INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Číslo projektu	CZ.1 .07/1.1 .36/02.0066
Autor	Ladislav Kašpárek
Předmět	Programování a vývoj aplikací
Téma	Test na prvočíselnost
Metodický pokyn	výkladový text s ukázkami

Test na prvočíselnost

Problém

Vstup: Máte zadáno kladné celé číslo.
Úkol: Máte najít algoritmus, který řekne zda toto číslo je, nebo není prvočíslo.

Co je to prvočíslo?

Prvočíslo je přirozené číslo, které je beze zbytku dělitelné právě dvěma různými přirozenými čísly, a to číslem jedna a sebou samým. Čili číslo 1, není prvočíslo.

Co nás zajímá?

U algoritmů nás zajímá jejich složitost. Čili jak roste výpočetní čas (doba trvání algoritmu), když zvyšuji požadavky (rozsah vstupních dat).
Složitost řazení se zapisuje jako funkce, která nám říká, kolik musíme udělat elementárních operací nad zadanými daty, abychom mohli uživateli s jistotou říci, že zadané číslo je nebo není prvočíslo.

Hrubá síla

Základní myšlenka:

Otestuji zda zadané číslo N je dělitelné jakýmkoliv číslem v rozsahu 2 až N-1. Pokud najdu takové číslo, pak N není prvočíslo.

Kód:

```
public static boolean bruteForce(int n) {
    if (n == 1) {
        return false;
    }
```

```
    for (int i = 2; i < n; i++) {
        if ((n % i) == 0) {
            return false;
        }
    }
    return true;
}
```

Poloviční hrubá síla

Základní myšlenka :

Nemá smysl testovat dělitelnost celým rozsahem.
Stačí si uvědomit že nejnižší číslo, které může dělit N je 2, takže naopak nejvyšší číslo, které může N dělit je $\frac{N}{2}$.

Kód:

```
public static boolean bruteForceBetter(int n) {
    if (n == 1) {
        return false;
    }
    for (int i = 2; i <= n / 2; i++) {
        if ((n % i) == 0) {
            return false;
        }
    }
    return true;
}
```

Odmocninná hrubá síla

Základní myšlenka :

Číslo 2, je jedinné sudé prvočíslo - můžeme ho vyřadit bokem.
Dále nemá smysl testovat dělitelnost celým rozsahem.
Testovat stačí pouze do odmocniny, protože pokud N je složené číslo, můžeme psát: n=a * b, pokud jsou a, b kladná celá čísla větší jak 1.

Pokud by nestačilo testovat do odmocniny, znamenalo by to, že a musí být větší jak $\sqrt{N}$ a současně b by muselo být větší jak $\sqrt{N}$, což je spor.

Kód:

```
public static boolean bruteForceBetter2(int n) {
    if (n == 1) {
        return false;
    } else if(n == 2) {
        return true;
    }
    int odm = (int) Math.sqrt(n);
    for (int i = 2; i <= odm; i++) {
        if ((n % i) == 0) {
```

```
            return false;
        }
    }
    return true;
}
```

Poloviční odmocninná hrubá síla

Základní myšlenka :

Číslo 2, je jedinné sudé prvočíslo - můžeme ho vyřadit bokem.
Jakékoliv jiné sudé číslo, není prvočíslo.
Dále nemá smysl testovat dělitelnost celým rozsahem.
Testovat stačí pouze do odmocniny, protože pokud N je složené číslo, můžeme psát: n=a * b, pokud jsou a, b kladná celá čísla větší jak 1.

Pokud by nestačilo testovat do odmocniny, znamenalo by to, že a musí být větší jak $\sqrt{N}$ a současně b by muselo být větší jak $\sqrt{N}$, což je spor.

Stačí testovat dělitelnost pouze lichými čísly, sudá už jsme vyřadili výše.

Kód:

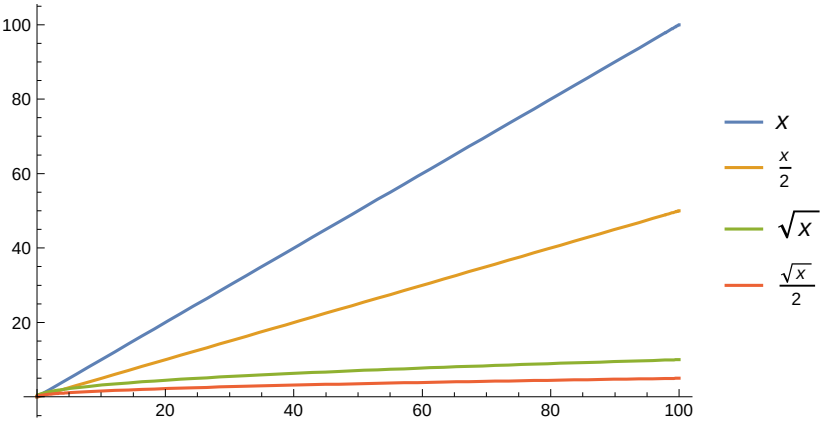
```
public static boolean bruteForceBetter3(int n) {
    if(n == 1) {
        return false;
    } else if(n == 2) {
        return true;
    } else if((n % 2) == 0) {
        return false;
    }
    int odm = (int) Math.sqrt(n);
    for (int i = 3; i <= odm; i = i + 2) {
        if ((n % i) == 0) {
            return false;
        }
    }
    return true;
}
```

Závěrečné porovnání

Porovnejme složitosti všech přístupů :

In[1]:=
```
Plot[{x, x/2, Sqrt[x], Sqrt[x]/2}, {x, 0, 100}, PlotLegends → "Expressions"]
```

Out[1]=



Závěrečné porovnání

Jak je vidět ze závěrečného grafu, složitost jednotlivých postupů klesá. V prvních dvou případech sice ne asymptoticky, ale třetí a čtvrtý způsob má již asymptoticky nižší složitost než oba předchozí.

Zdroje

http://reference.wolfram.com/mathematica/ref/Plot.html