TÖL403G GREINING REIKNIRITA

2. Umritun

Hjálmtýr Hafsteinsson

Vor 2022

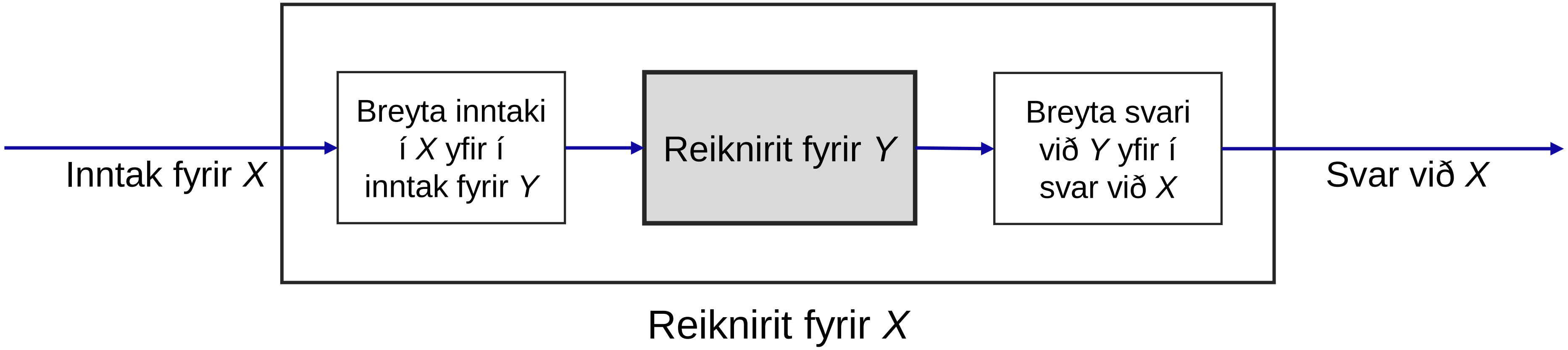
Í þessum fyrirlestri

- Umritun (*reduction*)
- Endurkvæmni (recursion)
- Bændamargföldun (*Peasant multiply*)
- Turnarnir í Hanoi
- Samrunaröðun (*Mergesort*)
- Quicksort

1.1 − 1.5

Umritun (*reduction*)

- Gagnleg aðferðafræði við hönnun reiknirita
- Umritum verkefni *X* yfir í verkefni *Y*:
  - Skrifum reiknirit fyrir *X* sem notar reiknirit fyrir *Y* sem svartan kassa



Dæmi um umritun

- Hanna minnstu löggenga stöðuvél *(DFA)* ***M*** fyrir reglulega segð ***R***
- Leysum þetta verkefni sjaldnast beint
  - Brjótum það upp í nokkur verkefni:

Fáum reglulega segð ***R***:

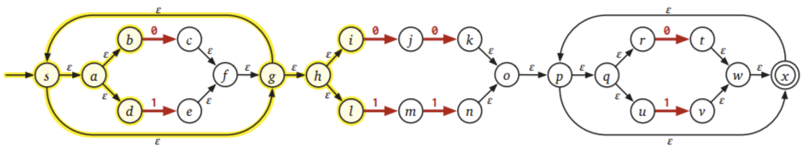
- Breyta reglulegu segðinni ***R*** yfir í brigðgenga stöðuvélina (*NFA*) ***N***
- Breyta ***N*** yfir í jafngilda löggenga stöðuvél ***M'***
- Breyta ***M'*** yfir í minnstu jafngildu löggengu stöðuvél ***M***

Skila ***M***

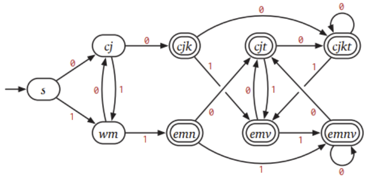
Hvert þeirra er sjálfstætt verkefni

$(0+1)^*(00+11)(0+1)^*$

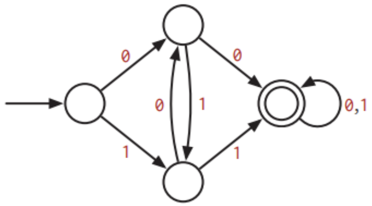
R



N



M'



M

Endurkvæmni (*recursion*)

- Endurkvæmni er ein tegund umritunar:

  Tvö skref:
  - Ef tilvikið er nógu einfalt þá leysa það beint
  - Annars umrita yfir í einfaldari útgáfu af <u>sama verkefni</u>

- Lykilatriði að einföldunin leiði að lokum til einfalds tilviks sem hægt er að leysa beint
  - Annars fáum við endalausa endurkvæmni

```
function haveGreatIdea() {
   startProject();
   loseMotivation();
   abandonProject();
   haveGreatIdea();
}
```

Frá DEV Community

Endurkvæm útgáfa af bændamargföldun

- Sáum síðast ítrunarútgáfu af bændamargföldun
- En reikniritið byggir samt á endurkvæmri formúlu!

$$x \cdot y = \begin{cases} 0, & \text{ef } x = 0 \\ \left\lfloor \frac{x}{2} \right\rfloor \cdot 2y, & \text{ef } x \text{ er jöfn} \\ \left\lfloor \frac{x}{2} \right\rfloor \cdot 2y + y, & \text{ef } x \text{ er odda} \end{cases}$$

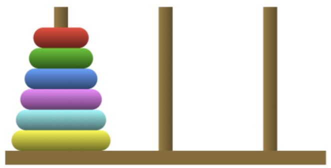
```
PEASANTMULTIPLY(x, y):
  if x = 0
      return 0
  else
      x' ← ⌊x/2⌋
      y' ← y + y
      prod ← PEASANTMULTIPLY(x', y')    ⟨⟨Recurse!⟩⟩
      if x is odd
          prod ← prod + y
      return prod
```

Nú er fjöldi endurkvæmra kalla háður gildinu á *x*

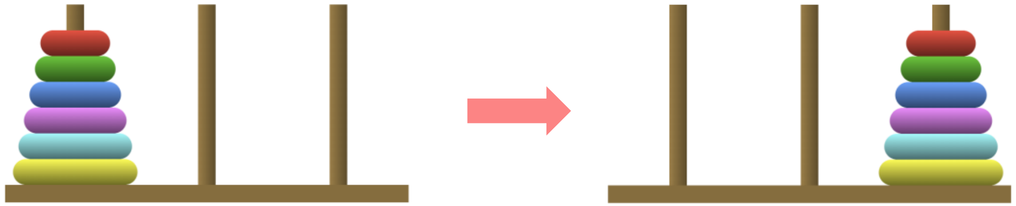
x helmingast í hvert sinn, svo það eru $\log_2 x$ köll

Tímaflækjan áfram $O(mn)$ fyrir margföldun á *m*- og *n*-tölustafa tölum

Turnarnir í Hanoi (*Tower of Hanoi*)



- Höfum 3 súlur og *n* skífur, allar af mismunandi stærð
  - Í upphafi er skífunum staflað í hækkandi röð á eina súluna
  - Markmiðið er að koma staflanum yfir á aðra súlu með eftirfarandi reglum:
    - Aðeins má færa eina skífu í einu
    - Aldrei má vera stærri skífa ofaná minni skífu
    - Það má nota þriðju súluna sem vinnusvæði



- Upphaflega sett fram af franska stærðfræðingnum Edouard Lucas árið 1883
  - Síðar búin til saga um munka sem vinna við að leysa þrautina með 64 skífum - verður heimsendir þegar þeir hafa lokið verkinu

Sjáum á eftir að við þurfum ekki að hafa miklar áhyggjur af heimsendi!

Reiknirit fyrir Turnana í Hanoi

- Auðvelt að leysa verkefnið endurkvæmt:

Höfum n skífur á upphafssúlu s, viljum færa á lokasúlu d:

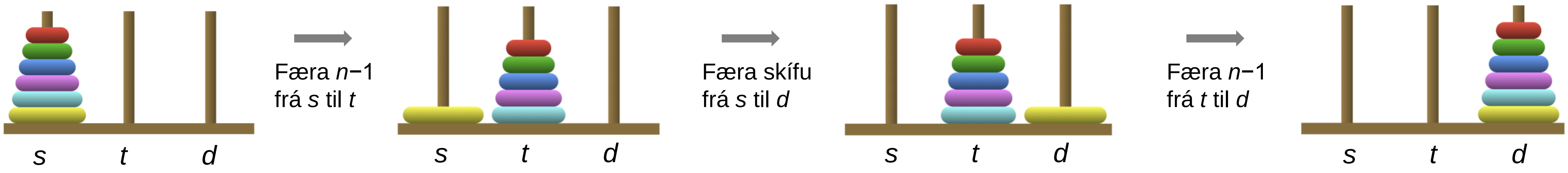
- Færa $n-1$ skífu frá súlu s til súlu t
- Færa neðstu skífu frá súlu s til súlu d
- Færa $n-1$ skífu frá súlu t til súlu d

Höfum umritað n-skífu verkefnið yfir í tvö $(n-1)$-skífu verkefni

Grunntilvikið er þegar $n = 0$, þá þarf ekkert að gera

[Hreyfimynd](#) frá [Ian Parberry](#)

[Önnur hreyfimynd](#)



Greining á Hanoi reikniriti

```
HANOI(n, src, dst, tmp):
  if n > 0
      HANOI(n − 1, src, tmp, dst)    ⟨⟨Recurse!⟩⟩
      move disk n from src to dst
      HANOI(n − 1, tmp, dst, src)    ⟨⟨Recurse!⟩⟩
```

Grunnaðgerð er að færa eina skífu

- Tímaflækja:
  - $T(n)$ = fjöldi færsla á skífum til að leysa verkefnið fyrir n skífur

$T(0) = 0$

$T(n) = 2*T(n-1) + 1$

Þarf ekkert að gera

Tvær færslur á $n-1$ stafla

Færsla á neðstu skífu

Ein leið til að leysa svona rakningarvensl:

- Giska á lausn og sanna hana svo með þrepun!

Ágiskun á lausn rakningarvensla

- Verðum að giska á "skynsaman" hátt
- Prófum að rekja okkur frá grunntilvikinu:

$T(0) = 0$
$T(n) = 2*T(n-1) + 1$

$T(0) = 0, \quad = 2^0-1$
$T(1) = 2*0 + 1 = 1, \quad = 2^1-1$
$T(2) = 2*1 + 1 = 3, \quad = 2^2-1$
$T(3) = 2*3 + 1 = 7, \quad = 2^3-1$
$T(4) = 2*7 + 1 = 15, \quad = 2^4-1$
...

Virðist vera almenn regla:

$T(k) = 2^k - 1$

Sönnum með þrepun:

Grunntilvik: $T(0) = 0$ og $2^0 - 1 = 0$ **OK!**

Þrepunartilvik: Gerum ráð fyrir að gildi fyrir $n-1$

$$T(n) = 2*T(n-1) + 1$$
$$= 2*(2^{n-1} - 1) + 1$$
$$= 2^n - 2 + 1$$
$$= 2^n - 1$$

Þrepunarforsenda

OK!

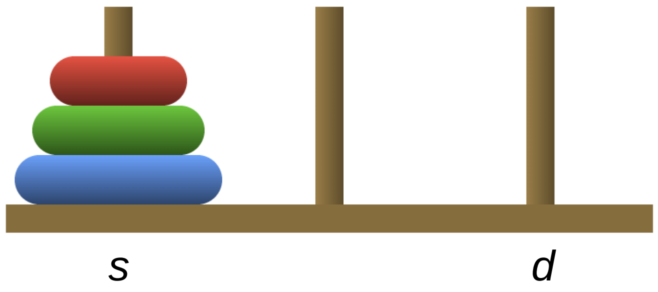
Fyrir verkefnið í sögunni er n=64, svo að fjöldi færsla er $2^{64} - 1 = 1.8*10^{19}$

Ef munkarnir færa eina skífu á sek. þá tekur það 585 þús. milljón ár að klára verkefnið!

Talið að núverandi aldur alheimsins sé um 14 þús. milljón ár

Æfing um turna í Hanoi

- Framkvæmið allar færslur við að færa þrjár skífur frá súlu *s* til súlu *d*
  - Hvert á efsta skífan að fara í fyrstu færslu?
  - Hversu margar eru færslurnar?



Samrunaröðun (*Mergesort*)

- Eitt af elstu reikniritunum fyrir röðun
  - Skipta inntakinu í tvo jafna hluta
  - Raða hvorum þeirra endurkvæmt
  - Sameina (*merge*) röðuðu listana í einn raðaðan lista

```
MERGESORT(A[1..n]):
  if n > 1
      m ← ⌊n/2⌋
      MERGESORT(A[1..m])        ⟨⟨Recurse!⟩⟩
      MERGESORT(A[m+1..n])      ⟨⟨Recurse!⟩⟩
      MERGE(A[1..n], m)
```

Grunntilvik: Eitt eða færri stök

Þá þarf ekki að gera neitt!

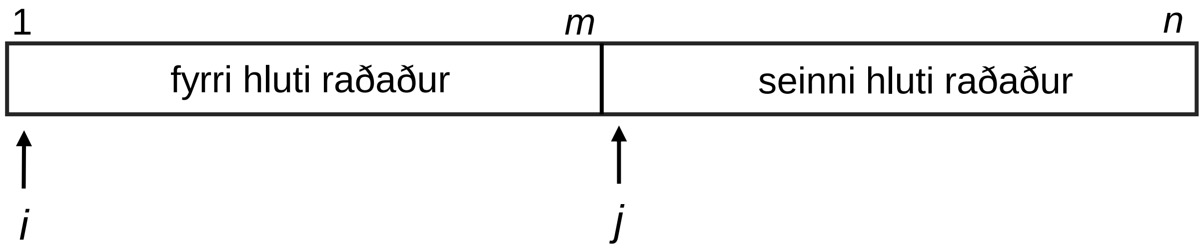
Skiptingin er aðeins að finna miðjuna

Ein deiling!

Tvö endurkvæm köll með hálfa inntaksstærð

Samruni (*merge*)

- Mesta vinnan er í samruna tveggja raðaðra lista



Höfum fjögur tilvik:

j* > *n [Seinni hlutinn er tómur]
Taka fremsta stakið úr fyrri hluta

i* > *m [Fyrri hlutinn er tómur]
Taka fremsta stakið úr seinni hluta

***A*[*i*] < *A*[*j*]** [Fremsta stak í fyrri hluta er minna]
Taka fremsta stakið úr fyrri hluta

***A*[*i*] ≥ *A*[*j*]** [Fremsta stak í seinni hluta er minna]
Taka fremsta stakið úr seinni hluta

```
MERGE(A[1..n], m):
  i ← 1;  j ← m + 1
  for k ← 1 to n
      if j > n
          B[k] ← A[i];  i ← i + 1
      else if i > m
          B[k] ← A[j];  j ← j + 1
      else if A[i] < A[j]
          B[k] ← A[i];  i ← i + 1
      else
          B[k] ← A[j];  j ← j + 1

  for k ← 1 to n
      A[k] ← B[k]
```

Tímaflækja Samrunaröðunar

Samruni (**Merge**-fallið) tekur greinilega $O(n)$ tíma

Ein **for**-lykkja frá 1 til n, með föstum fjölda aðgerða í hverri ítrun

Tímaflækjan er því $T(n) = T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + O(n)$

Við megum oftast sleppa efri og neðri mörkum (*floor*, *ceiling*) í stærðargráðureikningu, svo við fáum

$T(n) = 2T(n/2) + O(n)$, með $T(1) = 1$

Grunntilvikið er þegar það er eitt stak,
þá þarf aðeins einn samanburð

Getum leyst þessi rakningarvensl með "tréaðferðinni" sem við sjáum síðar, eða með því að giska á lausn og sanna hana með þrepun

Lausn á $T(n) = 2T(n/2) + O(n)$

Búa til ágiskun með því að ítra venslin:

$$
\begin{aligned}
T(n) = \;& 2T(n/2) + n \\
& 2(2T(n/4) + n/2) + n && = 4T(n/4) + 2n \\
& 4(2T(n/8) + n/4) + 2n && = 8T(n/8) + 3n \\
& 8(2T(n/16) + n/8) + 3n && = 16T(n/16) + 4n \\
& \;\ldots
\end{aligned}
$$

Virðist sem að almenna tilfellið sé
$T(n) = 2^k T(n/2^k) + kn$ fyrir heiltölu $k > 0$

Getum sannað það með þrepun á k (heimadæmi!)

Hvenær fáum við grunntilvikið?

Það kemur upp þegar $n/2^k = 1$ eða þegar $n = 2^k$ eða þegar $\underline{k = \log_2(n)}$

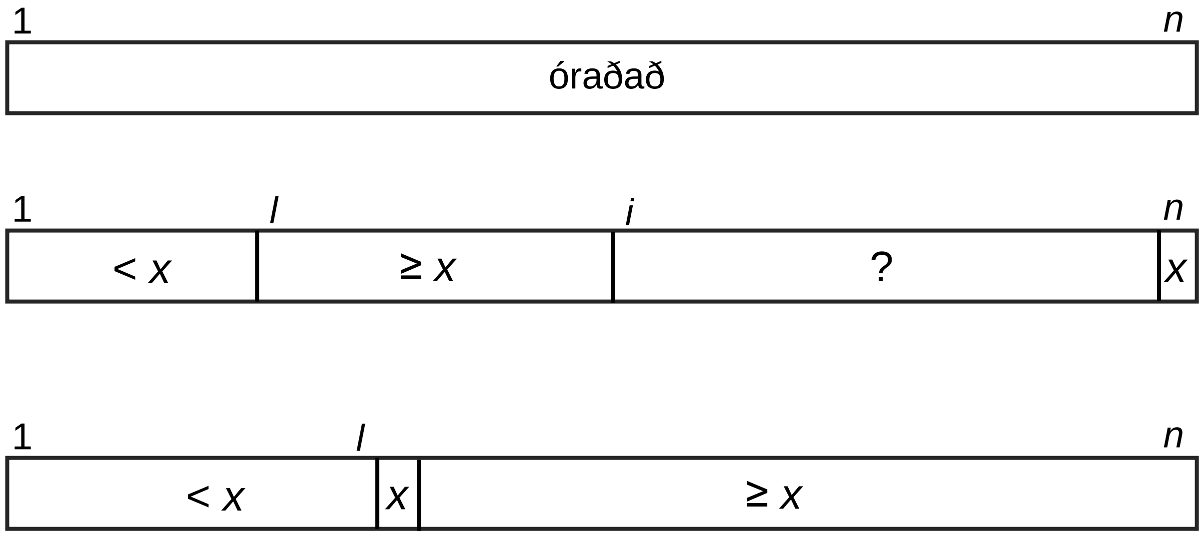
Setjum það inn í ágiskunina: $T(n) = nT(1) + n\log_2(n) = n\log_2(n) + n$

Í stærðargráðum: $\underline{T(n) = O(n\log(n))}$ ← Grunntala í logrum skiptir ekki máli í stærðargráðureikningi, munurinn er fasti

Quicksort

- Sett fram af Tony Hoare árið 1962
- Gerir mesta vinnuna á leiðinni niður endurkvæmnina
  - Samrunaröðun gerir vinnuna á leiðinni upp endurkvæmnina (við að sameina röðuðu listana)
- Quicksort velur <u>vendistak</u> (*pivot*) og <u>skiptir</u> (*partition*) listanum um það

```
QuickSort(A[1..n]):
  if (n > 1)
      Choose a pivot element A[p]
      r ← Partition(A, p)
      QuickSort(A[1..r − 1])      ⟨⟨Recurse!⟩⟩
      QuickSort(A[r + 1..n])      ⟨⟨Recurse!⟩⟩
```



Upphafsstaða

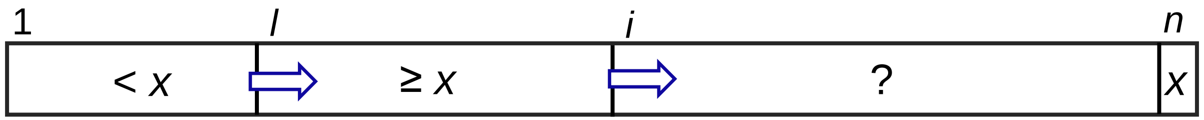
"Fastayrðing"

Lokastaða

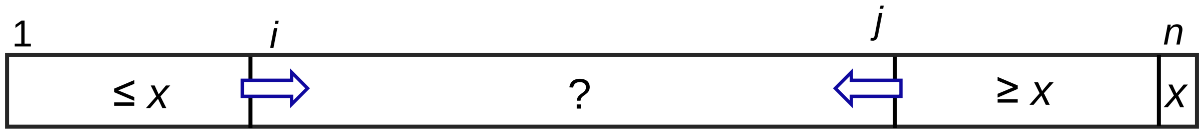
```
Partition(A[1..n], p):
  swap A[p] ↔ A[n]
  ℓ ← 0                 ⟨⟨#items < pivot⟩⟩
  for i ← 1 to n − 1
      if A[i] < A[n]
          ℓ ← ℓ + 1
          swap A[ℓ] ↔ A[i]
  swap A[n] ↔ A[ℓ + 1]
  return ℓ + 1
```

Skipting í Quicksort

- Það eru til nokkrar mismunandi aðferðir til að skipta listanum um vendistakið
- Aðferðin í bókinni er kennd við Lomuto
  - Vísirinn *l* telur fjölda staka sem eru minni en vendistakið *x*
  - Kostur: Auðveld að skilja og einföld í útfærslu
  - Galli: Tekur O(n^2) tíma þegar öll stökin hafa sama gildi



- Upphafleg aðferð Hoare:
  - Hækka *i* meðan $A[i] \leq x$, lækka *j* meðan $A[j] \geq x$
  - Víxla á $A[i]$ og $A[j]$ og halda áfram
  - Kostur: Ræður við lista með eins stökum
  - Galli: Mikil hætta á villum í útfærslu



Greining á Quicksort

Fáum rakningarvenslin

$T(n) = T(r-1) + T(n-r) + O(n)$ þar sem r er sætistala (*rank*) vendistaksins

Ef vendistakið væri alltaf miðgildið (þ.e. $r = n/2$), þá fáum við

$$T(n) = T(\left\lceil \frac{n}{2} - 1 \right\rceil + T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + O(n) \leq 2T\left(\frac{n}{2}\right) + O(n)$$

Lausnin á þessum venslum er $T(n) = O(n\log(n))$ ← Besta tilfelli

Í versta tilfelli er vendistakið alltaf stærsta/minnsta stakið í listanum, þá fáum við

$$T(n) \leq T(n-1) + O(n)$$

og lausnin er $T(n) = O(n^2)$ ← Versta tilfelli

Val á vendistaki í Quicksort

- Einfaldasta leiðin er að velja alltaf stak í tilteknu sæti
  - Til dæmis velja alltaf aftasta stakið $A[n]$
  - Raðaður listi er versta inntakið!!

Auðvelt fyrir andstæðing að búa til slæmt inntak

- Algeng leið er að velja miðgildi af $A[1]$, $A[n/2]$ og $A[n]$
  - Getum samt fengið mjög ójafna skiptinu og $O(n^2)$ tíma
  - Raðaður (eða næstum því) er núna gott tilfelli með $O(n \log(n))$ tíma
- Við getum fundið miðgildi n-staka lista á $O(n)$ tíma
  - Ef við veljum það sem vendistak þá er versta-tilfellis tíminn $O(n \log(n))$

Nokkuð hár fasti

- Getum valið sætisnúmer af handahófi (*random*) á bilinu 1 til n
  - Þá er ekkert inntak "betra" eða "verra" en annað − þau geta öll verið góð eða slæm
  - Fáum þá <u>væntan tíma</u> (*expected time*) reikniritsins (**ekki** meðaltíma!)

Ómögulegt fyrir andstæðing að búa til slæmt inntak

Venjuleg hegðun Quicksort

- Oftast er Quicksort mjög hraðvirkt
  - Slæmu tilvikin eru mjög fá (hlutfallslega!)
- Ef vendistak skiptir alltaf þannig að minni listinn er stærri en $n/10$:
  - Fáum þá rakningarvenslin $T(n) = T(n/10) + T(9n/10) + O(n)$
  - með lausn $\boldsymbol{O(n \log(n))}$

Reyndar nokkuð hærri fasti

- Sama gildir ef minni listinn er alltaf stærri en $n/100$, o.s.frv.

Ennþá hærri fasti

- Quicksort framkvæmir mjög fáar aðgerðir í innstu lykkju
- Quicksort "grófraðar" listanum fyrst
  - með því að skipta listanum um vendistakið

Getur fært stök
langa leið í listanum

- Til margar útgáfur af Quicksort
  - Nota tvö vendistök, Skipta í Innsetningarröðun við lítið inntak, Taka út endurkvæmni, ...

Fyrirlestraæfingar

1. Hver er næsta færsla í myndinni hér til hliðar?

2. Er Mergesort hraðvirkari ef inntakið er þegar í hækkandi röð? Hvað ef það er í lækkandi röð?

3. Ef vendistakið í Quicksort er valið sem miðgildi stakanna $A[1]$, $A[n/2]$ og $A[n]$ er þá öruggt að Quicksort taki ekki $O(n^2)$ tíma? Ef ekki, hvers vegna er þessi aðferð þá oft notuð í Quicksort?