ADATSZERKEZETEK ÉS ALGORITMUSOK

Típusok – Absztrakció

Adatszerkezetek megközelítése

- Nem fejlesztünk ki egy típuselméletet
- Szemléletmód kialakítása a fontos
- A könyvekben sokféle szemlélet és leírási mód található
- A (zavaró) sokféleség fő oka: a típus-szemlélet hiánya/mellőzése

Típusok

Modern definíció

- A típus egy adat által felvehető lehetséges értékek halmazán kívül megadja az adaton értelmezett műveleteket is.

Régi megközelítés

- Egy adat típusán csak az adat által felvehető értékek halmazát értik.

A típus-absztrakció szintjei

- Absztrakt adattípus (ADT)
  - Nem feltételez a belső szerkezetről semmit sem
  - Egységbezárás – enkapszuláció
- Absztrakt adatszerkezet (ADS)
  - Absztrakt szerkezet
  - Irányított gráf mutatja a rákövetkezéseket
    - Csúcsok – adatelemek
    - Élek – rákövetkezések

A típus-absztrakció szintjei

- Reprezentáció
  - ADS gráf az absztrakt memóriában
  - Fontos, hogy a rákövetkezések megmaradjanak!
    - Láncolt vagy aritmetikai ábrázolás
- Implementáció
  - programnyelven
- Fizikai ábrázolás
  - az illúzió vége: bitek

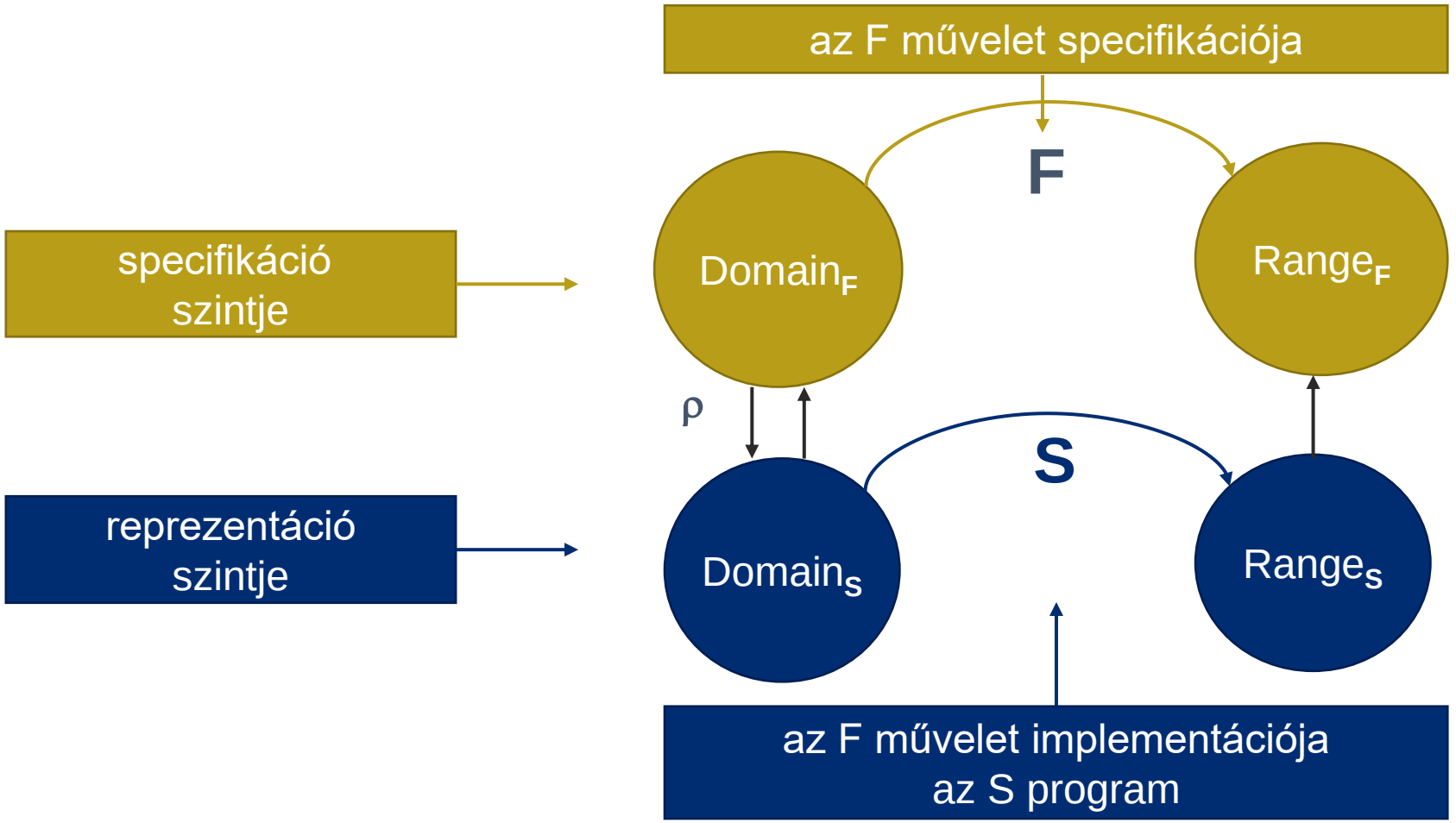
Típus-specifikáció

- Egy adat külső jellemzésére szolgál (interfész)
  - Típusérték-halmaz
    - Az adat által felvehető értékek T halmaza.
  - Típusműveletek
    - T -n értelmezett feladatok.
- Önálló szerepet játszik a programtervezésben
- Megadására többféle lehetőség van:
  - Algebrai specifikáció
    - Axiómák megadásával
  - Funkcionális specifikáció
    - Elő- és utófeltételekkel

Típus

- A típus-reprezentáció
  - Típusértékek ábrázolására
  - Ábrázoló elemek H halmaza
    - típus-szerkezet
  - Az ábrázoló elemek és a típusértékek kapcsolatát leíró leképezés
    - $\rho : H \rightarrow T, \rho \subseteq H \times T$
  - A típus-invariáns kiválasztja a hasznos ábrázoló elemeket
    - $\boldsymbol{I} : H \rightarrow \boldsymbol{L}, [\boldsymbol{I}]$
- A típus-implementáció
  - Műveletek helyettesítése
  - Nem a típusértékekkel, hanem az azokat ábrázoló elemekkel működő programok

A típus specifikáció és a típus kapcsolata



Példa a „SZÍN” típus és művelete.

az F művelet specifikációja

Specifikáció szintje

Értékek: 1, 2, 3,
4, 5, 6, 7, 8

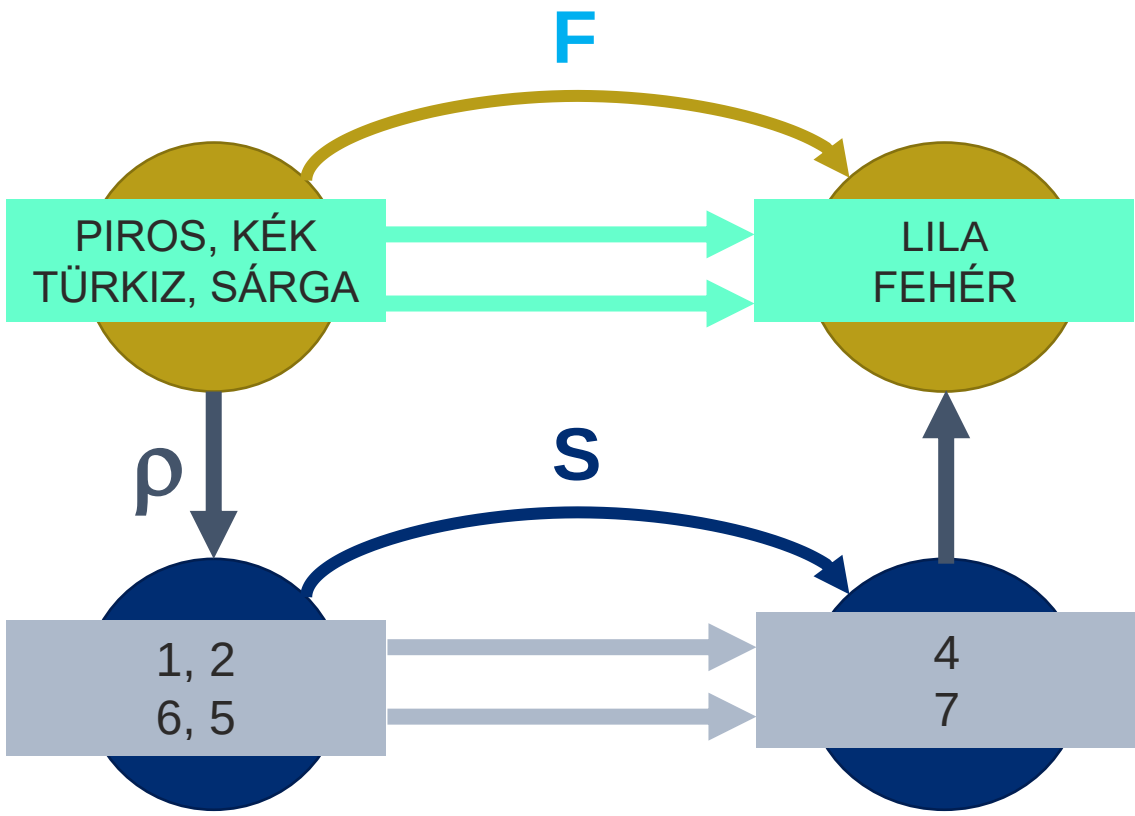
Legyen az F művelet a „KEVERÉS”

Szükséges egy függvény, amellyel az egyes típusértékekhez reprezentációs értéket rendelünk.

ρ

Reprezentáció szintje

A program végrehajtása után vissza kell térni az eredeti típusértékhez.
Ehhez az előző függvény inverzét alkalmazzuk.



S program végrehajtása

Egész értékekkel reprezentáljuk a színeket.

az F művelet implementációja az S program

Absztrakt adattípus

- A típus-specifikáció (közvetett) megadására szolgál
  - Nem szükséges, hogy egy konkrét programozási környezetben ábrázoljuk a típusértékeket.
  - Elég a műveletek programjainak csak a hatását ismerni.
- Absztrakt a programozási környezet számára és a megoldandó feladat számára.
- Szükség van egy őt kiváltó (konkrét) típusra.
  - Részfeladatokra bontás eszköze.

Absztrakt adattípus

- A típus szemléletének ez a legmagasabb szintje
- Semmilyen feltételezéssel nem élünk a típus szerkezetéről, megvalósításáról!
- A specifikációban csak tisztán matematikai fogalmakat használhatunk
- Ez a szint nem a formalizálás mértékétől absztrakt
  - lehet informálisan is gondolkodni, beszélni ADT szinten

Az ADT algebrai specifikációja

- Részei
  - Típusérték halmaz
  - Műveletek (mint leképezések)
  - Megszorítások (értelmezési tartományok)
  - Axiómák
- Kérdések:
  - Helyesség (ellentmondásmentesség)
  - Teljesség
    - Nehéz feladat megoldani
  - Redundancia
    - Nem feltétlen fontos

Az ADT funkcionális specifikációja

- A típus matematikai reprezentációját használjuk
- Ez semmilyen módon nem kell, hogy utaljon a típus ábrázolási módjának megválasztására a megvalósítás során!
- Részei:
  - típusérték halmaz
  - műveletek
  - állapottér
  - paramétertér
  - előfeltétel
  - utófeltétel

Algoritmusok ADT szinten

- A típus értékeit csak a típusműveletek segítségével lehet változtatni
- Azt nem tudjuk, hogy a műveletek hogyan működnek
- A típus „fekete doboz”
  - semmilyen megkötés nincs a szerkezetéről

Műveletek, hozzárendelések

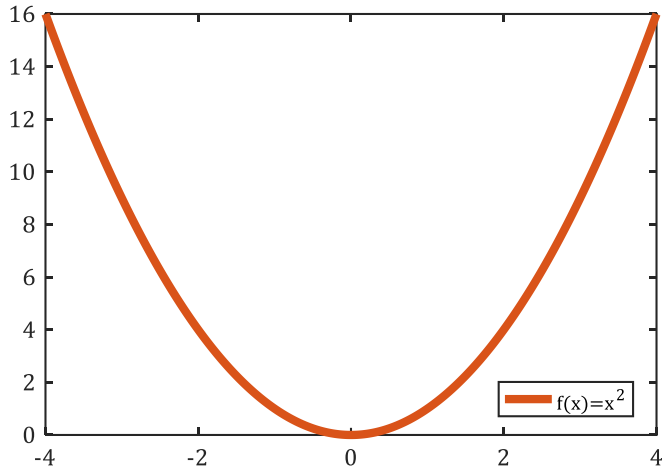
- A típushoz tartozó műveleteket formálisan specifikáljuk
- Ezek függvények
  - Egy adott halmaz (értelmezési tartomány) elemeihez rendelnek hozzá egy másik halmaz (értékkészlet) elemei közül
- Például
  - Négyzetre emelés: $\mathrm{x} \longmapsto x^2 \colon \mathbb{R} \to \mathbb{R}$

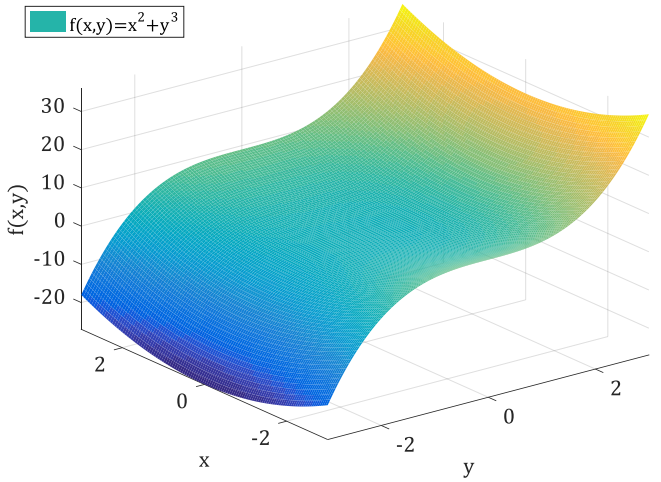
    $2 \to 4$

    $0{,}2 \to 0{,}04$
  - Ebben az esetben a párok könnyen ábrázolhatók.
- Másik példa
  - Két változó: $(\mathrm{x}, \mathrm{y}) \longmapsto x^2 + y^3 \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

    $(2; 3) \to 13$

    $(4; 0) \to 16$
  - Ez is ábrázolható.



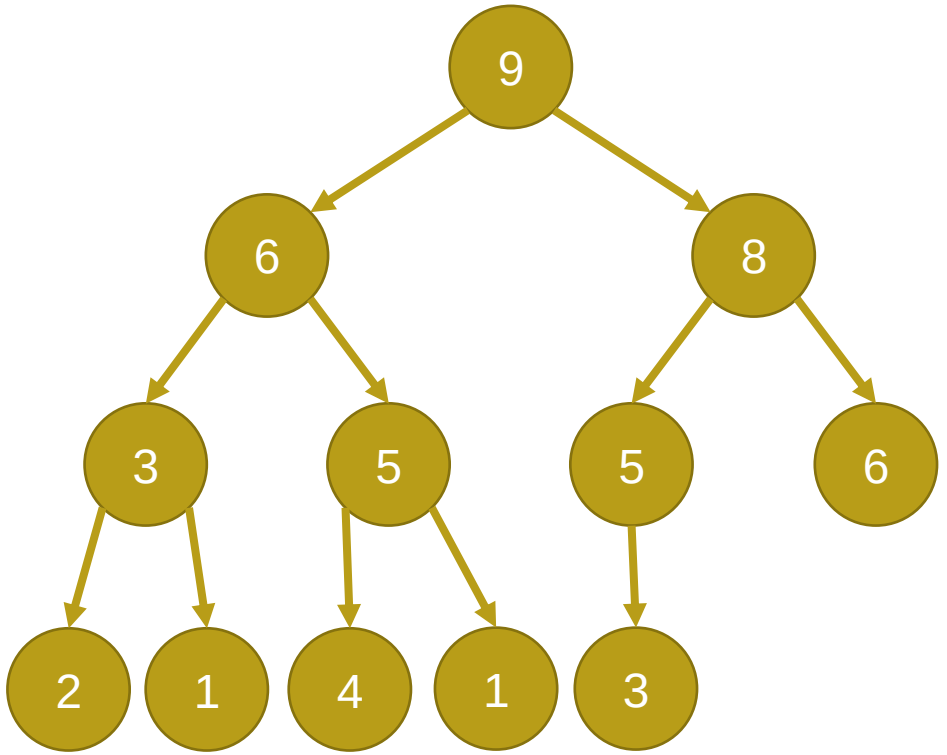


Absztrakt adatszerkezet (ADS)

- Széles körben használják az oktatásban, könyvekben
- A típus alapvető – absztrakt – szerkezetét egy irányított gráffal ábrázoljuk
  - A gráf jelentése
    - csúcsok: adatelemek
    - élek: rákövetkezési reláció
- A műveletek ezen a szinten is jelen vannak
  - ha egy típus ADS-éről van szó
- A műveletek hatása szemléltethető az ADS-gráf változásaival

Példa

- Kupac
  - A prioritásos sor szokásos reprezentációja
  - Bináris fa
  - Majdnem teljes
  - Balra tömörített
  - A szülő csúcsokban nagyobb értékek vannak, mint a gyerek csúcs(ok)ban

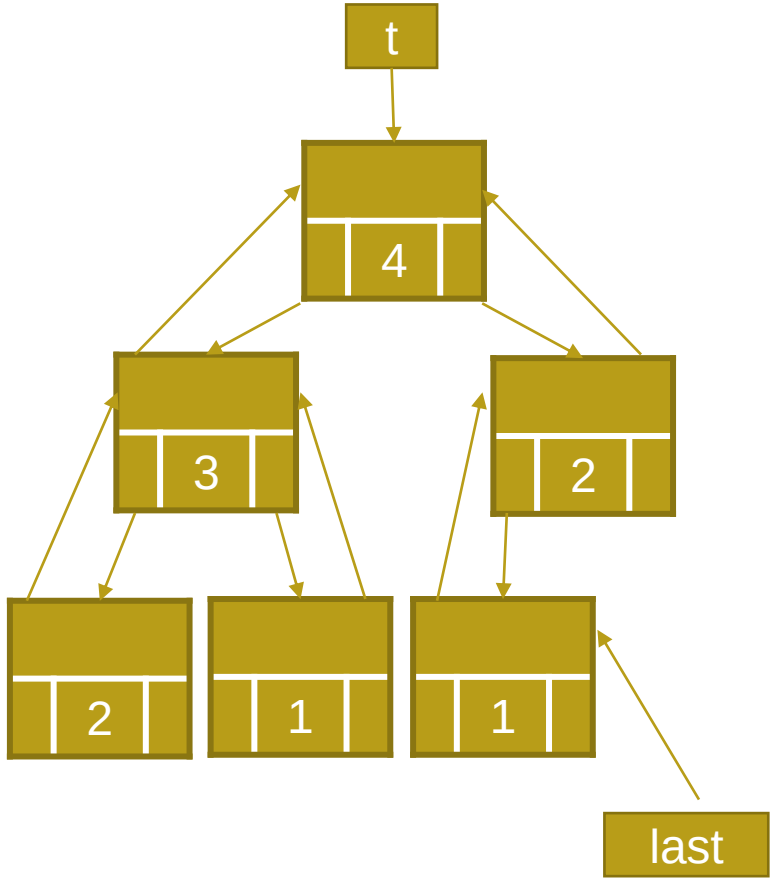


Reprezentáció

- Absztrakt reprezentáció – absztrakt memóriában
- Az absztrakt szerkezet ábrázolható
  - pointerekkel (láncolt ábrázolás), vagy
  - cím-/indexfüggvénnyel (aritmetikai reprezentáció)
  - (vegyes ábrázolás lehetséges)
- Mindkét reprezentáció megadja az adatelemek közötti rákövetkezési relációt
- További rákövetkezések is megadhatók pointerekkel, illetve kiolvashatók az index-függvényből, mint amelyet az ADS leírt!

Pointeres ábrázolás (láncolás)

- Példa: kupac láncolt ábrázolása
- Az ADS-gráf éleit pointerekkel ábrázoljuk
- A műveletek algoritmusait itt már meg kell adni
- A feladatban bevezetett függvények kiszámító algoritmusait is meg kell adni (szemben az ADS-sel)
- Következmény: egy feladat megoldása gazdagabb reprezentációt igényelhet, mint maga az ADS

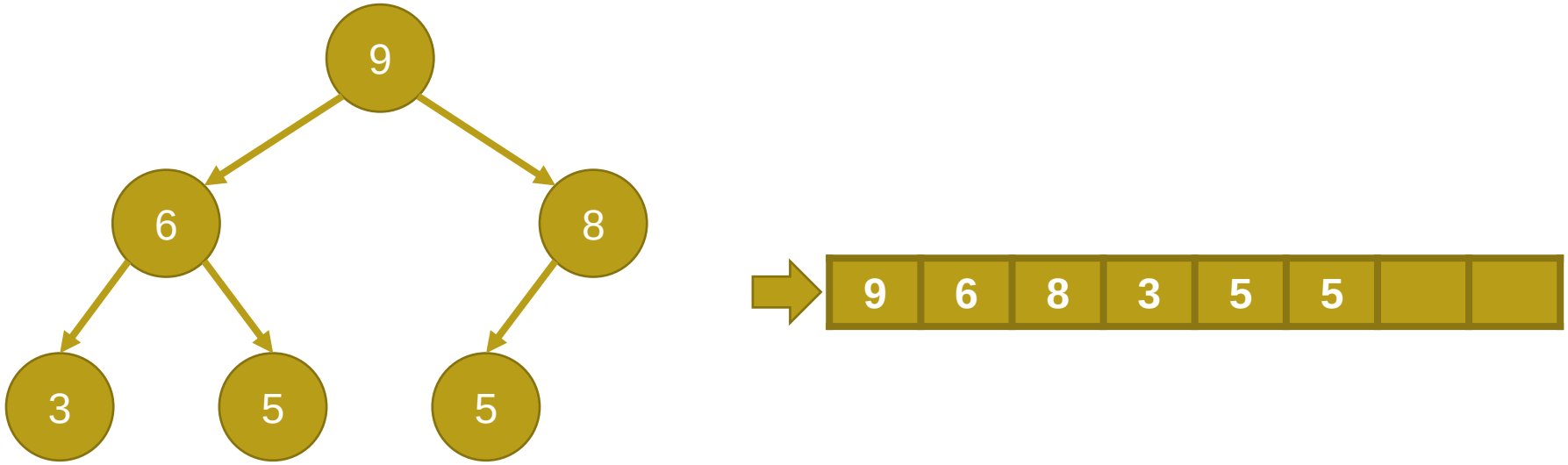


Aritmetikai reprezentáció

- Index- és címfüggvények megadása
  - Az adatelemeket folyamatosan elhelyezzük az absztrakt memóriában/ egy ugyanilyen alaptípusú vektorban
  - Az elemek közötti rákövetkezési relációt egy cím-/ indexfüggvénnyel adjuk meg
  - A címfüggvényből további rákövetkezések is kiolvashatók, nem csak az ADS-beli
  - A műveletek és a feladat-specifikus függvények algoritmusait meg kell adni

Példa

- (Majdnem) teljes fa indexelése tömbben:
  - Szintfolytonosan:
    - index(bal(a)) $= 2 *$ index(a)
    - index(jobb(a)) $= 2 *$ index(a) $+ 1$
  - Feltételezzük, hogy az indexelés 1-ről indul



Ami még hiányzik

- Az implementáció szintje egy programnyelv, illetve fejlesztő környezet megválasztását jelenti
  - gyakorlatokon lesz
- A fizikai ábrázolás szintjén azt vizsgáljuk, hogy az adatszerkezet hogyan képeződik le a memória bájtjaira
  - ezzel ebben a tárgyban alapvetően nem foglalkozunk
  - kicsit lesz róla szó a gyakorlatokon

Az adatszerkezetek osztályozása

- Az adatszerkezet egy $\langle A, R \rangle$ rendezett pár, ahol
  - A : az adatelemek véges halmaza
  - R : az A halmazon értelmezett valamilyen reláció
  - $R \subseteq (A \times A)$

Az adatszerkezetek osztályozása

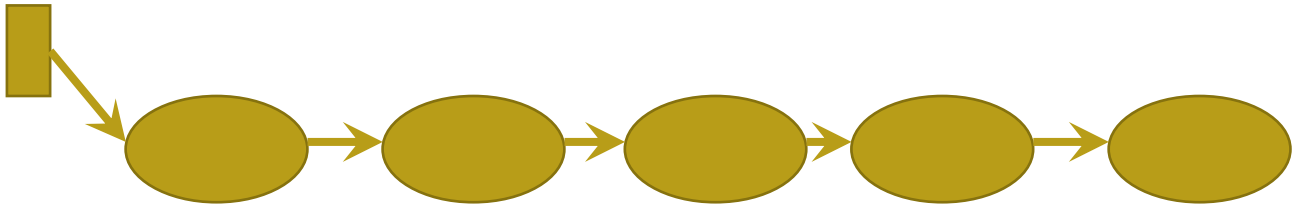
- Az adatelemek típusa szerint
  - Homogén
    - Az adatszerkezet valamennyi eleme azonos típusú.
  - Heterogén
    - Az adatszerkezet elemei különböző típusúak lehetnek.

Az adatszerkezetek osztályozása

- Az elemek közti R reláció szerint
  - Struktúra nélküli
    - Az egyes adatelemek között nincs kapcsolat. Nem beszélhetünk az elemek sorrendjéről (Pl. halmaz).
  - Asszociatív címzésű
    - Az adatelemek között lényegi kapcsolat nincs.
    - Az adatszerkezet elemei tartalmuk alapján címezhetők.
  - Szekvenciális
    - Szekvenciális adatszerkezetben az egyes adatelemek egymás után helyezkednek el.
    - Az adatok között egy-egy jellegű a kapcsolat: minden adatelem csak egy helyről érhető el és az adott elemről csak egy másik látható.
    - Két kitüntetett elem az első és az utolsó.

Az adatszerkezetek osztályozása

- Szekvenciális
  - Intuitív ADT és ADS szint



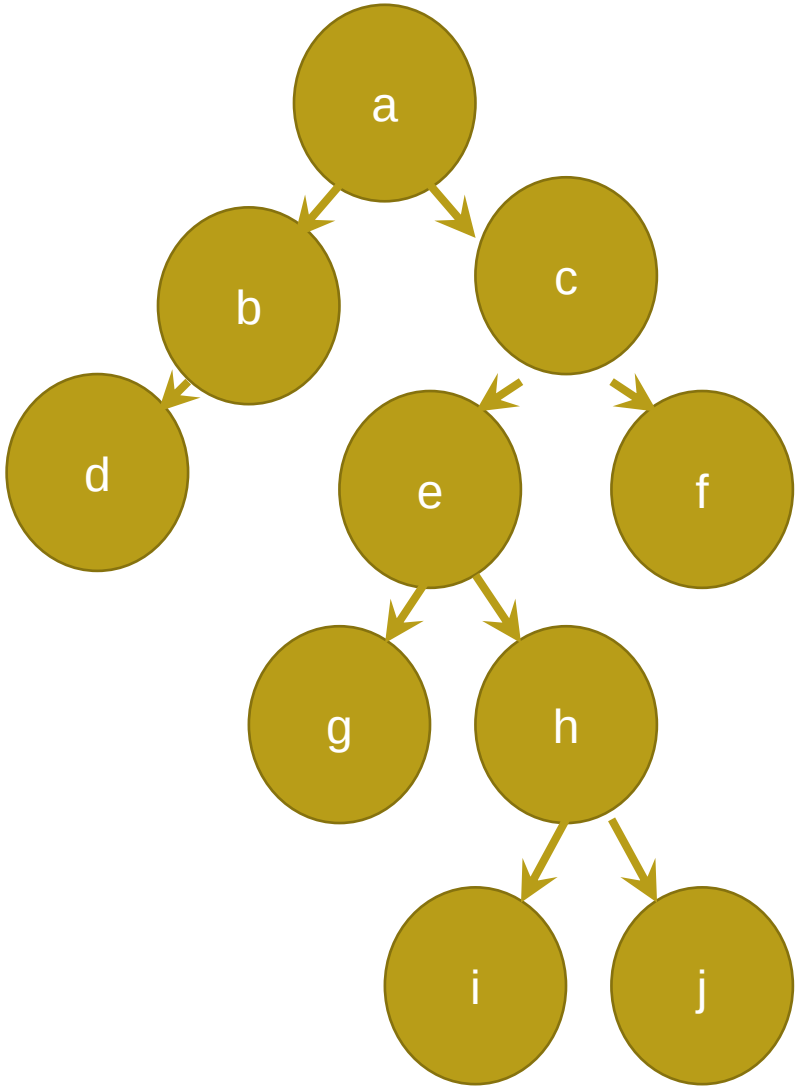
Végigmehetünk az elemeken egymás után.
Lehetőség van a módosítás, törlés, beszúrás műveletekre.

- Definíció: A szekvenciális adatszerkezet olyan $\langle A, R \rangle$ rendezett pár amelynél az $R \subseteq (A \times A)$ reláció tranzitív lezártja teljes rendezési reláció
- Az $R \subseteq (A \times A)$ reláció tranzitív lezártja az a reláció, mely tranzitív, tartalmazza R-et, és a lehető legkevesebb elemet tartalmazza
- Megadása:
  1. $R' = R \; U \; (R \circ R)$
  2. Ha $R \neq R'$, akkor folyt. 1.-nél,
     különben $R' = R_T$, a tranzitív lezárt

Az adatszerkezetek osztályozása

- Hierarchikus
  - A hierarchikus adatszerkezet olyan $\langle A, R \rangle$ rendezett pár, amelynél van egy kitüntetett r elem, ez a gyökérelem, úgy, hogy:
    - r nem lehet végpont
    - $\forall\, a \in A \setminus \{r\}$ elem egyszer és csak egyszer végpont
    - $\forall\, a \in A \setminus \{r\}$ elem r-ből elérhető
  - Az adatelemek között egy-sok jellegű kapcsolat áll fenn.
  - Minden adatelem csak egy helyről érhető el, de egy adott elemből akárhány adatelem látható
    - Például fa, összetett lista, B-fa

Bináris fa



Az adatszerkezetek osztályozása

- Hálós
  - A hálós adatszerkezet olyan $\langle A, R \rangle$ rendezett pár, amelynél az R relációra semmilyen kikötés nincs
  - Az adatelemek között a kapcsolat sok-sok jellegű: bármelyik adatelemhez több helyről is eljuthatunk, és bármelyik adatelemtől elvileg több irányban is mehetünk tovább
    - Például gráf, irányított gráf

Az adatszerkezetek osztályozása

- Az adatelemek száma szerint
  - Rögzített kapacitású (statikus elemszámú)
    - Egy statikus elemszámú adatszerkezetet rögzített számú adatelem alkot.
    - A feldolgozás folyamán az adatelemek csak értéküket változtathatják, de maga a szerkezet, az abban szereplő elemek száma változatlan
    - Következésképpen az adatszerkezetnek a memóriában elfoglalt helye változatlan a feldolgozás során
  - Dinamikus kapacitású (dinamikus elemszámú)
    - Egy dinamikus elemszámú adatszerkezetben az adatelemek száma egy adott pillanatban véges ugyan, de a feldolgozás során tetszőlegesen változhat
    - Dinamikus elemszámú adatszerkezetek lehetnek rekurzív vagy nem-rekurzív, lineáris vagy nem-lineáris struktúrák

Az adatszerkezetek osztályozása

- Az adatelemek száma szerint
  - Dinamikus kapacitású (dinamikus elemszámú)
    - Egy adatszerkezet rekurzív, ha definíciója saját magára való hivatkozást tartalmaz.
      - Ha egyetlen ilyen hivatkozás van, akkor lineáris a struktúra, ha több, akkor nem-lineáris
    - A dinamikus adatszerkezetek feldolgozása során az adatelemek száma változik így egy-egy elemnek területet kell allokálnunk, illetve a lefoglalt területeket fel kell szabadítanunk
      - Ezzel felvetődik a tárolóhely újrahasznosításának problémája

Az adatszerkezetek osztályozása

- Reprezentáció szerint
  - Az egyes adatszerkezetek tárolhatók
    - folytonosan
    - szétszórt módon
  - A leképezés és a műveletek megvalósítása annál egyszerűbb, minél jobban illeszkedik az adatszerkezet a tárolási szerkezetre.
    - Az asszociatív és a string szerkezetek nagyon jól tárolhatók folytonosan
    - A hierarchikus és hálós szerkezetek elsősorban szétszórt tárolással kezelhetők
    - De például a verem és a sor mindkét módon tárolható hatékonyan

Az adatszerkezetek osztályozása

- Reprezentáció szerint
  - Folytonos ábrázolású
    - A központi tárban a tárelemek egymás után helyezkednek el.
    - Az adattételek tárolási jellemzői (típus, ábrázolási forma, méret) azonosak.
    - Ismert az első elem címe, ehhez képest bármely elem címe számítható.
      - Legyen minden elem hossza H byte és jelölje $\mathrm{loc}(a1)$ az első adatelem címét.
      - Ekkor $\mathrm{loc}(aN) = \mathrm{loc}(a1) + (N-1) * H$
  - Szétszórt ábrázolású
    - A tárelemek véletlenszerűen helyezkednek el
    - Közöttük a kapcsolatot az teremti meg, hogy minden elem tartalmaz más elemek elhelyezkedésére vonatkozó információt
      - az elemek címét

OOP – Elvek, alapfogalmak

Következő téma