A logika és a számítástudomány alapjai mérnök informatikus hallgatóknak

13. előadás, algoritmusok

Mihálydeák Tamás, Aszalós László

A tananyag elkészítését az EFOP-3.4.3-16-2016-00021 számú projekt támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg.

2019. november 4.

Problémamegoldás

A problémamegoldás azzal kezdődik, hogy pontosan és világosan megfogalmazzuk a problémát. A gyakorlati problémából egy modellt alkotunk, azaz a kevésbé fontos részleteitől megszabadulunk, csak a lényegre összpontosítunk. Az úgy születő absztrakt modell rendszerint már ismert eszközökkel megoldható.
A feladat megoldásánál az egymást követő lépéseket egy meghatározott utasítás szabja meg.
(Lásd reguláris kifejezés alapján automata konstruálása)
Az eljárást más és más kezdőértékkel sok azonos típusú feladatot oldhatunk meg.

Algoritmus fogalma

Mivel nincs egy mindenki által elfogadott algoritmusfogalom, így konkrét definíciót nem adunk meg. Viszont közelíteni tudjuk az alábbi megfogalmazással:
Műveletek tartalmát és sorrendjét meghatározó egyértelmű utasításrendszer, amely a megfelelő kiinduló adatokból a kívánt eredményre vezet.

- A művelet egy olyan átalakítás, mellyel az adatok aktuális értékeit felhasználva azok új értékeit állítja elő
  - pl. változó értékének növelése, adatok sorba rendezése
- A műveletek sorrendjét vezérlő szerkezetek írják le:
  - szekvencia (egymás utáni végrehajtás)
  - szelekció (kiválasztás, döntés: `if-then-else`)
  - iteráció (ismétlés, ciklus: `while`, `for`)

Példa - Euklideszi algoritmus

Határozzuk meg két szám legnagyobb közös osztóját!

- Alapötlet: a lnko nem változik, ha a nagyobb számot a két szám különbségével helyettesítjük.
- Ezt a lépést ismételjük mindaddig, míg a két szám egyenlővé nem válik
- Ez a közös szám az eredeti számol legnagyobb közös osztója.

Formálisabb megadás

Jelölje a bemenő két pozitív egész számot m és n!

1. Osszuk el m-et n-nel, a maradék legyen r.
2. Ha $r = 0$, akkor az algoritmus véget ért, az eredmény n.
3. Legyen $m \leftarrow n$, $n \leftarrow r$, és folytassuk az algoritmust az 1. lépéssel.

Példa - Euklideszi algoritmus

Formális leírás - Python nyelven

```
def gcd(a, b):
    while b:
        a, b = b, a%b
    return a
```

Algoritmus tulajdonságai

Végesség Az algoritmus futása véges sok lépés után befejeződik. Esetünkben r kisebb mint n, azaz n értéke folyamatosan csökken az algoritmus végrehajtása során, és pozitív egészek egyre fogyó sorozata egyszer véget ér.

Meghatározottság Az algoritmus minden egyes lépésének pontosan definiáltnak kell lennie. Miután az élőnyelvi megfogalmazás esetenként nem egyértelmű, használhatunk különféle programnyelveket, ahol mindennek pontos, egyértelműen definiált jelentése van.

Bemenet/Input Az algoritmus vagy igényel vagy sem olyan adatokat, amelyeket a végrehajtása előtt meg kell adni. Az input mindig bizonyos meghatározott halmazból kerülhet ki, esetünkben m és n pozitív egész szám.

Algoritmus tulajdonságai

Kimenet/Output Az algoritmushoz egy vagy több output tartozhat, amelyek meghatározott kapcsolatban állnak az inputtal. Esetünkben az output az n utolsó értéke lesz, ami az input értékeknek legnagyobb közös osztója.

Elvégezhetőség Elvárjuk, hogy az algoritmust végre lehessen hajtani, azaz a végrehajtható utasítások elég egyszerűek ahhoz, hogy egy ember papírral és ceruzával véges idő alatt pontosan végrehajthassa. Például a végtelen tizedestörtek osztása nem ilyen, mert ez végtelen lépéssorozat.

Univerzális Az algoritmusnak működnie kell tetszőleges (a feltételeknek megfelelő) bemeneti értékek esetén. Az euklideszi algoritmusunk természetesen tetszőleges pozitív számpárnak meghatározza a legnagyobb közös osztóját.

Történeti előzmények

- 1928, David Hilbert - *van olyan algoritmus, mellyel eldönthető, hogy egy formula bebizonyítható az axiómákból?* (effektív kiszámolás)
- 1930, Alonzo Church - lambda kalkulus: formális számolási modell függvényalkalmazással és függvényabsztrakcióval.
- 1936, Stephen C. Kleene, rekurzív függvények (kiszámolható függvények)
- 1936, Alan Turing - Turing gép (író-olvasó fej, szalagok), elméleti kérdések
- 1951, A. A. Markov - szöveghelyettesítés

Markov algoritmus

Formális rendszer, az input és az output szöveg, helyettesítéseket hajtunk végre, amíg az lehetséges.
Az egyértelmű végrehajtás érdekében a helyettesítéseket sorba állítjuk, és mindig az első lehetséges helyettesítést hajtjuk végre. Ha ugyanazon helyettesítés a szöveg több részén is lehetséges, akkor csak az első előfordulást helyettesítjük. Az univerzalitás érdekében bevezetünk záróhelyettesítéseket, és ezek alkalmazása után a végrehajtás leáll.

Markov algoritmus

Definíció

- Az $M = \langle \Sigma, H, H' \rangle$ hármast Markov-féle normál algoritmusnak nevezzük, ahol Σ egy ábécé $H \subset \Sigma^\star \times \Sigma^\star$ helyettesítési szabályok véges, rendezett halmaza, és $H' \subseteq H$ a záróhelyettesítések halmaza.
- Azt mondjuk, hogy M alapján $x \in \Sigma^\star$-ből közvetlenül levezethető $y \in \Sigma^\star$, ha léteznek olyan u, v, w és $z \in \Sigma^\star$, hogy $x = uvw$, $y = uzw$, $(v, z) \in H$, és (v, z) az első olyan szabály, ahol v része x-nek – azaz ha $x = svt$, ahol s, $t \in \Sigma^\star$, akkor u része s-nek.
  A közvetlen levezethetőséget $x \Rightarrow y$ jelöli.
- A közvetlen levezetést záró levezetésnek nevezzük, ha a benne szereplő helyettesítés záróhelyettesítés – azaz $(v, z) \in H'$.
- Azt mondjuk, hogy M alapján x-ből y legalább 1 lépésben levezethető ($x \Rightarrow^+ y$), ha van olyan $n \geq 1$ és $x_0, x_1, \ldots, x_n \in \Sigma^\star$, hogy $x = x_0$, $y = x_n$, és $x_i \Rightarrow x_{i+1}$ ha $0 \leq i < n$.
- Azt mondjuk, hogy M alapján x-ből y levezethető ($x \Rightarrow^\star y$), ha $x = y$ vagy $x \Rightarrow^\star y$.

Markov algoritmus

Definíció

Azt mondjuk, hogy az M Markov-algoritmus az x bemenő szöveghez az y eredményt állítja elő, ha M alapján x-ből y levezethető, és

- a levezetésben szereplő közvetlen levezetések egyike sem záró közvetlen levezetés, és y a szabályok bal oldalainak egyikét sem tartalmazza.
- a levezetésben szereplő közvetlen levezetések egyike sem záró közvetlen levezetés, kivéve az utolsót.

Markov algoritmus Python-ban

```
def replace(text, replacements):
    while True:
        for pat, repl, term in replacements:
            if pat in text:
                text = text.replace(pat, repl, 1)
                if term:
                    return text
                break
        else:
            return text

replace("1011", [("|0", "0||", False),
                 ("1", "0|", False),
                 ("0", "", False)])
```