TÖL403G GREINING REIKNIRITA

6. Endurrakning 2

Hjálmtýr Hafsteinsson
Vor 2022

Í þessum fyrirlestri

- Lengsta hækkandi hlutruna (*longest increasing subsequence*)
  - Tvær endurrakningarútgáfur
  - ***LISbigger***
  - ***LISfirst***

2.6 − 2.8

- Bestu tvíleitartré (*optimal binary search trees*)
  - Veldistíma endurrakningarreiknirit

Hlutrunur (*subsequences*)

- Ef *S* er runa af hlutum, þá er <u>hlutruna í *S*</u> fengin með því að eyða út núll eða fleiri stökum úr *S*
  - Röð stakanna í *S* breytist ekki, en þau eru ekki öll með í hlutrununni
  - Tóma runan er hlutruna í *S* og *S* er hlutruna í sjálfri sér
- Dæmi:

  S: SUBSEQUENCEBACKTRACKING

  inniheldur m.a. hlutrunurnar: BENT SQUARING SUBSEQUENT

  en ekki hlutrunurnar: QUEUE eða EQUUS

- Hlutruna sem er samfelld í *S* kallast <u>hlutstrengur</u> (*substring*)

  í dæminu að ofan: SEQ og TRACK

Lengsta hækkandi hlutruna (*LIS*) (Longest Increasing Subsequence)

fylki af tölum

- Inntak: $A[1..n]$
- Finna lengstu röð vísa $1 \le i_1 < i_2 < \ldots < i_k \le n$ þannig að $A[i_j] < A[i_{j+1}]$ fyrir $j=1..k$
- Dæmi:

  [15, 8, 3, 5, 4, 8, 25, 12]

  Lengsta hækkandi hlutruna (*LIS*) er af lengd 4,
  t.d. [3, 4, 8, 25] eða [3, 5, 8, 12], ...

- Hugsum endurkvæmt:

  Ef inntakið er tómt, þá gerum við ekkert (*LIS* = 0)

  Annars: ákveða hvað á að gera við fremsta stakið í inntakinu
  og kalla endurkvæmt á rest

Endurkvæm framsetning

LIS af $A[1..n]$ er

A[1] ekki með

$max\{$ LIS af $A[2..n]$,

A[1] er með

$A[1]$ og síðan LIS af $A[2..n]$ með þeim stökum sem eru stærri en $A[1]$ $\}$

Þetta skilyrði flækir
málið aðeins

Útvíkkum aðeins skilgreininguna á LIS:

Finnum $LISbigger(p, A)$ = lengd lengstu hækkandi hlutrunu í A, með öll stökin hærri en p

Köllum á fallið í upphafi með:
$LISbigger(-\infty, A[1..n])$

```
LISBIGGER(prev, A[1..n]):
  if n = 0
      return 0
  else if A[1] ≤ prev
      return LISBIGGER(prev, A[2..n])
  else
      skip ← LISBIGGER(prev, A[2..n])
      take ← LISBIGGER(A[1], A[2..n]) + 1
      return max{skip, take}
```

A[1] ekki með

A[1] er með

Vísaútgáfa af *LISbigger*

- Þægilegra að gera ráð fyrir að fylkið *A* sé víðvær breyta
  - Þurfum þá aðeins að hugsa um tvo vísa (*indices*)

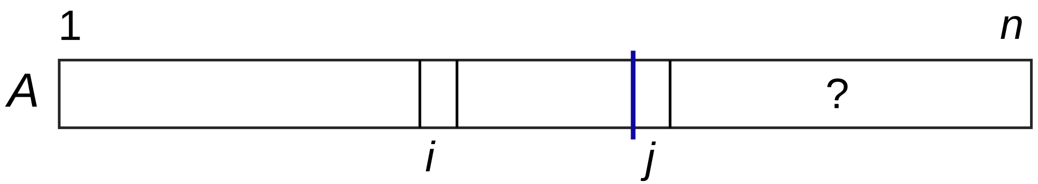
$LISbigger(i, j)$: lengd lengstu hækkandi hlutrunu í $A[j..n]$ með öll stök stærri en $A[i]$

$$
LISbigger(i,j): \begin{cases} 0 & \text{ef } j > n \\ LISbigger(i, j+1) & \text{ef } A[i] > A[j] \\ \max\begin{Bmatrix} LISbigger(i, j+1), \\ 1 + LISbigger(j, j+1) \end{Bmatrix} & \text{annars} \end{cases}
$$

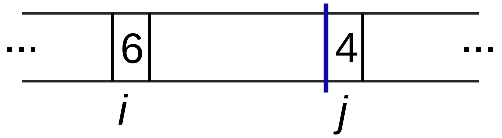
Reiknirit fyrir vísaútgáfu af *LISbigger*

```
LISBIGGER(i, j):
  if j > n
      return 0
  else if A[i] ≥ A[j]
      return LISBIGGER(i, j + 1)
  else
      skip ← LISBIGGER(i, j + 1)
      take ← LISBIGGER(j, j + 1) + 1
      return max{skip, take}
```

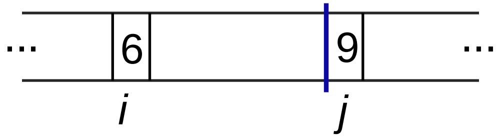
Almennt:



til dæmis:



til dæmis:



Setjum −∞ sem varðstak fremst í fylkið *A*[0..*n*]
og hjúpum með fallinu *LIS*

```
LIS(A[1..n]):
  A[0] ← −∞
  return LISBIGGER(0, 1)
```

Í seinna tilviki: Er betra að taka *A*[*j*] með?

Hvað með:

minnkum þá möguleikana á að bæta fleiri stökum úr A[j+1..n]

Æfing um *LISbigger*

	1	2	3	4	5	6	7	8

- Gefin runan 11, 7, 8, 3, 14, 2, 7, 5. Hvað gerist í *LISbigger*(*i*, *j*) ef:

  a. $i = 2$ og $j = 5$

  b. $i = 2$ og $j = 6$

```
LISBIGGER(i, j):
  if j > n
      return 0
  else if A[i] ≥ A[j]
      return LISBIGGER(i, j + 1)
  else
      skip ← LISBIGGER(i, j + 1)
      take ← LISBIGGER(j, j + 1) + 1
      return max{skip, take}
```

Annað endurrakningarreiknirit fyrir *LIS*

- Fyrra reikniritið:

  Er $A[j]$ næsta stakið í lengstu hækkandi hlutrunu?

Vinna með inntakið stak fyrir stak

- Skoðum núna:

  Hvar er næsta stakið í lengstu hækkandi hlutrunu?

Hugsa þetta út frá úttakinu

Dæmi:

Höfum ákveðið að 6 verði staki í hlutrunu, hvað stak hægra megin við strikið ætti að vera næst?

3	1	4	1	5	9	2	6	|	$5^?$	$3^?$	$5^?$	$8^?$	$9^?$	$7^?$	$9^?$	$3^?$	$2^?$	$3^?$	$8^?$	$4^?$	$6^?$	$2^?$	$6^?$

Ljóst að ekki öll stökin hægra megin koma til greina, bara þau sem eru > 6

3	1	4	1	5	9	2	6	|	5	3	5	$8^?$	$9^?$	$7^?$	$9^?$	3	2	3	$8^?$	4	6	2	6

Prófum þá bara alla möguleika!

Endurkvæm framsetning

Gefinn vísirinn *i*, finna lengstu hækkandi hlutrunu í *A*[*i*..n], sem byrjar á *A*[*i*]

Köllum þetta *LISfirst*(*i*)

6	5	3	5	8	9	7	9	3	2	3	8	4	6	2	6

Endurkvæm skilgreining:

$$LISfirst(i) = 1 + \max\{LISfirst(j) \mid j > i \text{ og } A[j] > A[i]\}$$

```
LISFIRST(i):
  best ← 0
  for j ← i + 1 to n
      if A[j] > A[i]
          best ← max{best, LISFIRST(j)}
  return 1 + best
```

Athugið að ef $A[j] \leq A[i]$ fyrir öll $j > i$, þá er $LISfirst(i) = 1$

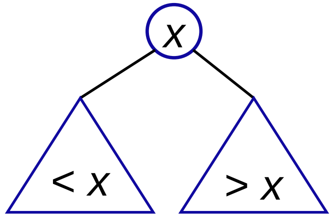
(því þá er A[i] eina stakið í rununni!)

Getum líka notað varðstakið $-\infty$ hér til að finna *LIS*(*A*[1..*n*])

```
LIS(A[1..n]):
  A[0] ← −∞
  return LISFIRST(0) − 1
```

Besta tvíleitartré (*optimal binary search tree*)

- Tvíleitartré eru tvíundartré sem viðhalda eftirfarandi reglu fyrir alla hnúta trésins:



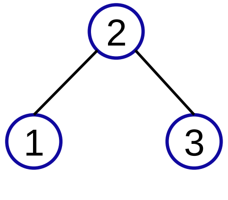
Leyfum oftast ekki endurtekin gildi í trénu

- Ef tréð er í jafnvægi (*balanced*) þá er aðgangstíminn í lágmarki
  - $O(\log(n))$ í versta tilfelli
- Stundum vitum við að sumir lyklar eru vinsælli en aðrir
  - Þá er tré í jafnvægi ekki endilega hagstæðasta tréð

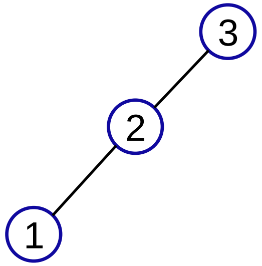
Ef mjög oft leitað að 3 þá gæti verið betra að hafa þann lykil ofar

Dæmi: Höfum lyklana 1, 2, 3

Tvíleitartré í jafnvægi:



Kostar tvo aðganga að finna lykil 3

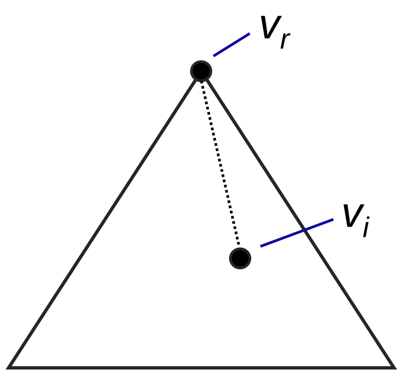


Finna besta tvíleitartré

- Höfum raðað fylki af lyklum (*keys*) $A[1..n]$ og tíðnifylki $f[1..n]$
  - Fjöldi leitana að lykli $A[i]$ er $f[i]$

- Viljum lágmarka heildarkostnað við allar leitirnar
  - Kostnaður við leit að lykli $A[i]$ fer eftir fjarlægð hans frá rótinni í tvíleitartrénu

Lát v_i vera hnútinn sem inniheldur lykil $A[i]$ í tvíleitartrénu T



Heildarkostnaður við leit að lykli i er þá: $f[i] \cdot$ (fjöldi forfeðra v_i í T)

Kostnaður við allar leitir:

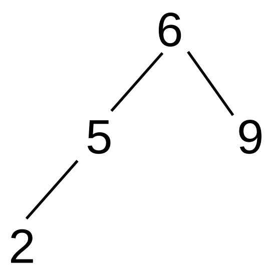
$$Cost(T, f[1,,n]) = \sum_{i=1}^{n} f[i] \cdot (\text{fjöldi forfeðra } v_i \text{ í } T)$$

dýpi hnútar v_i í T

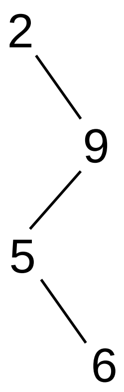
Æfing um tvíleitartré

- Höfum stökin *A* = [2, 5, 6, 9] og tíðnirnar *f* = [6, 1, 0, 3]. Hver er heildarkostnaður við leitirnar ef tvíleitartréð er

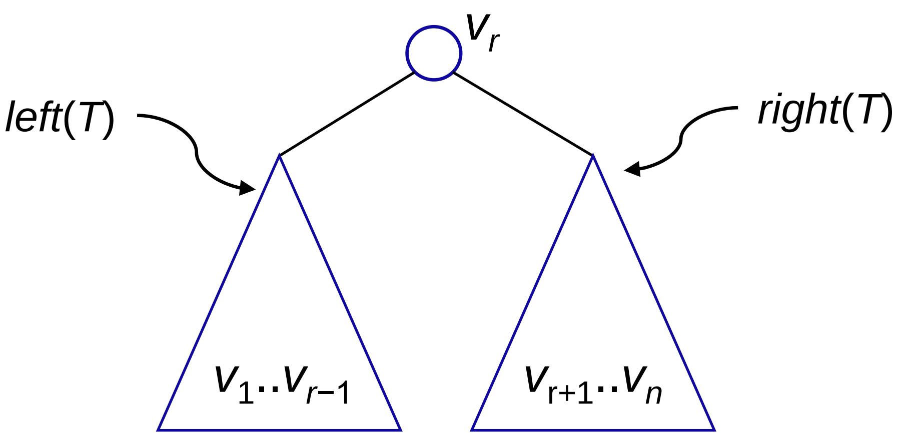
  a.



  b.



Endurkvæm framsetning



Tvíleitartréð T

Getum brotið heildarkostnaðinn upp í þrjá hluta:

Leita að lykli í rótinni: $f[r] \cdot 1$

Leita að lykli í vinstra hluttré: $\sum_{i=1}^{r-1} f[i] \cdot (\text{fjöldi forfeðra } v_i \text{ í } T)$

Leita að lykli í hægra hluttré: $\sum_{i=r+1}^{n} f[i] \cdot (\text{fjöldi forfeðra } v_i \text{ í } T)$

Alltaf:

fjöldi leita "sinnum" fjöldi skrefa

Endurkvæm framsetning

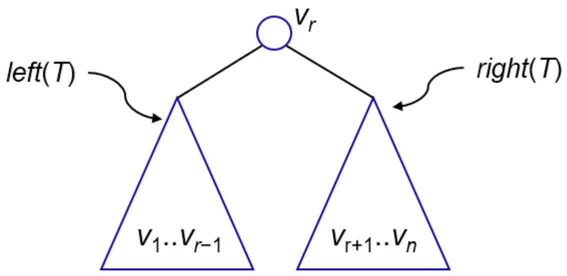
Heildarkostnaður:

$$f[r]\cdot 1 \;+\; \sum_{i=1}^{r-1} f[i]\cdot(\text{fjöldi forfeðra } v_i \text{ í } T) \;+\; \sum_{i=r+1}^{n} f[i]\cdot(\text{fjöldi forfeðra } v_i \text{ í } T)$$

rótin

vinstra hluttré

hægra hluttré



Tvíleitartréð T

Getum tekið kostnað við rótina út úr hinum summunum:

$$f[r]\cdot 1 + \sum_{i=1}^{r-1} f[i]\cdot 1 + \sum_{i=1}^{r-1} f[i]\cdot\big(\text{fjöldi forfeðra } v_i \text{ í } \mathit{left}(T)\big) + \sum_{i=r+1}^{n} f[i]\cdot 1 + \sum_{i=r+1}^{n} f[i]\cdot\big(\text{fjöldi forfeðra } v_i \text{ í } \mathit{right}(T)\big)$$

Einföldum:

$$\boxed{\sum_{i=1}^{n} f[i]\cdot 1 + \sum_{i=1}^{r-1} f[i]\cdot\big(\text{fjöldi forfeðra } v_i \text{ í } \mathit{left}(T)\big) + \sum_{i=r+1}^{n} f[i]\cdot\big(\text{fjöldi forfeðra } v_i \text{ í } \mathit{right}(T)\big)}$$

Umritun kostnaðarfalls

- Getum skrifað þessa formúlu endurkvæmt:

$$Cost(T, f[1..n]) = \sum_{i=1}^{n} f[i] + Cost(left(T), f[1..r-1]) + Cost(right(T), f[r+1..n])$$

Þetta er kostnaðurinn við leitirnar miðað við að lykill *r* sé í rót trésins

En hvaða lykill ætti að vera í rótinni?

Skilgreinum *OptCost*(*i*, *k*): heildarkostnaður við besta tvíleitartré fyrir lyklana *A*[*i*..*k*] með tíðnir *f*[*i*..*k*]

$$OptCost(i,k) = \begin{cases} 0 & \text{ef } i > k \\ \sum_{j=i}^{k} f[i] + \min_{i \le r \le k} \{OptCost(i, r-1) + OptCost(r+1, k)\} & \text{annars} \end{cases}$$

Prófum allar mögulegar rætur og fyrir hverja þeirra finnum við besta vinstra hluttré og besta hægra hluttré!

Tímaflækja

- Í næsta kafla bókarinnar munum við sjá hraðvirka aðferð til að leysa þetta verkefni
  - Það er reiknirit sem nota kvika bestun (*dynamic programming*)

- Aðferðin hér er tímafrek:

$$T(n) = \sum_{k=1}^{n} \big(T(k-1) + T(n-k)\big) + O(n)$$

Getum notað rakningartré til að sýna að lausnin er $T(n) = O(3^n)$

Reyndar er líka hægt að sýna að fjöldi ólíkra tvíleitartrjáa með *n* hnúta er $O(\frac{4^n}{\sqrt{n}})$

Þýðir að þessi aðferð skoðar a.m.k. ekki öll möguleg tvíleitartré!

Fyrirlestraæfingar

1. Hver þarf tíðnidreifing 5 lykla að vera til þess að besta tvíleitartréð sé af hæð 5?

2. Hver er lengd lengstu hækkandi hlutrunu (*LIS*) í eftirfarandi runu:
   [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]?

3. Lyklarnir [5, 10, 15] hafa leitartíðnirnar [5, 2, 3]. Hvaða tvíleitartré lágmarkar meðalleitarkostnað fyrir þá?