TÖL403G GREINING REIKNIRITA

10. Kvik bestun 4

Hjálmtýr Hafsteinsson
Vor 2022

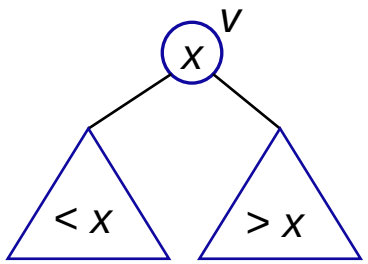
Í þessum fyrirlestri

- Bestu tvíleitartré (*optimal binary search trees*)
  - Endurkvæm formúla (úr kafla 2)
  - Undirverkefnið Tíðnisumma (*frequency count*)
  - Uppsetning sem kvik bestun
    - Gagnagrind og reiknirit

3.9

Bestu tvíleitartré (*binary search trees*)

- Tvíleitartré eru tvíundartré með gildum (lyklum) þar sem fyrir hvern hnút *v* gildir:
  - gildin í vinstra hluttré *v* eru minni en gildið í *v*
  - gildin í hægra hluttré *v* eru stærri en gildið í *v*

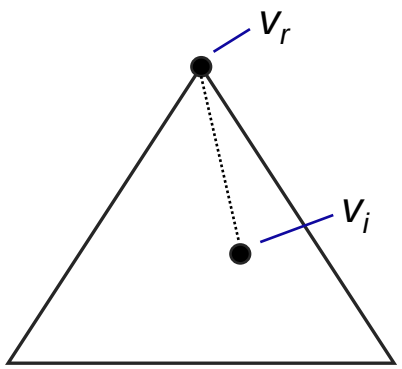


- Hentar vel sem einföld útfærsla á forgangsbiðröð (*priority queue*)
  - "Meðal"-leitartími fyrir *n* gildi er $O(\log(n))$ ← ef tréð er í jafnvægi
  - Versta tilfellis leitartími er $O(n)$ ← ef tréð er keðja af hnútum

- Ef við erum alltaf með sömu *n* lyklana þá getum við búið til tvíleitartré sem lágmarkar leitartímann
  - Þá verðum við að vita hversu oft er leitað að hverjum lykli ← Gætum notað söguleg gögn, eða ágiskun

Besta tvíleitartré

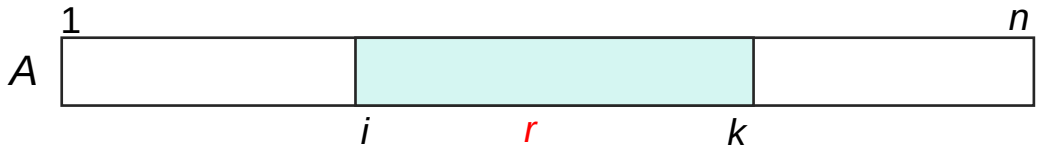
- Höfum raðað fylki af lyklum (*keys*) $A[1..n]$ og tíðnifylki $f[1..n]$
  - Fjöldi leitana að lykli $A[i]$ er $f[i]$
- Viljum lágmarka heildarleitartímann
  - Kostnaður við eina leit er fjöldi hnúta frá rótinni niður í hnútinn sem inniheldur lykilinn
- Heildarkostnaðurinn við allar leitir í tré T er þá:



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

Hnútur v_i er forfaðir sjálfs síns!

Endurkvæm formúla



- Skilgreindum fallið *OptCost*(*i*, *k*):

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

- Fengum svo endurkvæma formúlu fyrir það:

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

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

Tíminn er $O(3^n)$ ef við notum þessa formúlu sem reiknirit

Undirverkefnið Tíðnisumma

- Endurkvæma formúlan inniheldur summu
  - Tekur tíma að reikna hana út

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

- Getum reiknað hana með kvikri bestun!
  - Fáum einfalt og "skemmtilegt" verkefni til að æfa kvika bestun

Ef við förum beint eftir þessari formúlu þá mun þessi útreikningur taka $O(n^3)$ tíma!

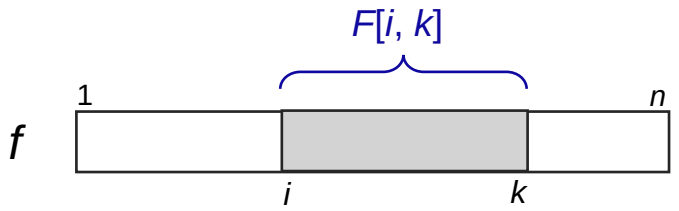
Við erum að gera sömu samlagningarnar aftur og aftur

```
fyrir i = 1 til n
   fyrir k = i til n
      F[i, k] = 0
      fyrir j = i til k
         F[i, k] = F[i, k] + f[j]
```

Tíðnisumma í kvikri bestun

- Erum að finna allar hlutsummur *n*-staka fylkis *f*

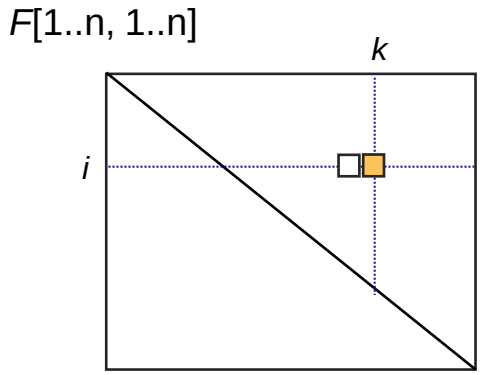
$$F[i, k] = \sum_{j=i}^{k} f[j]$$



Segjum að við höfum reiknað $F[i, k]$

Þegar við reiknum $F[i, k+1]$ þá leggjum við aftur saman allar tölurnar frá $F[i]$ upp í $F[k]$ og leggjum svo $f[k+1]$ við það

Við ættum að geyma milliniðurstöður og nýta þær



Endurkvæm framsetning:

$F(i, k) = F(i, k-1) + f[k]$

og $F(i, i) = f[i]$

Reiknirit:

```
fyrir i = 1 til n
    F[i, i] = f[i]
    fyrir k = i+1 til n
        F[i, k] = F[i, k-1] + f[k]
```

Nú er greinilegt að tíminn er $O(n^2)$

Aftur í bestu tvíleitartré

- Nú höfum við aðeins einfaldari endurkvæma formúlu:

$$OptCost(i, k) = \begin{cases} 0 & \text{ef } i > k \\ F[i,k] + \min\limits_{i \le r \le k} \{OptCost(i, r-1) + OptCost(r+1, k)\} & \text{annars} \end{cases}$$

Við reiknum þessi gildi út fyrirfram á $O(n^2)$ tíma

Undirverkefni:

Skilgreind af tveimur vísum *i* og *k*:
$1 \le i \le n+1$ og $0 \le k \le n$

Gagnagrind:

Tvívíða fylkið *OC*[1..*n*+1, 0..*n*] geymir öll möguleg gildi á fallinu *OptCost*

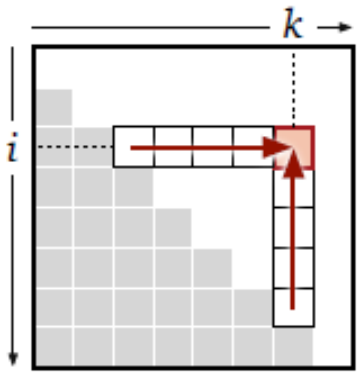
Kvik bestun á tvíleitartrjám

Tengsl milli staka:

Hvert stak *OC*[*i*, *k*] byggir á stökunum
OC[*i*, *j*−1] og *OC*[*j*+1, *k*] fyrir öll *j*, þ.a. $i \le j \le k$

Stök vinstra megin við *OC*[*i*, *k*]

Stök fyrir neðan *OC*[*i*, *k*]



Fall sem reiknar út *OC*[*i*, *k*]:

```
ComputeOptCost(i, k):
  OptCost[i, k] ← ∞
  for r ← i to k
      tmp ← OptCost[i, r − 1] + OptCost[r + 1, k]
      if OptCost[i, k] > tmp
          OptCost[i, k] ← tmp
  OptCost[i, k] ← OptCost[i, k] + F[i, k]
```

Upphafsstilla OC[i, k] með stóru gildi

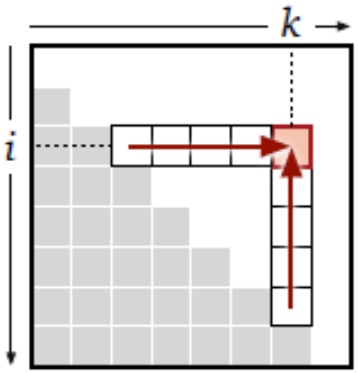
Reikna eina mögulega rót

Ef hún betri en það sem komið er, þá uppfæra *OC*[*i*, *k*]

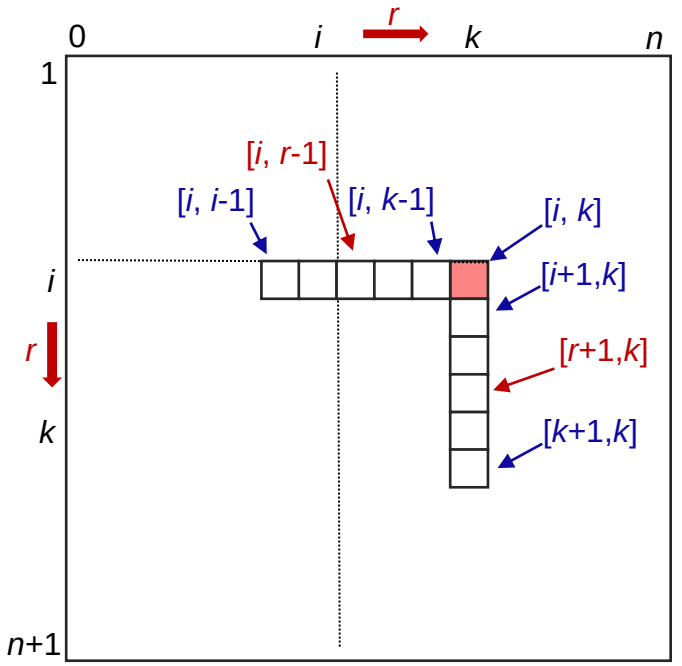
Bæta svo við hlutsummu tíðnigildanna (reiknað fyrirfram)

Útreikningur á einu staki í fylkinu

```
ComputeOptCost(i, k):
    OptCost[i, k] ← ∞
    for r ← i to k
        tmp ← OptCost[i, r − 1] + OptCost[r + 1, k]
        if OptCost[i, k] > tmp
            OptCost[i, k] ← tmp
    OptCost[i, k] ← OptCost[i, k] + F[i, k]
```



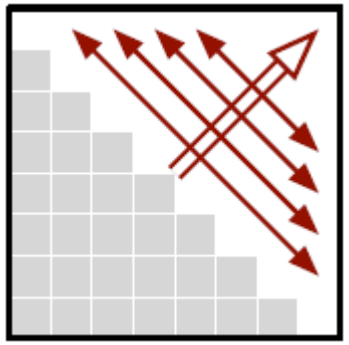
Prófum allar mögulega hnúta sem rót og finnum þann sem gefur lægstan kostnað



Útreikningsröð

- Eðlilegasta útreikningsröð:

Skálína fyrir skálínu



Vitum að *OptCost*(*i*, *k*) = 0 ef *i* > *k*
Setjum því *OC*[*i*, *i*−1] = 0

Reiknirit:

```
OptimalBST(f[1..n]):
  InitF(f[1..n])
  for i ← 1 to n+1
      OptCost[i, i−1] ← 0
  for d ← 0 to n−1
      for i ← 1 to n−d        ⟨⟨... or whatever⟩⟩
          ComputeOptCost(i, i+d)
  return OptCost[1, n]
```

Reikna allar hlutsummur tíðnigilda

Hornalínan er öll 0

Hver skálínan á fætur annari

Niðurstaðan kemur efst til hægri

Sýnidæmi

- Höfum lyklafylkið A = [1, 2, 3, 4, 5] og tíðnifylkið f = [10, 2, 5, 2, 3]
- Reiknum fyrst út hlutsummur tíðnigildanna:

F

	1	2	3	4	5
1	10	12	17	19	22
2		2	7	9	12
3			5	7	10
4				2	5
5					3

Núllstillum hornalínuna

Næsta skálína er bara tíðnifylkið f eða $F[i, i]$

OC

	0	1	2	3	4	5
1	0	10				
2	-	0	2			
3	-	-	0	5		
4	-	-	-	0	2	
5	-	-	-	-	0	3
6	-	-	-	-	-	0

Sýnidæmi, frh.

F

	1	2	3	4	5
1	10	12	17	19	22
2		2	7	9	12
3			5	7	10
4				2	5
5					3

OC[1, 2] = *F*[1, 2] + **min**{ *OC*[1, 0] + *OC*[2, 2],
OC[1, 1] + *OC*[3, 2] }

= 12 + min{ 0+2, 10+0 }
= 12 + 2
= 14

OC

	0	1	2	3	4	5
1	0	10	**14**			
2	-	0	2			
3	-	-	0	5		
4	-	-	-	0	2	
5	-	-	-	-	0	3
6	-	-	-	-	-	0

Erum að ákveða hér:

Ef við höfum lyklana [1, 2] með tíðnirnar [10, 2] hvor lykillinn á að vera rót?

Ef *r* = 1 þá kostnaðurinn 10*1 + 2*2
= 12 + (0+2) = **14**

1
2

Ef *r* = 2 þá kostnaðurinn 10*2 + 2*1
= 12 + (10+0) = **22**

2
1

Æfingadæmi

F

	1	2	3	4	5
1	10	12	17	19	22
2		2	7	9	12
3			5	7	10
4				2	5
5					3

Reiknið *OC*[2, 5]:

Það er *F*[2, 5] + lággildi af fjórum gildum:

min{ ____ + ____,
____ + ____,
____ + ____,
____ + ____ }

Hvert af þessum fjórum gildum er lægst?

Hvaða lykil er best að hafa sem rót á [2, 3, 4, 5]?

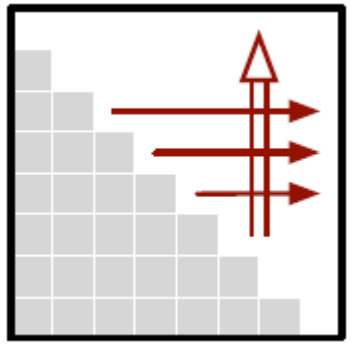
OC

	0	1	2	3	4	5
1	0	10	14	26	32	
2	-	0	2	9	13	
3	-	-	0	5	9	17
4	-	-	-	0	2	7
5	-	-	-	-	0	3
6	-	-	-	-	-	0

Önnur útreikningsröð

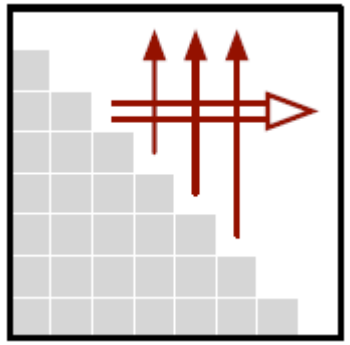
- Það þarf ekki endilega að reikna fylkið eftir skálínum
- Getum reiknað það eftir línum:

```
OptimalBST2(f[1..n]):
    InitF(f[1..n])
    for i ← n+1 downto 1
        OptCost[i, i−1] ← 0
        for j ← i to n
            ComputeOptCost(i, j)
    return OptCost[1, n]
```



- eða eftir dálkum:

```
OptimalBST3(f[1..n]):
    InitF(f[1..n])
    for j ← 0 to n+1
        OptCost[j+1, j] ← 0
        for i ← j downto 1
            ComputeOptCost(i, j)
    return OptCost[1, n]
```



Tímaflækja

- Í öllum útreikningsröðum þarf að fylla í ~$n^2/2$ hólf fylkisins
- Útreikningur á hverju hólfi tekur $O(n)$ tíma

- Heildartímaflækja: $O(n^3)$
- Minnisnotkun: $O(n^2)$

Bæði fyrir *OC*-fylkið og *F*-fylkið

Donald Knuth náði að bæta tímaflækjuna niður í $O(n^2)$ með viðbótarnokun á kvikri bestun

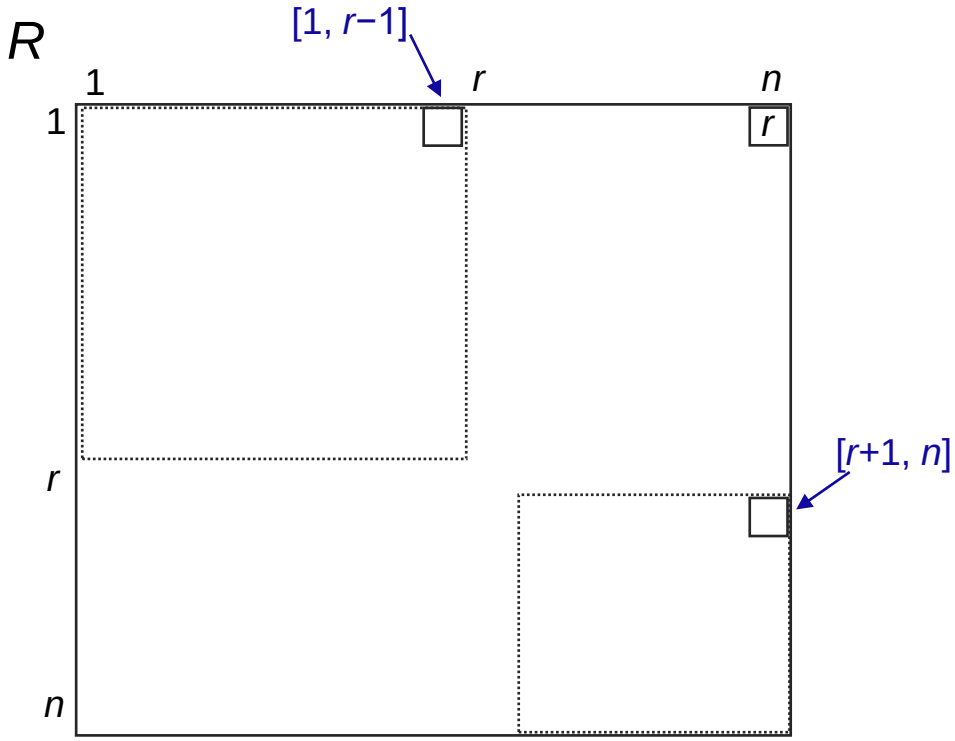
Ef n er mjög stórt gildi þá er $O(n^2)$ ennþá frekar tímafrekt reiknirit

Kurt Mehlhorn sýndi $O(n)$ tíma gráðugt reiknirit sem finnur oftast besta tréð

Velur rótina þannig að tíðnisumma vinstri og hægri hluttrjáa sé sem líkust

Að finna best tréð

- Geymum í hvert sinn það *r* sem gefur lægsta kostnaðinn í öðru tvívíðu fylki *R*
  - Þá er R[*i*, *k*] það *r* sem gaf lægsta kostnað við útreikning á *OC*[*i*, *k*]
- Getum þá rakið okkur til baka:
  - Finnum fyrst rót trésins í *R*[1, *n*] = *r*
  - Þá er rót vinstra hluttrésins í *R*[1, *r*−1]
  - og rót hægra hluttrésins í *R*[*r*+1, *n*]
- Finnum síðan rætur hluttrjánna á sama hátt
- Ef við höfum fylkið *R* þá tekur þetta aðeins *O*(*n*) tíma



Fyrirlestraæfingar

1. Hvernig væri besta tvíleitartréð í laginu ef jafnar líkur eru á öllum lyklum?

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

3. Hvaða gildi kemur í hornalínuna á fylkinu *OptCost*, þ.e. hvert er gildið á *OptCost*[*i*, *i*]?