Keerukus ja Andmestruktuurid

(algajad)

Margus Niitsoo

Mis täna toimub

- Tänane töö korraldus
  - Saame tuttavaks
  - Korda N korda:
    - Mina räägin
    - Praktika
  - Kokkuvõte
    - Üldistavad mõtted

Saame tuttavaks

- Tere, mina olen Margus
  - Keskkoolis käisin korra IOI-l
  - PhD arvutiteaduses
  - 3+ aastat õppejõud TÜ-s
  - 5 aastat ettevõtja/progeja
    - Enda firma nimega MatchMySound
- Keeled:
  - C, JavaScript, Python, SQL, R, HaXe

Saame tuttavaks

- Kes teie olete?
  - Kaua olete programmeerinud?
  - Mis keeles?
  - Kas olite eelmisel õppesessioonil?
  - Kas UVA ütleb midagi?

Saame tuttavaks

- Kas olete kuulnud mõisteid:
  - Kiire sorteerimine (Quicksort)
  - “ruutkeerukus” või $O(n^2)$
  - Kahendpuu (binary tree)
  - Pinu ja järjekord (stack/queue)
  - Kujutis (map)

Mis täna toimub

- Sorteerimine
  - Bogosort, Insertion sort, Quicksort
  - Keerukus
- Andmestruktuurid
  - Süsteemitüüpide hingeelu
  - Heap ja heapsort

Teie omavahel

- “Paaris programmeerimine”
  - Ehk ma jagan teid paaridesse
- Miks?
  - Kahekesi lihtsam aru saada,
    millest te aru ei saa, ja julgem küsida!
  - Õpetamine aitab enda teadmistes auke täita
  - Lisaks ülikoolis ACM selles formaadis ;)
- Tutvumiseks 3+3 minutit:
  - Räägi raskeimast ülesandest,
    mida sa lahendanud oled?

Sorteerimine

Ülesande püstitus

- Pikkuse järjekorras
  Antud klassis olevate laste arv, ja nende pikkused
  - Esimesel real arv: N. Igal järgmisel N real lapse nimi (sõne) + üks täisarv
- Väljastada:
  - Laste nimed nende pikkuse järjekorras

Esimene idee

- Proovime juhuslikult ümber järjestada ja kontrollime, kas sai õige.

Esimene idee

- Proovime juhuslikult ümber järjestada ja kontrollime, kas sai õige.

- Ei tundu nagu hea mõte? Aga miks?

Bogosort sammude arv

- Erinevaid järjestusi on N!
  - 1*2*3*4*...*N
- Tõenäosus et neist juhuslikult valitu on õige on 1/N!
- Seega kulub keskmiselt 1/(1/N!) = N! Sammu
- See kasvab kiiresti!

Insertion sort

- Uus plaan:
  - Liigume mööda järjendit
  - Kui leiame elemendi, mis väiksem kui eelmine, lükkame seda nii kaua ettepoole kuni enam ei ole
- Soojendusharjutus: kirjutage see valmis!

Insertion sort sammud

- Samme parimal/halvimal juhul?

Insertion sort sammud

- Samme parimal/halvimal juhul?
- Parim juht: N sammu
- Halvim juht: N(N+1)/2
  - 1+2+3+..+(N-1) = N(N+1)/2

Insertion sort sammud

- Samme parimal/halvimal juhul?
- Parim juht: N sammu
- Halvim juht: N(N+1)/2
  - 1+2+3+..+(N-1) = N(N+1)/2
- Kummast me rohkem hoolima peaks?

Kas saab veel paremini?

- Mida paremini üldse tähendab?
  - Kas “alati paremini”?
  - Vs “teatud N korral”
    - Joonistame graafikuid
  - “Keskmiselt” vs “alati”?

Quicksort

- Jaga-ja-valitse
  - Vali suvaline element
  - Jaga ülejäänud kahte: sellest väiksemad ja suuremad
  - Sorteeri väiksemad
  - Sorteeri suuremad
  - Pane nad kokku

Quicksort sammude hulk

- Parimal juhul:
  - Loeb kihtide arv * sammud kihis
    - (Joonis kihtidest)

Quicksort sammude hulk

- Parimal juhul:
  - Sammud kihis * kihtide arv
    - (Joonis kihtidest)
  - N * log N
- Halvimal juhul?

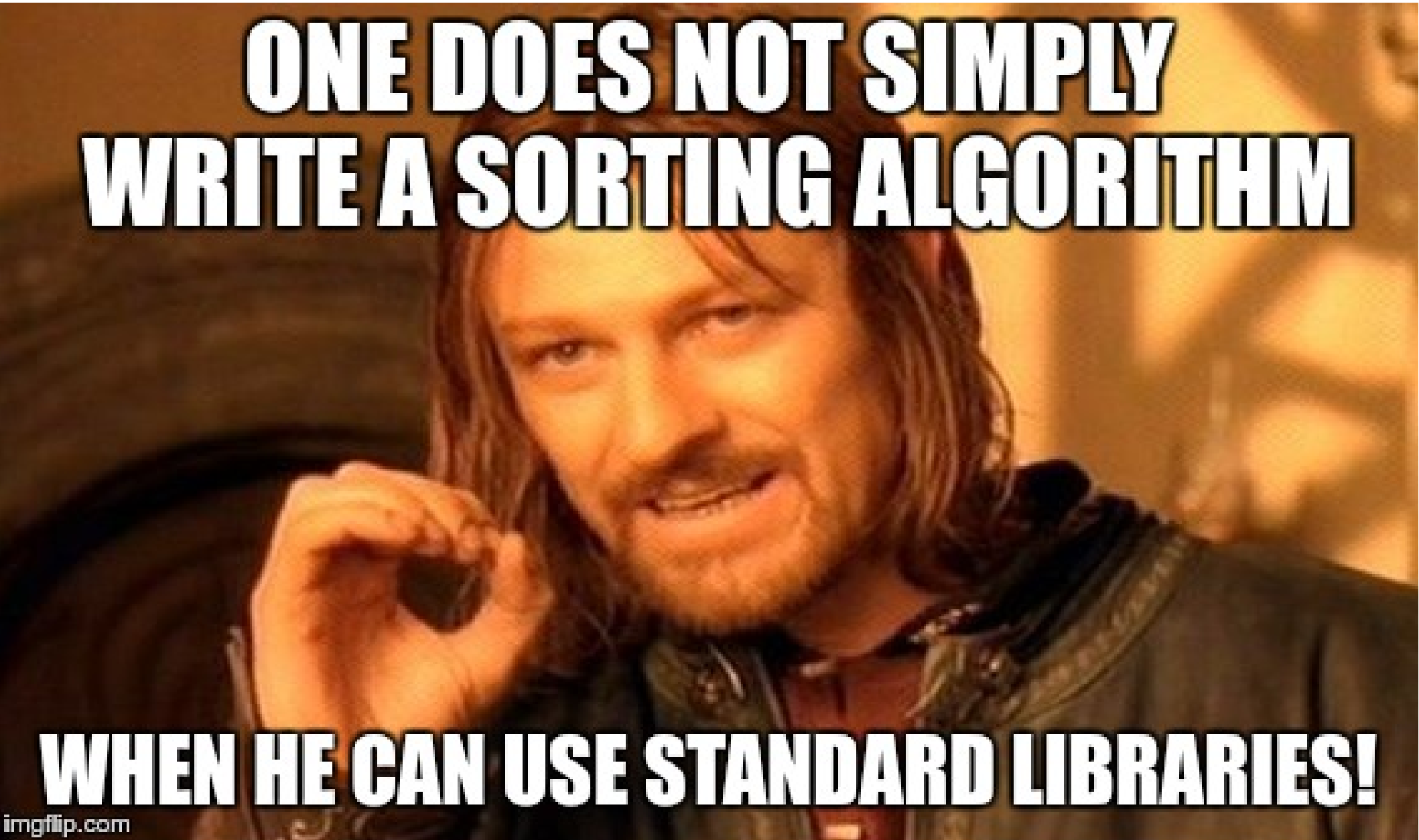
Quicksort sammude hulk

- Parimal juhul:
  - Sammud kihis * kihtide arv
    - (Joonis kihtidest)
  - N * log N
- Halvimal juhul
    - 1+2+3+..+(n-1) = n(n-1)/2
- Keskmiselt?

Sorteerimisest veel

- Algoritme on palju:
  - “Aeglased”: Selection, Bubble
  - “Kiired”: Heap, Merge, Quick
  - Erijuhud: Counting, Radix
- Sorteerimise “stabiilsus”
  - Teekides stable_sort eraldi
    - Merge sort tüüpiliselt

Meanwhile, in real life:



Quicksorti on hea teada

- Ülesande variatsioon: väljasta pikkuse järjekorras K-s laps

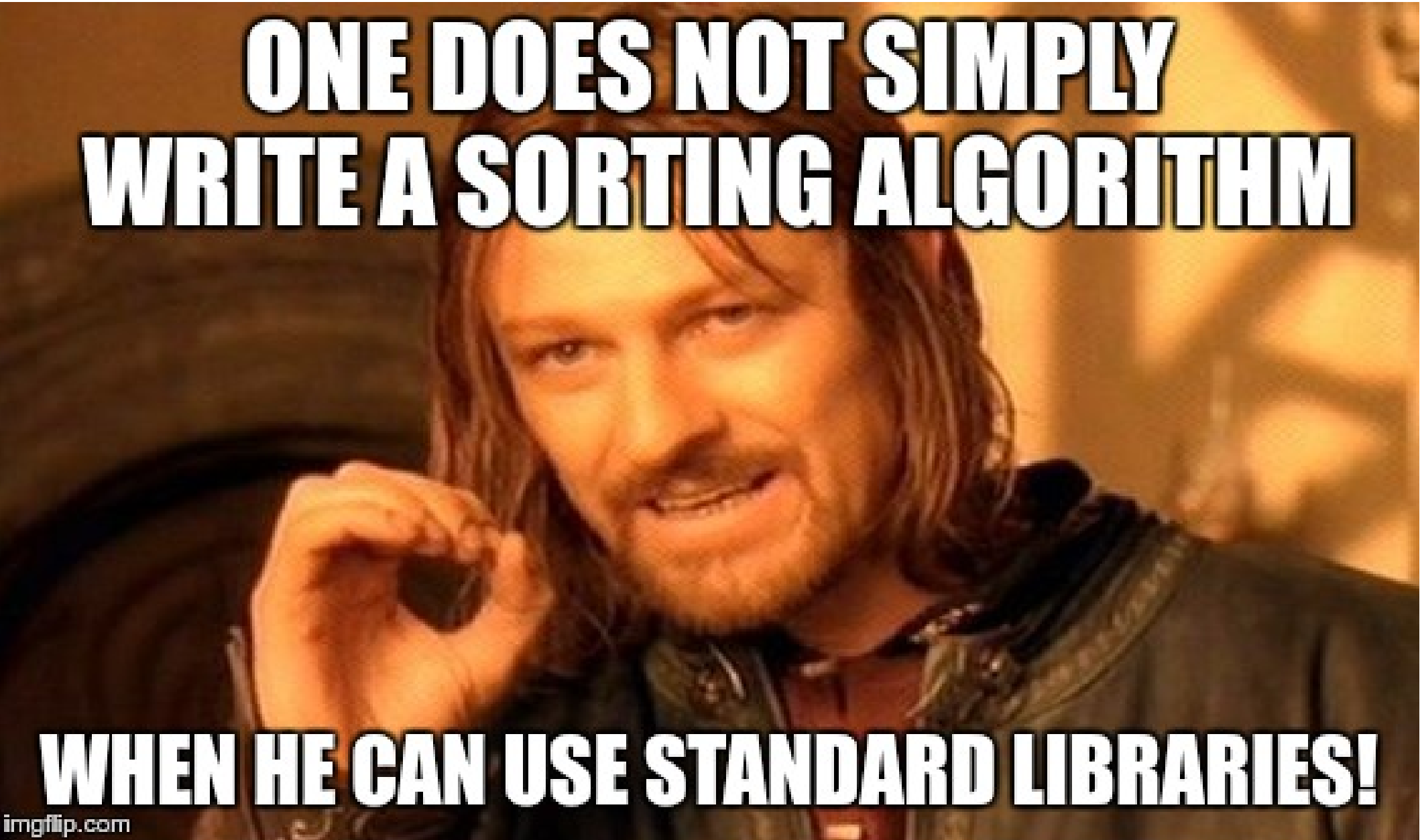
Quicksorti on hea teada

- Ülesande variatsioon: väljasta pikkuse järjekorras K-s laps
- Ühe poolega quicksordist piisab!
- Ja sammude hulk?

Quicksorti on hea teada

- Ülesande variatsioon: väljasta pikkuse järjekorras K-s laps
- Ühe poolega quicksordist piisab!
- Ja sammude hulk
  - n+n/2+n/4+n/8+.. = 2n

Kuigi ka siin:



Keerukus?

- Algoritmi “Keerukus” ehk “Ajaline keerukus” on selle sammude arv
- “Halvima juhu” keerukus vs “Keskmine” keerukus
- Räägitakse veel ka mälukeerukusest

O-notatsioon

- Me lugesime samme täpselt
  - N(N+1)/2, N log N
- “Samm” ise pole täpne mõõtühik
- Meid huvitab peamiselt käitumine suurte N-idega
- Suurusjärgu “hinnang”

O-notatsioon

- Definitsioon
  f(n) = O(g(n)) kui leidub konstant c et mingist n väärtusest alates alati f(n) < c*g(n)
- Sarnaselt Ω (“<” asemel “>”)
- Θ on mõlemad koos
  - f(n) = Θ(g(n)) <=>
    f(n) = O(g(n)) and f(n) = Ω(g(n))

O-notatsioon

- O(1) – ei sõltu n-ist
- O(N) - “lineaarne” e. “for tsükkel”
- O(NlogN) – kiired sorteerimised
- $O(N^2)$ - “ruutkeerukus”
  - Kaks for tsüklit üksteise sees
  - Nt “aeglased” sortimised
- $O(2^N)$, $O(3^N)$, $O(N!)$

Andmestruktuurid

Andmete salajane siseelu

- Mis on järgneva koodi keerukus?

```
ar = [1,2,3]
for i in range(100):
    ar.insert(i,1)
```

  - (C++ inimesed: ar on std::vector)

Andmete salajane siseelu

- Pythoni dokumentatsioonist: ar.insert is O(N)
  - Ehk siis kood on $O(N^2)$
  - Mis toimub?
- Las vana mees räägib kuidas tema ajal oli

C Array

- Blokk mälu järjest
- Pikkus fikseeritud, muuta ei saa
  - Niiet elemendi lisamist/eemaldamist ei ole
- N-inda elemendi kätte saamine VÄGA kiire O(1)

C Array

- Pikkuse muutmine?
- Elemendi lisamine lõppu?
- Elemendi lisamine keskele?

C Array

- Pikkuse muutmine?
  - Loo uus array ja kopeeri sinna.
- Elemendi lisamine lõppu?
  - Korruta pikkus kahega uue loomisel
- Elemendi lisamine keskele?
  - Nihuta kõiki järgmiseid ühe el jagu edasi: O(n)

Linked list

- Ehk mida teha kui tihti vaja keskele elemente lisada ja random access pole oluline
- Pange iga elemendi külge viide järgmisele ja eelmisele.
- Lihtne struktuur, võite ise kirjutada

Dict / HashMap

- O(1) elemendi otsimine ja lisamine
  - Paisktabel
  - Kollisioonid
  - Suurus/Itereerimine
- Väga kasulik asi praktikas

Sorted binary tree

- O(logN) elemendi otsimine ja lisamine
- O(1) miinimum ja maksimum
- O(1) järjest läbi käimine

Heap

- Andmestruktuur kus kaks tehet:
  - Vähima leidmine ja eemaldamine
  - Uue elemendi lisamine
- Rakendused
  - Graafialgoritmid
    - Dijikstra ja Prim
  - **Heapsort**

Kahendkuhi

- kahendpuu + reegel
  - Vanem on oma lastest väiksem
  - Puu juur seega vähim väärtus kogu puus

Heap lisamine

- Uue elemendi lisamine:
  - Pane uus element mõneks leheks
  - Kuni vanem on suurem kui uus:
    - Vaheta uus ja vanem omavahel. Korda

Heap eemaldamine

- Vähima elemendi eemaldamine
  - Tõsta kuhja mõni leht juure asemele
  - Kuni see suurem kui lapsed, vaheta ta väiksema lapsega ja korda.

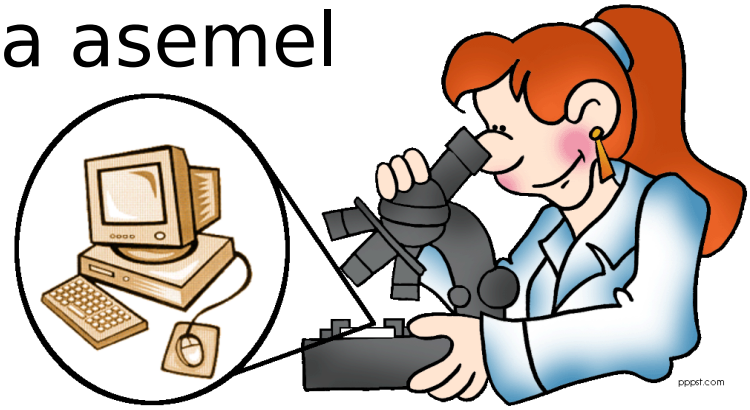
Heap implementeerimine

- Järjendis “mõttelise”puuna
  - Kui hetke tipp on kohal i siis:
    - Vasak laps: 2*i+1
    - Parem laps: 2*i+2
    - Vanem: floor((i-1)/2)
  - “mõni” leht = alati viimane element

Heapsort

- Sisesta kõik elemendid ükshaaval
- Võta nad järjest ükshaaval välja

- Saab teha in-place kui max-heap
  - Max-heap = suurim vähima asemel
  - Array algus on heap,
    lõpp sorteeritud väärtused



Kokkuvõte

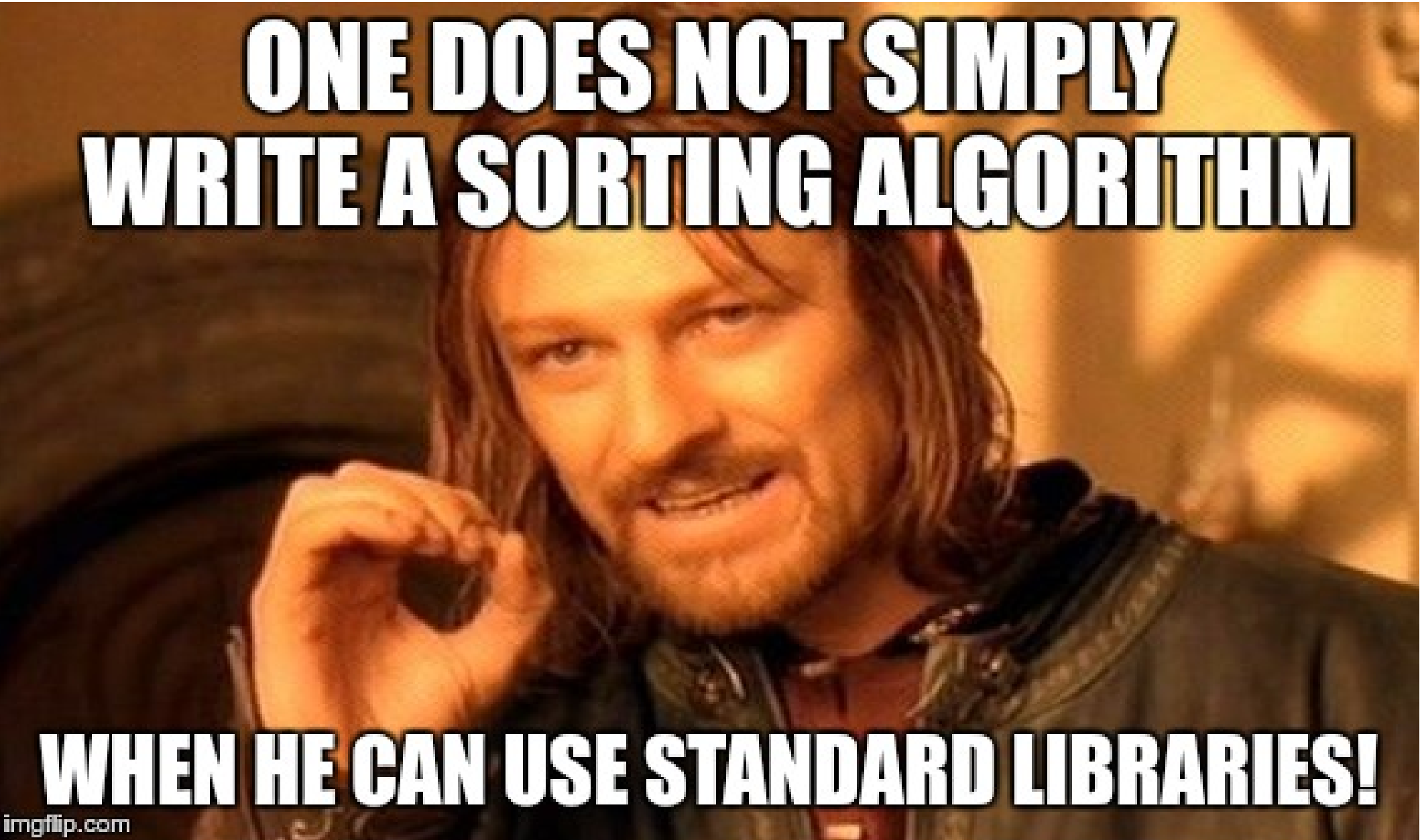
Mis täna toimus

- Sorteerimine
  - Bogosort, Insertion sort, Quicksort
  - Keerukus
- Andmestruktuurid
  - Süsteemitüüpide hingeelu
  - Heap ja heapsort

Õppetunnid

- Mõtle alati keerukusest:
  - Sisendandmete suurus annab lahenduse jaoks tihti vihjeid, sest seab piirid
- Kui sinu lahendus aeglane:
  - Siis mõtle, ehk saab kusagil paremat andmestruktuuri kasutada
  - Või olemasolevaid efektiivsemalt rakendada

And remember:



Tänan osalemast!

Küsimused on teretulnud!