Alga2 fák, kupacok, fapacok
+linkes jegyzettám...

Bináris keresőfa:
- kisebbet tesszük az elem helyére tölrésnél
- Inorder: növekvő sorrend
- Preorder: balra le (minden csúcsot érintve kiír) amíg tud, majd vissza 1 és jobbra, vissza 1 jobbra...
- Posztorder minden részfát: bal-jobb-apa módon, lentről indulva, szintenként
- Amúgy ea jegyzetben van rá pszeudokód, meg alga1-es anyag

Rendezettmintafa:
- Kieg infó: adott részfa mérete (kulcsainak száma) (két gyerek kieg infó összeg + 1)
- Rang keresése:
  - rekurzívan
  - ha balra lépsz, ugyanazt a rangot keresed
  - Ha jobbra, akkor a **bal** fiú kieg infójával csökken a keresett rang
- Rang meghatározása:
  - ugyanez felfele
  - ha az adott kieginfó nagyobb, mint a csúcsé (gyökér felé haladva)
  - Ha az adott **kulcs** kisebb, akkor rang+=balfiú+1
- Törlés: kieg infókat a gyökér felé vezető úton frissíteni kell

Intervallumfa:
- bináris keresőfa, rendezés prior: alsó végpont, felső végpont
- kieg infó: az adott részfa abszolút felső végpontja
- átfedő keresés: gyökérből:
  - átfedő az adott csúccsal? igen -> return
  - ha lehet balra menni, és a bal kieg nagyobb, mint az i-nk alsó végpontja, akkor balra
  - egyébként jobbra

AVL fa:
- legrosszabb esetben O(logN) magasságú
- csúcs kieg1: egyensúlyi faktor: a fiai magasságának különbsége (tkp a kieg infóiké)
- kieg infó: magasság: nagyobb (kieg infójú) fiú kieg + 1
- csúcsok a fibonacci számok-1 módon növekszik a magassággal:
  - 1 2-1
  - 2 3-1
  - 3 5-1
  - 4 8-1 stb

    - m = m(h-2) + m(h-1) + 1
- h < 1.44 log2(N) korlát. Ez amúgy pont az 1,618 (aranymetszés, meg a fibo számok aránya) 10-esről 2-es logba írva…
- Helyreállítás minden lépés után, forgatások:
    - ha adott helyen az egyensúlyi faktor absz értéke legalább 2
    - a kisebb gyerek felé forgatunk
        - Ha a magasabb gyerek ellentétes (pl bal gyerek esetén jobb) fiának nagyobb a magassága, akkor ott is forgatni kell előtte (cikkcakk)

B-fa:
- csúcs elemszám (ha nem gyökér - haha) akkor [t, 2t] (t a fa rendje)
- rang == fapont csúcsai, elemszáma, tagjai, whatever
- gyökér elemszáma is <= 2t (ofc minimum 1)
- minden levél azonos mélységű… vagy magasságú
- minden csúcsnak rang+1 gyereke van
- Keresés kinda triviális. A csúcsok elemei növekvő sorrendben vannak, a gyerekek meg két elem közötti értéket vehetnek fel adott ágon
- magassága logt(N) ahol t a fa randje
- Beszúrás:
    - levélként
    - ha sok a csúcs rangja, akkor a középső elemet felviszed az apjába és kettébontod ementén. (ezt rekurzívan)
- Törlés:
    - Ha alulcsordul, a szomszédjától kell kérni (kisebb szomszédtól by def)
        - Azaz a bal szomszéd legnagyobb eleme megy az apába, az apa adott eleme pedig le
    - Ha kell összeolvasztható a kicsi szomszéddal (ehhez le kell hozni a szülő megfelelő elemét)
        - Így egy 2t rangú csúcs fog keletkezni, és a szülő rangja csökken
- (Elméleti cucc még ide a tail rekurzió, ami csak annyit takar, hogy a fv utolsó parancsa a rekurzív hívás)

2-3-4 fa:
- általános keresőfa
- minden csúcsának a rangja 1 vagy 2 vagy 3.

Piros-fekete fa:
- gyökér fekete
- minden levél fekete
- piros csúcsnak csak fekete gyereke lehet
- a fa fekete magassága állandó
- A fa magassága n kulcsra <= 2 (logn + 1)
- A fa magassága >= fa fekete magassága
- Kicsit rosszabb a legrosszabb magassága, mint az AVL fának (keres esetén AVL fa lehet jobb)

- Beszúr, töröl esetén inkább piros-fekete fa
- Műveletek https://github.com/begab/alga2/blob/master/gyakorlatok/gyak04/alga2_gyak04.pdf
- <=> 2-3-4 fa. Ha a fekete csúcsokat a piros gyerekeikkel összeolvasztjuk

==
=====================

Binomiális kupac:
- kupac - prisor
- ezekből fák építve
- Fontos műveletek:
  - max vagy min elem hatékony visszaadása
  - egy adott kulcs értékének módosítása
- Kupactulajdonság fogalma (minimum vagy maximum kupac, azaz minden szülőnél kisebbek a gyerekei vagy nagyobbak a gyerekei)
- Kupac implementációja -> tömb
- Bináris kupac: teljes bináris fa, amire teljesül a kupactulajdonság (azaz a leveleken kívül minden csúcsnak 2 fia van)

Fapac:
- kulcsok keresőfa tul. szerint (bal kisebb jobb nagyobb)
- extra adattag

Binomiális fa:
- 2^k csúcsa van
- i-edik mélységben k alatt i csúcsa van (gyökér i=0)
- Rend: i-edik gyerek 1-1 Bi részfa gyökere
- n csúcsú fa minden csúcsának fokszáma legfeljebb log(n)

Binomiális kupac:
- ha minden fa rendelkezik a (min/max) kupactulajdonsággal
- nincsenek azonos fokszámú fák
- Tehát legfeljebb floor(logn) + 1 binom fából áll
- Minimális kulcs keresés: gyökérlistán kell végigiterálni
- egyesítés logn
- Törlés: gyökér helyére írjuk, majd kupac helyreállít, és törlés
- binom vs bináris kupac: https://github.com/begab/alga2/blob/master/eloadas/05/alga2_05.pdf

Amortizált költségelemzés:
- ea: https://github.com/begab/alga2/blob/master/eloadas/06/alga2_06.pdf
- gyak: https://github.com/begab/alga2/blob/master/gyakorlatok/gyak07/alga2_gyak07.pdf
- k bites számlálón Növel művelet amortizáltan konstans

Fibonacci kupac:

- egy csúcs megjelölt, ha vesztett gyereket azóta, hogy másik csúcs gyereke lett
- minimumpointer
- gyökérlista random sorrend
- elméleti dolgok, potenciálfüggvény az ea slide-okon
- kupacösszehasonlítás ea slideon
- Felhasználás:
  - Ha NEM használunk sok Törlést és Pop-ot
  - pl Dijkstrára jó
  - Ha nem használjuk a műveleteket, logn
  - Ha igen, akkor log(fí)n (fí az 1,618) - ez rosszabb mint a log2 just sayin

Önszerveződő fa:

- Keresőfa
- minden művelet után az utoljára érintett csúcsot forgatásokkal a gyökérbe viszi
- O(n) magas, amortizáltan csak Logn

==

Innentől csak elméleti anyag lesz, gyakanyagra solver!

Geom
Ea slideok:
https://github.com/begab/alga2/blob/master/eloadas/07_08/alga2_07.pdf
Kereséshez jobb a tex:
https://github.com/begab/alga2/blob/master/eloadas/07_08/alga2_07.tex

Fonya, dimat:
https://github.com/begab/alga2/blob/master/eloadas/09/alga2_09.pdf
Kereséshez tex:
https://github.com/begab/alga2/blob/master/eloadas/09/alga2_09.tex