)^2}{x^5}\right)^3\cdot\frac{x^{20}}{(x+1)^3}\cdot\left(\frac{x+1}{x^3}\right)^3} = \sqrt[6]{\frac{x^4}{(x-1)^2}\cdot\frac{(x-1)^6}{x^{15}}\cdot\frac{x^{20}}{(x+1)^3}\cdot\frac{(x+1)^3}{x^9}} =$$

$$= \sqrt[6]{(x-1)^4\cdot x^5\cdot\frac{1}{x^5}} = \sqrt[6]{(x-1)^4} = \sqrt[3]{(x-1)^2}$$

Zanimivost Želim preveriti, kako taka optimizacija vpliva v resničnem življenju. Recimo, da se ta račun v računalniku izvede desetmilijonkrat v sklopu nekega procesa. Koliko procesorskega časa smo pridobili s tem, da smo štiri korene spremenili v enega? Kljub vsemu smo na koncu optimizacije dobili šesti koren, kar nikakor ni mačji kašelj za X86 arhitekturo. Najlažje je računalniku računati korene kvadratnih eksponentov (2, 4, 16, ...).

```c
/* gcc kompleksnost.c -pedantic -Wall -Wextra -g -okompleksnost -lm -O0 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define KOLIKOKRAT 10000001
#define nrx(x, y) powl((x), 1.0 / (y)) /* nti koren izpeljemo iz potence na 1/eks */
int main (void) { /* za kvadratne in kubicne korene uporabimo sqrt in cbrt */
  long double rezultat = 0; /* da prevajalnik ne bi optimiziral */
  long double rezultat2 = 0;
  clock_t cas_zacetek, cas_konec, cas2_konec;
  cas_zacetek = clock();
  for (unsigned long long int x = 2; x <= KOLIKOKRAT; x++)
    rezultat += cbrtl(powl(x, 2)/(x-1))*sqrtl(powl(x-1, 2)/powl(x, 5))*nrx(powl(x, 20)/
    powl(x+1, 3), 6)*sqrtl((x+1)/powl(x, 3));
  cas_konec = clock();
  for (unsigned long long int x = 2; x <= KOLIKOKRAT; x++)
    rezultat2 += nrx(pow(x-1, 2), 3);
  cas2_konec = clock();
  fprintf(stdout, "neoptimizirana enacba je za 10e6 iteracij porabila %lf sekund,
    optimizirana pa %lf\n"
      "rezultat = %Lf, rezultat2 = %Lf\n",
      ((double) (cas_konec-cas_zacetek))/CLOCKS_PER_SEC,
      ((double) (cas2_konec-cas_konec))/CLOCKS_PER_SEC,
      rezultat,
      rezultat2
    );
  return 0;
}
```

```
a@upor:[0]$ ./kompleksnost
neoptimizirana enacba je za 10e6 iteracij porabila 15.330899 sekund, optimizirana pa 4.157521
rezultat = 278495353224.555294, rezultat2 = 278495353224.555814
a@upor:[0]$
```

Ojej! Mislim, da je razlika očitna. Neoptimizirana enačba je porabila 10 sekund več. Optimizacija je torej zelo pomembna. Ampak zakaj tega namesto nas ne delajo računalniki? V GNU prevajalniku C jezika sem avtomatilno optimizacijo nastavil na nivo `0`, kar pomeni, da prevajalnik ne dela optimizacij, v navodilih za uporabo pa sem, poleg drugih, zasledil še eno opcijo, `fast`. Opcija `fast` je sicer eksperimentalna in nepriporočena, vendar dela čudeže! Če program prevedem z optimizacijskim nivojem `fast`, bo računanje neoptimizirane enačbe in optimizirane trajalo približno enako dolgo.

```
a@upor:[0]$ gcc kompleksnost.c -pedantic -Wall -Wextra -g -okompleksnost -lm -Ofast
a@upor:[0]$ ./kompleksnost
neoptimizirana enacba je za 10e6 iteracij porabila 5.326709 sekund, optimizirana pa 4.017244
rezultat = 278495353224.555294, rezultat2 = 278495353224.555814
a@upor:[0]$
```

1 Zaključek

Ta dokument je informativne narave in se lahko še spreminja. Najnovejša različica, torej PDFji in LaTeX[1] izvorna koda, zgodovina sprememb in prejšnje različice, je na voljo v mojem šolskem Git repozitoriju na `https://git.sijanec.eu/sijanec/sola-gimb-2` v mapi /mat/domace_naloge/30/. Povezava za ogled zadnje različice tega dokumenta v PDF obliki je `http://razor.arnes.si/~asija3/files/sola/gimb/2/mat/domace_naloge/30/dokument.pdf` in/ali `https://git.sijanec.eu/sijanec/sola-gimb-2/raw/branch/master/mat/domace_naloge/30/dokument.pdf`.

[1] Za izdelavo dokumenta potrebujete `TeXLive 2020`.

Razhroščevalne informacije

Te informacije so generirane, ker je omogočeno razhroščevanje. Pred objavo dokumenta izklopite razhroščevanje. To naredite tako, da nastavite ukaz `razhroscevanje` na 0 v začetku dokumenta.

Grafi imajo natančnost 100 točk na graf.

Konec generiranja dokumenta: 12. april 2021 ob 12:10:24[2]

Dokument se je generiral 3 s.

[2]To ne nakazuje dejanskega časa, ko je bil dokument napisan, temveč čas, ko je bi dokument generiran v PDF/DVI obliko. Isto velja za datum v glavi dokumenta. Če berete direktno iz LaTeX datoteke, bo to vedno današnji datum.