TÖL403G GREINING REIKNIRITA

7. Kvik bestun 1

Hjálmtýr Hafsteinsson
Vor 2022

Í þessum fyrirlestri

- Fibonacci tölur
  - Útreikningur með endurrakningu
  - Minnisfesting (*memoization*) − geyma milliniðurstöður
  - Enn hraðvirkari útreikningur

3.1 − 3.5

- Strengskipting
  - **NÝ ENDURBÆTT** − Nú með kvikri bestun!

- Almennt um kvika bestun

Fibonacci tölur

- Skoðum first mjög einfalt dæmi um hvernig kvik bestun vinnur

- Fibonacci tölur: $F_0 = 0$, $F_1 = 1$ og $F_n = F_{n-1} + F_{n-2}$ fyrir $n \geq 2$

Getum skrifað þetta sem endurkvæmt endurrakningarreiknirit:

```
RecFibo(n):
  if n = 0
      return 0
  else if n = 1
      return 1
  else
      return RecFibo(n − 1) + RecFibo(n − 2)
```

Tími: $T(n) = T(n-1) + T(n-2) + 1$

með $T(0) = T(1) = 1$

Með því að rekja okkur í gegnum venslin sést að

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

Hægt að sanna að $T(n) = O(\varphi^n)$ ← Veldistímaflækja!

þar sem $\varphi = \frac{\sqrt{5}+1}{2} \approx 1.61803$ (gullna hlutfallið)

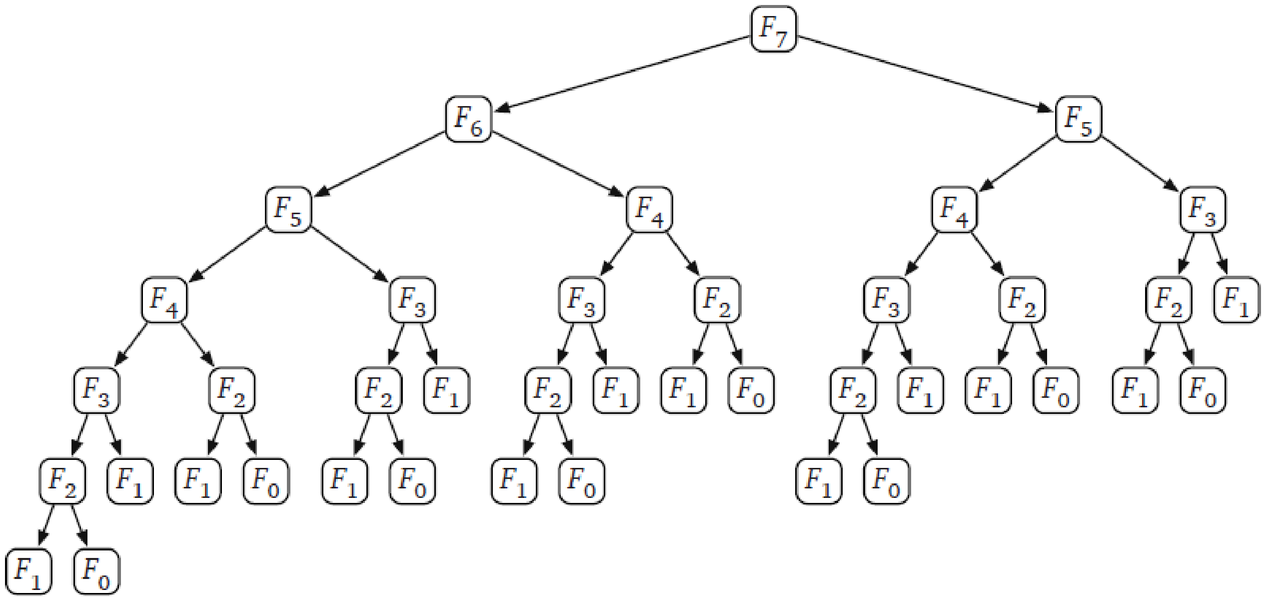
Veldistímareiknirit

- Hver er ástæðan fyrir veldistíma?

Sömu gildin eru reiknuð aftur og aftur!

Þegar við reiknum F_7 með þessu reikniriti, þá

- reiknum F_6 einu sinni
- reiknum F_5 tvisvar
- reiknum F_4 þrisvar
- reiknum F_3 fimm sinnum
- reiknum F_2 ____ sinnum
- reiknum F_1 ____ sinnum



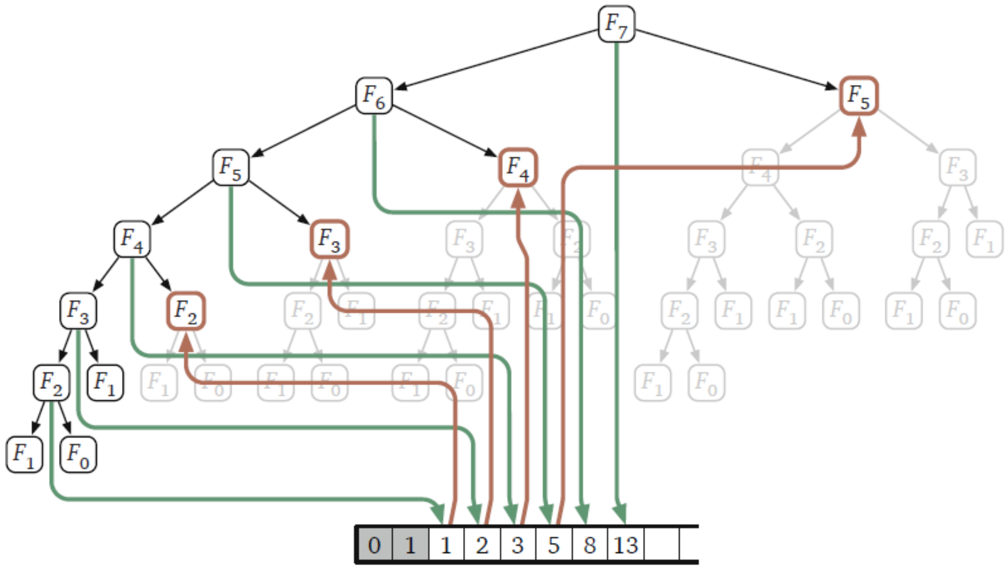
Við ættum að geyma gildin og fletta þeim upp:

<u>Minnisfesting</u> (*memoization*)

Geyma milliniðurstöður

- Geymum útreiknaðar Fibonacci tölur í fylki og flettum þeim upp
  - Skilgreinum víðværa fylkið F[0..n]

```
MEMFIBO(n):
  if n = 0
      return 0
  else if n = 1
      return 1
  else
      if F[n] is undefined
          F[n] ← MEMFIBO(n − 1) + MEMFIBO(n − 2)
      return F[n]
```



Grænar örvar sýna skrift í fylkið F[ ],
en rauðar örvar sýna lestur úr fylkinu

Reiknum fyrst út F_2, síðan út frá því F_3, o.s.frv.

Almennt: Reiknum aðeins út F_i, eftir að búið er að reikna F_{i-1}

Óendurkvæm útgáfa

- Þurfum ekki að nota endurkvæmni í reikniritinu:

```
ITERFIBO(n):
    F[0] ← 0
    F[1] ← 1
    for i ← 2 to n
        F[i] ← F[i−1] + F[i−2]
    return F[n]
```

Hér fyllum við inn í fylkið á beinni hátt

Augljósara hvernig við flettum upp í eldri gildum

Tími: $O(n)$ samlagningar

Þetta er fyrsta kvika bestunar (*dynamic programming*) reikniritið okkar!

Munum sjá síðar flóknari kvika bestun, en gott að hafa
svona einfalda útgáfu til að skilja grunnskipulagið

Minnkun á minnisnotkun

- Þurfum ekki alltaf að geyma allar milliniðurstöður
- Í Fibonacci reikniritinu *IterFibo* notum við aðeins síðustu tvö gildi
  - Hin gildin eru ekki notuð meira, svo við getum gleymt þeim!

```
ITERFIBO2(n):
  prev ← 1
  curr ← 0
  for i ← 1 to n
      next ← curr + prev
      prev ← curr
      curr ← next
  return curr
```

Fastayrðing lykkjunnar:

curr er alltaf núverandi Fibonacci tala

prev er alltaf síðasta Fibonacci tala

Athugið að til að fallið geti skilað F_0,
þá skilgreinum við F_{-1} sem 1

Enn hraðvirkari Fibonacci reikningur

- Getum fengið skemmtilega framsetningu á *IterFibo2* með því að nota fylki

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} prev \\ curr \end{bmatrix} = \begin{bmatrix} curr \\ prev + curr \end{bmatrix}$$

Sama útkoma og ein ítrun á lykkju í *IterFibo2*

Getum þá reiknað F_n með $$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{n-1} \\ F_n \end{bmatrix}$$

Upphafsstilling
á *prev* og *curr*

Ef *n* er ekki heilt veldi af 2, þá
þarf bara að geyma tiltekin fylki
og leggja þau svo saman

Hægt að reikna $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n = A^n$ með því að reikna $A^1, A^2, A^4, A^8, \cdots, A^n$

Þurfum því bara *O*(log(*n*)) margfaldanir (og saml.) á 2x2 fylkjum

Hver fylkjamargföldun er fastur fjöldi heiltöluaðgerða

Þurfum því *O*(log(*n*)) heiltöluaðgerðir

Raunveruleg tímaflækja

- En er heiltölumargföldun grunnaðgerð?
- Er hægt að margfalda hvaða tvær heiltölur saman á föstum tíma?

Nei, auðvitað ekki!

- Fibonacci tölurnar stækka mjög hratt
  - F_n er með u.þ.b. $n/5$ tölustafi í tugakerfinu, eða ~$2n/3$ bita í tvíundarkerfinu

Við þurfum $O(n)$ tíma bara til að skrifa F_n út

Það getur því varla tekið $O(\log(n))$ að reikna hana!

Sjá glæru 16
í fyrirlestri 4

Ef það tekur M(n) tíma að margfalda tvær *n*-stafa tölur þá er fylkisútgáfan með tímann:

$T(n) = T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + M(n)$ með lausn $T(n) = O(M(n))$ Besta gildi á $M(n)$ er $O(n \cdot \log n)$

Hvað með ítrunarútgáfuna?

- Ítrunarútgáfan *IterFibo2* notar samlagningu á allt að *n*-stafa tölum
- Hver samlagning kostar $O(n)$ tíma
- Heildartíminn á *IterFibo2* er því $O(n^2)$ grunnaðgerðir

þ.e. aðgerðir sem taka fastan tíma, margföldun á eins-stafs tölum

- Reyndar er F_n næsta heiltala við $\frac{\varphi^n}{\sqrt{5}}$, þar sem $\varphi = \frac{\sqrt{5}+1}{2}$ (sjá Wikipedia)

En það tekur $O(M(n))$ tíma að finna φ^n með endurtekinni margföldun

Tímaflækjan á þeim útreikningi er þá líka $O(n \cdot \log n)$

Þessar aðferðir eru því hraðvirkari en ítrunaraðferðirnar
$O(n \cdot \log n)$ á móti $O(n^2)$ grunnaðgerðir

Æfingar um Fibonacci

- Við getum útvíkkað Fibanacci tölurnar yfir í neikvæða vísa, F_{-1}, F_{-2}, F_{-3}, ...
  - Ef við setjum $F_{-1} = 1$, hver eru þá næstu gildi (í mínus-átt)?
  - Hvað þarf F_{-2} að vera til að $F_0 = F_{-1} + F_{-2}$ gangi upp?
  - Hvað þarf F_{-3} að vera til að $F_{-1} = F_{-2} + F_{-3}$ gangi upp?

Venjuleg skilgreining:

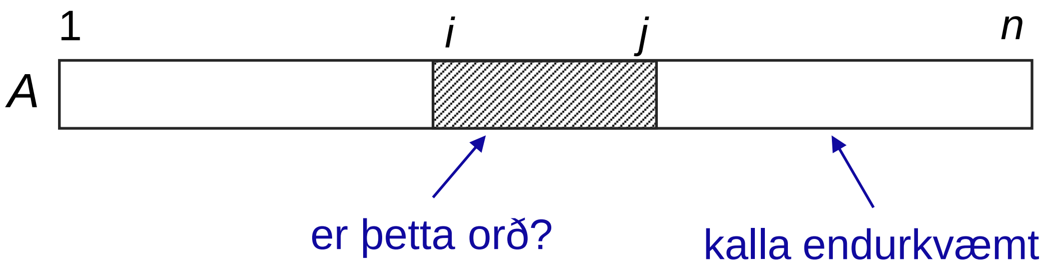
$F_0 = 0$, $F_1 = 1$ og $F_n = F_{n-1} + F_{n-2}$ fyrir $n \geq 2$

Orðskipting (*text segmentation*)

- Sáum í síðustu viku endurrakningarreiknirit fyrir orðskiptingu

Skilgreindum *Splittable*(*i*) = **satt** þþaa eftirstrengurinn (*suffix*) *A*[*i*..*n*] er skiptanlegur í orð

$$\mathit{Splittable}(i) = \begin{cases} \underline{\textbf{satt}} & \text{ef } i > n \\ \displaystyle\bigvee_{j=1}^{n} \mathit{IsWord}(i,j) \wedge \mathit{Splittable}(j+1) & \text{annars} \end{cases}$$



Í endurkvæmu útgáfunni köllum við aftur og aftur á *Splittable* með sömu viðföngum

Svipað og við gerum í fyrsta endurkvæma Fibonacci fallinu

Veldistími á endurrakningu

- Vitum ekki tímann á fallinu *IsWord*
  - Má gera ráð fyrir það sé margliða í stærð strengsins (uppfletting í gagnagrind)
- Teljum því fjölda kalla á *IsWord* til að meta tíma endurrakningarreikniritsins

- Reikniritið kallar á *IsWord* fyrir sérhvern forstreng (*prefix*) intaksins
  - og kallar hugsanlega á sjálft sig fyrir sérhvern eftirstreng af afgangnum

Fáum rakningarvenslin $$T(n) \le \sum_{i=0}^{n-1} T(i) + O(n)$$

Getum einfaldað í $$T(n) = 2T(n-1) + \alpha$$

með lausn: $$T(n) = O(2^n)$$

```
⟨⟨Is the suffix A[i..n] Splittable?⟩⟩
SPLITTABLE(i):
  if i > n
      return TRUE
  for j ← i to n
      if ISWORD(i, j)
          if SPLITTABLE(j + 1)
              return TRUE
  return FALSE
```

Endurrakningarútgáfa

Orðskipting með kvikri bestun

- Notum sömu hugmynd og í Fibonacci:
- Notum fylkið *SplitTable*[1..*n*+1] og flettum upp í því í stað þess að kalla endurkvæmt á fallið *Splittable*

Fyllum í fylkið "aftan frá", þ.e. byrjum á staki *n*+1

SplitTable[*n*+1] = er hægt að skipta strengnum *A*[*n*+1..*n*] upp í orð?

A[*n*+1..*n*] er tómi strengurinn!

```
FastSplittable(A[1..n]):
  SplitTable[n+1] ← True
  for i ← n down to 1
      SplitTable[i] ← False
      for j ← i to n
          if IsWord(i, j) and SplitTable[j+1]
              SplitTable[i] ← True
  return SplitTable[1]
```

Rekjum okkur svo niður fylkið

Við útreikning á *SplitTable*[*i*] notum við aðeins <u>uppflettingar</u> á stökum í *SplitTable*[*i*+1..*n*+1]

þegar búið að reikna þessi gildi

Tímaflækja *FastSplittable*

- Hér er engin endurkvæmni
- Bara tvöföld ***for***-lykkja
  - *i* gengur frá *n* til 1 í ytri lykkju
  - *j* gengur frá *i* til *n* í innri lykkju

```
FASTSPLITTABLE(A[1..n]):
  SplitTable[n+1] ← TRUE
  for i ← n down to 1
      SplitTable[i] ← FALSE
      for j ← i to n
          if ISWORD(i, j) and SplitTable[j+1]
              SplitTable[i] ← TRUE
  return SplitTable[1]
```

Fjöldi framkvæmda á innri lykkju: $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} = O(n^2)$

Í innri lykkju er eitt kall á fallið *IsWord*

Höfum því $O(n^2)$ köll á *IsWord* í stað $O(2^n)$ köll á *IsWord*

Tímaflækja *IsWord* fer eftir útfærslu þess

Almennt um kvika bestun

- Kvik bestun er <u>endurkvæmni án endurtekningar</u>
  - Geymir lausnir á hlutverkefnum í fylki (ein- eða tvívíðu)

- Mjög mikilvægt að finna réttu endurrakningarformúluna (*recurrence*) fyrir verkefnið fyrst
  - Gerðum það í síðustu viku með nokkur verkefni
  - Nú erum við að skoða hvernig við breytum þeim í kvika bestun

- Munum þróa kvik bestunarreiknirit í tveimur þrepum:
  1. Setja verkefnið fram á endurkvæman hátt
  2. Smíða lausnina neðan frá (*bottom-up*)

Ráðleggingar um kvika bestun

- Smíða endurkvæma formúlu fyrir lausn verkefnisins
  - Sbr. endurkvæma Fibonacci formúla
- Skoða hlutverkefni
  - Sbr. Fibonacci: F_{i-1} og F_{i-2}
- Velja gagnagrind fyrir lausnir á hlutverkefnum
  - Sbr. fylkið $F[0..n]$ í Fibonacci
- Finna tengsl milli hlutverkefna
  - Sbr. Fibonacci: þarf að reikna F_{n-2} áður en F_{n-1}, o.s.frv.
- Finna góða útreikningsröð
  - Byrja á grunntilvikum, síðan á hlutverkefni sem eru aðeins háð þeim, o.s.frv.
  - Sbr. Fibonacci: first F_0 og F_1, síðan F_2, F_3, ...
- Skrifa upp sem reiknirit

Oft óhagkvæmt að nota
þessa formúlu beint

Oftast ein- eða tvívítt
fylki, en **ekki** alltaf

Hvaða hlutverkefni þarf að leysa
á undan hvaða öðrum hlutverkefnum

Oftast með ítrun, án endurkvæmni

Græðgi virkar (nær) aldrei

- Þegar við erum að leysa bestunarverkefni er freistandi að nota gráðugt reiknirit
  - Gráðugt reiknirit tekur staðværar ákvarðanir sem virðast bestar á hverjum tíma
  - Oftast fyrsta hugmynd að reikniriti sem okkur dettur í hug

Gráðugt reiknirit fyrir LIS:

Finna lægsta gildi fylkisins, festa það sem fyrsta stak hlutrunu og leita síðan að lægsta gildinu hægra megin við það, o.s.frv.

- Gráðug reiknirit eru sjaldnast best
  - Það eru örfá verkefni (sjáum síðar) þar sem gráðugt reiknirit finnur alltaf bestu lausn
  - En lang lang lang oftast finna þau bara staðvært bestu lausn
  - Oftast mjög auðvelt að klekkja á þeim, þ.e. láta þau skila mjög slæmri lausn!

Gráðugt LIS: Hvað með inntakið [2, 3, 4, 5, 6, 1]?

Fyrirlestraæfingar

1. Sýnið að eftirfarandi gildir um Fibonacci tölur: $F_n = F_{n+2} - F_{n+1}$
2. Notið formúluna að ofan til að reikna Fibonacci tölur F_{-1}, F_{-2}, F_{-3}, F_{-4}, F_{-5}
3. Skrifið út fyrstu 12 Fibonacci tölurnar. Hvar koma jöfnu tölurnar fyrir (þ.e. heilt margfeldi af 2)? Hvar koma þær tölur sem eru heilt margfeldi af 3 fyrir? Hvert er mynstrið?