Obsah

19 Algoritmus

- konečná sekvence podrobně definovaných instrukcí řešící určitý problém
- výpočty, zpracování dat, automatické rozhodování
- možno použít stejný algoritmus s různými vstupními daty

19.1 Vlastnosti algoritmu

- **správnost** – výsledek algoritmu musí být správný
- **resultativnost** – algoritmu v konečném čase dosáhne řešení a skončí
- **determinovanost** – v každém kroku jednoznačně určeno pokračování algoritmu
- **hromadnost** – možno algoritmus použít na řešení obecné úlohy
- **opakovatelnost** – se stejnými vstupními daty má algoritmus stejný výsledek

19.2 Zápis algoritmu

- slovní – slovní popis postupu v mluvené řeči
- grafický diagram – vývojový diagram, kroky zakresleny pomocí značek
- matematický – zápis pomocí veličin a rovnic
- v programovacím jazyku – vyjádření v programovacím jazyce pomocí funkcí, keywords, proměnných…

19.3 Rozdělení algoritmů

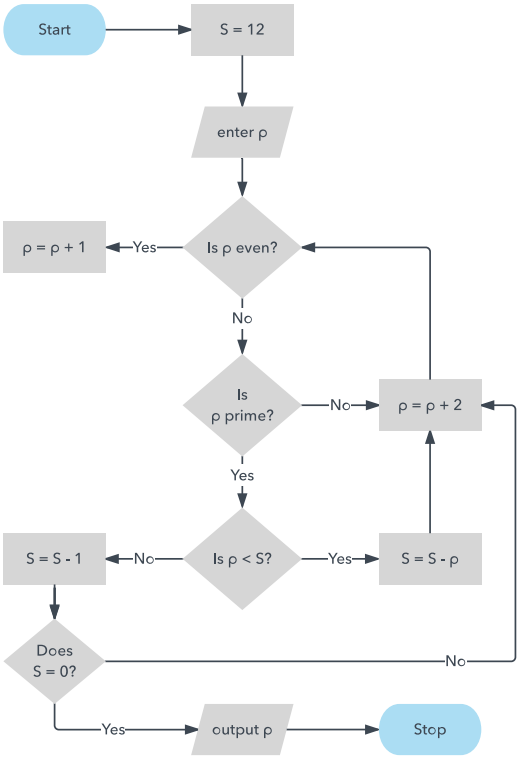
19.3.1 Podle implementace

Rekurzivní / iterativní

- rekurzivní
  - volá sám sebe dokud není splněna podmínka
  - rozdělení problému na části a postupné řešení
  - všechna potřebná data specifikována jako parametry funkce
- iterativní
  - opakování kódu v cyklech
  - použití externích datových struktur pro přenos dat mezi cykly
- každý problém možno přepsat z rekurze do iterace a naopak

Logický

- kontrolovaná logická dedukce
- specifikace komponentů jako axiomů, ty následně použity v dedukci



Obr. 19.1: Grafický diagram algoritmu

Sériové a Paralelní

- sériový alg.
  - spouští instrukce postupně jednu po druhé
- paralelní
  - umožněno spuštění instrukcí na více jádrech/procesorech najednou
  - rozdělení problému na symetrické či nesymetrické podproblémy
  - možno použít pouze při zpracování nezávislých dat
  - využití distribuovaných systémů

Deterministické a nedeterministické

- deterministické – přesné rozhodnutí v každém kroku
- nedeterministické – řešení problému hádáním, zavedení náhody

Přesné a přibližné (approximate)

- přesné alg. – řešení problému najdou přesně
- přibližné
  - řešení problému pouze aproximují (deterministicky či náhodně)
  - řešení problému neřešitelných přesně či značně rychlejší řešení
  - Monte Carlo, Runge Kutta

19.3.2 Podle filozofie designu

Brute-force

- naivní metoda řešení problému
- vyzkoušení každé možnosti a nalezení té nejlepší

Divide and conquer (rozděl a panuj)

- opakované rozdělení problému do jednoho či více menších problému, dokud nejsou problémy dostatečně malé na vyřešení
- typicky rekurzivní
- merge sort, binary search (decrease and conquer)

Vyhledávání a výčty

- problém vyjádřen grafem/stromem
- využití při hledání cest nebo např. hledání možnosti při hraní šach
- search algorithms…
- backtracking
  - více řešení vytvářeno postupně
  - zahození nevyhovujících
  - vrácení k poslední validní hodnotě

Algoritmy s náhodou

- některé rozhodnutí (pseudo)náhodně
- hledání přibližných řešení
- Monte Carlo, Las Vegas algoritmus

Redukce komplexity

- transformace složitého problému na více známý, řešitelný problém
- cílem najít algoritmus, jehož složitost není přesahována složitostí algoritmu řešící známý problém
- např. medián neseřazeného seznamu
  1. seřazení seznamu – expensive
  2. nalezení mediánu – cheap

19.3.3 Optimalizační problémy

Lineární programování

- nalezení maxima/minima lineární funkce
- určitý počet proměnných na množině popsané soustavou lineárních nerovnic

Dynamické programování

- problém složen z podproblémů, které se překrývají (mají stejná řešení)
- zapamatování výsledků
- vyhnutí se přepočítávání řešení, které již byly vypočítány
- např. floyd-washallův algoritmus vyhledávání nejkratší vzdálenosti v grafu

The greedy method

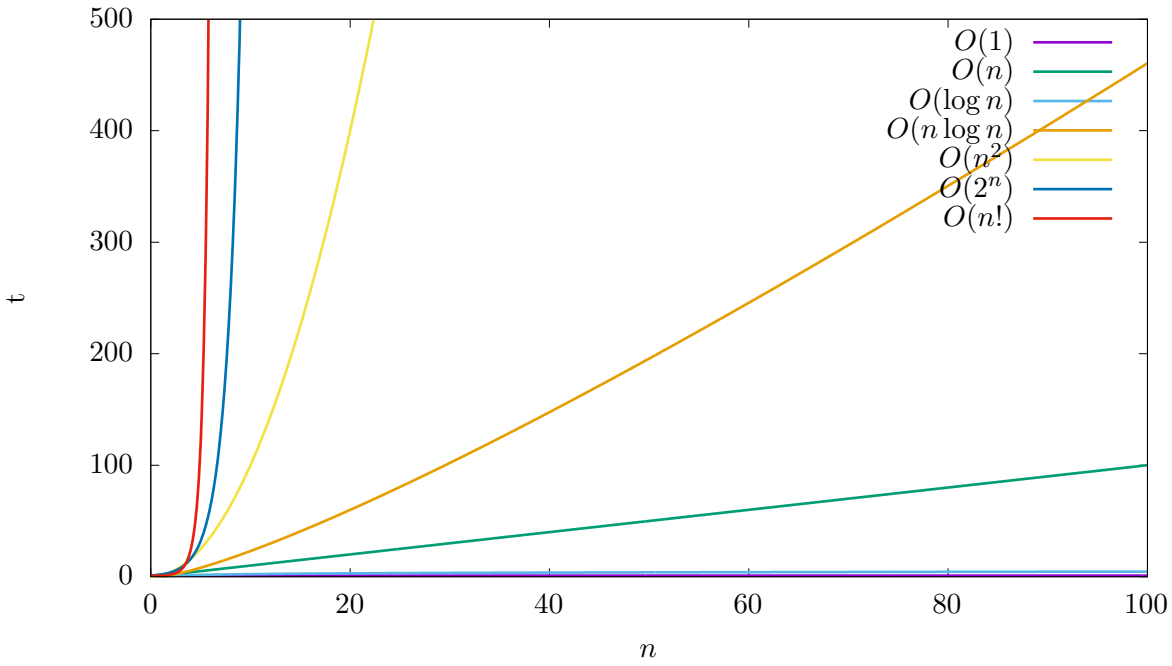
- podobný dynamickému programování, ovšem pracuje již se získaným výsledkem
- snaha vylepšit řešení pomocí malý změn
- možno najít optimální řešení nebo taky jen lokální minimum
- Huffmanův strom, Kruskal, Prim, Sollin

Heuristická metoda

- nalezení řešení blízké optimálnímu řešení
- v případech, že hledání optimálního řešení není praktické
- dostávání se blíže a blíže výsledku
- při nekonečném čase nalezení optimálního řešení
- local search, tabu search, genetické algoritmy

19.4 Komplexnost algoritmu

- vlastnost udávající výpočetní náročnost
- především sledován čas potřebný na spuštění, někdy i velikost potřebné paměti
- vyjádřeno pomocí Big-O notace
  - vyjádření nejhorší/nejdelší kalkulace
  - zápis pomocí matematické funkce; n – počet prvků
  - brán v potaz pouze nejrychleji rostoucí komponent
    - z předpokladu $n \to \infty$
    - $O(n^2 + n) = O(n^2)$
- související veličiny
  - big-omega $\Omega(n)$ – nejlepší případ
  - bit-theta $\Theta(n)$ – průměrný případ



Obr. 19.2: Porovnání průběhu jednotlivých funkcí

19.4.1 Příklady komplexnosti

- konstantní čas – $O(1)$ – získání array elementu, hledání v hashmapě
- logaritmický čas – $O(\log n)$ – binary search
- lineární čas – $O(n)$ – projití všech prvků v poli
- polynomiální čas – $O(n^2), O(n^3)$ – bubble sort ($O(n^2)$)
- exponenciální čas – $O(x^n)$ – brute-force search

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

Obr. 19.3: Porovnání složitosti řadících algoritmů