

15. TÉTEL: Algoritmus szemléletes és pontos fogalma. Turing gép. Megállási probléma. Algoritmikusan kiszámítható számok és számelméleti függvények. Példa algoritmussal kiszámolhatatlan függvényre.

Az algoritmus több azonos jellegű, egymástól csak a kiinduló adatokban különböző (pl. matematikai) feladat megoldására szolgáló eljárás, amelynek során előre meghatározott, véges számú lépést adott sorrendben végrehajtva jutunk el a feladat megoldásához.

- Algoritmus: egy függvény, amely
 - Bemenete: értelmezési tartomány (adatszerkezet)
 - Kimenete: az értékkészlet (adatszerkezet)
 - Specifikáció: a függvény definíciója, amely megadja, hogy milyen bemenethez milyen kimenetet kell előállítania.

Algoritmus kifejezése/megfogalmazása: Természetes nyelven

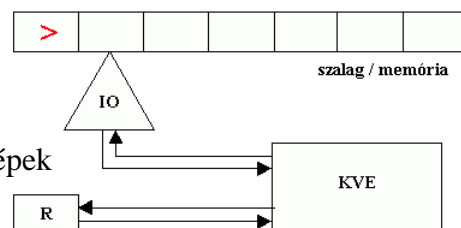
- Pszeudokóddal
- Blokkdiagrammal
- Tervezési nyelvvel (UML)
- Programnyelvvel
- Adott specifikációhoz többféle megfogalmazás is létezhet

A "jó" algoritmusok alapvető jellemzői:

- végeesség
 - az algoritmus véges számú lépésből áll
 - nincs az algoritmusban "végtelen ciklus"
- egyértelműség
 - minden lépés után egyértelműen meghatározható a következő lépés
- teljesség
 - minden lépésre van rákövetkező, kivéve az utolsó, befejező lépést
 - a "kivételek" kezelve vannak (pl. a nullával való osztás)
- determinisztikusság vagy meghatározottság
 - adott bemenő adatok mindig ugyanazokat a kimenő adatokat fogják szolgáltatni
 - az un. "rejtett paraméterek" ki vannak küszöbölve

Turing-gép

Megjegyzés: a Turing-gépnek több "változata" is létezik a szakirodalomban (pl. balról is végtelen szalag, un. "igen" és "nem" típusú elfogadó, ill. elutasító megállási állapotok a halt vagy stop állapot mellett, stb.), a különbözőképpen definiált Turing-gépek azonban ekvivalensek egymással.



A Turing-gép egy végtelenig leegyszerűsített számítógép elméleti modelljének tekinthető, amellyel egyszerűsége ellenére minden adatfeldolgozási algoritmus leírható. Ezt fejezi ki az un. **Church-tézis:** Minden intuitív értelemben kiszámíthatónak tekinthető (azaz algoritmizálható) függvényt ki lehet számítani a Turing-géppel.

A Turing-gép részei:

- egy jobbra végtelen hosszúságú szalag, a Turing-gép memóriája, amely cellákból (vagy rekeszekből) áll
 - minden cella egy karaktert tartalmaz a Turing-gép ábécéjéből
 - a bal szélső cella a start karaktert vagy szimbólumot (>) tartalmazza
 - ez az első karakter, amit az író-olvasó fej be fog olvasni
 - a start karaktert beolvasása után az író-olvasó fej mindig visszaírja (azaz nem cseréli ki más karakterre)
 - a start karakter beolvasása után az író-olvasó fej mindig jobbra fog lépni (azaz nem "lép le" balról a szalagról)
 - bináris ábécéjű Turing-gép esetén a további cellákban a bináris számjegyek (0 vagy 1) szerepelhetnek (ill. az ezeket elválasztó üres karakter, ld. alább)
 - a nem használt cellák az üres karakterrel () vannak feltöltve
 - a szalag tartalmazza
 - a gép indulásakor a Turing-gép inputját
 - a gép működése közben a képződött részeredményeket
 - a gép leállása után a Turing-gép outputját
 - a szalagon levő adatok a Turing-gép működését is vezérlik (azaz a Turing-gépben az adatok és a gépet működtető program nem választható élesen szét)
- egy író-olvasó fej, amely minden lépésben elmozdulhat egy cellával jobbra vagy balra (de helyben is maradhat)
- egy állapotregiszter (R), amely a Turing-gép aktuális állapotát tárolja
 - a Turing-gép lehetséges állapotainak halmazát a Turing-gép alapprogramja tartalmazza
 - az állapotok egyfajta "címke" funkciót látnak el, azaz az utasítások egy csoportját adják meg
 - minden utasítás megadja, mi lesz a Turing-gép következő állapota
- egy (központi) vezérlő egység (KVE), amely
 - tárolja a Turing-gép alapprogramját
 - vezérli a Turing-gép **utasításciklusát**

A Turing-gép alapprogramja (az un. átmenetfüggvény) legegyszerűbben egy táblázattal adható meg, amelynek sorai a Turing-gép lehetséges állapotainak, oszlopai pedig a Turing-gép ábécéjének felelnek meg. A táblázat cellái azokat az utasításokat tartalmazzák, amelyek a Turing-gép egyes állapotainak és a Turing-gép ábécéjében levő karaktereknek felelnek meg.

A Turing-gép utasításai három részből állnak, amelyek az utasítás végrehajtása után meghatározzák

- a Turing-gép új állapotát; ez lehet
 - a lehetséges állapotok közül valamelyik
 - a h megállási állapot (halt vagy stop állapot), amely a Turing-gép leállítását eredményezi
- a szalag aktuális cellájába beírandó új karaktert
- az író-olvasó fej mozgatását; ez lehet
 - léptetés egy cellával balra
 - léptetés egy cellával jobbra
 - helyben maradás; ilyenkor a gép nem lépteti a fejet

A Turing-gép utasításciklusa

1. egy karakter beolvasása a szalag aktuális cellájából (ez megfelel egy utasítás beolvasásának)
2. a Turing-gép aktuális állapotának beolvasása az állapotregiszterből
3. a Turing-gép aktuális állapotának és a beolvasott karakternek megfelelő utasítás megkeresése a Turing-gép alapprogramjában
4. az utasítás értelmezése, azaz részeire bontása (három részre, ld. fent)
5. az értelmezett utasítás végrehajtása
 - o az állapotregiszter tartalmának beállítása
 - o egy karakter kiírása a szalag aktuális cellájába
 - o az író-olvasó fej léptetése
6. az állapotregiszter tartalmától függően a Turing-gép működésének leállítása (ez megfelel a megszakításkérések vizsgálatának)
7. az utasításciklus újratekintése

KILIÁNOS ANYAG (mivel ott lesz államvizsgán beraktam a diáit):

- A TM elfogadja a szalag kezdeti tartalmát, ha elfogadó állapotban megáll, elutasítja, ha egyéb állapotban áll meg.
- TM megáll, ha az adott állapotra és bemenő szimbólum(ok)ra nincs illeszkedő feltételrészű szabály
- TM nemdeterminisztikus, ha létezik olyan állapot és bemenő szimbólum pár, amire több szabály is létezik
- Nemdeterminisztikus TM elfogadja a szalag bemeneti tartalmát, ha létezik olyan mozgássorozat, ami elfogadja.
- Church-tézis: minden, ami algoritmusokkal megvalósítható, az T-géppel kiszámítható
- Többszalagos TM-et egyszalagos szimulálhatja.
- Eljáráshívás: T1 hívja T2-t. Egyesítsük az állapotaikat. Induljon a T1, hívás: olyan szabály, ami a T2-t indítja, Befejezés: T2 végállapotai után térjen valahová T1-be vissza
- Beszúrás: T állapothalmaza legyen képes egy (beszúrandó) szimbólum eltárolására. A szalagra férjen még egy jelölő is. Jelöljük meg a helyet. Cseréljük ki a tároltat a szalagon levővel. Lépünk jobbra. Ha üres szimbólumot olvasunk, akkor visszatekerünk a jelölőig; ha nem, akkor a cserétől újratekintjük a ciklust.
- A T-gép megáll, ha nincs a helyzetnek (állapot-szalag) megfelelő szabály. Elfogadja a bemeneti nyelvet, ha elfogadó állapotban áll meg. Elutasítja, ha nem elfogadó állapotban áll meg, vagy végtelen ciklusba esik.
- Neumann-elv: program-adat ekvivalencia
- T-gép képes egy (másik) T-gépet szimulálni úgy, hogy induláskor a szalagján van a (másik) gép leírása, plusz a másik gép bemenete. (kb. 60 mozgási szabály elég hozzá).
➔ Programbetöltés, Univerzális T-gép
- A T-gép egyenértékű egy egyállapotú T-géppel.
- Számítógép: Random Access Machine egyenértékű a T-géppel
- Egy T-gép által elfogadott bemenetek egy nyelvet alkotnak
- Tétel: A T-gépek a Chomsky 0 osztályú (legszerkeletesebb) nyelvek elemzésére képesek

MEGÁLLÁSI PROBLÉMA

- Megállapítható-e, hogy mikor esik a T-gép végtelen ciklusba? Módszer: univerzális T-gépnek odaadjuk a T-gép(ek) leírását bemenetként...

- Odaadhatjuk-e univerzális T-gépnek a saját leírását bemenetként?
- Odaadhatjuk-e egyes T-gépeknek a saját leírásukat bemenetként? Mi történik? → lesz olyan, amelyik elfogadja/elutasítja/ciklusba esik
- Legyen: L1 azon T-gép leírások nyelve, amelyek sajátmagukat elfogadják, L2 pedig azoké, amelyek sajátmagukat elutasítják
- Az univerzális T-gépet egészítsük ki egy a leírást megkettőző (gép+adat) előkészítő algoritmussal. Ez a T-gép éppen az L1 nyelvet fogja elfogadni.
- Készíthetünk-e olyan T-gépet, amely az L2 nyelvet fogadja el? → Nem
- Tfh. létezik ilyen (a saját leírását elutasító gépeket elfogadó) gép. Mit csinál ez a gép a saját leírásával? → Ha elfogadja, akkor a saját leírását elutasítónak kell lennie, vagyis el kellene utasítania
→ Ha elutasítja, akkor a specifikáció miatt el kellene fogadnia
- → Nincs az L2 nyelvet elfogadó T-gép. Az L2 nyelv nem írható le generatív módon
- Eldönthető-e minden T-gépre és minden bemenetre, hogy a T-gép megáll-e? → Nem. (eldönthető egy kérdés, ha megáll)
- Tfh. igen. Ekkor szerkeszthető lenne olyan (univerzális) T-gép, ami minden gépleírást-bemenet párt elfogad, ha a leírt gép a bemenetre ciklusba esne, és elutasít vagy ciklusba esik, ha a gép az adatra megállna.
- Mit tenne a gép a saját megkettőzött leírására? → elfogadná? (akkor, ha ciklusba kellene esnie) → ellentmondás
→ elutasítaná vagy ciklusba esne? (akkor a specifikáció miatt meg kellene állnia...
→ ellentmondás
- → nem dönthető el minden gépre és bemenetre, hogy a gép megáll-e → Léteznek nem eldönthető problémák...

Algoritmikusan kiszámítható számok és számelméleti függvények

Algoritmikusan kiszámíthatóak azok az egész értékű függvények, amelyeknek az értékeit valamilyen algoritmussal elő tudjuk állítani.

Church 1931: Minden kiszámítható függvény kiszámítható Turing géppel... → nem kell megjelölni, hogy egy függvény milyen géppel kiszámítható.

Példa algoritmussal kiszámolhatatlan függvényre

Például:

- két természetes szám relatív prím-e,
- egy Turing-gép mindig megáll-e véges sok lépés után, stb.

Algoritmikus probléma: általános módszert kell megadnunk, amellyel minden konkrét esetben meg tudjuk válaszolni az adott típusú kérdést. Az ilyen kérdések arra vezethetők vissza, hogy egy adott ábécéből alkotott szó benne van-e egy adott nyelvben, vagy sem.

Egy algoritmikus problémát **megoldhatónak nevezzük**, ha a hozzá tartozó formális nyelv rekurzív, és **megoldhatatlannak**, ha nem rekurzív. Tehát akkor mondunk egy ilyen problémát megoldhatatlannak, ha nem létezik olyan Turing-gép, amely bármely konkrét eset kódjáról véges sok lépésben eldönti, hogy az adott rögzített nyelvben benne van-e, vagy sem.

Annak felismerése, hogy egy rekurzívan felsorolható nyelv mikor üres, véges, rekurzív, stb., *algoritmikusan megoldhatatlan problémát* jelent.