Příklad

Je dán následující program:

```
int n = …;
double result = 0;
int index = n*n;
while (index>0) {
      result += someFunction(index, n);
      result += someFunction(index, n+1);
      index--;
}
```

Zapište funkci $f(n)$ popisující dobu výpočtu programu v závislosti na proměnné n. Započtěte **pouze** čas nutný pro **vyhodnocení funkce** `someFunction(…)`, která trvá **18ms**. Určete, do jaké množiny Θ patří funkce $f(n)$ a proveďte **důkaz**.

Řešení

- Uvnitř smyčky se funkce volá 2x
- Samotná smyčka provede index cyklů
- Jelikož index = n*n, je celkový počet volání k(n) = 2n*n
- Jelikož každé volání trvá 18ms, je celkový čas f(n)=18*2*n*n ms

Pokusíme se o důkaz O(n)

36*n*n < c*n   dělíme nerovnici n

36*n < c

- n roste nade všechny meze, žádné takové c pro které by nerovnice platila neexistuje, funkce tedy do O(n) nepatří
- tento důkaz v rámci řešení uvádět není třeba, je zde jen pro ilustraci co by nám vyšlo, pokud bychom složitost odhadli špatně

Pokusíme se o důkaz O(n*n)

36*n*n < c*n   dělíme nerovnici n*n

36 < c

⇨ pro c>36 nerovnice platí, na n0 nezáleží. Zvolíme tedy c=37, n0=1, důkaz O(n*n) je hotov

Pokusíme se o důkaz Ω(n*n)

36*n*n > c*n*n   dělíme rovnici n*n

36 > c

⇨ pro c<36 nerovnice platí, na n0 nezáleží. Zvolíme tedy c=35 (c musí být hlavně kladné), n0=1, důkaz Ω(n*n) je hotov

Funkce patří do O(n*n) i Ω(n*n), proto patří i do Θ(n*n)

Příklad

Je dán následující program:

```
int n = …;
double result = 0;
int index = 0;
while(index<n) {
      for (int i = 0;i<n*1000;i++) {
            result *= someFunction(i,index);
            result *= someFunction(index,i);
      }
      index++;
}
```

Zapište funkci $f(n)$ popisující dobu výpočtu programu v závislosti na proměnné n. Započtěte **pouze** čas nutný pro **vyhodnocení funkce** `someFunction(…)`, která trvá **3ms**. Určete, do jaké množiny Θ patří funkce $f(n)$ a proveďte **důkaz**.

Řešení

- Uvnitř smyčky for se funkce volá 2x
- Smyčka provede 1000n cyklů, celkem tedy 2000n volání
- Smyčka while provede n cyklů, v každém cyklu se provede 2000n volání
- Celkový počet volání k(n) = n*2000n
- Jelikož každé volání trvá 3ms, je celkový čas f(n)=3*n*2000n = 6000n*n ms

Pokusíme se o důkaz Ω(n*n*n)

6000*n*n > c*n*n*n   dělíme nerovnici n*n*n

6000/n > c

- 6000/n klesá s n k nule, žádné kladné c pro které by nerovnice platila neexistuje, funkce tedy do Ω(n*n*n) nepatří
- tento důkaz v rámci řešení uvádět není třeba, je zde jen pro ilustraci co by vyšlo, pokud bychom složitost odhadli špatně

Pokusíme se o důkaz O(n*n)

6000*n*n < c*n   dělíme nerovnici n*n

6000 < c

⇨ pro c>6000 nerovnice platí, na n0 nezáleží. Zvolíme tedy c=6003, n0=1, důkaz O(n*n) je hotov

Pokusíme se o důkaz Ω(n*n)

6000*n*n > c*n*n   dělíme rovnici n*n

6000 > c

⇨ pro c<6000 nerovnice platí, na n0 nezáleží. Zvolíme tedy c=0.042 (c musí být hlavně kladné), n0=1, důkaz Ω(n*n) je hotov

Funkce patří do O(n*n) i Ω(n*n), proto patří i do Θ(n*n)

Příklad

Je dán následující program:

```
double result = 0;
for (int i = 0;i<n*100;i++)
      for (int j = 0;j<n*10;j++)
            for (int k = n;k>=0;k--)
                  result += someFunction(i,j,k);
```

Zapište funkci $f(n)$ popisující dobu výpočtu programu v závislosti na proměnné n. Započtěte **pouze** čas nutný pro **vyhodnocení funkce** `someFunction(…)`, která trvá **12ms**. Určete, do jaké množiny Θ patří funkce $f(n)$ a proveďte **důkaz**.

Řešení

- Uvnitř poslední smyčky for se funkce volá 1x
- Poslední smyčka provede n+1 cyklů, celkem tedy n+1 volání
- Prostřední smyčka for provede n*10 cyklů, a v každém cyklu se provede n+1 volání, celkem tedy prostřední smyčka představuje n*10*(n+1) volání
- Vnější smyčka provede n*100 cyklů, a v každém cyklu se provede n*10*(n+1) volání, celkem tedy vnější smyčka představuje k(n) = n*100*n*10*(n+1) = 1000*n*n*n + 1000*n*n volání
- Jelikož každé volání trvá 12ms, je celkový čas f(n)=12000*n*n*n + 12000*n*n ms

Pokusíme se o důkaz O(n*n)

12000*n*n*n + 12000*n*n < c*n*n   dělíme nerovnici n*n

12000*n + 12000 < c

- 12000*n roste nade všechny meze, žádné kladné c pro které by nerovnice platila neexistuje, funkce tedy do O(n*n) nepatří
- tento důkaz v rámci řešení uvádět není třeba, je zde jen pro ilustraci co by vyšlo, pokud bychom složitost odhadli špatně

Pokusíme se o důkaz O(n*n*n)

12000*n*n*n + 12000*n*n< c*n*n*n   dělíme nerovnici n*n*n

12000 + 12000/n < c

- člen 12000/n se s rostoucím n monotóně blíží k nule
- např. pro n=6000 je levá část nerovnosti rovna 12002, pro n>6000 bude levá část ještě menší
- můžeme tedy volit n0=6000, c = 12003, důkaz O(n*n*n) je hotov

Pokusíme se o důkaz Ω(n*n*n)

12000*n*n*n + 12000*n*n > c*n*n*n dělíme nerovnici n*n*n

12000 + 12000/n > c

- člen 12000/n se s rostoucím n blíží k nule, levá část nerovnosti je tedy vždy větší než 12000
- můžeme tedy volit c = 11999, na n0 nezáleží, zvolíme třeba 42, důkaz Ω (n*n*n) je hotov

Funkce patří do $O(n*n*n)$ i $\Omega(n*n*n)$, proto patří i do $\Theta(n*n*n)$