

Algoritmus má složitost:

- Statickou – jak velký je jeho zdrojový kód v určitém jazyce (zpravidla nezajímavá),
- dynamickou – jak dlouho běží.
- implicitně máme na mysli složitost **dynamickou!**

Definition

Označuje-li hodnota n délku vstupu, pak složitostí algoritmu \mathcal{A} nazveme funkci f takovou, že pro každý vstup délky n algoritmus \mathcal{A} zastaví po nejvýše $f(n)$ elementárních krocích.

Definition

Označuje-li hodnota n délku vstupu, pak složitostí algoritmu \mathcal{A} nazveme funkci f takovou, že pro každý vstup délky n algoritmus \mathcal{A} zastaví po nejvýše $f(n)$ elementárních krocích.

- Rozklad na prvočísla: Zdánlivě kvadratický vůči dekadickému zápisu, ve skutečnosti lineární.
- Převod do dvojkové soustavy: lineární v délce binárního zápisu.
- Pozor! Složitost počítáme vůči délce zápisu čísla!
- Mínótaurus v bludišti – lineární v počtu hran grafu.
- Stabilní párování (volenkový algoritmus) – nejvýše kvadratické v počtu pánů (resp. dam).

Jelikož počítat přesnou složitost je obtížné a jelikož není zajímavé, zda pro daný vstup budeme čekat hodinu nebo dvě, ale že algoritmus doběhne, používáme přibližné odhady složitosti:

Definition

Řekneme, že:

- $f \in O(g)$, jestliže existuje c a x_0 tak, že pro každé $x > x_0$ platí $f(x) \leq cg(x)$.
- $f \in \Omega(g)$, jestliže existuje $c > 0$ a x_0 tak, že pro každé $x > x_0$ platí $f(x) \geq cg(x)$.
- $f \in \Theta(g)$, pokud $f \in O(g)$ a současně $f \in \Omega(g)$.

Příklady:

- Je $n \in O(n^2)$?
- Je $n^2 \in O(n)$?
- Je $3n^5 + 2n^3 + 1000 \in \Theta(n^5)$?
- Je $n^{1000} \in O(2^n)$?
- Je $2^n \in n^{2000}$?

Cvičení s kartami aneb jak rychle roste exponenciála.

- Složitost v nejlepším případě,
- složitost v průměrném případě – průměrný počet kroků algoritmu přes jednotlivé instance dané délky,
- amortizovaná složitost – průměrný počet kroků při potenciálně nekonečně mnoha po sobě jdoucích operacích – uvažujeme nejhorší možnou posloupnost,
- složitost problému – řeší, jak efektivní algoritmus existovat může a jak efektivní algoritmus už existovat nemůže.

- Neseříděné pole \Rightarrow snadný horní i dolní odhad (prohledat celé pole),
- seříděné pole:
 - unární hledání (listuj jako knihou, dokud nenajdeš),
 - binární hledání (začni uprostřed, pul intervaly),
 - kvadratické vyhledávání, zobecněné kvadratické vyhledávání...

- Snadný algoritmus, snadná analýza, složitost:
- $\Theta(n)$.

- Jaká je složitost algoritmu? Kdy je potřeba udělat o krok navíc?
- $\Theta \log n$.

Hledání největší nulové podmatice:

Naivní algoritmus: $O(n^6)$

Kdo má lepší nápady?

Definice funkcí a procedur

- Mnoho operací provozujeme opakovaně, proto je hloupé programovat je při každém použití znovu.
- Procedury a funkce představují možnost je naprogramovat jednou a použít mnohokrát.
- Procedura: Součást programu schopná přijmout parametry a zpracovat.
- Funkce: Součást programu schopná přijmout parametry, zpracovat a vrátit výsledek.
- Příklady: Přejdi ulici, vypiš hlášení; dojeď vlakem někam, spočítej faktoriál.

function nazev(argument :typ;...):navratovy_typ

- Zahajujeme klíčovým slovem function, následuje název funkce,
- argumenty (parametry) zapisujeme do kulatých závorek jako bychom definovali proměnné.
- Jednotlivé parametry oddělujeme středníkem,
- za dvojtečkou pak uvedeme návratový typ dotyčné funkce.
- Návratová hodnota se přiřadí do proměnné jmenující se stejně jako příslušná funkce.

Příklad

```
program x;  
var a:integer;  
  
function secti(a:integer; b:integer):integer;  
begin  
    secti:=a+b;  
end;  
  
begin  
    a:=secti(5,10);  
    writeln(a);  
end.
```