Algoritmusok gyakorlat

6. hét, október 21.

6. gyakorlat

Az *adatszerkezet* adatok tárolására és szervezésére szolgáló módszer (konstrukció), amely lehetővé teszi a(z adatokhoz való) hatékony hozzáférést és (a) módosításokat.

Tanult absztrakt adatszerkezetek: lista, verem, sor, prioritás sor, kupac

6. gyakorlat

Leggyakoribb műveletek (H: adatok dinamikus halmaza, k: kulcsmező)

KERES(H,k)

BESZÚR(H,k) (Push)

TÖRÖL(H,k) (Pop)

MIN(H)

MAX(H)

ELŐZŐ(H,k)

KÖV(H,k)

6. gyakorlat

Adatszerkezet: absztrakt adatszerkezet konkrét megvalósítása.

- Egy programozási nyelven is lehet több megvalósítás,
- bizonyos műveletek hatékonyabbak,
- megfelelő absztrakt adatszerkezet megválasztása fontos az algoritmushoz,
- megfelelő adatszerkezet megválasztása kulcs az implementáció optimális futásidejéhez!

6. gyakorlat

Lista: az adatok lineáris sorrendben követik egymást. Egy kulcs többször is előfordulhat.

Példa: közvetlen elérésű tömb megvalósítás esetén mi a műveletigénye a keresésnek, törlésnek, beszúrásnak, *i*. elem kiolvasásának?

Átméretezés?

Láncolt listás megvalósításnál mik az előző műveletek igényei?

6. gyakorlat

Verem és sor: Olyan listák ahol a beszúrás és törlés csak meghatározott pozícióban történhet.

- Verem: legutoljára beszúrt elemet vehetjük csak ki (last-in, first-out = LIFO)
- Sor: legrégebbi elemet vehetjük csak ki (first-in, first-out = FIFO)

Prioritás sor: Kulcsok érkezésének sorrendje lényegtelen, mindig a maximális (minimális) elemet akarjuk kivenni

6. gyakorlat

Adott a 3; 6; 10; 8; 1; 9; 7 számsorozat, melyeket ebben a sorrendben tárolunk el. Adjuk meg milyen sorrendben vehetjük ki az elemeket verem, sor, prioritás sor esetén!

6. gyakorlat

Egy V veremben kezdetben egyetlen szám, az 1 van. Utána n-szer végrehajtjuk a

Veremből (V, x)

y:=2x;

z := 2x+1;

Verembe (V, y);

Verembe (V, z);

műveleteket. Mi lesz ezután a Veremből (V, x) eredménye? Mi lesz az eredmény, ha sort használunk verem helyett?

6. gyakorlat

Szimuláljunk veremmel sort!

6. gyakorlat

Elem hozzáadása a sorhoz **(Sorba** művelet):

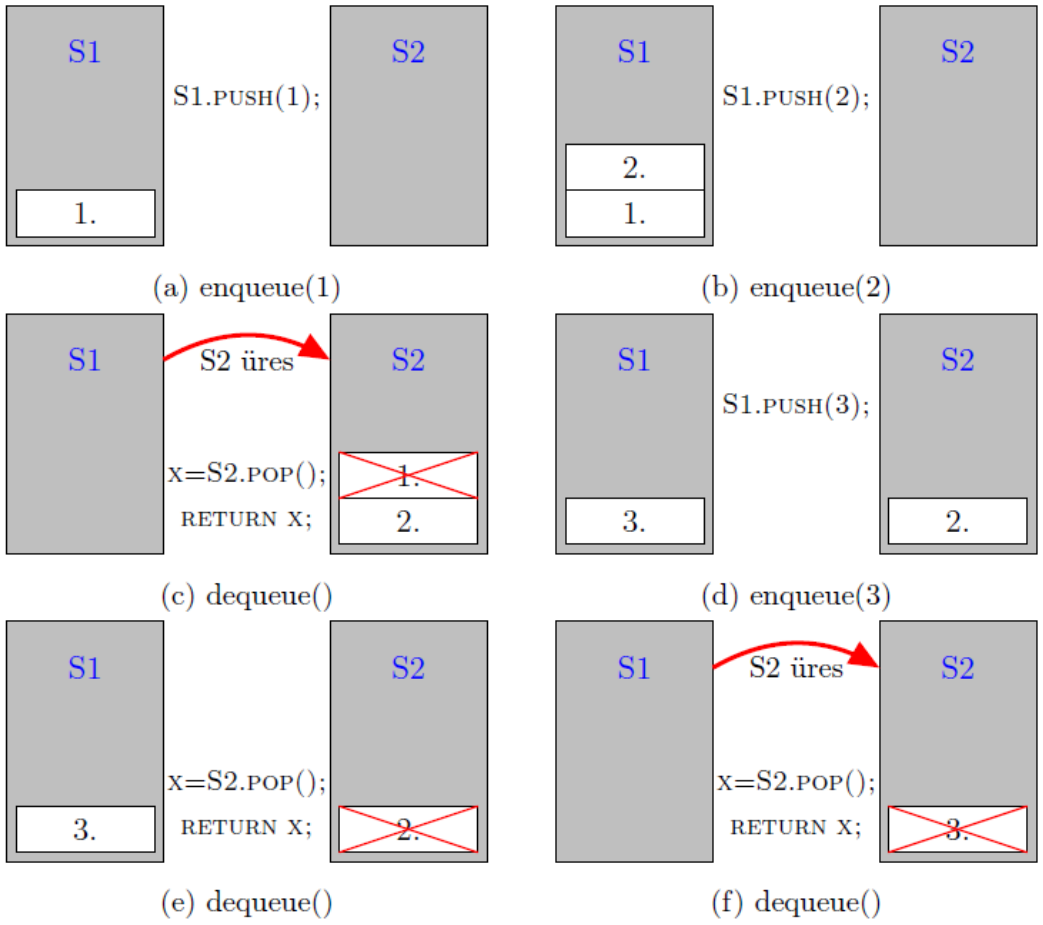
A sor vége az első veremben van, az eleje pedig a második veremben fordított sorrendben. Ezért az első veremhez Verembe művelettel hozzáadjuk az elemet.

Elem eltávolítása a sorból (**Sorból** művelet):

1. Ha a második verem üres, pakoljuk át az összes elemet az elsőből.

2. Vegyük ki a második veremből az utoljára betett elemet (mivel az elemek fordított sorrendben vannak, éppen az első elem lesz). Ha üres dobjunk hibát.

6. gyakorlat



Példa az adatszerkezetre (forrás: Gelle Kitti: Algoritmusok és adatszerkezetek I. jegyzet)

6. gyakorlat

Hogyan valósítható meg két verem egyetlen $A[1 . . n]$ tömbben úgy, hogy egyik verem sem csordul túl addig, amíg együttes elemszámuk nem haladja meg az n-et? A Verembe és Veremből műveletek végrehajtási ideje maradjon $O(1)$.

6. gyakorlat

```
// Kezdeti értékek beállítása, létrehozás
int n = 10;
int top1 = -1;
int top2 = size;
int arr[] = new int[n];
```

6. gyakorlat

```
// Verembe művelet az 1. veremre
 void push1(int x)
{
 if (top1 < top2 - 1) {
       top1++;
       arr[top1] = x;
    }
    else {
          System.out.println(„Túlcsordulás");
          System.exit(1);
    }
}
```

```
// Verembe művelet az 2. veremre
void push2(int x)
{
    if (top1 < top2 - 1) {
          top2--;
          arr[top2] = x;
       }
       else {
          System.out.println(„Túlcsordulás");
          System.exit(1);
       }
 }
```

6. gyakorlat

```
// Veremből művelet az 1. veremre
int pop1()
{
   if (top1 >= 0) {
      int x = arr[top1];
      top1--;
      return x;
   }
   else {
      System.out.println(„Alulcsordulás");
      System.exit(1);
   }
   return 0;
}
```

```
// Veremből művelet az 2. veremre
int pop2()
   {
      if (top2 < size) {
         int x = arr[top2];
         top2++;
         return x;
      }
      else {
         System.out.println(„Alulcsordulás”);
         System.exit(1);
      }
      return 0;
   }
```

6. gyakorlat

Adjunk meg egy olyan verem adatszerkezetet, amely támogatja a push es pop művelet mellett a minimum lekérdezését is O(1) időben és plusz O(n) tárban!

6. gyakorlat

Két vermet használunk. Az egyikben tároljuk az összes elemet. A másik veremben csak a részvermek minimumát tároljuk.

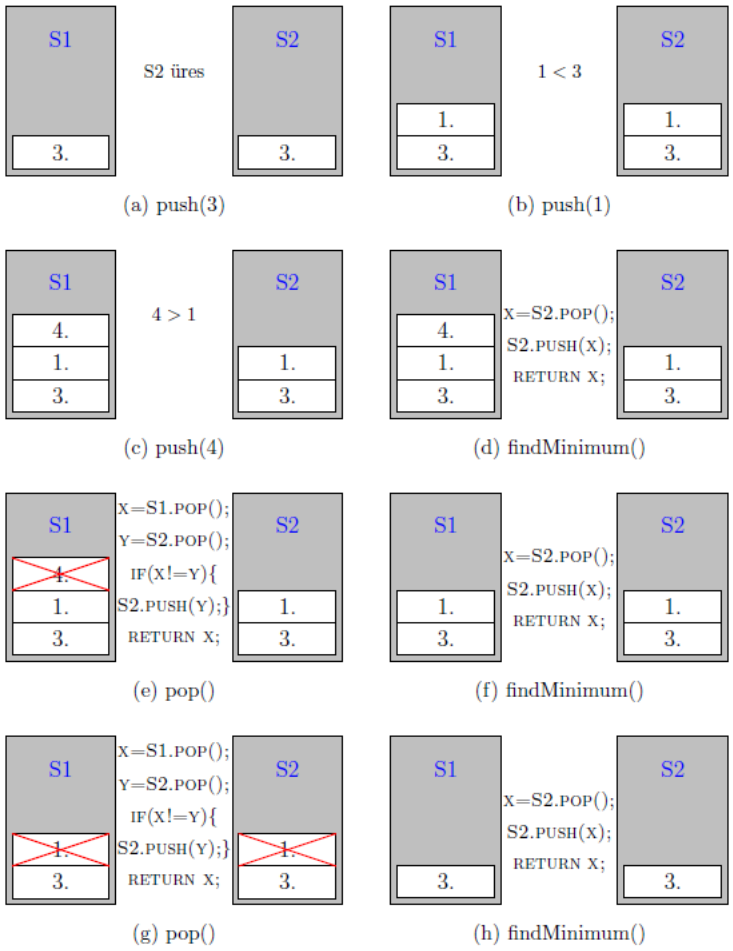
A műveletek:

Verembe: rakjuk be az elemet az első verembe. Ha a második verem üres oda is rakjuk be, különben nézzük meg, hogy kisebb vagy egyenlő-e, mint a tetején lévő elem. Ha igen, szintén rakjuk be.

Veremből: Vegyünk ki egy elemet az első veremből. Ha ez az elem megegyezik a minimumokat tároló verem tetején lévő elemmel, vegyük ki azt is.

Minimum: Adjuk vissza a minimumokat tároló verem tetején lévő elemet, de ne vegyük ki.

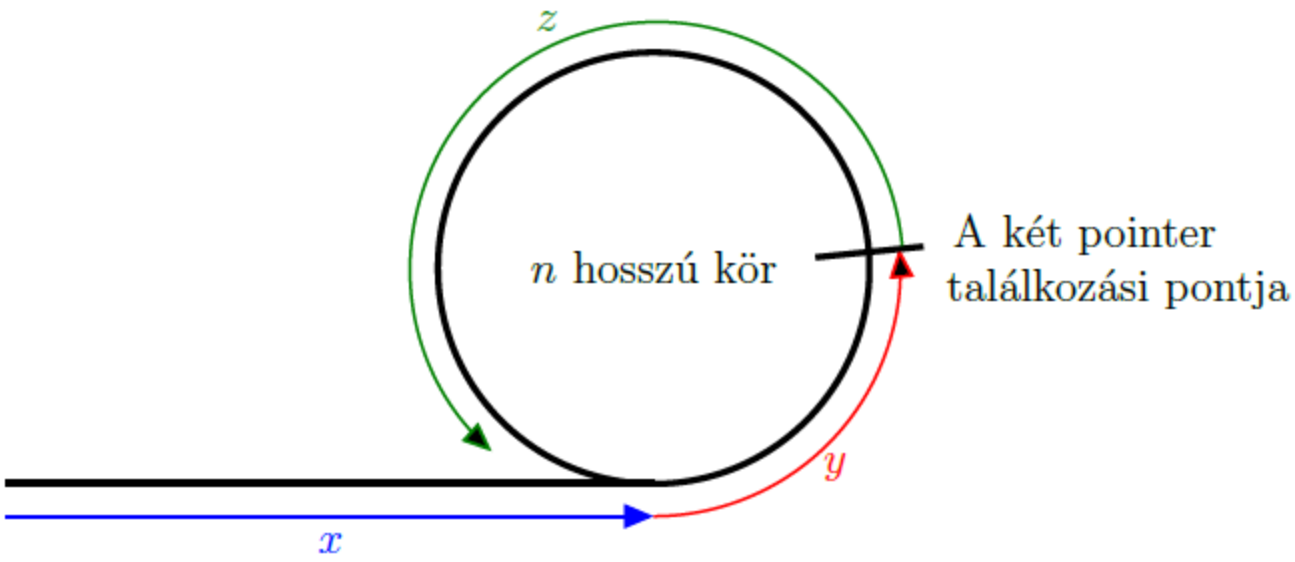
6. gyakorlat



Példa a működésre (forrás: Gelle Kitti: Algoritmusok és adatszerkezetek I. jegyzet)

6. gyakorlat

Hogyan tudjuk egy egyszeresen láncolt listában O(n) időben es konstans tárban ellenőrizni, hogy van-e benne kör?



6. gyakorlat

A **kupac** egy *majdnem teljes bináris fa*, amelynek minden csúcsa legalább akkora, mint gyerekei → maximális elem a gyökérben van

Házi feladatok

Megvalósítás?

Műveletigények?

6. gyakorlat

Egy *h* magasságú kupacnak legalább (ill. legfeljebb) hány eleme van?

Hol helyezkedhet el a legkisebb elem a maximum-kupacban, ha minden elem különböző?

Maximum-kupac-e a 23, 17, 14, 6, 13, 10, 1, 5, 7, 12 sorozat?

6. gyakorlat

Hogyan működik a Kupacrendezés eljárás az

A = 5, 13, 2, 25, 7, 17, 20, 8, 4 tömbön?

6. gyakorlat

- Magyarázatok, megoldások:
- Gelle Kitti: Algoritmusok és adatszerkezetek I. gyakorlati jegyzet.
- http://www.inf.u-szeged.hu/~kgelle/sites/default/files/upload/alga-gyak-06.pdf