Základní algoritmy

Euklidův algoritmus
Vlastnosti algoritmu
Erastothenovo síto
Algoritmy třídění
Algoritmy vyhledávání

Euklidův algoritmus

- *Eukleidés též **Euklides** nebo Euklid (asi 325 př. n. l. – asi 260 př. n. l) byl řecký matematik a geometr, většinu života strávil v Alexandrii v Egyptě*
- tento algoritmus je uveden v Euklidově díle ***Základy***
- přibližně do roku 1950 bylo slovo algoritmus spojováno nejčastěji právě s Euklidovým algoritmem
- zjištění největšího společného dělitele dvou čísel
- *popis algoritmu:*
  - *jsou dána dvě kladná celá čísla m a n, platí, že m > n.*
  1. ***Nalezení zbytku**: vydělte m číslem n a nechť z je zbytek*
  2. ***Je zbytek roven nule?** Pokud z = 0, pak n je největší společný dělitel a algoritmus končí.*
  3. ***Redukce:** přiřaďte n do m, z do n, a opakujte od kroku 1.*

Euklidův algoritmus

- při zahájení algoritmu obsahují proměnné **m** a **n** původně zadaná čísla,
- ale při skončení algoritmu tomu tak obvykle není
- ve třetím kroku algoritmu je důležité pořadí akcí, protože jedna z hodnot se ztratí

Co musí splňovat algoritmus

1. **Konečnost**: algoritmus musí vždy po konečném počtu kroků skončit. (např. Euklidův algoritmus vytváří klesající posloupnost kladných celých čísel a každá taková posloupnost musí jednou skončit)
2. **Určitost**: každý krok algoritmu musí být přesně definován. (např. u Euklidova algoritmu musí být jasné, co znamená zbytek po dělení)
3. **Vstup**: algoritmus musí mít definována vstupní data. (např. Euklidův algoritmus potřebuje na vstupu dvě kladná celá čísla)
4. **Výstup**: výstupní veličiny mají zadaný vztah ke vstupům. (např. u Euklidův algoritmus je výstupem největší společný dělitel pro zadaná dvě vstupní kladná čísla)
5. **Efektivita**: všechny operace musí být v rozumné míře jednoduché a skončit v rozumném čase.

Erastothenovo síto

- **Eratosthenés** z Kyrény (před 276/272 – 194 př. n. l. v Alexandrii) byl matematik, astronom a byl zřejmě největším geografem antického Řecka. Působil též jako správce alexandrijské knihovny. Věnoval se také literární činnosti jako básník.
- Eratosthenovo síto je jednoduchý algoritmus pro nalezení všech prvočísel menších než zadaná horní mez.

Erastothenovo síto

- Algoritmus funguje „prosíváním“ seznamu čísel:
  - na počátku seznam obsahuje všechna čísla v daném rozsahu (2, 3, 4, …, zadané maximum),
  - poté se opakovaně první číslo ze seznamu vyjme, toto číslo je prvočíslem;
  - ze seznamu se pak odstraní všechny násobky tohoto čísla (což jsou čísla složená)
  - tak se pokračuje do doby, než je ze seznamu odstraněno poslední číslo (nebo ve chvíli, kdy je jako prvočíslo označeno číslo vyšší než odmocnina nejvyššího čísla – v takové chvíli už všechna zbývající čísla jsou nutně prvočísly)
  - (časová složitost tohoto algoritmu je O(N.log(log N)), kde N je horní mez rozsahu)

Erastothenovo síto

- Pro nalezení prvočísel mezi prvními 20 čísly:

Krok 1: Seznam obsahuje všechna čísla v rozsahu 2–20:

- Seznam: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Krok 2: Odebereme první číslo ze seznamu (2) a označíme ho jako prvočíslo:

- Známá prvočísla: 2
- Seznam: 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Krok 3: Odebereme ze seznamu všechny násobky právě odebraného prvočísla (4, 6, 8, 10, …):

- Známá prvočísla: 2
- Seznam: 3 5 7 9 11 13 15 17 19

Krok 4: Pokračujeme opět krokem 2, dokud zbývají nějaká čísla (první číslo v seznamu a také prvočíslo je tentokrát 3):

- Známá prvočísla: 2 3
- Seznam: 5 7 11 13 17 19

....

....

- Výsledný seznam prvočísel v rozsahu 2–20: 2, 3, 5, 7, 11, 13, 17, 19.

Algoritmy třídění

- Select Sort
- Insert Sort
- Bubble Sort
- PF Sort
- Heap Sort
- Shell Sort
- Quick Sort
- Merge Sort

SelectSort

- Select Sort (třídění přímým výběrem)
- výběr mezního prvku (maximum nebo minimum) z tříděné posloupnosti a jeho záměna s prvním (posledním) prvkem (Tím dojde k **rozdělení posloupnosti na dvě části**. Setříděná část obsahuje pouze jeden prvek, nesetříděná n-1.)
- v dalším kroku máme pole o (n-1) prvcích a opakujeme totéž
- takto postupujeme až do úplného seřazení posloupnosti (Získaná posloupnost bude seřazená vzestupně v případě, že budeme vybírat z nesetříděné posloupnosti nejmenší prvek a zařadíme ho na začátek posloupnosti, a sestupně v případě, že vybereme největší prvek z nesetříděné posloupnosti a zařadíme ho na začátek.)

SelectSort

- Při tomto řazení se vlastně setříděná část pole postupně zvětšuje a nesetříděná část naopak zmenšuje.
- Algoritmus končí v případě, že nesetříděná část již neobsahuje žádný prvek.
- Tato metoda je vnitřní, přímá a jednoduchá.
- Nevýhodou tohoto algoritmu je, že jestliže se na vstupu nachází již částečně setříděná posloupnost, algoritmus to nerozlišuje a proběhne v maximálním počtu koků.
- Složitost algoritmu SelectSort je $O(n^2)$.

Nejdříve vybereme z celé posloupnosti nejmenší číslo tj. 4 a vyměníme ho s prvním prvkem tj. 15. Poté vybereme nejmenší číslo z posloupnosti o jeden prvek menší tj. prvek 9 z šesti čísel. A vyměníme ho s druhým číslem setříděné části posloupnosti tj. s 15. Takto postupujeme až do úplného setřídění posloupnosti.

15	4	28	9	51	21	14
4	15	28	9	51	21	14
4	9	28	15	51	21	14
4	9	14	15	51	21	28
4	9	14	15	51	21	28
4	9	14	15	21	51	28
4	9	14	15	21	28	51
4	9	14	15	21	28	51

BubbleSort

- bublinkové třídění
- třídění výměnami:
  - Třídění je založeno na následujícím principu: postupně se systematicky porovnávají dvojice sousedních prvků a vyměňují se spolu vždy, když menší číslo následuje po větším.
  - Tak vlastně maximální prvek **probublá** na správné místo na konci posloupnosti.
  - Výpočet končí, neprovedla-li se při průchodu posloupností žádná změna. Je to pomalý a jednoduchý algoritmus ze skupiny vnitřní přímé řazení.

Bubble sort

Popis algoritmu v krocích:

1. posloupnost rozdělíme na dvě části, setříděnou a nesetříděnou, na začátku je nesetříděná část prázdná,
2. postupně porovnáváme všechny sousední prvky v nesetříděné části a pokud nejsou v požadovaném pořadí, prohodíme je,
3. krok 2 opakujeme tak dlouho, dokud nesetříděná část obsahuje více než jeden prvek, jinak algoritmus končí

Složitost algoritmu BubbleSort je $O(n^2)$.

Při prvním průchodu posloupností se posloupnost prochází od druhého prvku. Druhý prvek tj. 4 se porovná s prvkem o jeden index menším, tj. s prvkem 15, a protože nejsou ve správném pořadí prohodí se. Dále se porovná číslo 28 s číslem o jeden index menším, tj. s 15. Tato čísla jsou ve správném pořadí, nedochází k žádné výměně. Poté se porovná prvek 9 s číslem 28. Tyto prvky nejsou ve správném pořadí, proto se prohodí. Pak se porovná číslo 51 s číslem 21 a k žádné výměně nedojde. Dále se porovná číslo 21 a 51, čísla se vymění. Nakonec se číslo 51 porovná s číslem 14 a obě čísla se prohodí. Tím největší číslo 51 probublalo na konec posloupnosti.

15	4	28	9	51	21	14
4	15					
4	15	28				
4	15	9	28			
4	15	9	28	51		
4	15	9	28	21	51	
4	15	9	28	21	14	51

Při druhém průchodu posloupností se opět začíná druhým prvkem, ale končí se prvkem s indexem o jedničku menším než v předchozím cyklu. Nejprve se porovná 15 se 4 (nic se nezmění), dále se porovná 9 s 15 a prvky se prohodí. Pak se porovná 28 s 15 (nic se nezmění), dále 21 s 28 a prvky se prohodí. Poté 14 s 28, prvky se prohodí. Tím cyklus končí. Největší číslo z nesetříděné posloupnosti probublalo na konec posloupnosti před číslo 51.

4	15	9	28	21	14	51
4	15					
4	9	15				
4	9	15	28			
4	9	15	21	28		
4	9	15	21	14	28	51

Při třetím průchodu se porovná 9 se 4 (nic se nezmění), pak 15 s 9, dále 21 s 15 (nic se nezmění). Poté se porovná 14 s 21 a prvky se prohodí. Tím největší číslo 21 probublalo na pozici před číslo 28.

4	9	15	21	14	28	51
4	9					
4	9	15				
4	9	15	21			
4	9	15	14	21	28	51

Při čtvrtém průchodu se porovná 9 se 4, 15 s 9 (nic se nezmění). Pak se porovná 14 s 15 a čísla se prohodí. Tak číslo 15 probublá na správné místo.

A tak dále. Algoritmus proběhne v maximální počtu kroků. Tento algoritmus se dá vylepšit tak, že budeme testovat jestli došlo při průchodu cyklem alespoň k jedné výměně. Jestliže ano, pak algoritmus pokračuje, jinak končí. Tím se v některých případech zkrátí doba třídění. Vyplatí se to zejména tehdy, když je již posloupnost částečně setříděná. Přesto tento algoritmus patří mezi jedny z nejpomalejších.

4	9	15	14	21	28	51
4	9					
4	9	15				
4	9	14	15	21	28	51

MergeSort

Algoritmus pracuje následovně:

1) rozdělí neseřazenou množinu dat na dvě podmnožiny o přibližně stejné velikosti
2) seřadí obě podmnožiny
3) spojí seřazené podmnožiny do jedné seřazené množiny

- Algoritmus vytvořil v roce 1945 John von Neumann.
- *von Neumannova koncepce počítače:*
  - *Tato koncepce digitálního počítače vznikla kolem roku 1946.*
  - *Základní moduly jím navrženého počítače jsou: procesor, řadič, operační paměť, vstupní a výstupní zařízení.*
  - *Tato koncepce tvoří základ architektury současných počítačů.*

Vyhledávací algoritmy

lineární (sekvenční) vyhledávání

- sekvenční vyhledávání je nejjednodušší způsob vyhledávání
- při vyhledávání prvku v posloupnosti se tato posloupnost postupně prohledává od prvního prvku až do nalezení hledaného prvku
- v případě, že dojde k prohledání celé posloupnosti až do konce a žádný prvek posloupnosti neodpovídá hledanému prvku, pak se tento prvek v posloupnosti nenachází

binární vyhledávání

- algoritmus vyžaduje seřazená data

Lineární vyhledávání

- procházení všech prvků seznamu
- lineární vyhledávání má časovou složitost O(N), v případě náhodného rozložení je průměrně potřeba N/2 porovnání

nejlepší případ

- nastane tehdy, když se hledaná hodnota nachází na prvním místě v seznamu, v tomto případě je potřeba pouze jedno porovnání

nejhorší případ

- nastane tehdy, když se hodnota v seznamu vůbec nevyskytuje; v tom případě je potřeba N porovnání
- výhodou lineárního vyhledávání je možnost použití i na neuspořádaných seznamech
- V případě, že je potřeba vykonat na seznamu více vyhledávání, je vhodné použít efektivnější údajovou strukturu. Jedno z řešení je uspořádat seznam a použít binární vyhledávání.

Lineární sekvenční vyhledávání

V posloupnosti 10, 15, 2, 36, 14, 89, 52, 6, 47, 12 máme vyhledat např. prvky 52 a 32.

Vyhledání prvku 52:

- 10, 15, 2, 36, 14, 89, 52, 6, 47, 12

Vyhledávání prvku 32

- 10, 15, 2, 36, 14, 89, 52, 6, 47, 12

Binární vyhledávání

- Binární vyhledávání (nebo také metoda půlení intervalu) je vyhledávací algoritmus na nalezení zadané hodnoty **v uspořádaném seznamu** pomocí zkracování seznamu o polovinu v každém kroku. Binární vyhledávání najde medián, porovná s hledanou hodnotou a na základě výsledku porovnání se rozhodne o pokračování v horní nebo dolní části seznamu a rekurzivně pokračuje od začátku.
- Binární vyhledávání je algoritmus s logaritmickou časovou složitostí O(log n). Je značně rychlejší než lineární vyhledávání, které má časovou složitost O(n). Nicméně vyžaduje, aby data byla setříděna, je tudíž vhodný jen pro určitou množinu problémů.
- Binární vyhledávání je příkladem algoritmu typu **rozděl a panuj.**

Binární vyhledávání

V posloupnosti 2, 5, 6, 8, 9, 15, 23, 45, 67, 89, 95, 96, 97
máme vyhledat např. prvek 67.

Číslo 67 se porovná s číslem uprostřed

- 2, 5, 6, 8, 9, 15, 23, 45, 67, 89, 95, 96, 97

s číslem uprostřed pravé poloviny

- 2, 5, 6, 8, 9, 15, 23, 45, 67, 89, 95, 96, 97

s číslem uprostřed pravolevé poloviny

- 2, 5, 6, 8, 9, 15, 23, 45, 67, 89, 95, 96, 97

prvek je nalezen

Příští týden: Složitost algoritmů