Pázmány Péter Katolikus Egyetem
Információs Technológiai és Bionikai Kar

Adatszerkezetek és algoritmusok

Bevezetés

2017. szeptember 12.

Általános tudnivalók

- Előadás
  - 2 óra, kedd 10.15-12.00
- Laborgyakorlat
  - 3 óra, szerda 10.15-13.00
- Előismeretek:
  - Kötelező: Bevezetés a programozásba I-II.
  - Diszkrét matematika bizonyos fogalmai
- Kreditpont: 6
- A Wiki oldalról letölthetők a fájlok az órák után

Általános tudnivalók

- Irodalom:
  - Cormen, T. H. – Leiserson, C. E. – Rivest, R. L. – Stein, C.: Új algoritmusok. Scolar Kiadó, Budapest, 2003.
  - Rónyai, L. – Ivanyos, G. – Szabó, R.: Algoritmusok. Typotex Kiadó, Budapest, 1999.

Előadások

- Tornai Kálmán
- tornai.kalman@itk.ppke.hu
- 232-es szoba
  - Előre egyeztetett időpontban
  - Kedden előadás után

Gyakorlatok

- Szerdán, párhuzamosan, minden csoportnak
- Csoportbeosztás a felmérő alapján

Követelmények

- Jegy: Vizsga + gyakorlati jegy (1:1)
- Vizsga előfeltétele az érvényes (>1) gyakorlati jegy
- Az előadások látogatása kötelező
- A követelményrendszerben foglaltakat mindenkinek önállóan kell teljesítenie
  - Házi feladat, ZH, vizsga, stb. esetén a csoportmunka nem elfogadható

Követelmények

- Gyakorlati jegy
  - Részletes leírás megtalálható a Wiki oldalon

TESZT MEGÍRÁSA

Félév időrendje – terv

Hét	Kedd	Előadás anyaga	Szerda	Gyakorlat anyaga
1	2017.09.12.	Bevezetés	2017.09.13.	Ismétlés - Pointer, tömbök, referencia
2	2017.09.19.	OOP	2017.09.20.	SPORTNAP
3	2017.09.26.	Verem, Sor, Prioritásos sor, Lengyelforma	2017.09.27.	OOP, Verem
4	2017.10.03.	Tömbök, Láncolt Lista, Hierarchikus adatszerkezetek	2017.10.04.	Sor, Lengyelforma
5	2017.10.10.	Bináris Fa, Bináris keresőfa	2017.10.11.	Dinamikus Láncolt lista
6	2017.10.17.	Kiegyensúlyozott fák, AVL Fa	2017.10.18.	Bináris keresőfa
7	2017.10.24.	Papíros ZH	2017.10.25.	AVL fa
8	2017.10.31.	ŐSZI SZÜNET	2017.11.01.	ŐSZI SZÜNET
9	2017.11.07.	Piros-Fekete fa	2017.11.08.	Piros-fekete fa
10	2017.11.14.	Rendezés, Lassú rendezők, gyorsrendezés	2017.11.15.	Lassú rendezések, gyorsrendezés
11	2017.11.21.	Kupac, Kupacrendezés, Edényrendező, Radix rendező	2017.11.22.	Kupacrendezés, prioritásos sor
12	2017.11.28.	Összefésülő rendező, Batcher-féle páros páratlan rendező, Leszámláló rendezés	2017.11.29.	Összefésülő rendezés, edény, radix
13	2017.12.05.	Hasításos technikák	2017.12.06.	Hash tábla
14	2017.12.12.	Külső rendezők, 2-3 fa, B-fa	2017.12.13.	Géptermi ZH
			2017.12.20.	Géptermi pót ZH

A tárgy célja

- Egyszerű, hasznos típusok/adatszerkezetek és
- Általánosan használt algoritmusok megismerése,
- Az adatszerkezetek és algoritmusok hatékonyságának elemzése
- Programozási gyakorlat szerzése

Mi kell ehhez

- Előadásra járni
  - Megtanulni az elhangzottakat
- Gyakorlatra járni
  - Kipróbálni az előadáson tanultakat
- Házi feladatokat írni
  - Gyakorolni – sokat!

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

Tematika

- I. ALAPFOGALMAK
  - Az adattípus absztrakciós szintjei
- II. ALAPVETŐ ADATSZERKEZETEK
  - Tömb
  - Verem
  - Sor
  - Elsőbbségi (prioritásos) sor és a kupac (heap)
  - Listák
  - Hierarchikus adatszerkezetek és bináris fák

Tematika

- III. KERESÉSEK
  - Bináris keresőfák
  - AVL fák
  - Piros-fekete fák
  - 2-3 fák, B-fák
  - Hasításos technikák (hash-elés)

Tematika

- IV. RENDEZÉSEK
  - Algoritmusok műveletigényének elemzése - ismétlés
  - Az összehasonlításos rendezők alaptétele
  - Három „lassú" rendezés:
    buborék, beszúró és max. kiv. rendezés
  - Kupacrendezés (heap sort)
  - Gyorsrendezés (quick sort)
  - Összefésülő rendezés (merge sort) és külső rendezések
  - Edényrendezések

Típusok

Kezdetben

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

Ma

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

A típus-absztrakció szintjei

- Absztrakt adattípus (ADT)
  - semmit nem feltételez a belső szerkezetről
  - 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

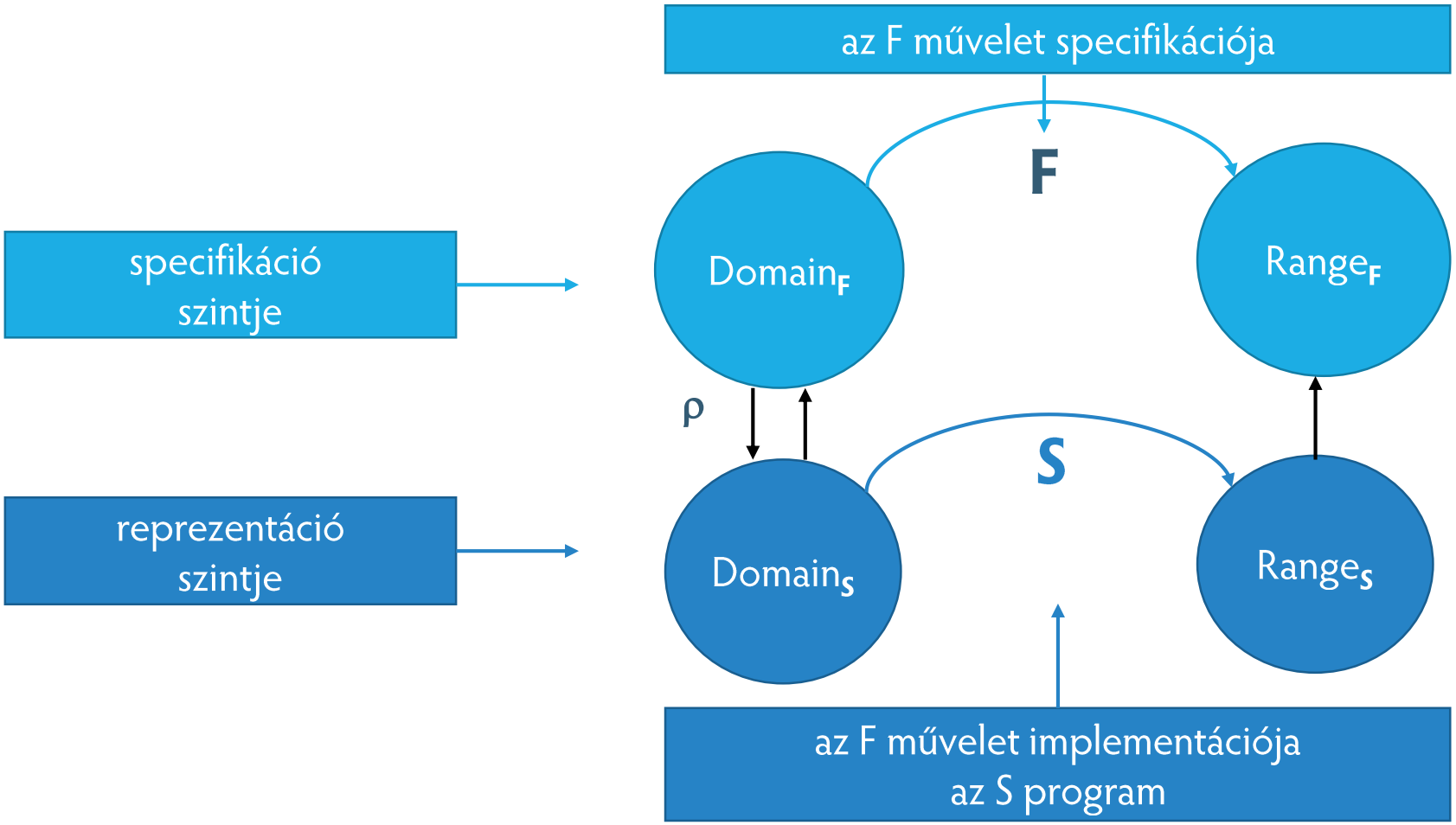
Típus-specifikáció

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

A típus specifikáció és a típus kapcsolata



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

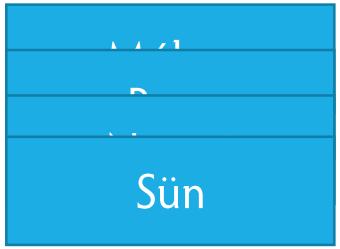
- A típus szemléletének ez a legmagasabb szintje
- Semmilyen feltételezéssel nem élünk a típus szerkezetéről, megvalósításáról!
- A specifikációban csak tisztán matematikai fogalmakat használhatunk
- Ez a szint nem a formalizálás mértékétől absztrakt; lehet informálisan is gondolkodni, beszélni ADT szinten!

Az ADT algebrai specifikációja

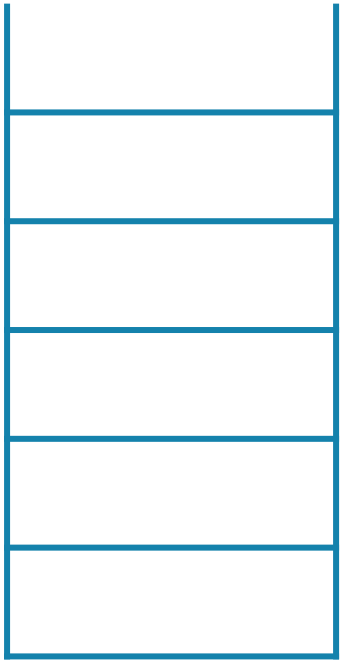
- Részei:
  - típusérték halmaz
  - műveletek (mint leképezések)
  - megszorítások (értelmezési tartományok)
  - axiómák
- Kérdések:
  - helyesség (ellentmondásmentesség)
  - teljesség /nehéz
  - redundancia /nem fontos

Példa: VEREM

- LIFO: last-in, first-out
- Köznapi fogalma



- Kimászás sorrendje



Példa: A verem ADT axiomatikus leírása

E alaptípus feletti V verem típus jellemzése

- Műveletek
  - empty $\rightarrow V$
  - isempty $V \rightarrow L$
  - push $V \times E \rightarrow V$
  - pop $V \rightarrow V \times E$
  - top $V \rightarrow E$
- Megszorítások:
  - pop és top értelmezési tartománya
    - $V \backslash \{\text{empty}\}$

- Műveletek jelentése
  - Üres verem létrehozása
  - Üres a verem?
  - Elem betétele a verembe
  - Elem kivétele a veremből
  - Felső elem lekérdezése
- Figyelem!
  - A műveletek között nem szerepel „isfull" művelet!

Példa: A verem ADT axiomatikus leírása

- Axiómák:
  - isempty(empty)
    - Vagy: $v = \text{empty} \rightarrow \text{isempty}(v)$
  - $\text{isempty}(v) \rightarrow v = \text{empty}$
  - $\neg\text{isempty}(\text{push}(v, e))$
  - $\text{pop}(\text{push}(v, e)) = (v, e)$
  - $\text{push}(\text{pop}(v)) = v$
  - $\text{top}(\text{push}(v, e)) = e$

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

Példa: A verem ADT funkcionális leírása

- Matematikai reprezentáció
  - a verem rendezett párok halmaza
    $v = \{(e1, t1), \ldots (ei, ti) |\ a\ tj$ komponensek különbözőek$\}$
  - komponens: a veremben elhelyezett (push) érték
  - komponens: a verembe helyezés (push) időpontja
- Megszorítás (invariáns)
  - az idő értékek különbözők
- A valóságban nem így implementáljuk!

Példa: A verem ADT funkcionális leírása

- A „pop" specifikációja:
  - Állapottér
    - $V \times E$ (Állapottér típusai: $V = \{(e_i, t_i), \dots\}$
    - $v \quad e$ (Állapottér változói)
  - Paramétertér
    - V és v'
  - Előfeltétel és utófeltétel:
  - Ef $= (v = v' \wedge v' \neq \emptyset)$
  - Uf $= \begin{pmatrix} (v = v' \backslash \{(e_j, t_j)\}) \wedge (e = e_j) \wedge ((e_j, t_j) \in v') \\ \wedge (\forall i((e_i, t_i) \in v' \wedge i \neq j) : t_j > t_i) \end{pmatrix}$

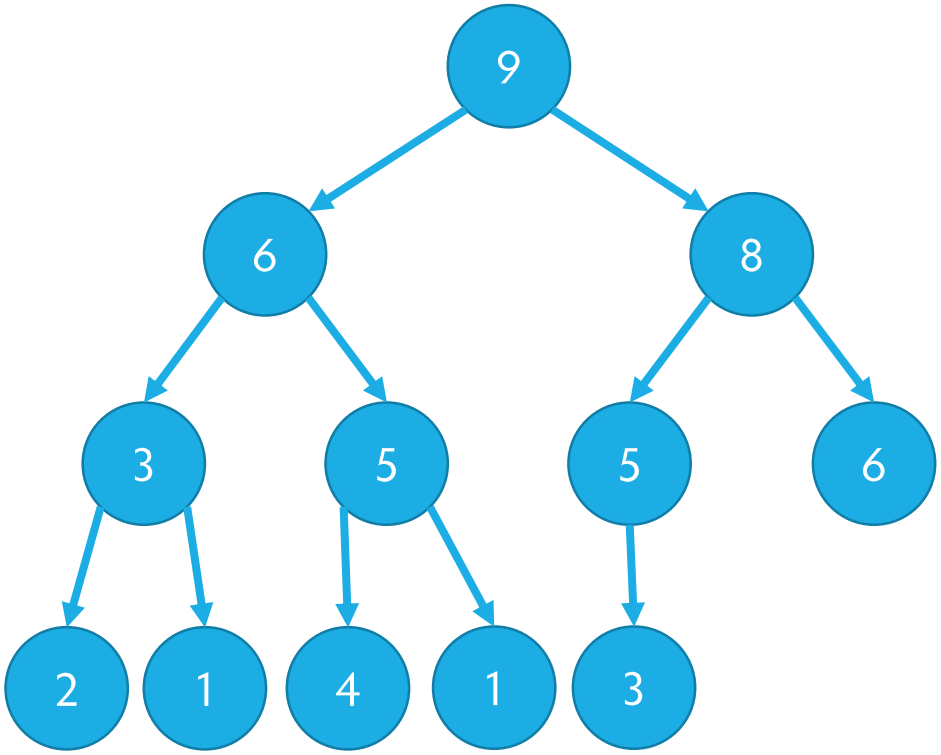
Algoritmusok ADT szinten

- 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

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



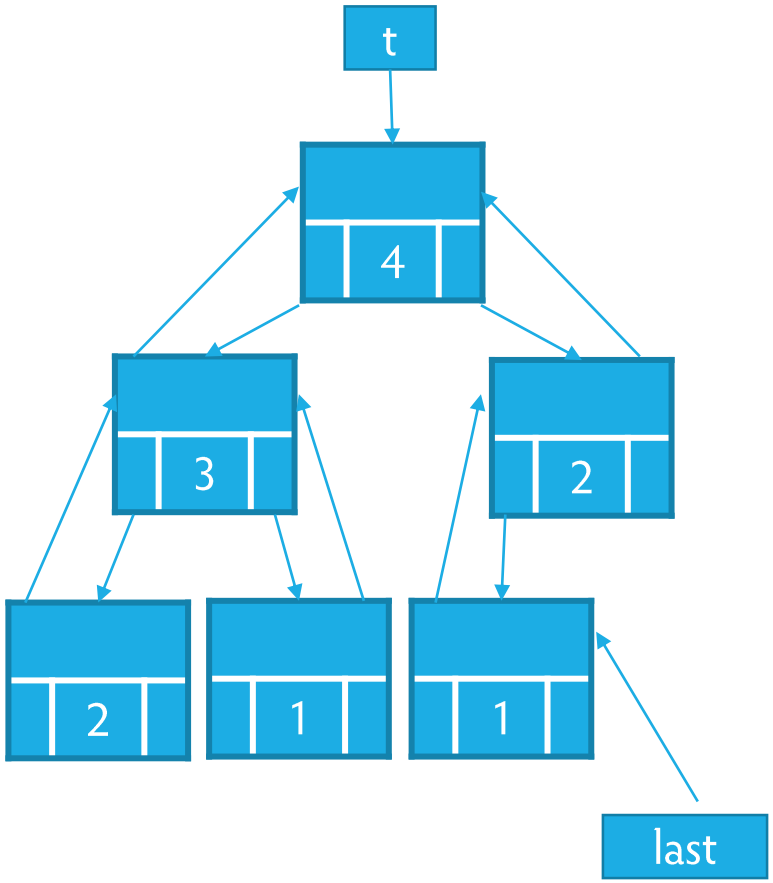
- Kupac
  - A prioritásos sor szokásos reprezentációja
  - bináris fa
  - majdnem teljes
  - balra tömörített
  - a szülő csúcsokban nagyobb értékek vannak, mint a gyerek csúcs(ok)ban

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

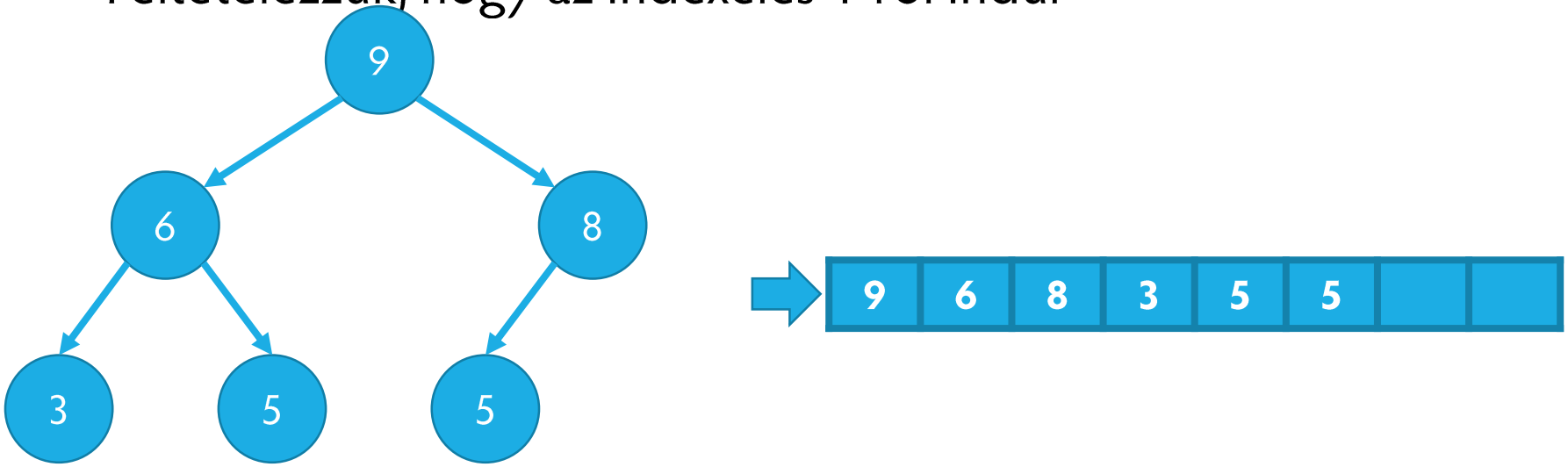


Aritmetikai reprezentáció

- 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:
    - $\text{index}(\text{bal}(a)) = 2 * \text{index}(a)$
    - $\text{index}(\text{jobb}(a)) = 2 * \text{index}(a) + 1$
  - 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

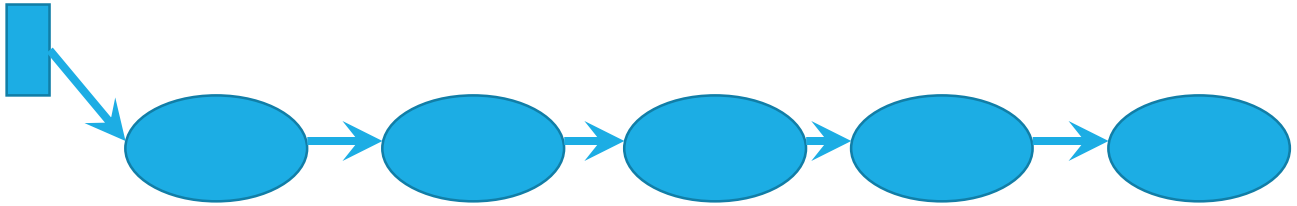
- Az adatelemek típusa szerint
  - Homogén
    - Az adatszerkezet valamennyi eleme azonos típusú.
  - Heterogén
    - Az adatszerkezet elemei különböző típusúak lehetnek.

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



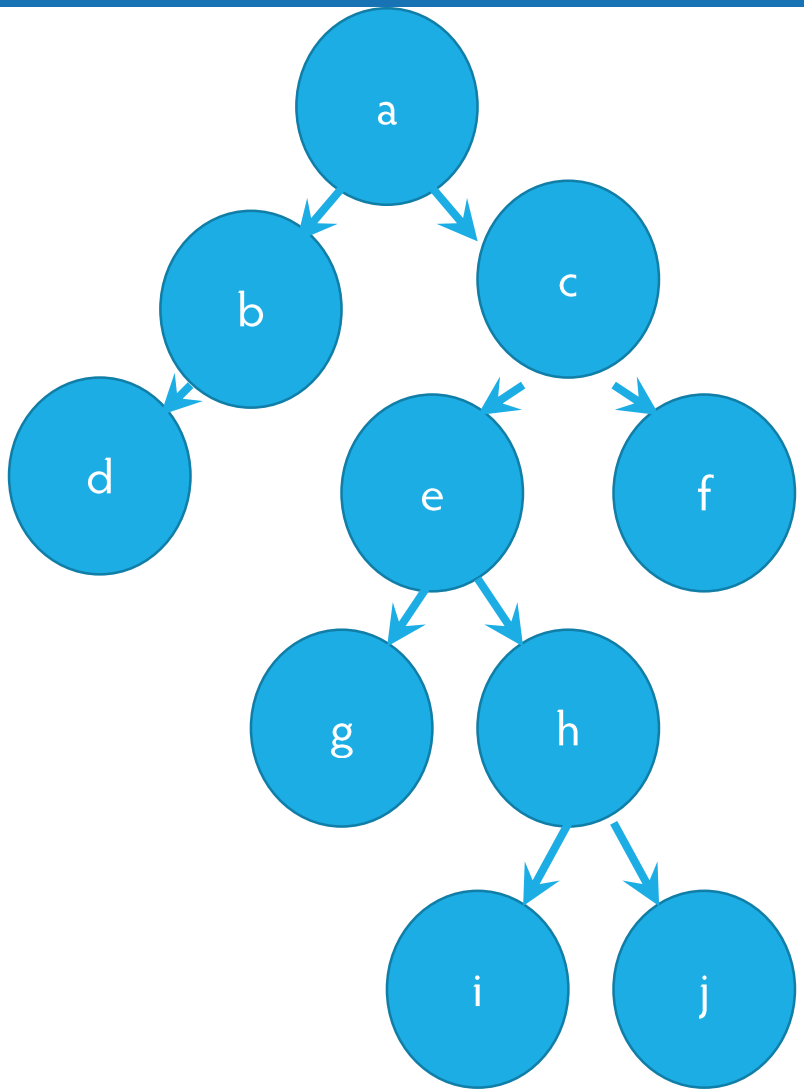
Végigmehetünk az elemeken egymás után.
Lehetőség van a módosítás, törlés, beszúrás műveletekre.

  - 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
  - Statikus
    - Egy statikus 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
    - Egy dinamikus adatszerkezetben az adatelemek száma egy adott pillanatban véges ugyan, de a feldolgozás során tetszőlegesen változhat
    - Dinamikus adatszerkezetek lehetnek rekurzív vagy nem-rekurzív, lineáris vagy nem-lineáris struktúrák

Az adatszerkezetek osztályozása

- Az adatelemek száma szerint
  - Dinamikus
    - 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 loc$(a1)$ az első adatelem címét.
      - Ekkor loc$(aN) =$ 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 Összefoglaló

Következő alkalommal