# 10. gyakorlat – Sor

## ADT

( igaz vagy hamis)

|  |
| --- |
|  |
| (kétféle reprezentációt is veszünk, lent) |
|  |
|  |
|  |

Sok helyen használjuk, például szimulációs modelleknél, eloszlás szerint jönnek a felhasználók a kiszolgálóegységekhez, kiszolgálási modellek. Szimuláció, gráf​bejárás.

Veremhez hasonló: Közepéhez nem tudunk nyúlni.

Eltérés: Nem egy végén dolgozunk, hanem kettőn.

|  |  |  |
| --- | --- | --- |
|  |  |  |

## ADS

Reprezentáció: Valamilyen lineáris adatszerkezet. Vektor vagy láncolt lista.

### Sor vektoros reprezentációja

 ( – first, – length)

Kell még egy kezdeti méret: , konstans.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| α | α | α | α |  |  |  |  |

Kitörlések után nem akarjuk balra tolni az egész tömböt, drága lenne sok elem esetén. Helyette hernyó:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | α | α |  |  |  |  |

Ekkor egy hármas írja le az adatszerkezetet: A tömb, az első elem helye () és a sor hossza (). Ha körbeér a hernyó, elég egy -nel helyrerakni (lent).

Konstruktor:

|  |
| --- |
|  |
|  |  |
|  |

Felvétel:

|  |
| --- |
|  |
|  |  |
|  |
|  | Megtelt. Lefoglalunk kétszer akkora tárhelyet. |
|  |
|  |  |
|  |

Törlés:

|  |
| --- |
|  |
|  |  |
|  |
|  |  |

### Sor láncolt listás reprezentációja

Milyen legyen? Nézzük meg mire lesz szükség.

Törlés: Kiláncolja és kész. Add: Fejelemet felülírjuk (bele kerül ), majd begenerálunk mögé egy új fejelemet.

Tehát egyirányú ciklikus lista elég. Szellemes reprezentáció nagyon sokat számít. Hihetetlenül felgyorsítja a programot.

Másik lehetséges megoldás: Két pointer: eleje, vége. Tárolni kell a hosszt is (). Ciklikusság nem kell, fejelem helyett őrszemelem (farok​elem). Új elemet mindig elé rakjuk, a fenti trükkel: beírjuk x-et a régi fejelembe, felveszünk új fejelemet, berakjuk a régi fejelem után. Ez kicsit hatékonyabb.

Struktogram HF.

Prioritási sor előadáson részletesen lesz, gyakorlaton nem.

# Set (Halmaz)

|  |
| --- |
|  |
| (reprezentáció) |
|  (beszúr) (benne van-e) |

Triviális reprezentációk: Lineáris adatszerkezetek nyilván.

### Rendezetlenek hátránya

Keresés () drága, mert nem rendezett. Lineáris.

Törlés és beszúrás olcsó, mert nem kell újrarendezni.

Halmaz vektoros/tömbös reprezentációja: Törlés: Utolsó elemet olcsón átrakjuk a törölt elem helyére. De itt is kell keresni.

Halmaz láncolt listás reprezentációja: Hova szúrjuk be az új elemet, mármint melyik végére? Elejére.

### Halmaz bináris keresőfás reprezentációja

Minden csúcsra igaz, hogy bal részfájában nála csupa kisebb elemek vannak, jobb részfájában pedig csupa nagyobbak.

 (: Minden csúcsnak van adatrésze () és két pointere (, ). Mind publikus.)

|  |
| --- |
|  |
|  |
|  |

A keresés () olcsóbb. Nem a halmaz elemszámával arányos, hanem /\*annak logaritmusával.\*/

|  |
| --- |
|  |
|  |  |
|  |
|  |
|  |  |
|  |  |
|  |

()

Beszúrás:

|  |
| --- |
|  |
|  |  |
|  |

^TODO^

|  |
| --- |
|  |
|  |  |
|  |
|  |  |  |  |
|  |  |  |

Törlés HF

Gyakorlat vége.