GAaDO
Algoritmy

Technická univerzita v Košiciach

KATEDRA MATEMATIKY A
TEORETICKEJ INFORMATIKY

2025–2026

Algoritmy

Pod pojmom **algoritmus** rozumieme postupnosť jednoduchých a jednoznačných krokov, ktorá nás dovedie k žiadanému riešeniu konkrétneho problému.

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,
2. determinovanosť – každý krok jednoznačne a presne definovaný,

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,
2. determinovanosť – každý krok jednoznačne a presne definovaný,
3. konečnosť – vykonávanie procesu skončí vždy po konečnom počte krokov,

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,
2. determinovanosť – každý krok jednoznačne a presne definovaný,
3. konečnosť – vykonávanie procesu skončí vždy po konečnom počte krokov,
4. rezultatívnosť – postup vedie po konečnom počte krokov k vyriešeniu úlohy a pre rovnaké vstupy dáva vždy rovnaké výsledky.

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,
2. determinovanosť – každý krok jednoznačne a presne definovaný,
3. konečnosť – vykonávanie procesu skončí vždy po konečnom počte krokov,
4. rezultatívnosť – postup vedie po konečnom počte krokov k vyriešeniu úlohy a pre rovnaké vstupy dáva vždy rovnaké výsledky.
5. efektívnosť – riešenie v čo najkratšom čase použitím čo najmenšieho počtu prostriedkov,

Vlastnosti algoritmov

1. elementárnosť – jednoduché, zrozumiteľné kroky,
2. determinovanosť – každý krok jednoznačne a presne definovaný,
3. konečnosť – vykonávanie procesu skončí vždy po konečnom počte krokov,
4. rezultatívnosť – postup vedie po konečnom počte krokov k vyriešeniu úlohy a pre rovnaké vstupy dáva vždy rovnaké výsledky.
5. efektívnosť – riešenie v čo najkratšom čase použitím čo najmenšieho počtu prostriedkov,
6. hromadnosť – použiteľný na celú triedu prípadov.

Zápis algoritmu

1. jednoznačnosť, použitím pseudokódu, (štandardné kľúčové slová),

Zápis algoritmu

1. jednoznačnosť, použitím pseudokódu, (štandardné kľúčové slová),
2. bloková štruktúra,

Zápis algoritmu

1. jednoznačnosť, použitím pseudokódu, (štandardné kľúčové slová),
2. bloková štruktúra,
3. 1 veta $=$ 1 riadok,

Zápis algoritmu

1. jednoznačnosť, použitím pseudokódu, (štandardné kľúčové slová),
2. bloková štruktúra,
3. 1 veta $=$ 1 riadok,
4. krátke mená premenných.

Zložitosť algoritmov

- **Pamäťová zložitosť algoritmov** - závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému.

Zložitosť algoritmov

- **Pamäťová zložitosť algoritmov** - závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému.
  - počet pamäťových miest, ktoré algoritmus pri výpočte potrebuje,

Zložitosť algoritmov

- **Pamäťová zložitosť algoritmov** - závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému.
  - počet pamäťových miest, ktoré algoritmus pri výpočte potrebuje,
  - vyjadrujeme ju funkciou, ktorá veľkosti vstupných údajov priraďuje potrebný počet pamäťových miest.

Zložitosť algoritmov

- **Pamäťová zložitosť algoritmov** - závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému.
  - počet pamäťových miest, ktoré algoritmus pri výpočte potrebuje,
  - vyjadrujeme ju funkciou, ktorá veľkosti vstupných údajov priraďuje potrebný počet pamäťových miest.

- **Časovú zložitosť algoritmu** - závislosť časových nárokov algoritmu na veľkosti vstupných údajov-

Zložitosť algoritmov

- **Pamäťová zložitosť algoritmov** - závislosť pamäťových nárokov algoritmu na veľkosti riešeného problému.
  - počet pamäťových miest, ktoré algoritmus pri výpočte potrebuje,
  - vyjadrujeme ju funkciou, ktorá veľkosti vstupných údajov priraďuje potrebný počet pamäťových miest.
- **Časovú zložitosť algoritmu** - závislosť časových nárokov algoritmu na veľkosti vstupných údajov-
  - vyjadrujeme počtom elementárnych operácií.

Časová zložitosť algoritmov

- algoritmus vyrieši úlohu U v čase T, ak potrebuje urobiť T elementárnych krokov,

Časová zložitosť algoritmov

- algoritmus vyrieši úlohu U v čase T, ak potrebuje urobiť T elementárnych krokov,
- dĺžka úlohy - množstvo vstupných dát,

Časová zložitosť algoritmov

- algoritmus vyrieši úlohu U v čase T, ak potrebuje urobiť T elementárnych krokov,
- dĺžka úlohy - množstvo vstupných dát,
- funkcia $T(n)$ vyjadruje hornú hranicu počtu krokov algoritmu pre najhorší prípad s dĺžkou n.

Časová zložitosť algoritmov

- algoritmus vyrieši úlohu U v čase T, ak potrebuje urobiť T elementárnych krokov,
- dĺžka úlohy - množstvo vstupných dát,
- funkcia $T(n)$ vyjadruje hornú hranicu počtu krokov algoritmu pre najhorší prípad s dĺžkou n.
- asymptotická časová zložitosť.

Časová zložitosť algoritmov

Definícia

Nech $g(n), h(n)$ sú dve kladné funkcie definované na množine $\mathbb{N}$. Budeme písať $g(n) = \mathcal{O}(h(n))$ a hovoriť, že funkcia $h(n)$ ***asymptoticky dominuje*** *funkcii $g(n)$, ak existuje konštanta $k \in \mathbb{N}_0$ a prirodzené číslo $n_0 \in \mathbb{N}_0$ také, že $g(n) \leq k \cdot h(n)$ pre $\forall n \geq n_0$.*

Časová zložitosť algoritmov

Definícia

Nech $g(n), h(n)$ sú dve kladné funkcie definované na množine $\mathbb{N}$. Budeme písať $g(n) = \mathcal{O}(h(n))$ a hovoriť, že funkcia $h(n)$ ***asymptoticky dominuje*** *funkcii $g(n)$, ak existuje konštanta $k \in \mathbb{N}_0$ a prirodzené číslo $n_0 \in \mathbb{N}_0$ také, že $g(n) \leq k \cdot h(n)$ pre $\forall n \geq n_0$.*

Definícia

Hovoríme, že algoritmus A ***má zložitosť*** *$\mathcal{O}(f(n))$, ak pre horný odhad $T(n)$ počtu krokov algoritmu A úlohy dĺžky n platí $T(n) = \mathcal{O}(f(n))$.*

Časová zložitosť algoritmov

Definícia

Nech $g(n), h(n)$ sú dve kladné funkcie definované na množine $\mathbb{N}$. Budeme písať $g(n) = \mathcal{O}(h(n))$ a hovoriť, že funkcia $h(n)$ ***asymptoticky dominuje*** *funkcii $g(n)$, ak existuje konštanta $k \in \mathbb{N}_0$ a prirodzené číslo $n_0 \in \mathbb{N}_0$ také, že $g(n) \leq k \cdot h(n)$ pre $\forall n \geq n_0$.*

Definícia

Hovoríme, že algoritmus A ***má zložitosť*** *$\mathcal{O}(f(n))$, ak pre horný odhad $T(n)$ počtu krokov algoritmu A úlohy dĺžky n platí $T(n) = \mathcal{O}(f(n))$.*

Definícia

Ak $f(n) \leq n^k$ pre nejaké konštantné $k \in \mathbb{N}$, hovoríme, že A je ***polynomiálny*** *algoritmus.*

Časová zložitosť algoritmov

Definícia

- *Hovoríme, že problém P je* ***NP–ťažký****, ak úlohu pre ktorú existuje polynomiálny algoritmus možno polynomiálne redukovať na P.*

Časová zložitosť algoritmov

Definícia

- *Hovoríme, že problém P je **NP–ťažký**, ak úlohu pre ktorú existuje polynomiálny algoritmus možno polynomiálne redukovať na P.*
- *Hovoríme, že problém P je **NP–ľahký**, ak problém P možno polynomiálne redukovať na úlohu pre ktorú existuje polynomiálny algoritmus.*

Časová zložitosť algoritmov

Definícia

- *Hovoríme, že problém P je **NP–ťažký**, ak úlohu pre ktorú existuje polynomiálny algoritmus možno polynomiálne redukovať na P.*
- *Hovoríme, že problém P je **NP–ľahký**, ak problém P možno polynomiálne redukovať na úlohu pre ktorú existuje polynomiálny algoritmus.*
- *Hovoríme, že problém P je **NP–úplný**, ak platia obe implikácie.*

Časová zložitosť algoritmov

Zrýchľovanie algoritmov:

- prepočítanie výsledkov a ich uloženie do pamäte,
- výpočet hodnôt na základe predchádzajúcich výpočtov,
- využitie predchádzajúcich hodnôt,
- predspracovanie výsledku,
- odstránenie rekurzie,
- odstránenie opakujúcich sa výpočtov.

Aproximačné algoritmy a heuristiky

- Aproximačné algoritmy a heuristiky - pre NP-ťažké, keď nám stačí dobré riešenie, ktoré, aj keď nebude práve optimálne, bude blízko optimálnemu.

Aproximačné algoritmy a heuristiky

- Aproximačné algoritmy a heuristiky - pre NP-ťažké, keď nám stačí dobré riešenie, ktoré, aj keď nebude práve optimálne, bude blízko optimálnemu.
- Heuristiky:
  1. vytvárajúce
  2. zlepšujúce

Vytvárajúca heuristika:

Vytvárajúca heuristika:

- začne najjednoduchším objektom (nar. $\emptyset$),

Vytvárajúca heuristika:

- začne najjednoduchším objektom (nar. $\emptyset$),
- ten rozširuje (v teórii grafov o vrchol alebo hranu) podľa nejakého pravidla tak, aby výsledný objekt bol blízko optimálnemu,

Vytvárajúca heuristika:

- začne najjednoduchším objektom (nar. $\emptyset$),
- ten rozširuje (v teórii grafov o vrchol alebo hranu) podľa nejakého pravidla tak, aby výsledný objekt bol blízko optimálnemu,
- snaha o dosiahnutie najlepšej kriteriálnej funkcie,

Vytvárajúca heuristika:

- začne najjednoduchším objektom (nar. $\emptyset$),
- ten rozširuje (v teórii grafov o vrchol alebo hranu) podľa nejakého pravidla tak, aby výsledný objekt bol blízko optimálnemu,
- snaha o dosiahnutie najlepšej kriteriálnej funkcie,
- pažravá metóda – greedy algorithm.

Zlepšujúca heuristika:

Zlepšujúca heuristika:

- vychádzajú z existujúceho prípustného riešenia,

Zlepšujúca heuristika:

- vychádzajú z existujúceho prípustného riešenia,
- snaha jednoduchými operáciami dostať iné blízke riešenie s lepšou hodnotou kriteriálnej funkcie.

Zlepšujúca heuristika:

- vychádzajú z existujúceho prípustného riešenia,
- snaha jednoduchými operáciami dostať iné blízke riešenie s lepšou hodnotou kriteriálnej funkcie.
- pokračuje dovtedy, kým nedospeje k takému riešeniu, ktoré už nemá žiadne lepšie susedné riešenie,

Zlepšujúca heuristika:

- vychádzajú z existujúceho prípustného riešenia,
- snaha jednoduchými operáciami dostať iné blízke riešenie s lepšou hodnotou kriteriálnej funkcie.
- pokračuje dovtedy, kým nedospeje k takému riešeniu, ktoré už nemá žiadne lepšie susedné riešenie,
- prehľadávanie okolí – neighborhood search.

Metódu prehľadávania okolí možno kombinovať s pravdepodobnostnou metódou tak, že ju spustíme na niekoľko náhodne vytvorených štartovacích riešení a za výsledok berieme to najlepšie z nájdených lokálnych optimálnych riešení.