1. tétel

Adatszerkezetek és algoritmusok vizsga

Frissült: 2013. január 28.

Az adattípus absztrakciós szintjei

Típus: egy adat által felvehető lehetséges értékek + a rajta értelmezett műveletek.

Típusspecifikáció

Egy adat külső jellemzésére szolgál (interfész).
Áll:

- típusérték-halmaz - Az adat által felvehető értékek T halmaza
- típusműveletek - T-n értelmezett feladatok.

Megadása:

- Algebrai - axiómák megadásával
- Funkcionális - elő és utófeltétel megadása

Típus

A típus reprezentációja (típusértékek ábrázolása):

- Ábrázoló elemek H halmaza. (típus-szerkezet)
- Az ábrázoló elemek és a típusértékek kapcsolatát leíró leképezés:

$$\varrho : H \to T, \qquad \varrho \subseteq H \times T$$

- Típus invariáns - kiválasztja a hasznos ábrázoló elemeket

$$I : H \to L, [I]$$

Típus implementációja (műveletek helyettesítése):
Nem a típusértékekkel, hanem az azokat ábrázoló elemekkel működő *programok*.

Absztrakt adattípus és megadási módjai

Absztrakt adattípus (ADT): Absztrakt, mert semmilyen feltételezéssel nem élünk a típus szerkezetéről, megvalósításáról!

A specifikációban csak tisztán matematikai fogalmakat használunk.

Ez a típus szemléletének a legmagasabb szintje.

Egységbe van zárva (enkapszláció.)

ADT algebrai specifikációja

Részei

- típusérték halmaz
- műveletek (leképezések)
- megszorítások (értelmezési tartományok)
- axiómák

Kérdés lehet specifikációnál: helyesség, teljesség, redundancia?

ADT funkcionális specifikációja

A típust matematikailag írjuk le, a matematikai reprezentációját használjuk.

Részei

- típusérték-halmaz
- műveletek
- állapottér
- paramétertér
- előfeltétel
- utófeltétel

Ez semmilyen módon nem kell, hogy utaljon a típus ábrázolási módjának megválasztására a megvalósítás során!

Példa: Verem

Verem, mint absztrakt adatszerkezet axiomatikus leírása

E adattípus feletti V verem típus jellemzése:
Műveletek:

- empty: $\rightarrow V$ (üres verem - létrehozás)
- isempty: $V \rightarrow L$ - üres-e a verm?
- push: $V \times E \rightarrow V$ - elem betétele a verembe
- pop: $V \rightarrow V \times E$ - elem kivétele a veremből
- top: $V \rightarrow E$ - legfelső elem lekérdezése

Megszoriítások: *pop* és *top* értelmezési tartománya: $V \setminus \{\text{empty}\}$
Axiómák:

- isempty(empty) vagy v=empty $\rightarrow$ isempty(v)
- isempty(v) $\rightarrow$ v = empty
- $\neg$ isempty(push(v,e))
- pop(push(v,e))=(v,e)
- push(pop(v))=v
- top(push(v,e))=e

ADT funkcionális specifikációja

Matematikai reprezentáció:
a verem rendezett párok halmaza: $v = \{(e_1, t_1), ...(e_i, t_i) | \text{a} t_j \text{komponensek különbözőek}\}$

- 1. komponens: a veremben elhelyezett (push) érték
- 2. komponens: a verembe helyezés (push) időpontja

Megszoriítás (invariáns): az időértékek kbözők.
Nem így szokás implementálni!

Példa - a "pop" specifikációja

Absztrakt adatszerkezet

Széles körben használják az oktatásban, könyvekben. A típus alapvető (absztrakt) szerkezetét egy irányított gráffal ábrázoljuk, ahol a csúcsok az adatelemeket, az élek pedig a rákövetkezési relációt jelentik.
Műveletek itt is vannak, hatásuk szemlélhethető az ADS gráf változásaival.
Absztrakt adatszerkezetre példa: kupac.

$A = V \times E$ - állapottér

$v \quad e$

$B = V$ - a kezdeti állapot értékei

v'

$Ef = (v = v' \wedge v' \neq \emptyset)$ - előfeltétel

- utófeltétel:

$Uf = ((v = v' \setminus \{(e_j, t_j)\}) \wedge (e = e_j) \wedge$
$((e_j, t_j) \in v') \wedge (\forall i ((e_i, t_i) \in v' \wedge i \neq j): t_j > t_i))$

Reprezentációja

Absztrakt memóriában. Ábrázolható:

- láncolt ábrázolás pointerekkel
- aritmetikai ábrázolás cím/index függvényekkel
- vegyes

Mindegyik megadja az adatelemek közti rákövetkezési relációt, és további rákövetkezések is kiolvashatók.

Láncolt ábrázolás

Az ADS gráf éleit pointerekkel ábrázoljuk.

Itt már meg kell adni a műveletek algoritmusait is, beleértve a kiszámító algoritmusokat is.

Példa: kupacokat láncoltan ábrázoljuk.

Aritmetikai ábrázolás

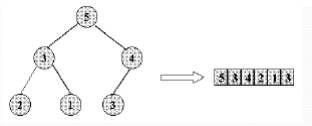
Az adatelemeket folyamatosan elhelyezzük az absztrakt memóriában vagy egy ugyanilyen alaptípusú **vektorban**. Az elemek közti rákövetkezési relációt egy cím/index függvénnyel adjuk meg (ezekből további rákövetkezések is kiolvashatók). A műveletek, és a feladat specifikus függvények algoritmusait is meg kell adni.

Példa: majdnem teljes bináris fa indexfüggvénye:

Szintfolytonosan

$$\text{index}((bal(a)) = 2 * \text{index}(a)$$

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



Adatszerkezetek osztlályozása

Definíció. Adatszerkezet: egy $\langle A, R \rangle$ rendezett pár ahol A: adatelemek (véges) halmaza, R halmazon értelmezett reláció ($R \subseteq A \times A$)

Adatelemek típusa szerint

- Homogén - adatszerkezet minden eleme azonos típusú
- Heterogén - lehetnek kböző típusúak is

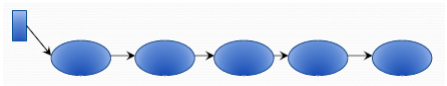
Elemek közti R reláció szerint

- Struktúra nélküli - elemek közt nincs kapcsolat, nincs sorrend.
- Asszociatív címzésű - adatelemek között lényegi kapcsolat nincs, az elemek tartalmuk alapján címezhetők.
- Szekvenciális - az egyes adatelemek egymás után helyezkednek el. 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 elemeken végigmehetünk egymás után, tudunk módosítani, beszúrni, törölni.

  Definíció szerint: A szekvenciális adatszerkezet olyan $\langle A, R\rangle$ rendezett pár, melynél az R reláció tranzitív lezártja teljes rendezési reláció.

  Példa: egyszerű lista

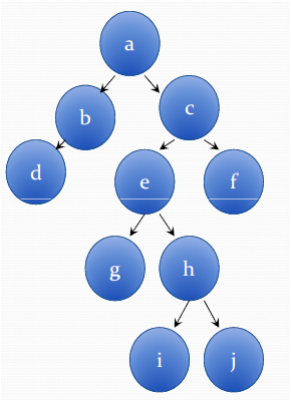


- Hierarchikus - adatelemek közt egy-sok jellegű kapcsolat áll fenn, minden adatelem csak egy helyről érhető el, de egy adott elemből akárhány adatelem látható.

  Def. A hiearchikus adatszerkezet olyan $\langle A, R\rangle$ rendezett pár, amelynél van egy kitüntett r elem, a **gyökérelem**, és

  - r nem lehet végpont
  - $\forall a \in A \setminus \{r\}$ elem egyszer és csak egyszer lehet végpont
  - $\forall a \in A \setminus \{r\}$ elem r-ből elérhető

  Példa: fa, összetett lista, bináris fa (képen), B-fa.



- Hálós - az adatelemek közt a kapcsolat sok-sok jellegű, bármelyik adatelemhez több helyről is eljuthatunk, bármely adatelemtől elvileg több irányból is mehetünk tovább.

  A hálós adatszerkezet olyan $\langle A, R \rangle$ rendezett pár, melynél az R relációra semmilyen kikötés nincs.

  Példa: gráf, irányított gráf.

Adatelemek száma szerint

- Statikus - rögzített számú adatelem. A feldolgozás során az adatelemek csak az értéküket változtathatják, a szerkezet változatlan. A memóriában elfoglalt hely is változatlan.

- Dinamikus - az adatelemek száma adott pillanatban véges, DE a feldolgozás során tetszőleges változhat. Itt lehetnek rekurzív (definíciója saját magára való hivatkozást tartalmaz) / nem rekurzív , lineáris (rekurzív, csak egy db belső hivatkozással) / nem lineáris struktúrák.

Reprezentáció szerint

- Folytonos - a központi tárban a tárelemek egymás után helyezkednek el (tárolási jellmezők azonosak). Ismert az első elem címe, ehhez képest bármely elem címe számítható.

  Legyen minden elem hossza H byte és jelölje $loc(a_1)$ az első adatelem címét. Ekkor $loc(a_N) = loc(a_1) + (N-1) * H$

  Pl. sztring, vektor - asszociatív szerkezetek

- Szétszórt ábrázolású - tárelemek véletlenszerűen helyezkednek el, minden elem tartalmaz más elemek elhelyezkedésére vonatkozó információt - mutatót.

  Pl. fa, gráf - hiearchikus, hálós szerkezetek

  Verem, sor - mindkettő lehet.