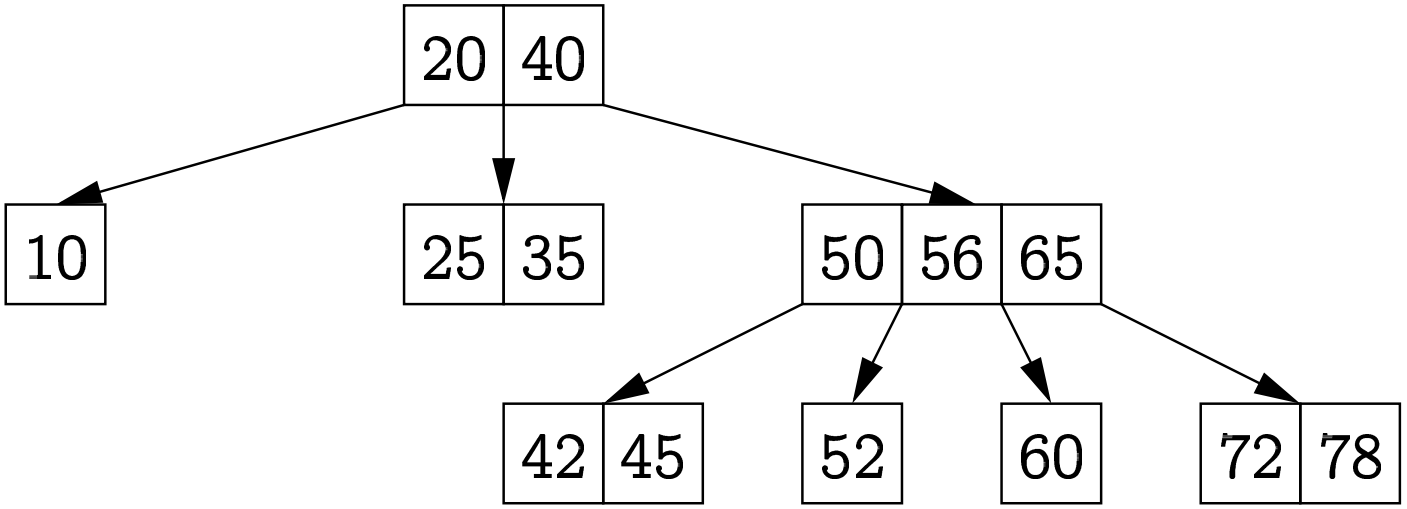
T on *m-rajaline otsimispuu*, kui

- Igas tipus on 0 kuni $m-1$ kirjet võtmetega $k_1 \leqslant k_2 \leqslant \cdots \leqslant k_j$ ning kui see tipp ei ole leht, siis $j+1$ alluvat, mis on alampuude $T_0, \ldots, T_j$ juurteks.
- Kui k on võti alampuus T_0, siis $k \leqslant k_1$.
- Kui k on võti alampuus T_j, siis $k \geqslant k_j$.
- Kui k on võti alampuus T_i $(1 \leqslant i \leqslant j-1)$, siis $k_i \leqslant k \leqslant k_{i+1}$.



Otsimine m-rajalises otsimispuus on sarnane otsimisega kahendotsimise puus.

m-järku B-puu on m-rajaline otsimispuu, kus

- m on paaritu;
- kõik lehed on juurest samal kaugusel;
- kõigis tippudes, v.a. juures on vähemalt $\lfloor m/2 \rfloor$ kirjet;
- juures on vähemalt üks kirje.

2-3 puu on 3. järku B-puu.

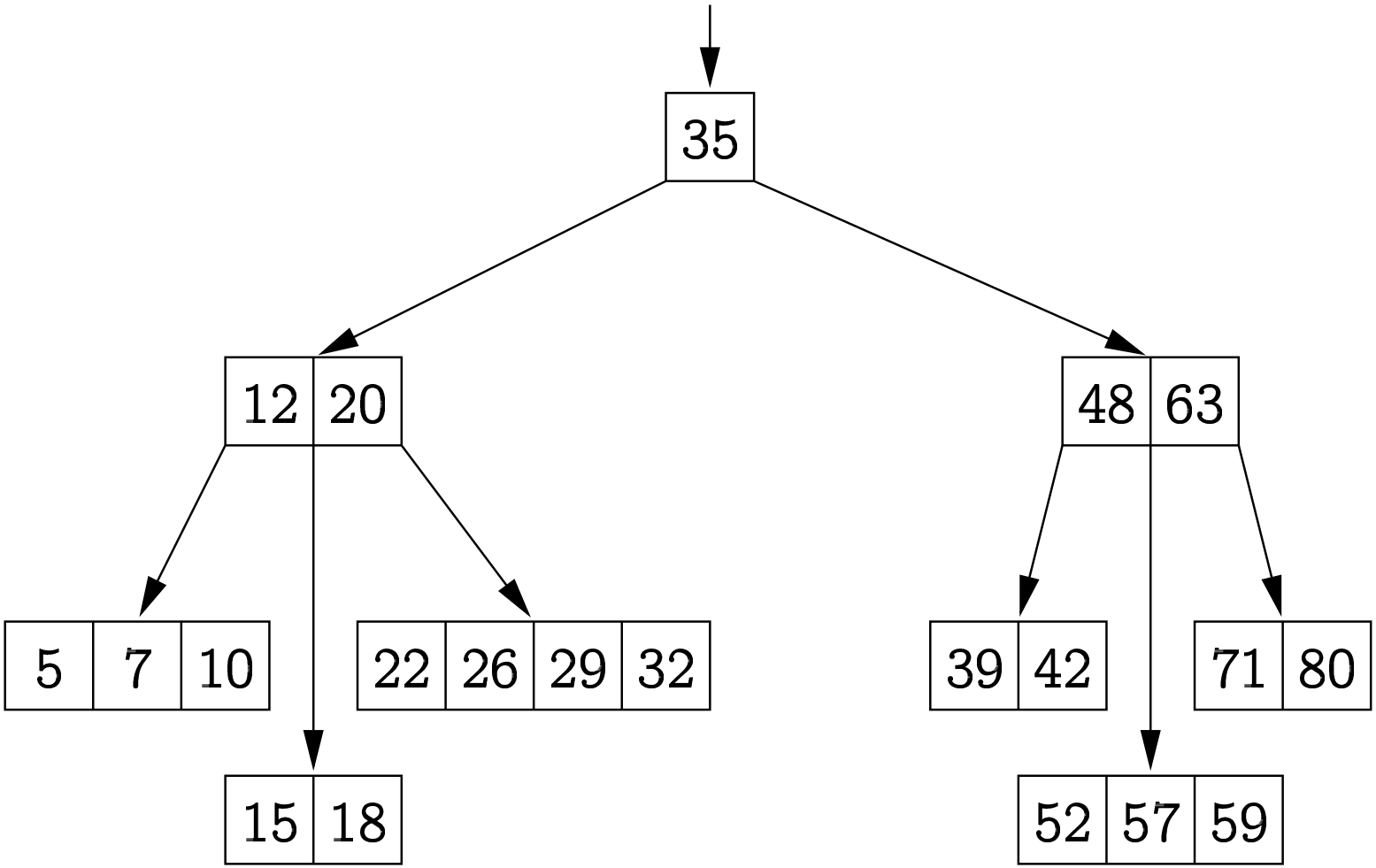
- Sama kasutusvaldkond, mis AVL-puul.

m-järku B-puus kõrgusega h on vähemalt $2\frac{(m/2)^h - 1}{m/2 - 1} =$ $2^{\Theta(h)}$ tippu.

Kujutagu m-järku B-puu tippu järgmine kirje:

Siin *.pikkus* näitab, mitu kirjet tipus tegelikult on.

Näide 5. järku B-puust



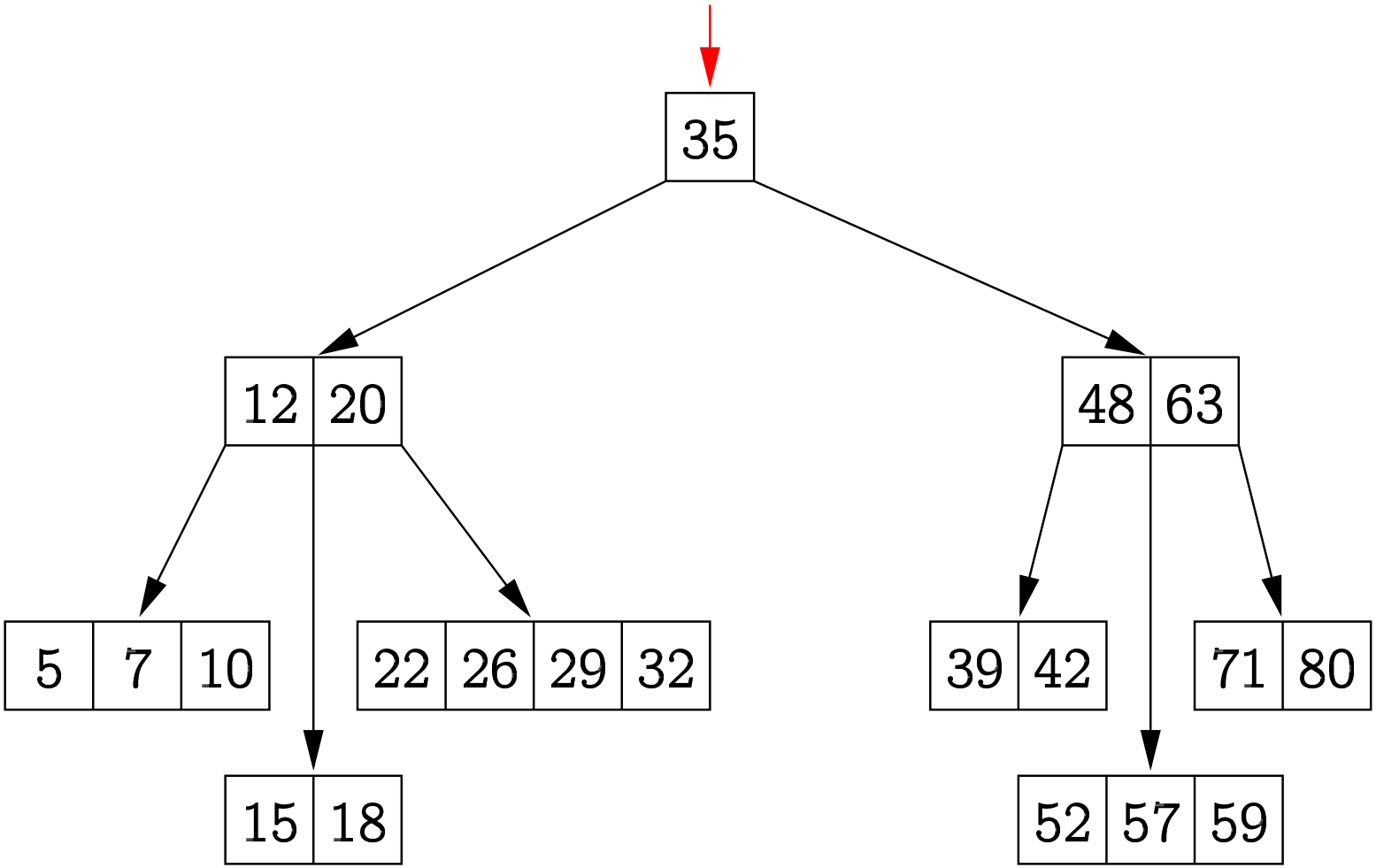
Otsimine B-puus on sarnane otsimisega kahendpuus.

Kui otsitavat võtit x antud tipus p ei ole, siis võrdleme x-i väärtustega $p.võti[1]$ kuni $p.võti[m-1]$.

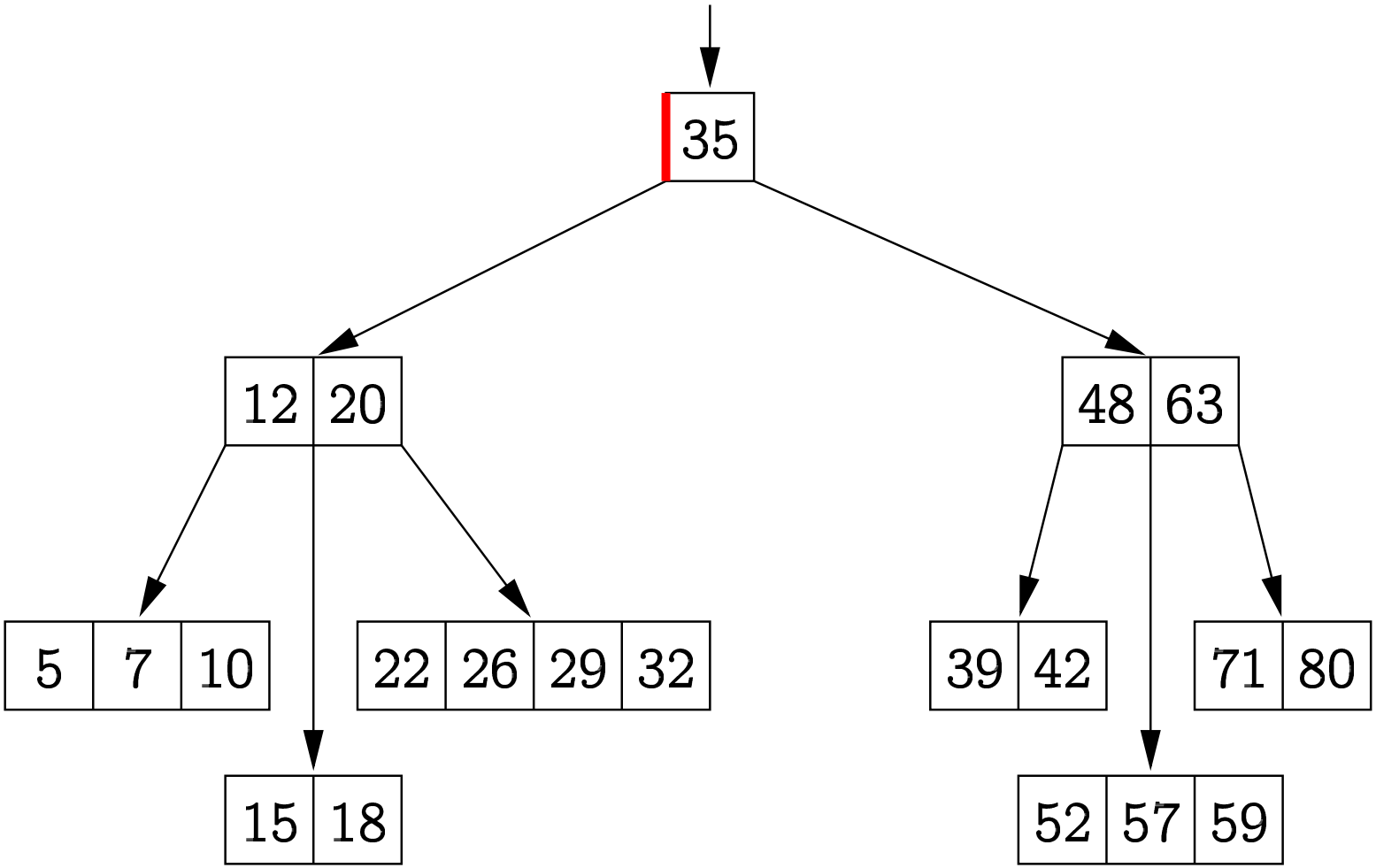
Keerukus: $O(m)$, aga m on konstant.

Võrdlemistulemus määrab, millisest alluvast otsimist jätkame.

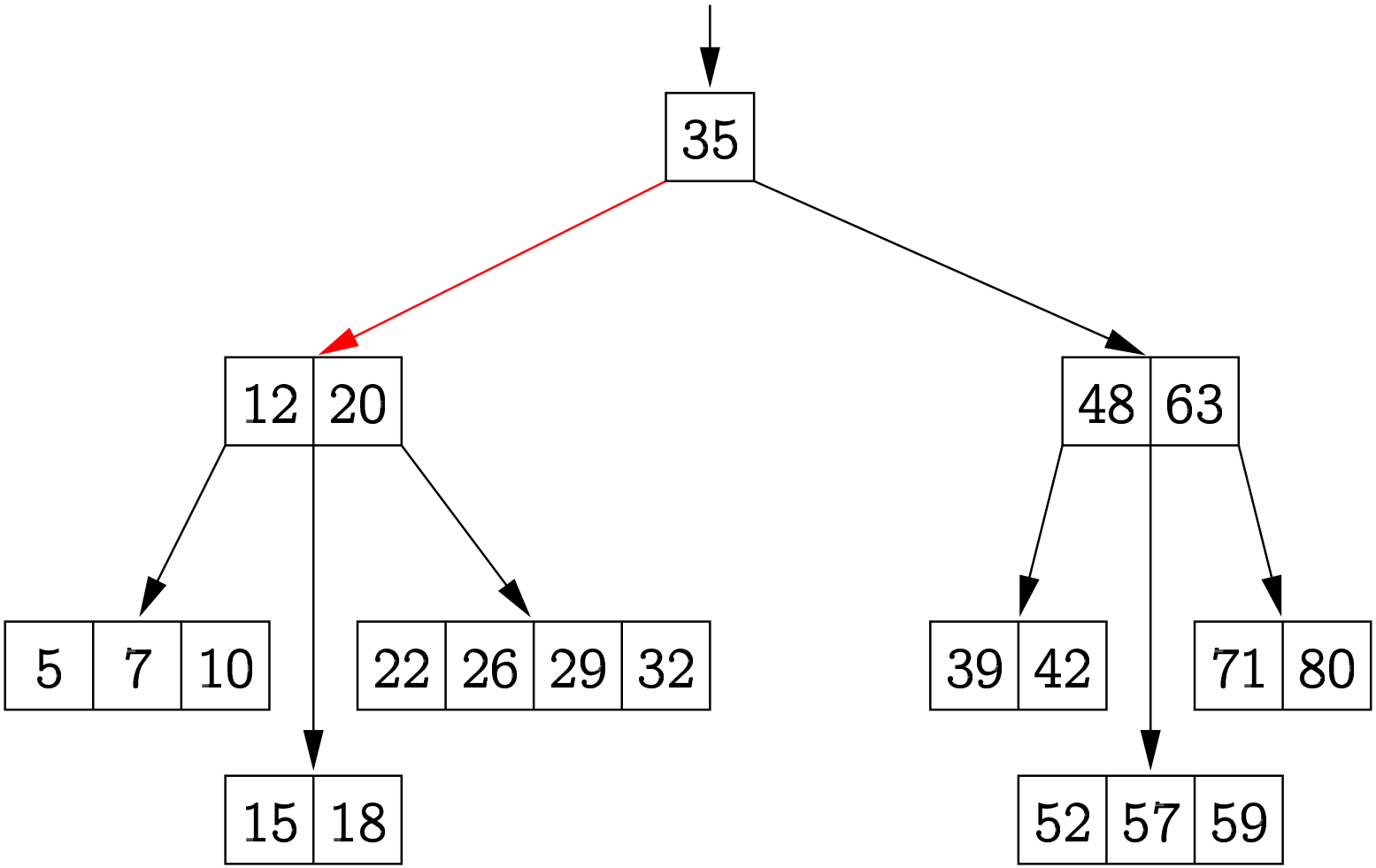
Otsime võtit 18



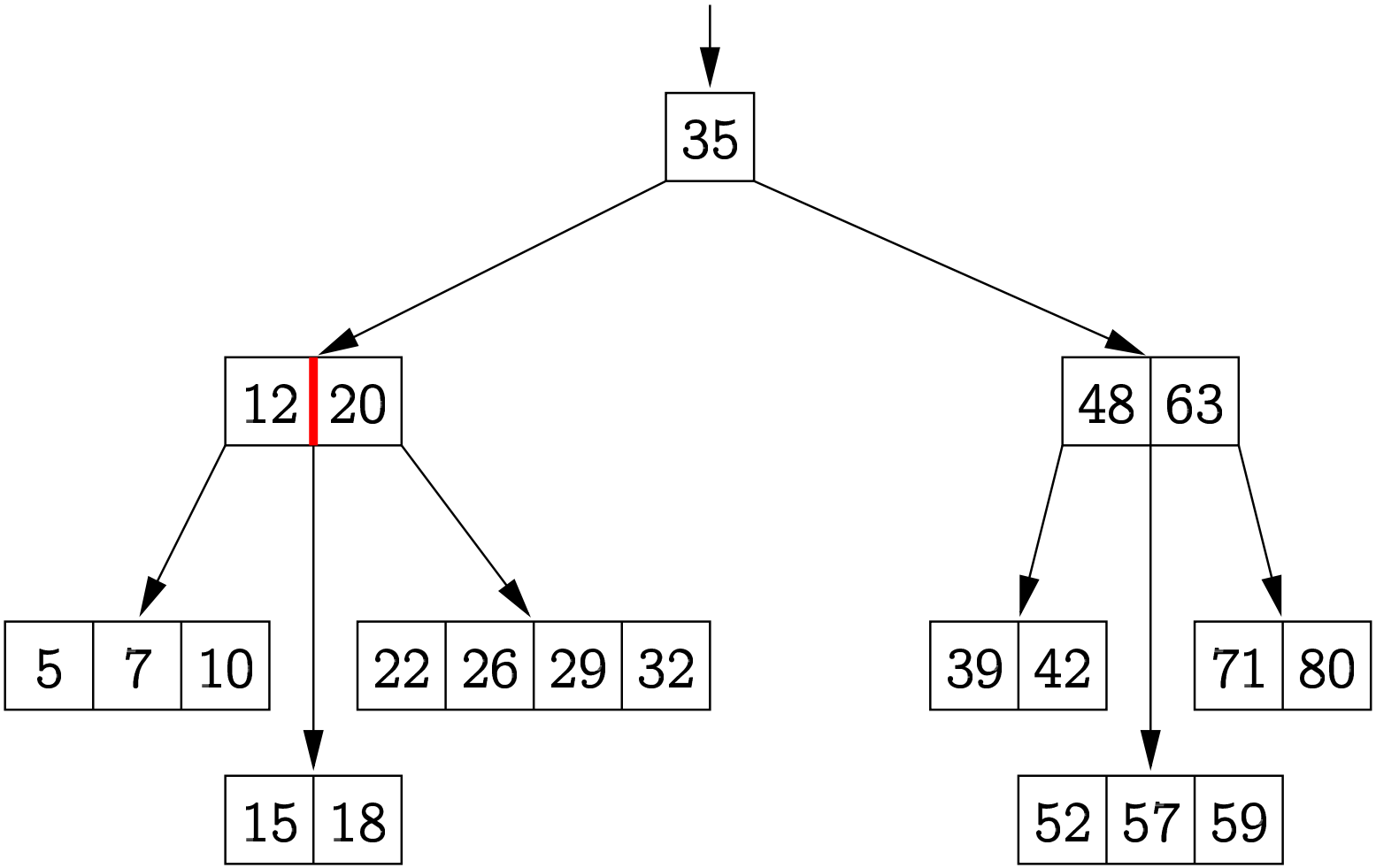
Otsime võtit 18



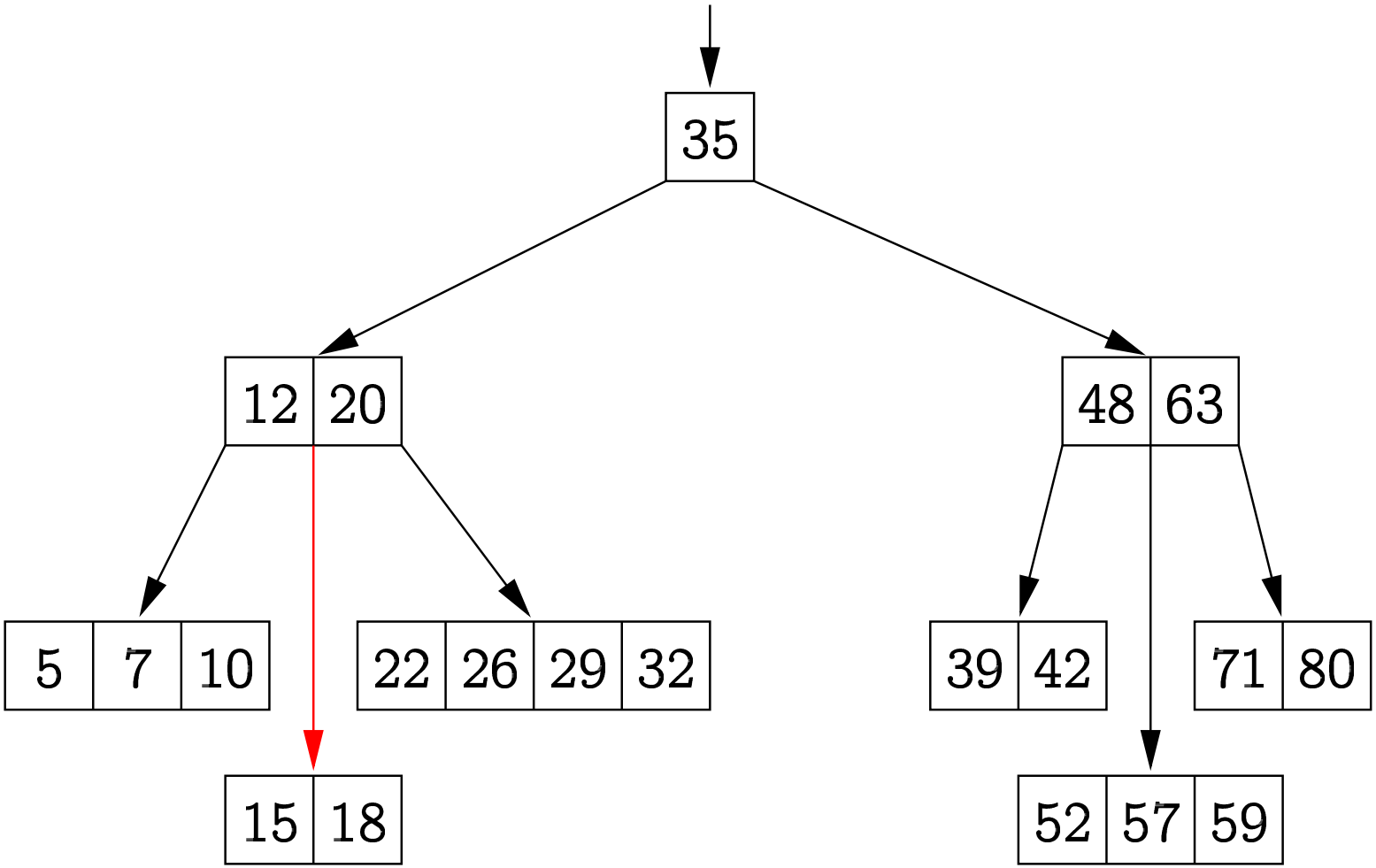
Otsime võtit 18



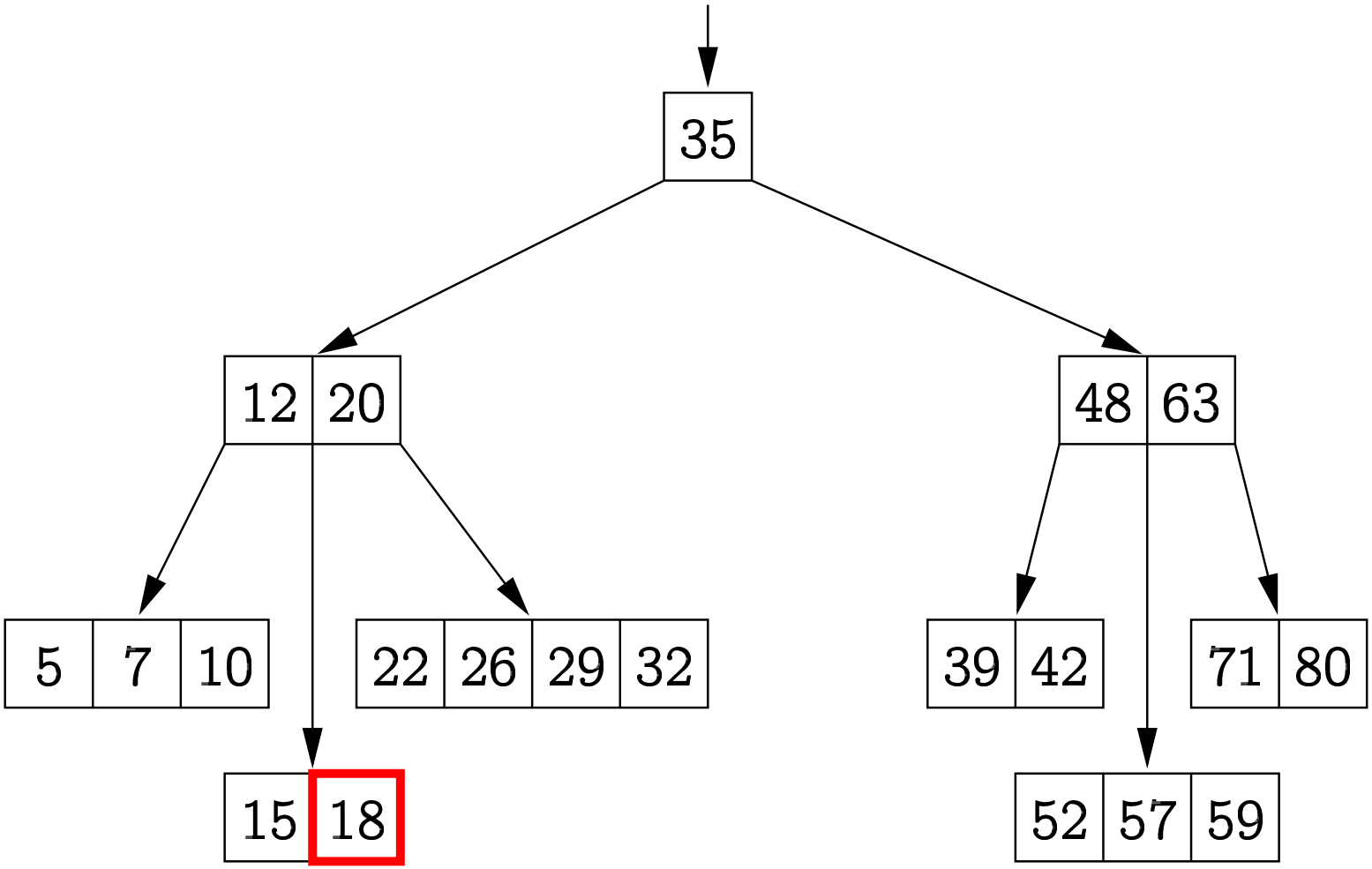
Otsime võtit 18



Otsime võtit 18



Otsime võtit 18



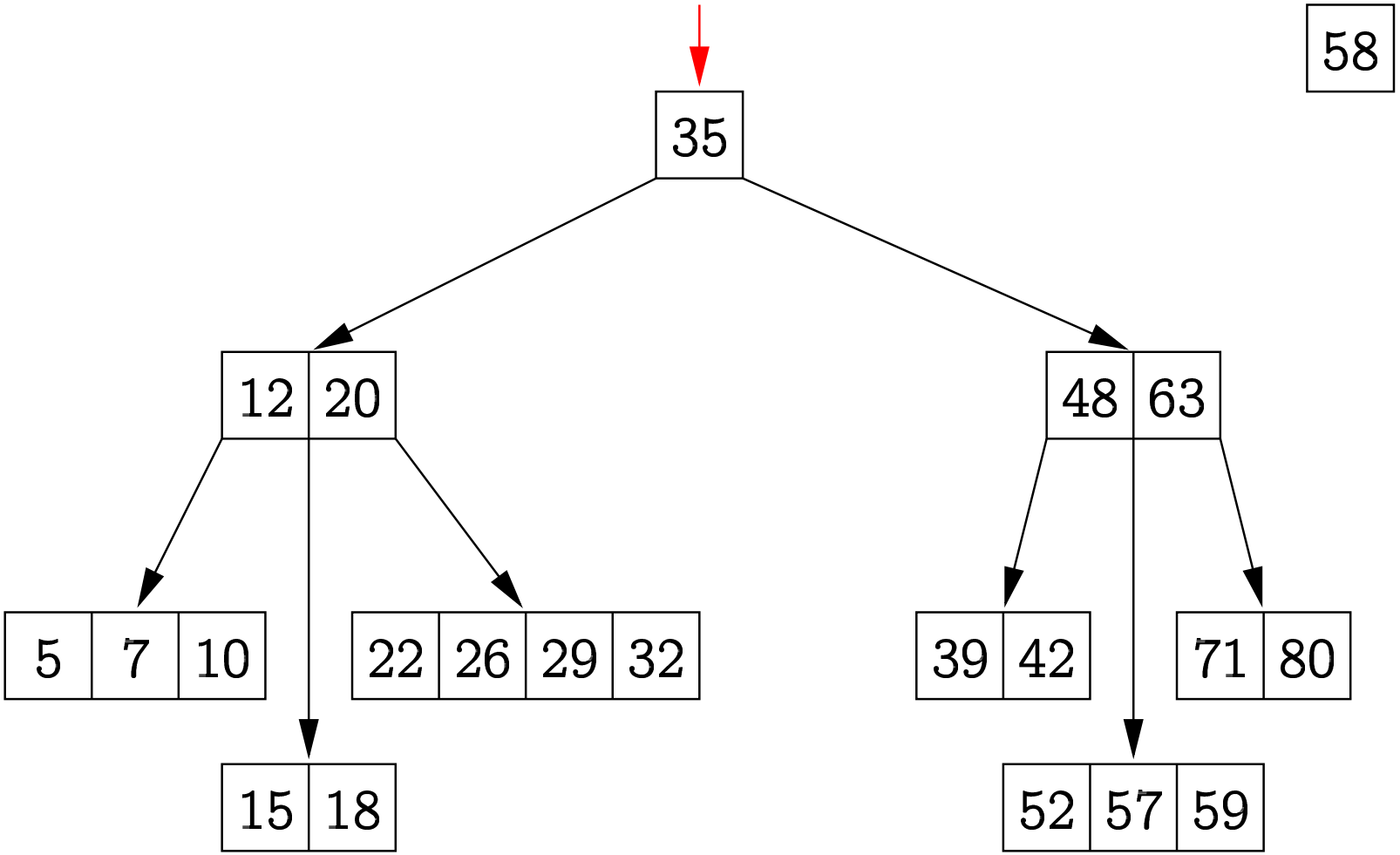
Lisades uut kirjet, eeldame, et vastavat võtit puus veel ei ole.

Kui on, siis liidame lisatava kirje võtmele ε-i.

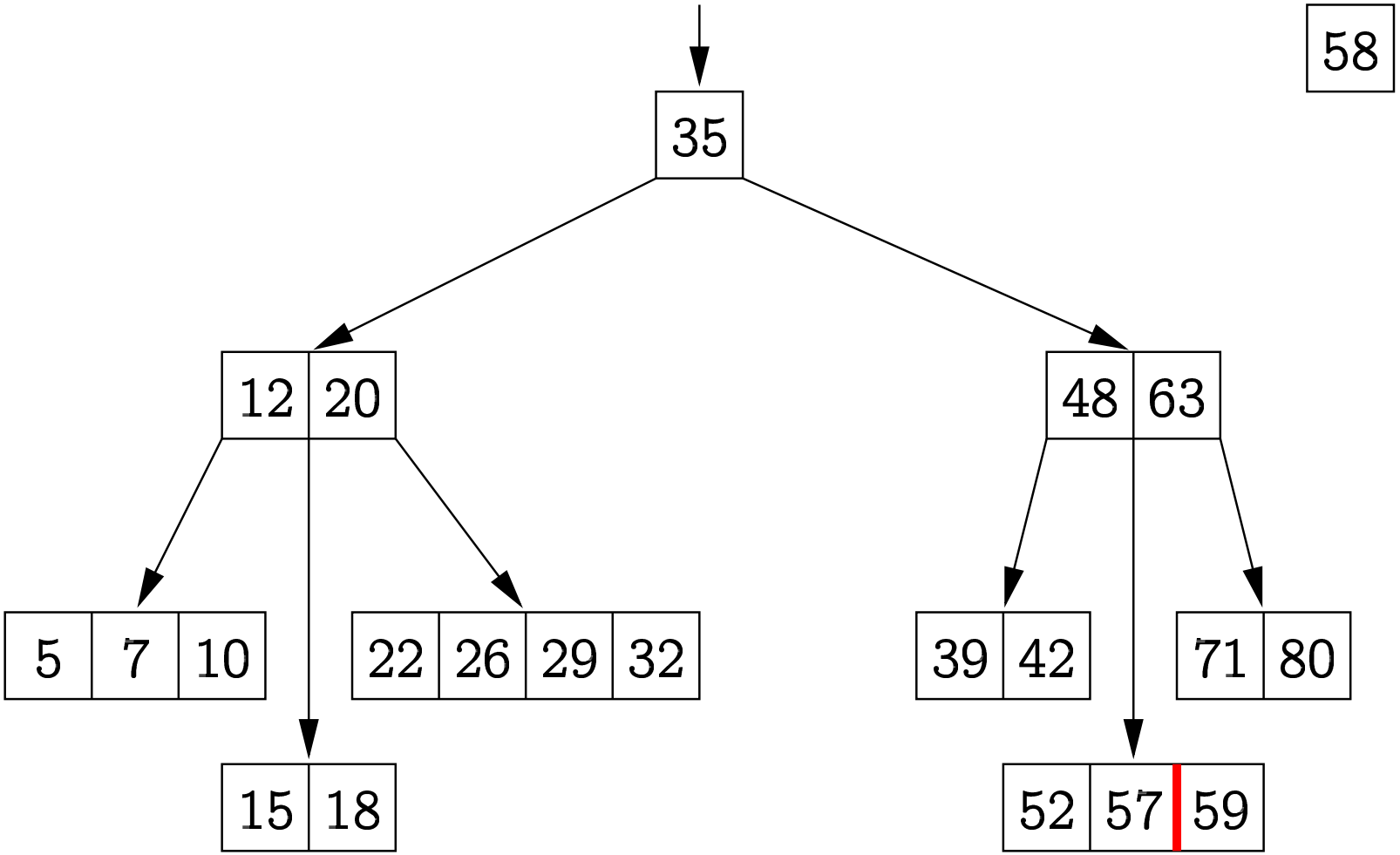
Lisamisel leitakse kõigepealt koht, kuhu vastav kirje käib. See koht on alati mingis lehes.

Kui selles lehes on vähem kui $m - 1$ kirjet, siis suurendatakse lihtsalt seda lehte.

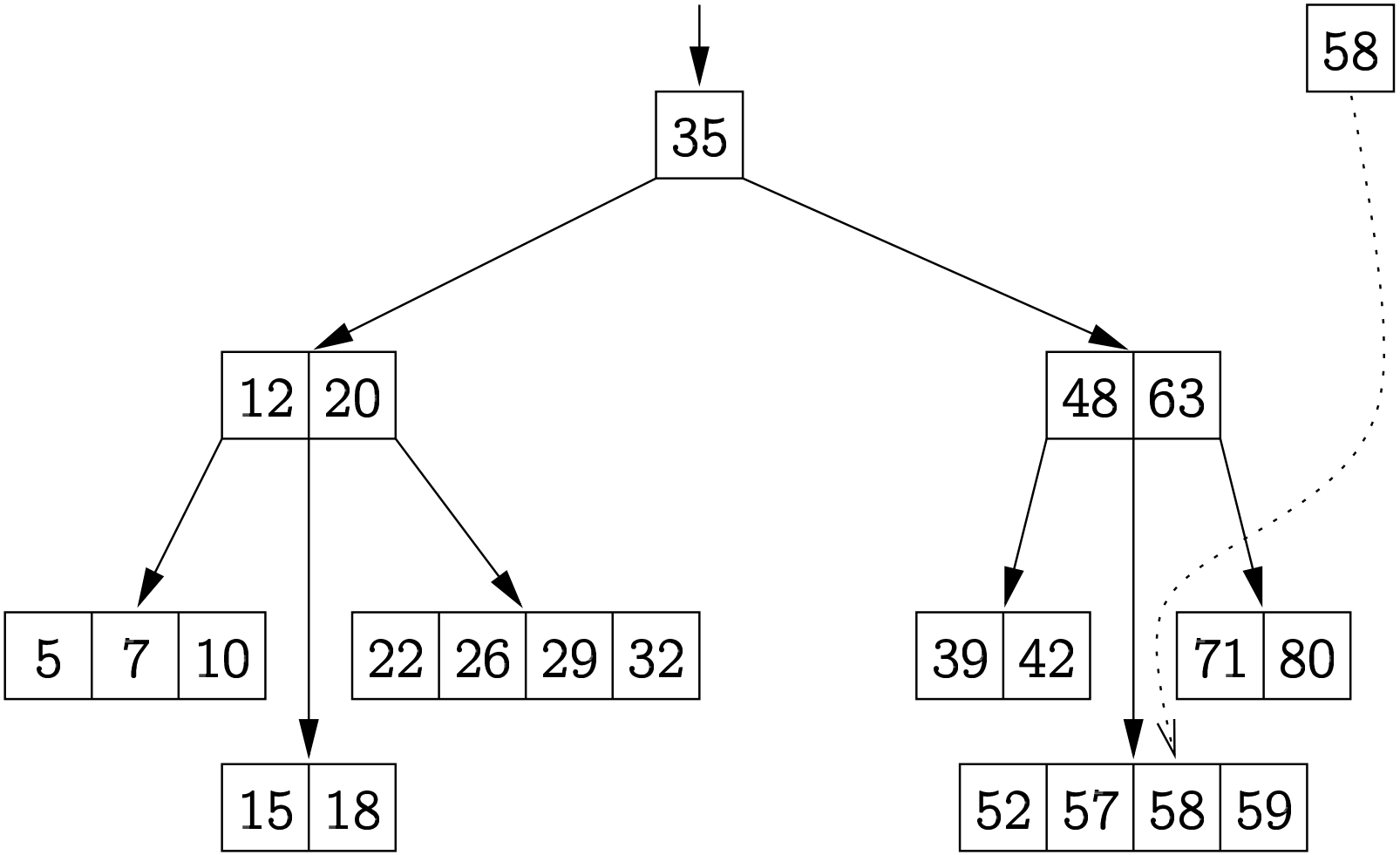
Lisame võtme 58



Lisame võtme 58



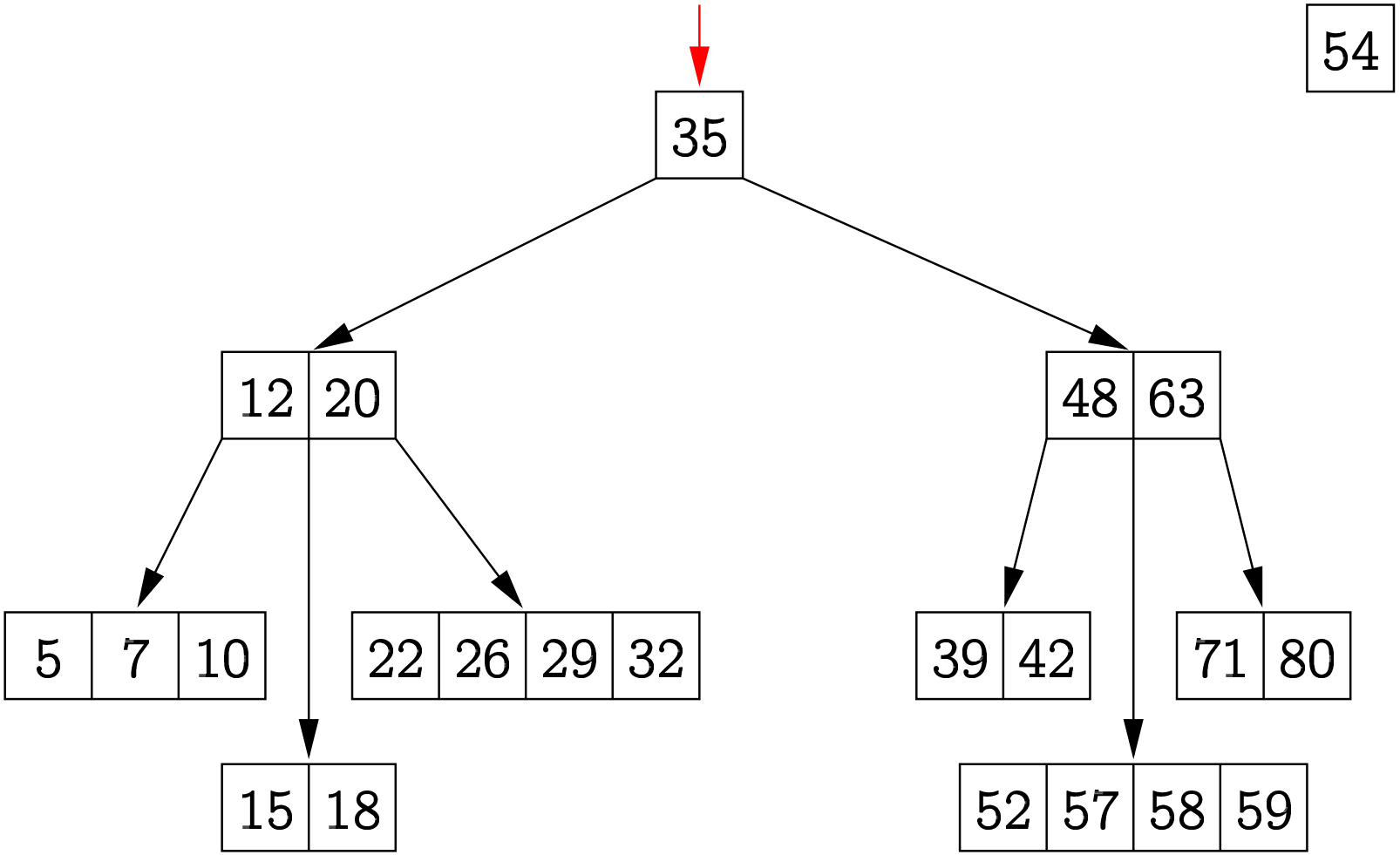
Lisame võtme 58



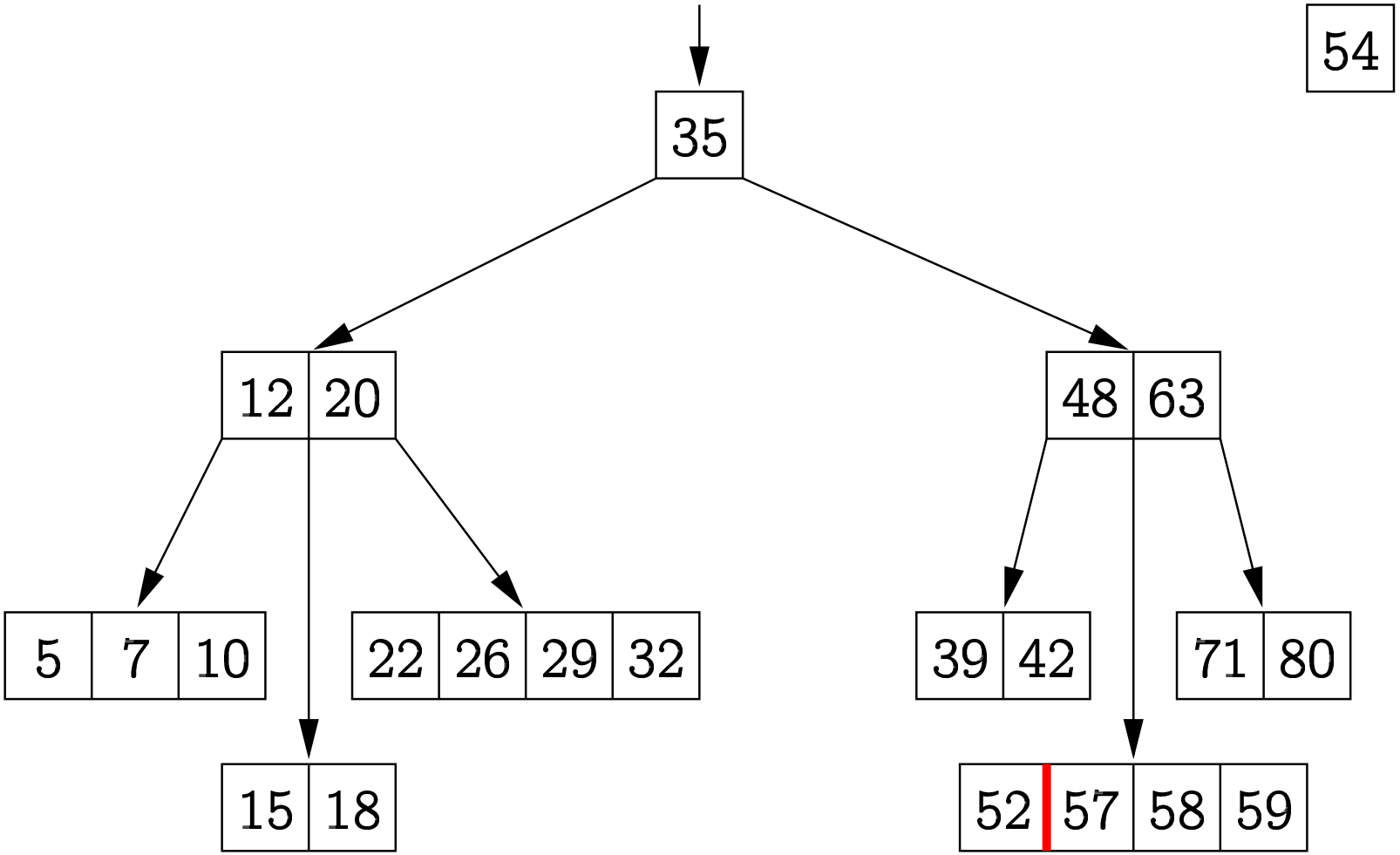
Mingi tipu ületäitumisel (kui sinna tuleb m kirjet) viiakse keskmine kirje ülemusse.

Tipu vasakust ja paremast poolest tehakse kaks tippu, milles on $\lfloor m/2 \rfloor$ kirjet.

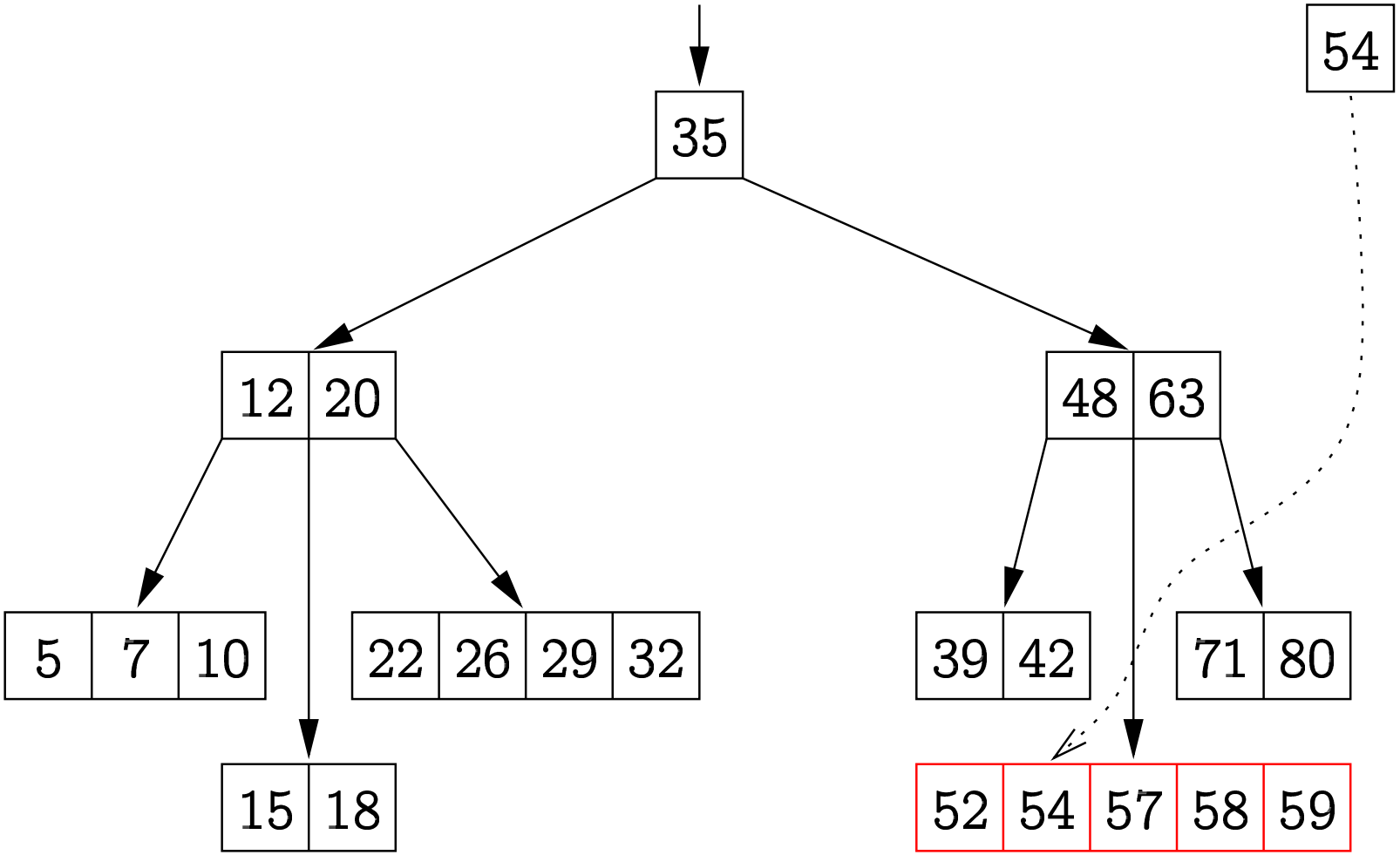
Lisame võtme 54



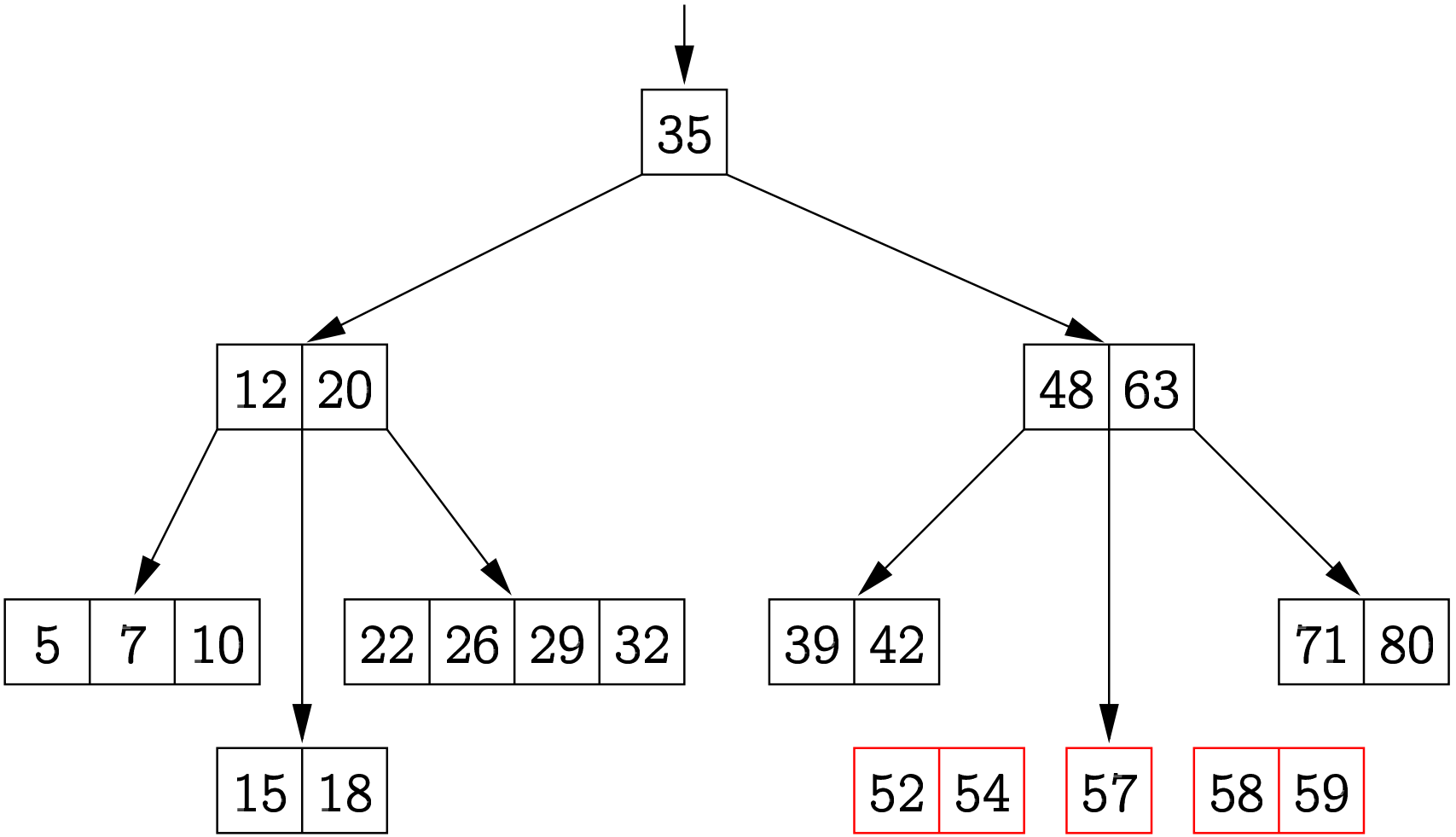
Lisame võtme 54



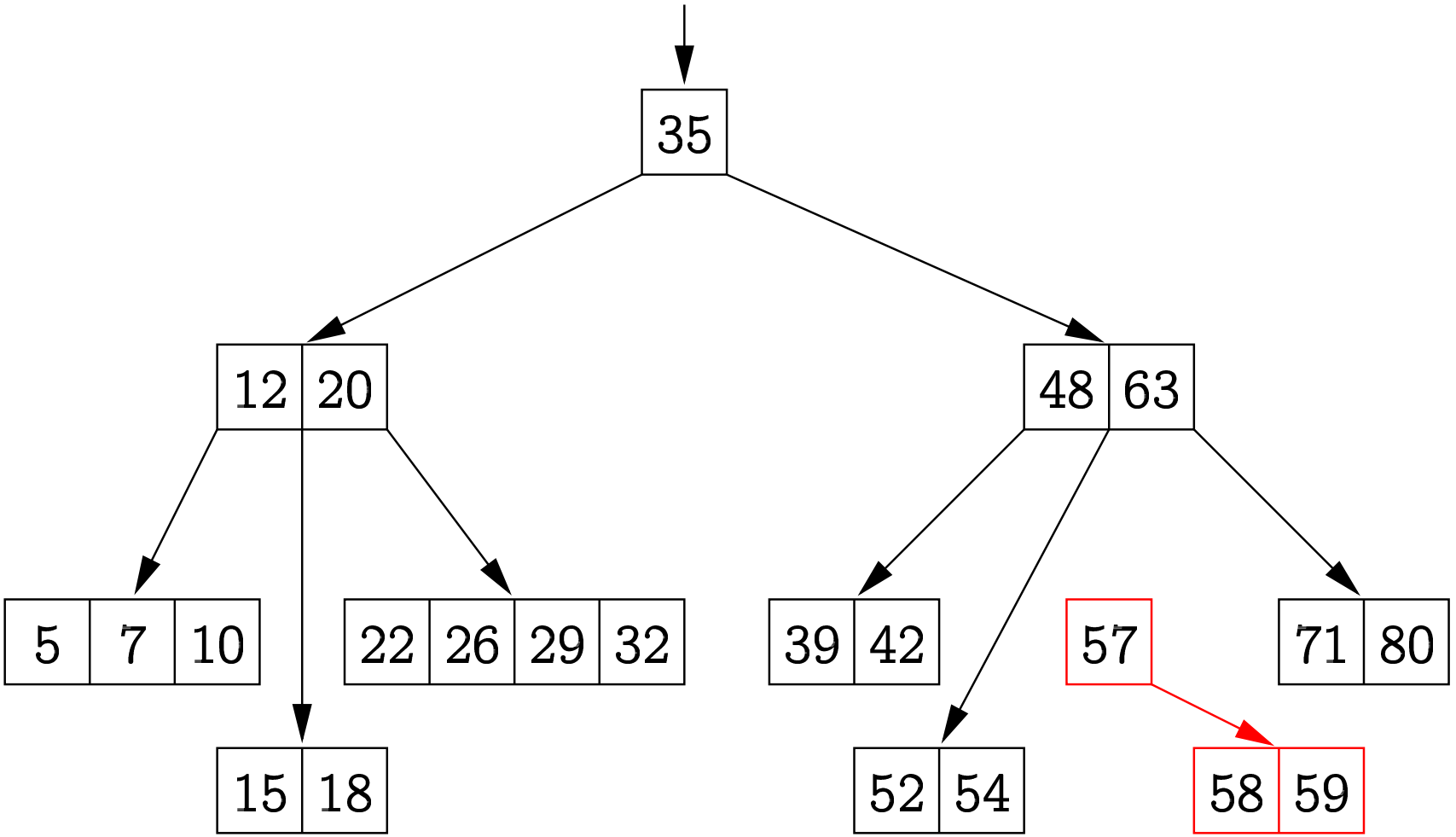
Lisame võtme 54



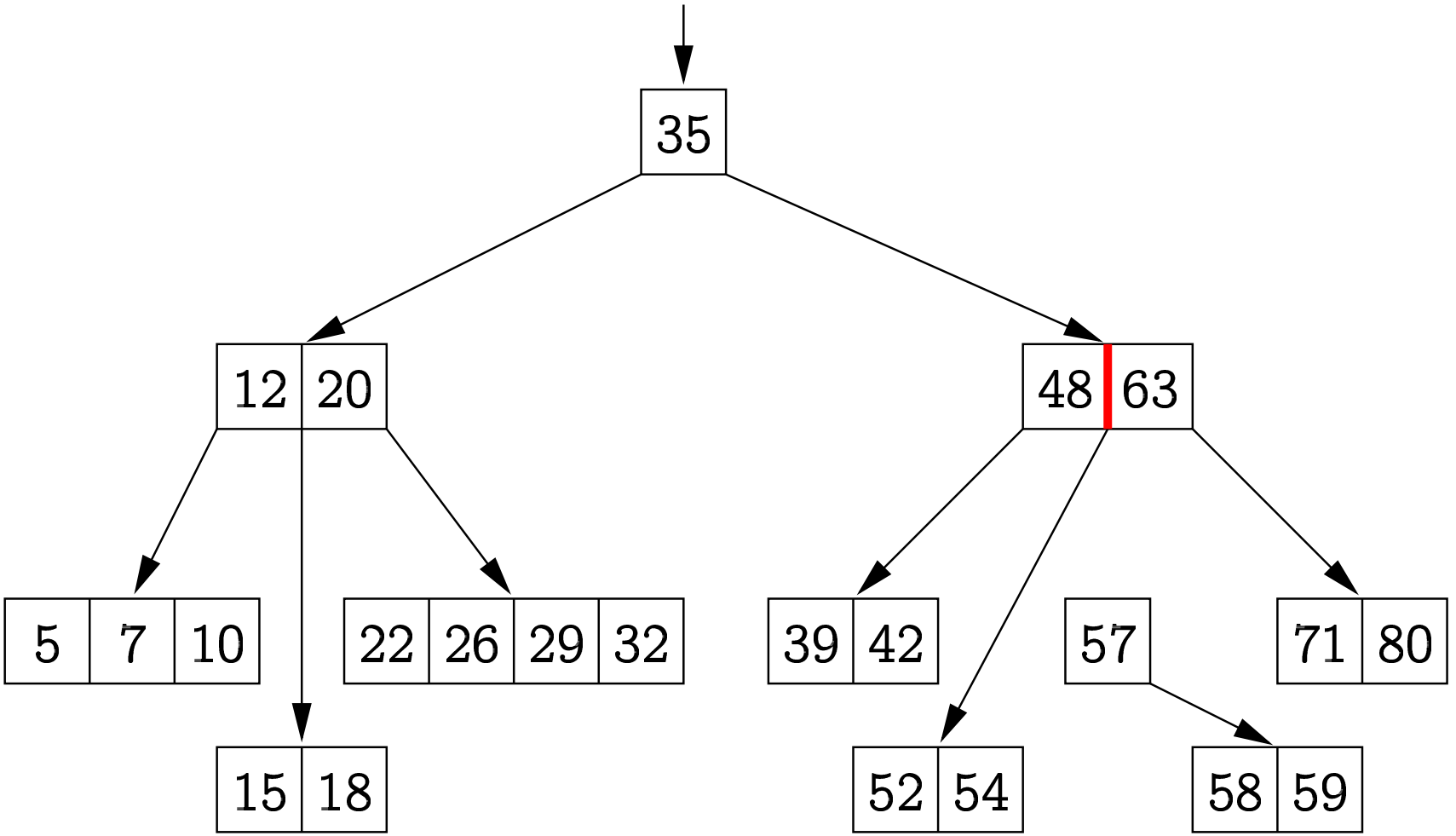
Lisame võtme 54



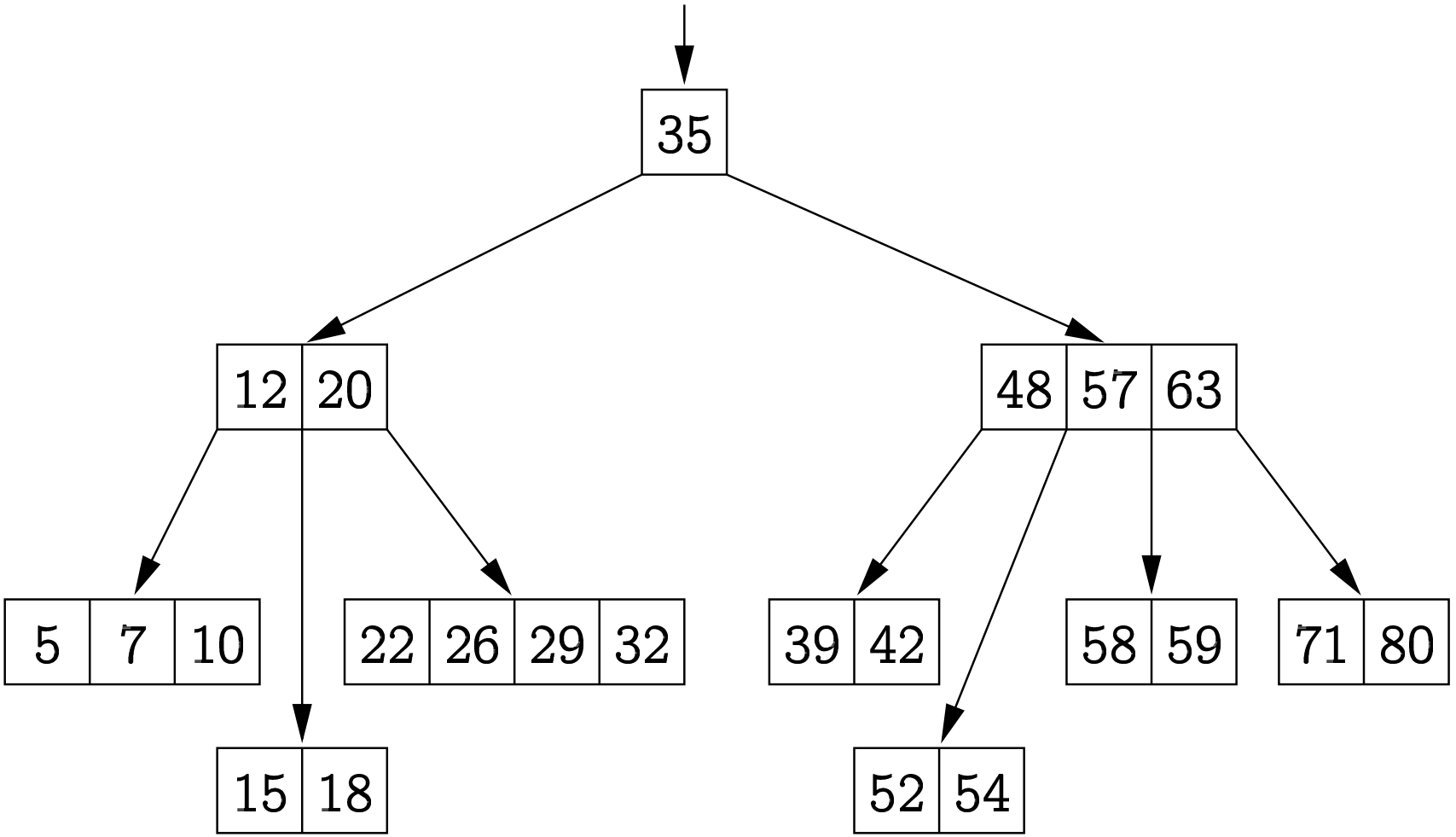
Lisame võtme 54



Lisame võtme 54

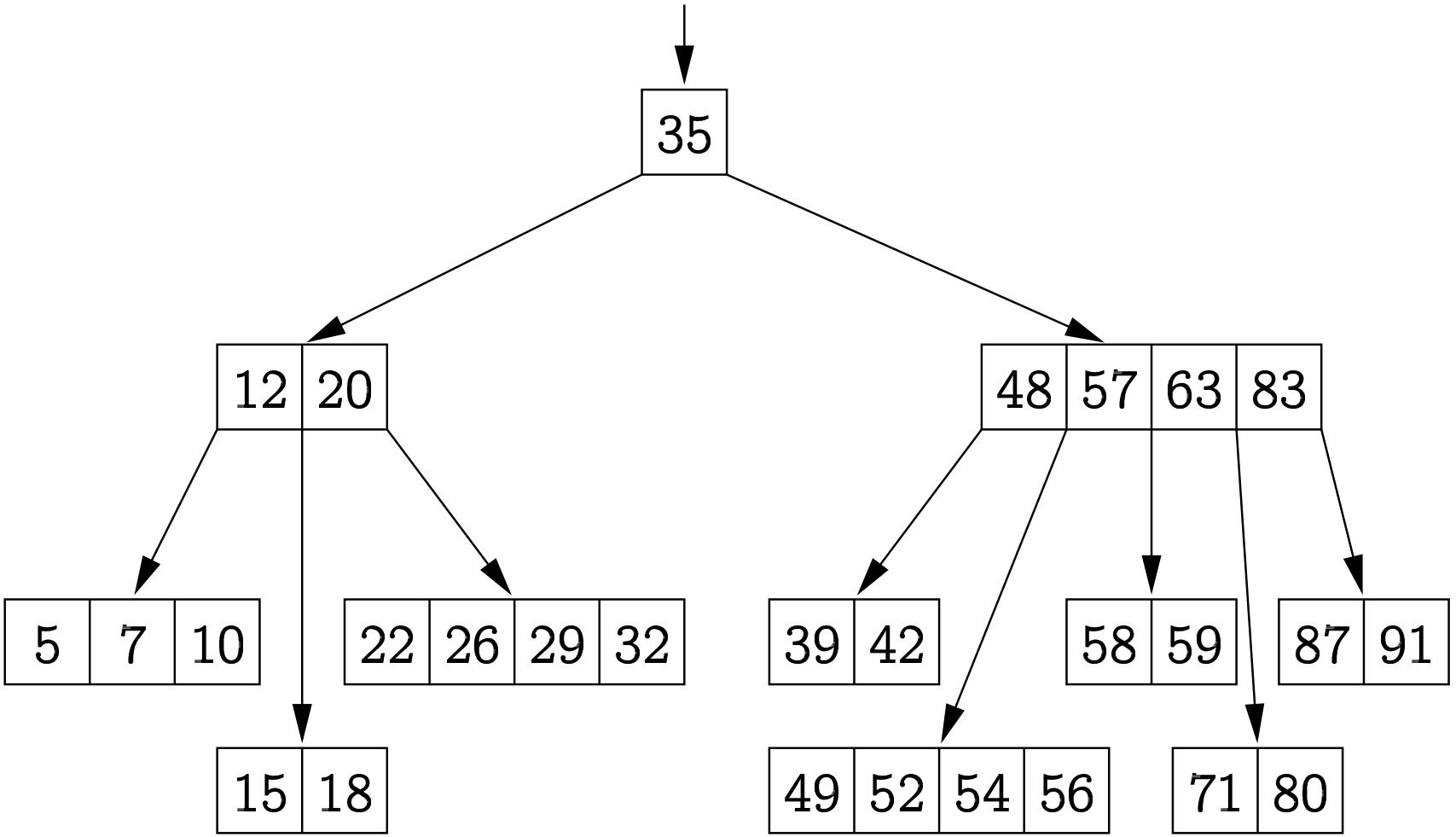


Lisame võtme 54

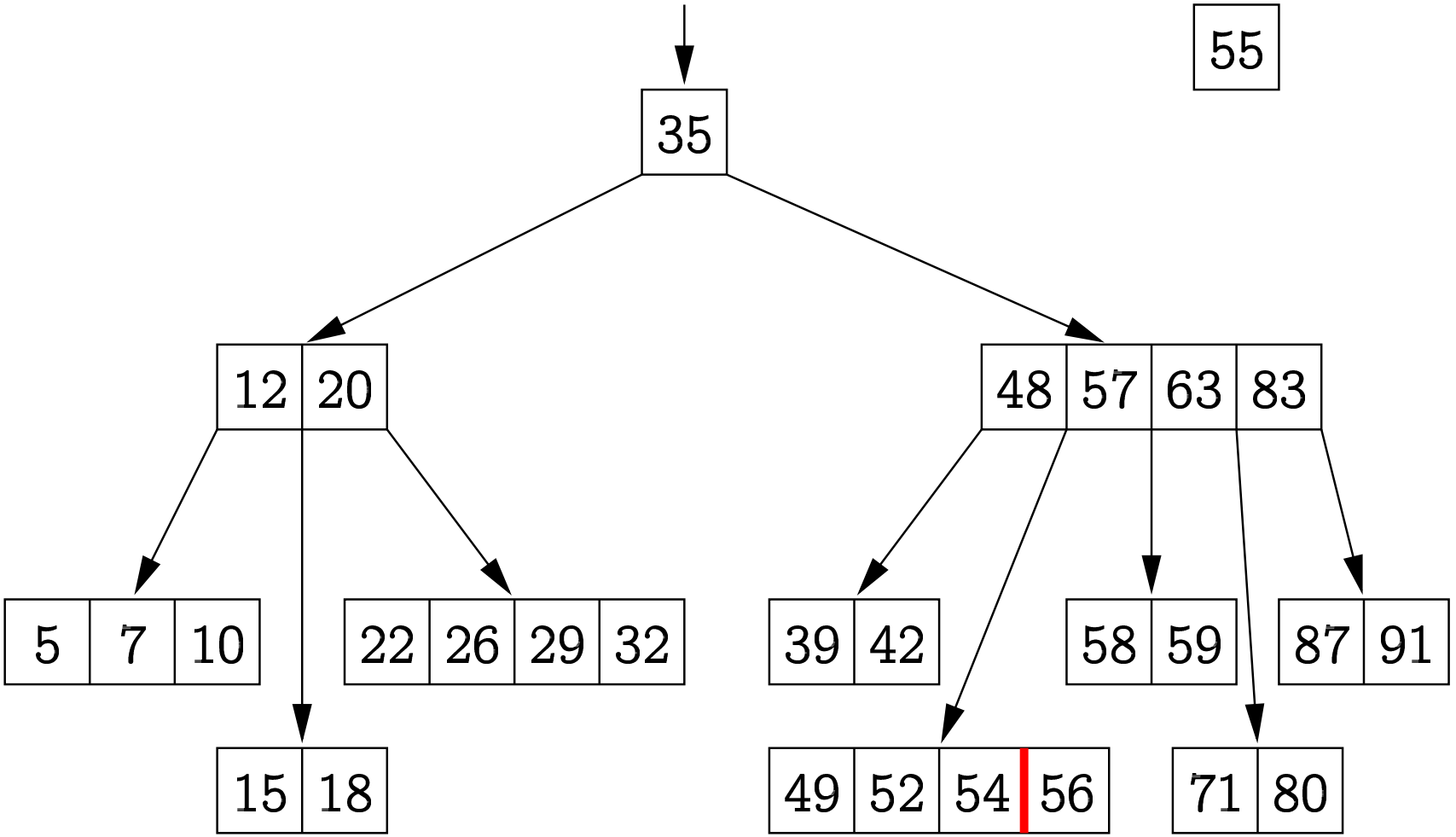


Ülemusse tipu lisamisel võib omakorda ülemus ületäituda.

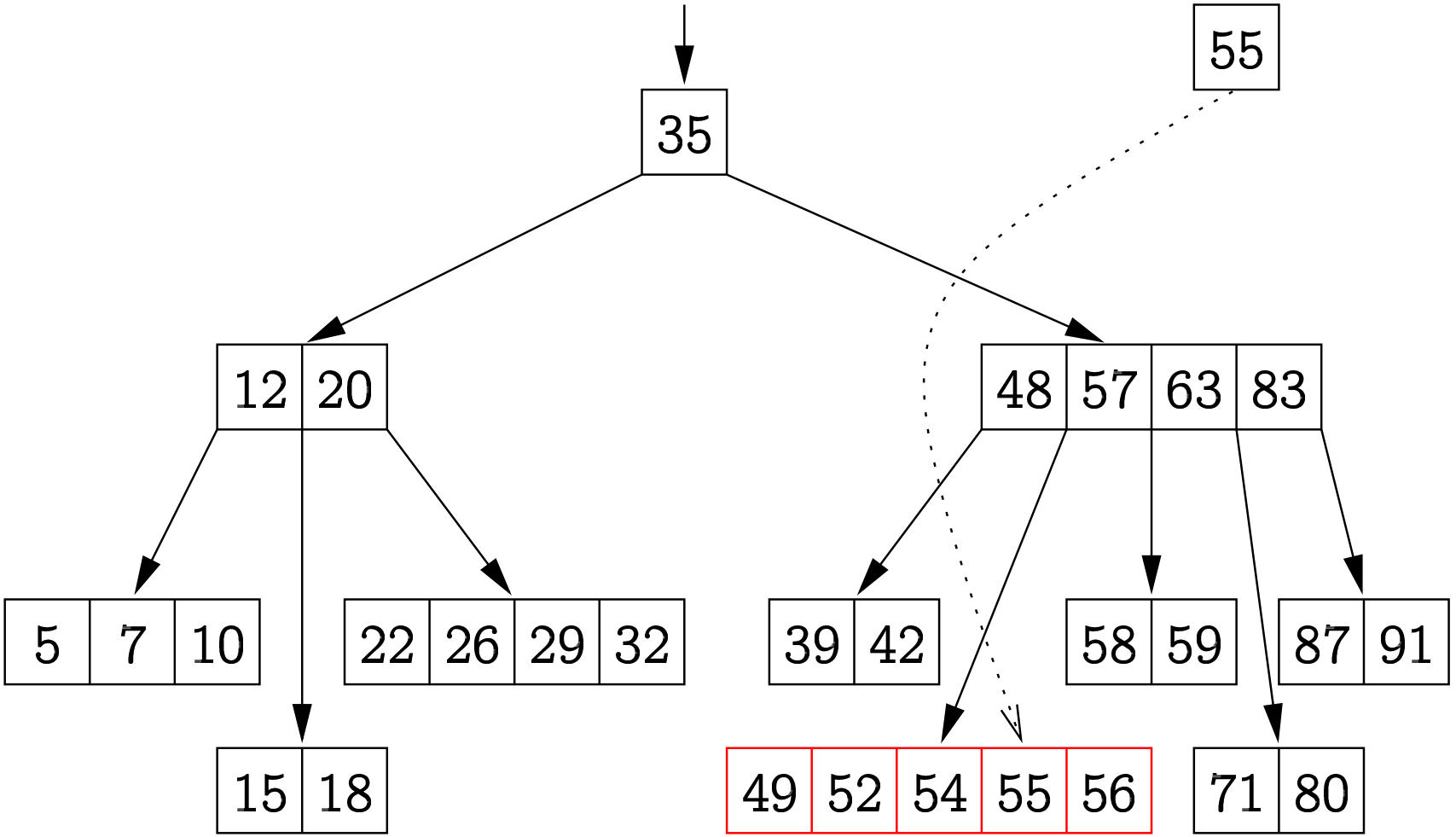
. . .


35
12
20
48
57
63
83
5
7
10
22
26
29
32
39
42
58
59
87
91
15
18
49
52
54
56
71
80

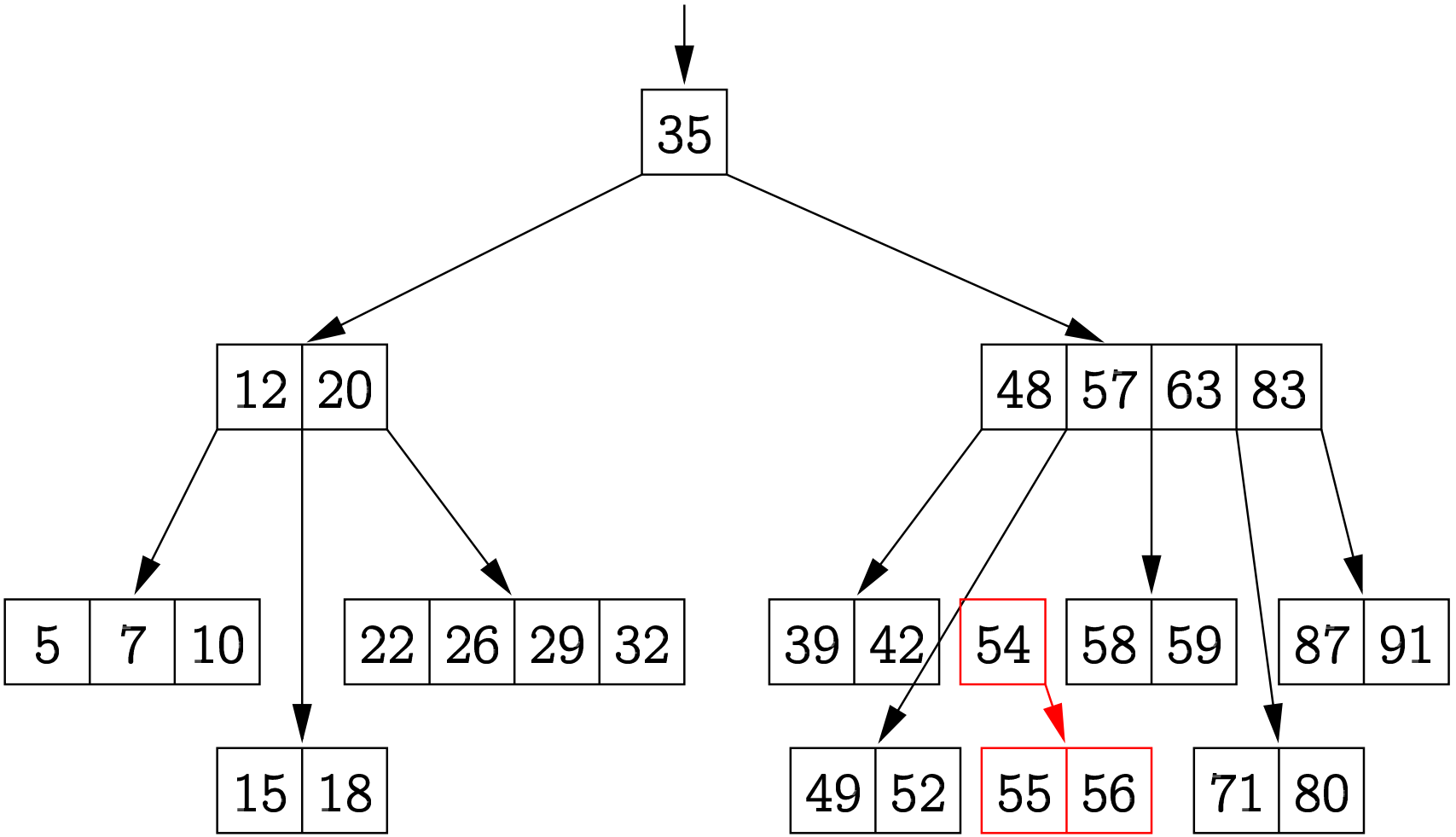
Lisame võtme 55



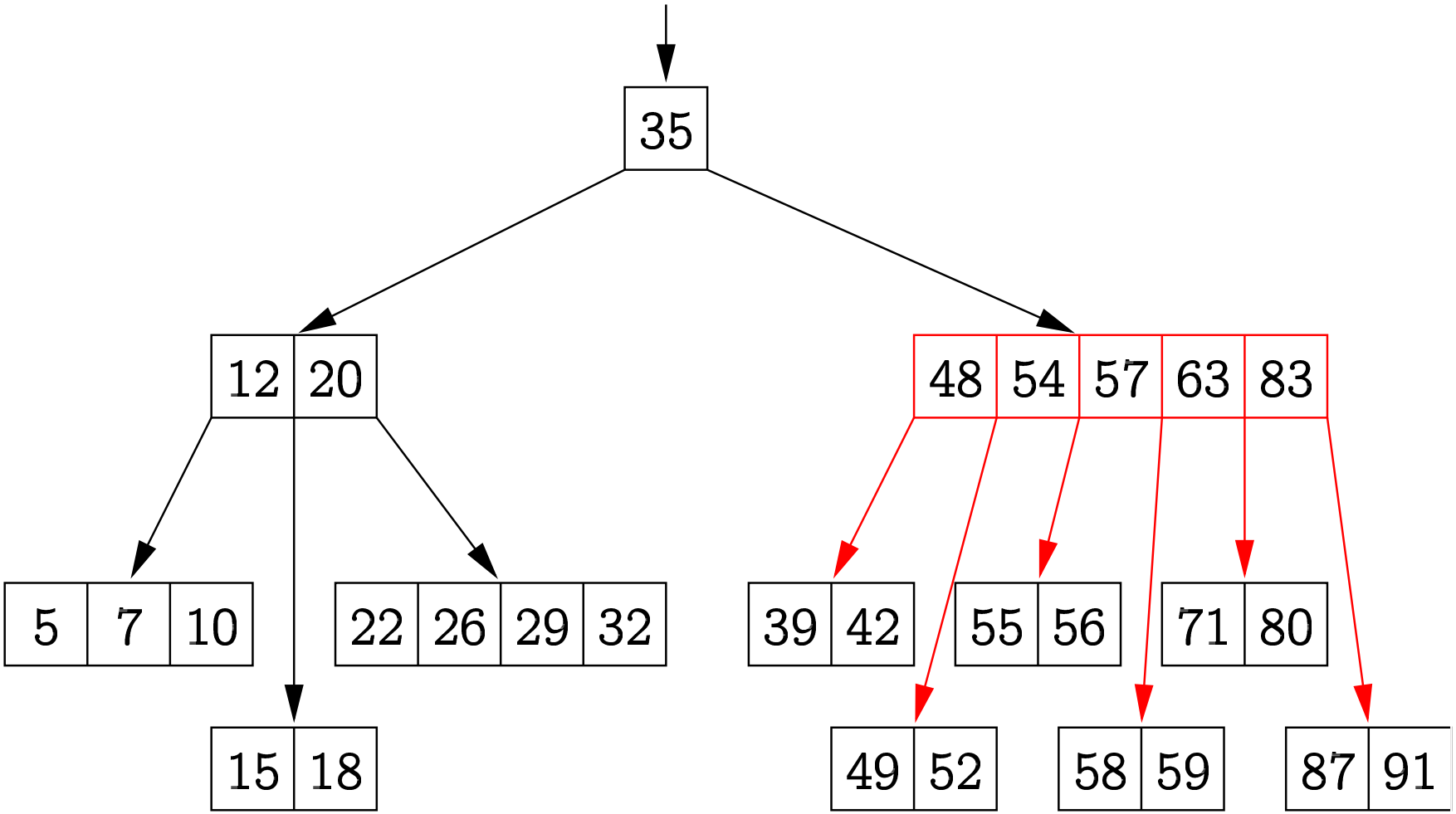
Lisame võtme 55



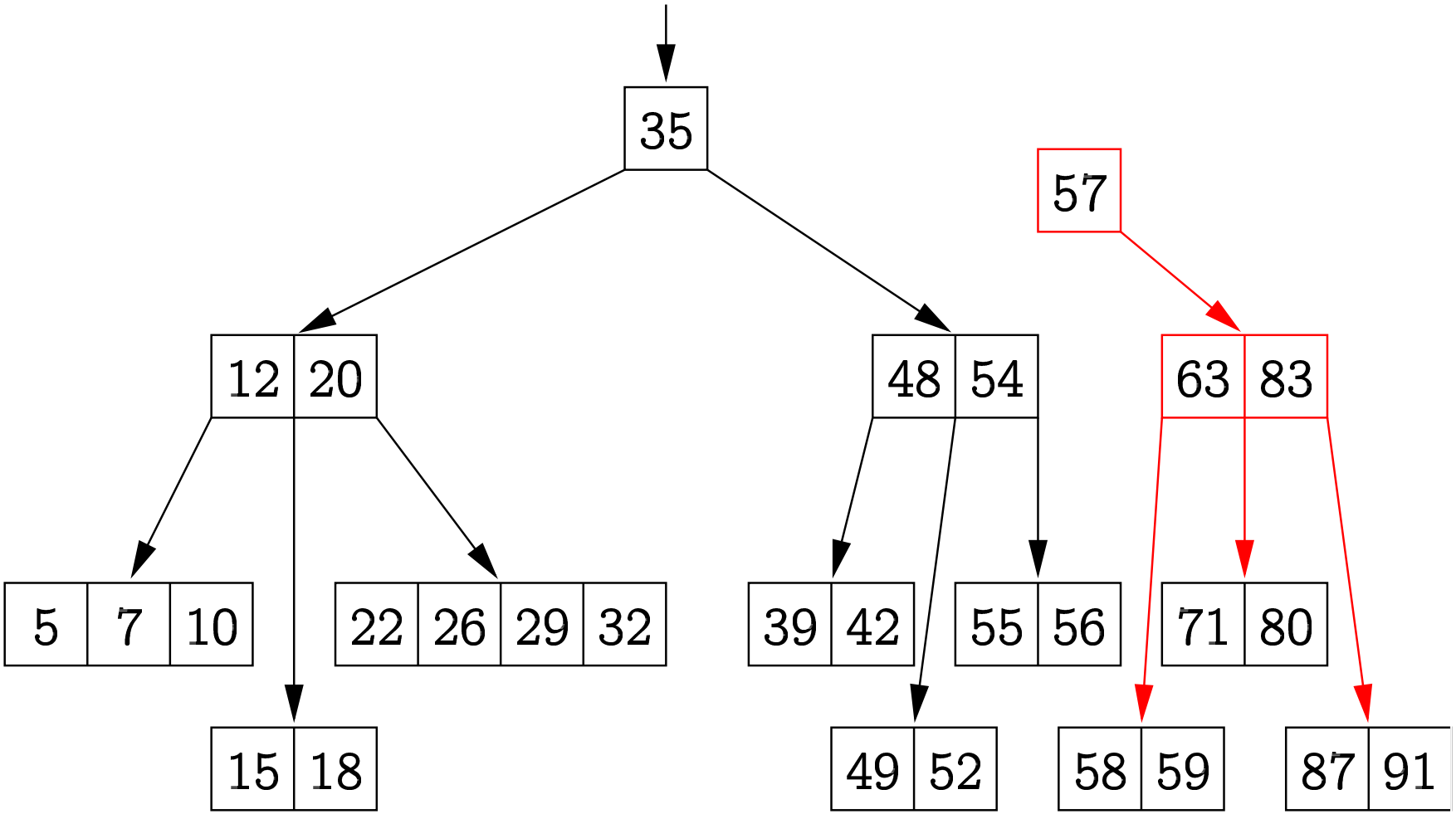
Lisame võtme 55



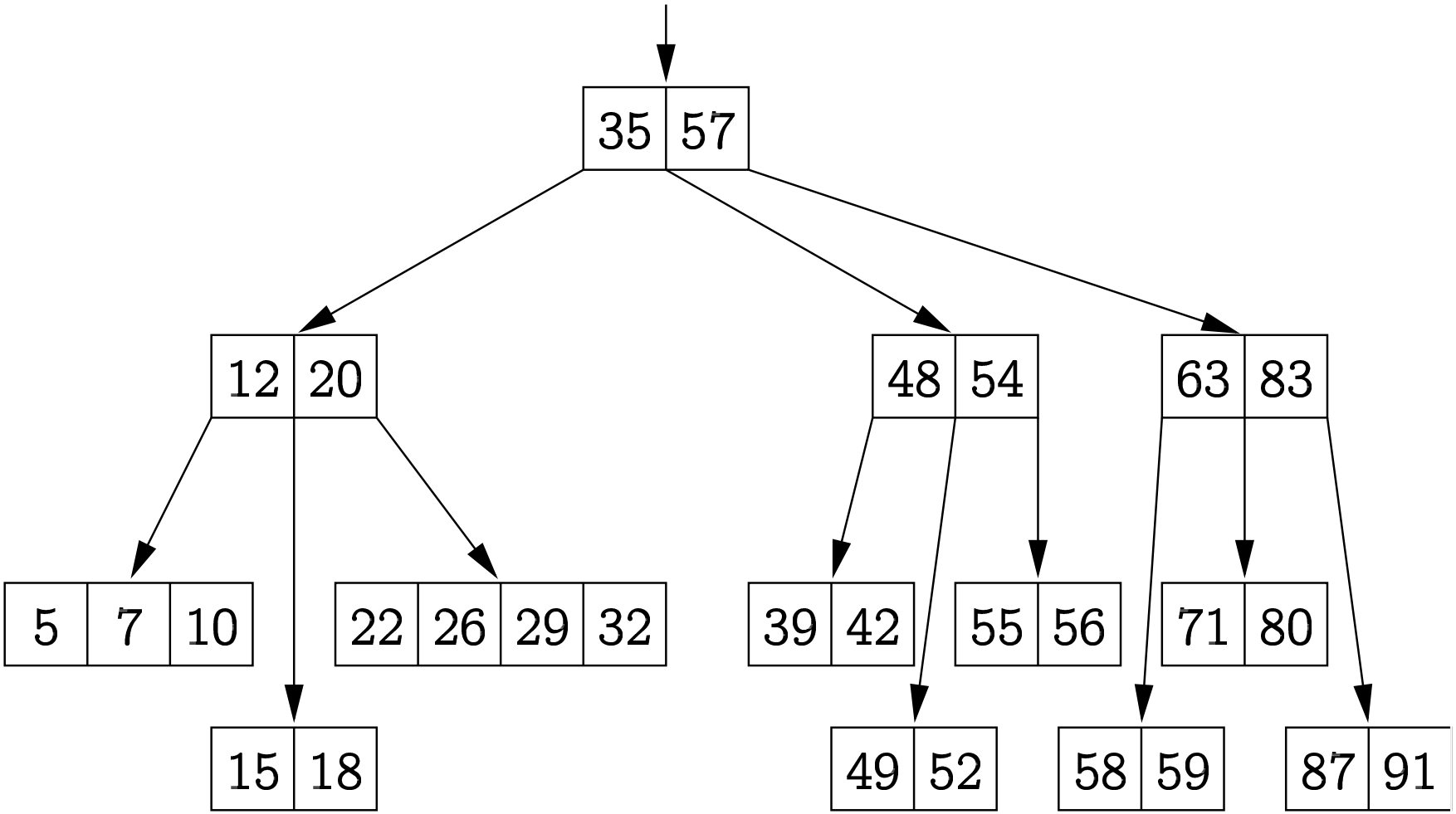
Lisame võtme 55



Lisame võtme 55



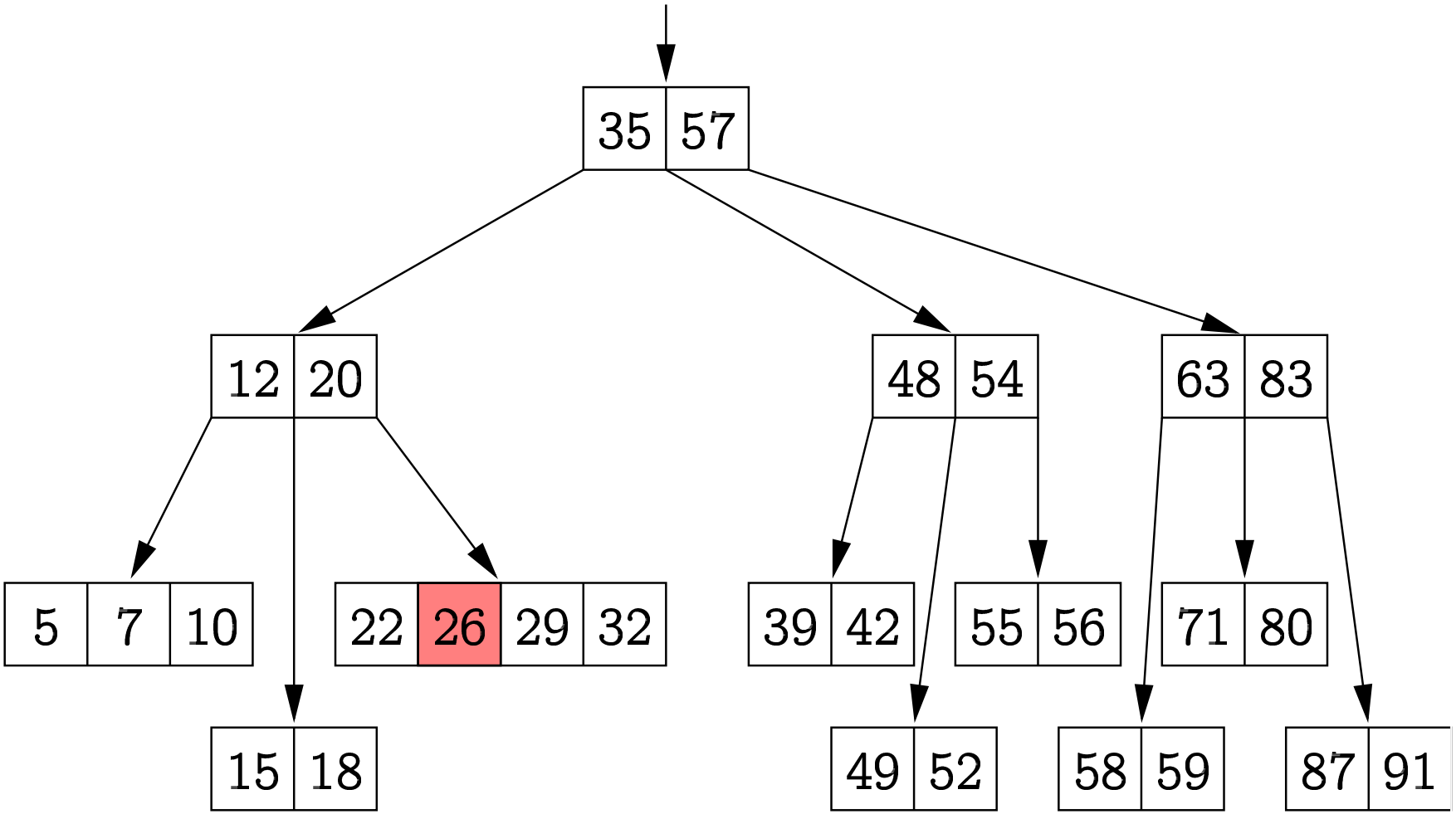
Lisame võtme 55



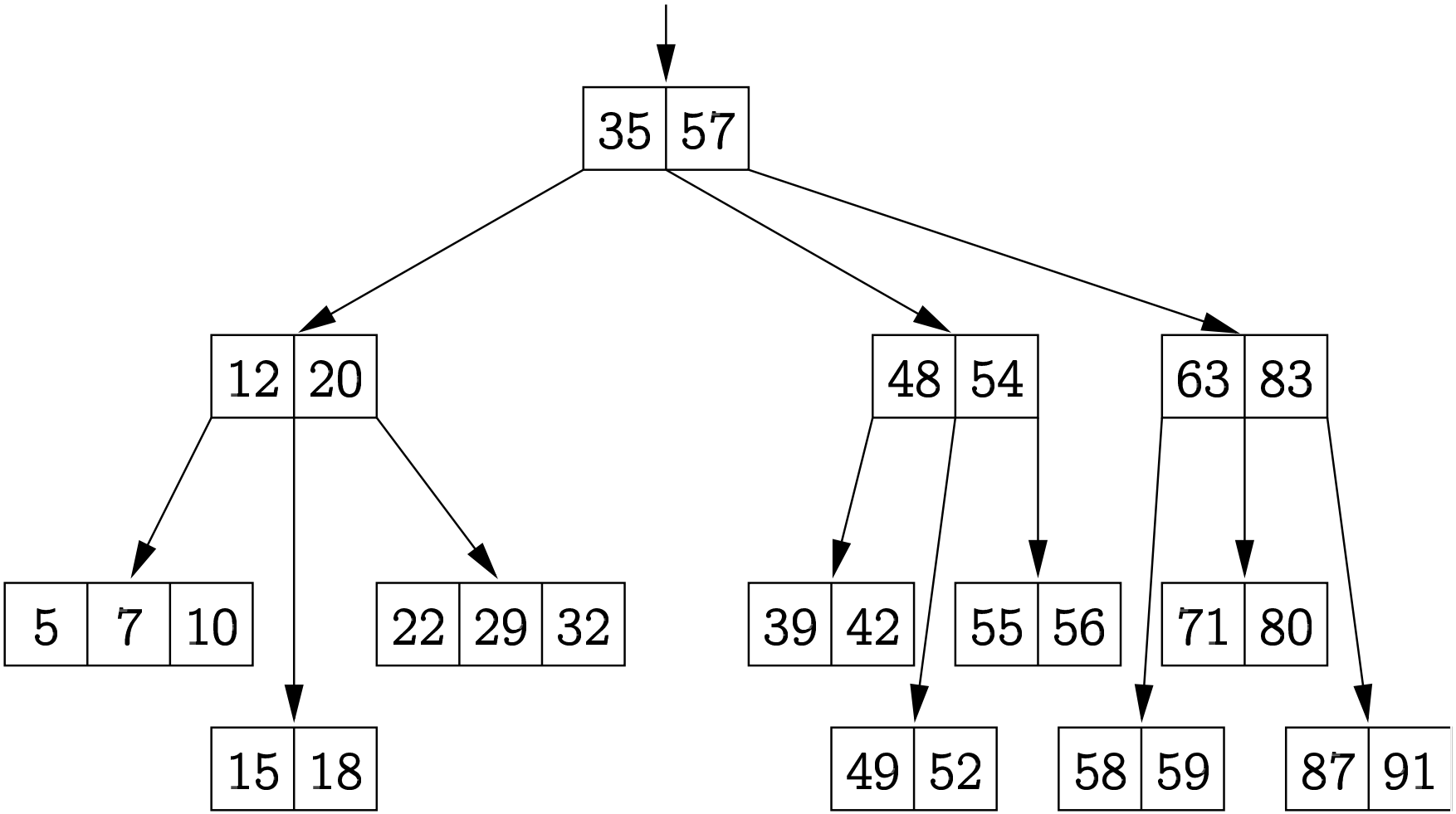
Juure ületäitumisel tekib uus juur.

Kirje eemaldamisel lehest, kus on rohkem kui $\lfloor m/2 \rfloor$ kirjet, teeme selle lehe lihtsalt väiksemaks.

Eemaldame võtme 26

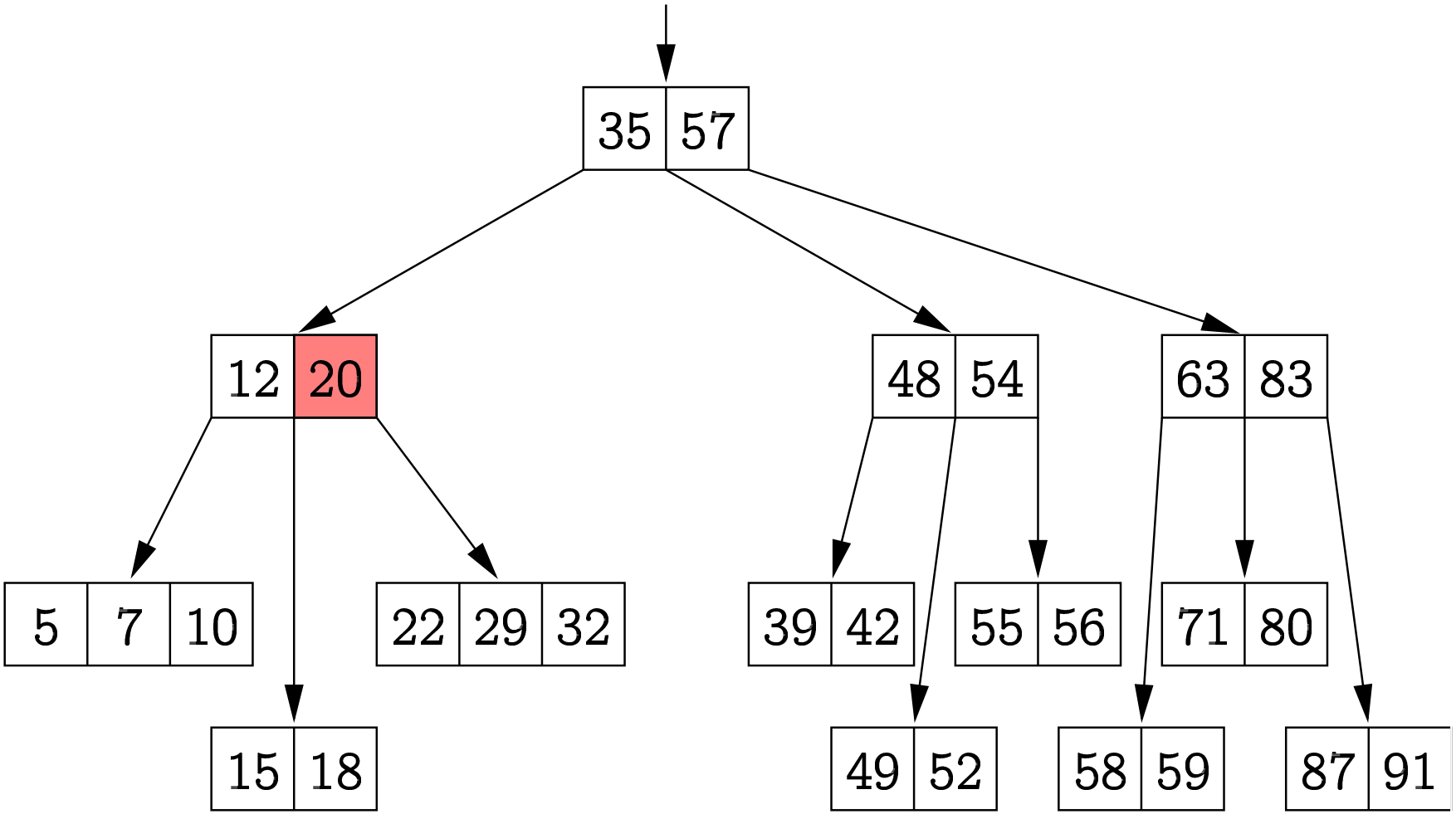


Eemaldame võtme 26

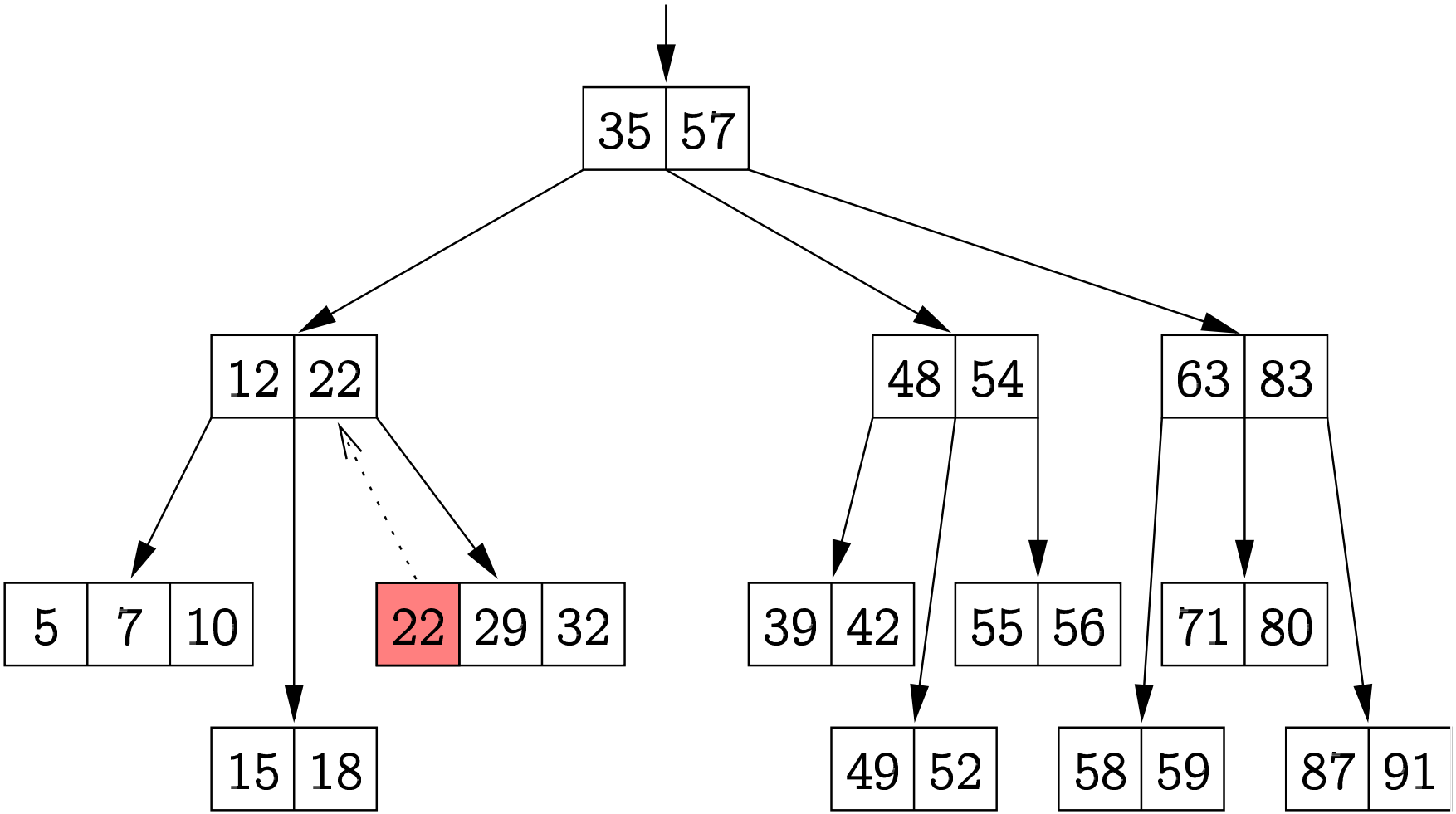


Kirje eemaldamisel sisemisest tipust kirjutame ta üle talle vahetult järgneva kirjega (see asub lehes) ja kustutame hoopis selle kirje.

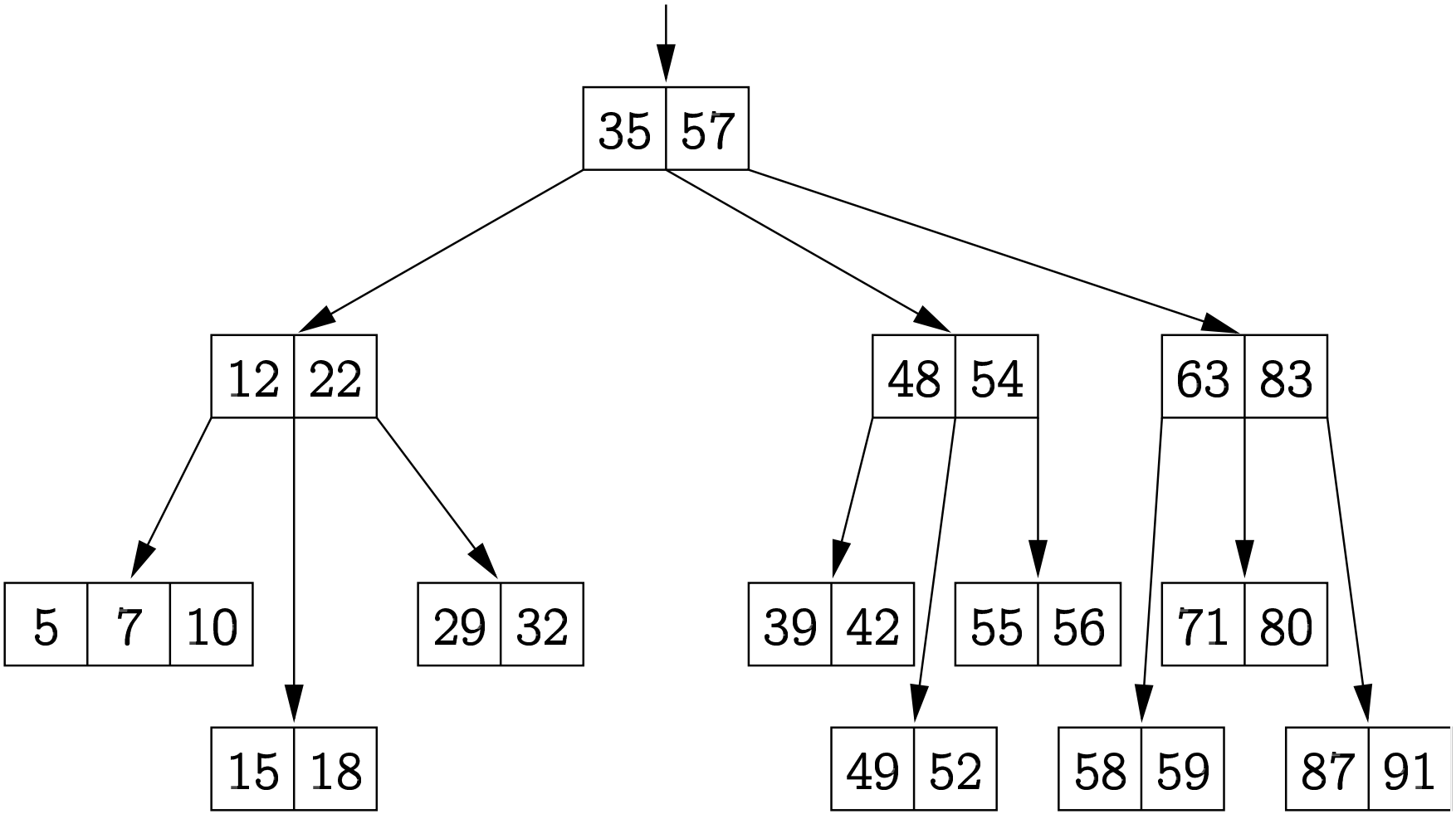
Eemaldame võtme 20



Eemaldame võtme 20

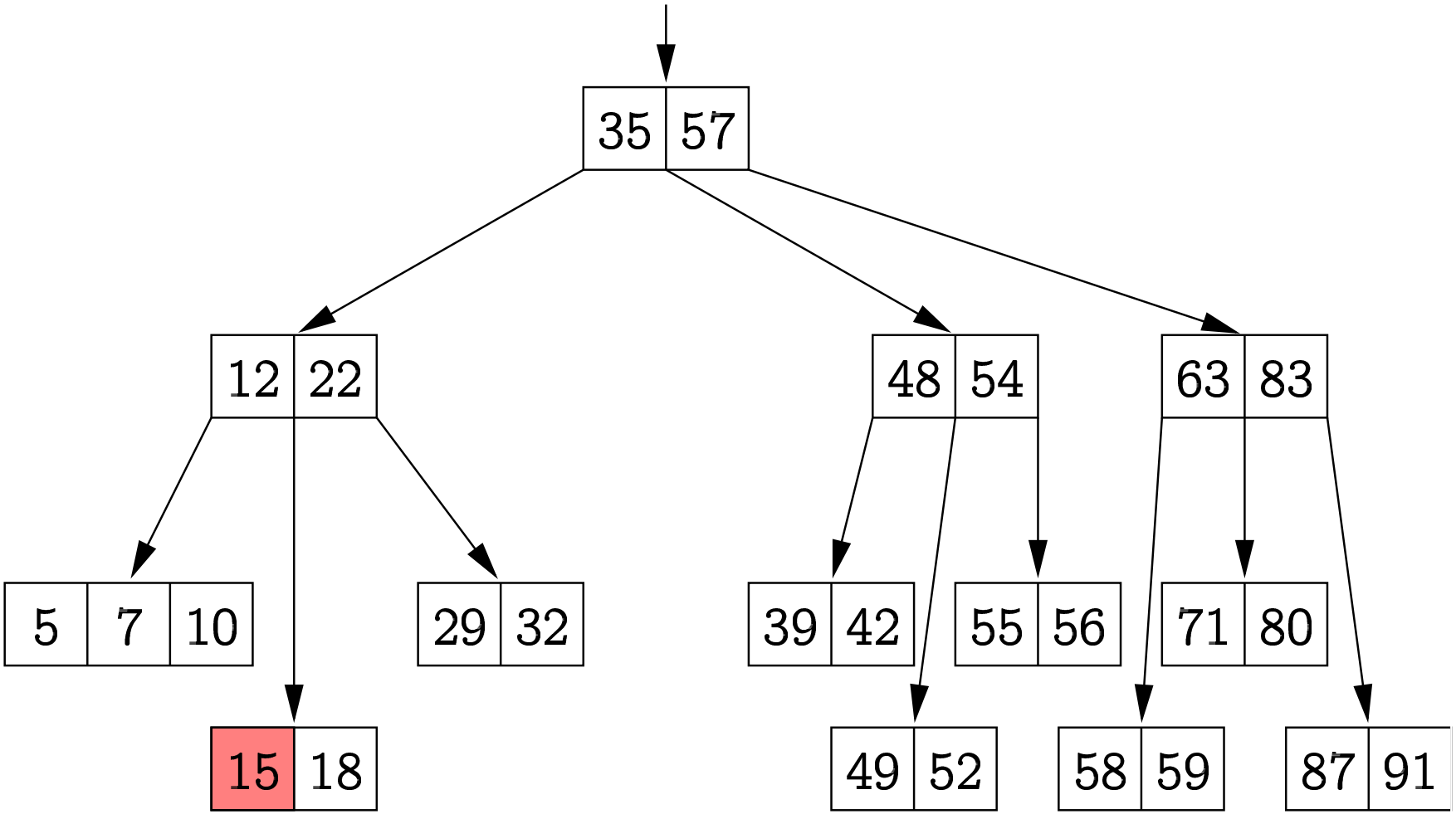


Eemaldame võtme 20

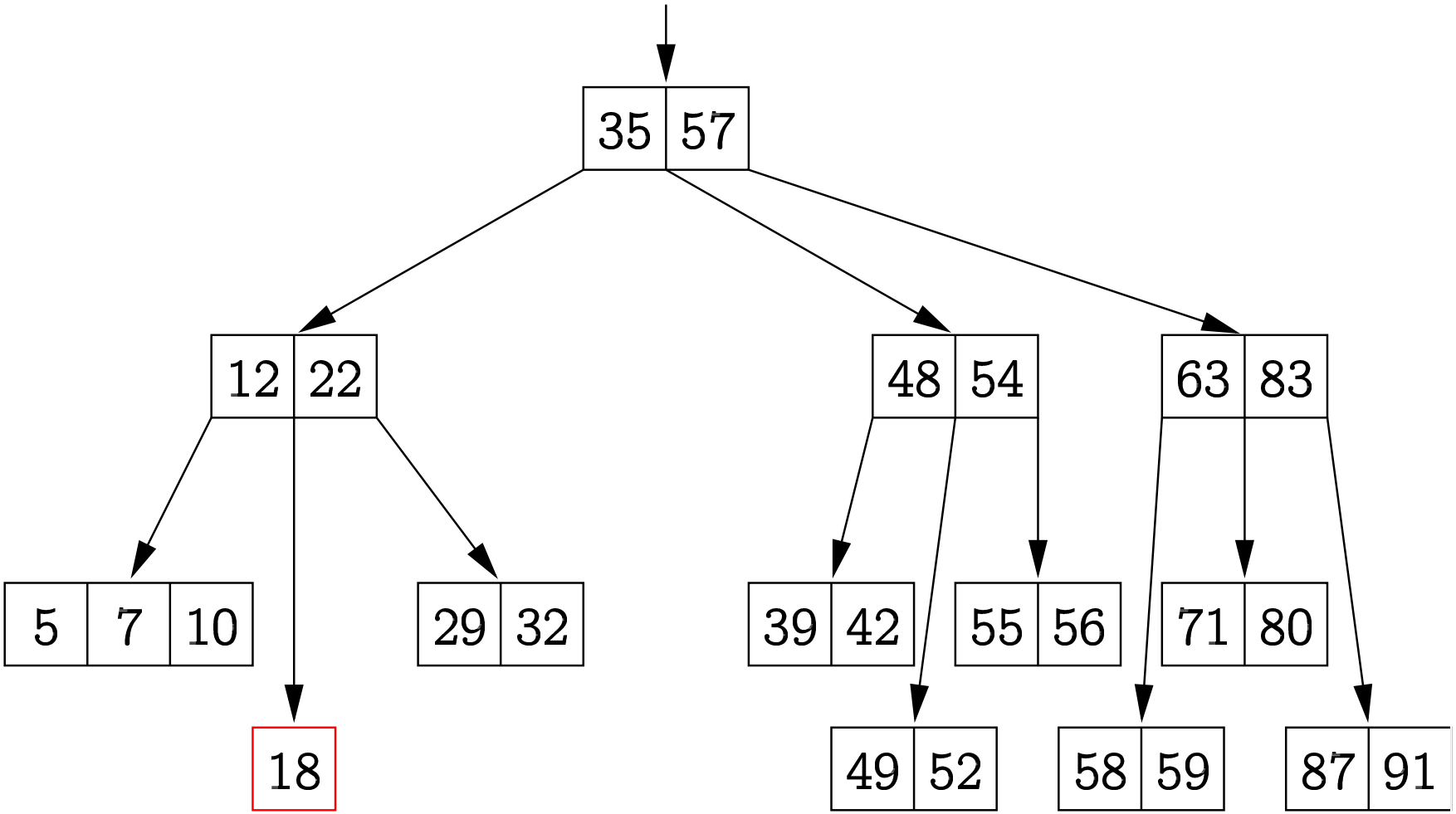


Kui kirje eemaldamisel jääb mõnda tippu vähem kui $\lfloor m/2 \rfloor$ kirjet ja selle tipu vasakus või paremas naabris on rohkem kui $\lfloor m/2 \rfloor$ kirjet, siis võtame sellest naabertipust ühe kirje (ülemuse kaudu).

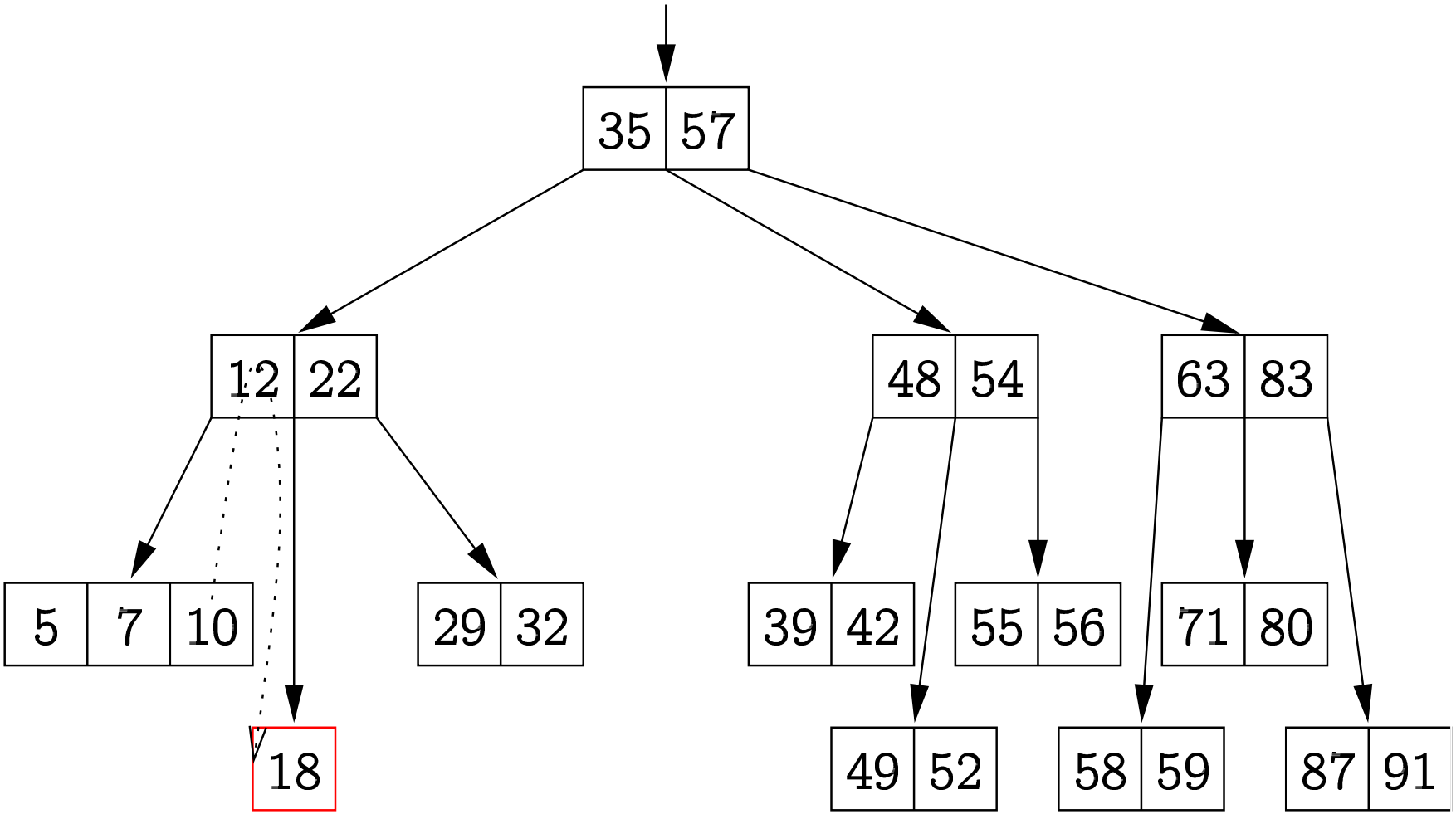
Eemaldame võtme 15



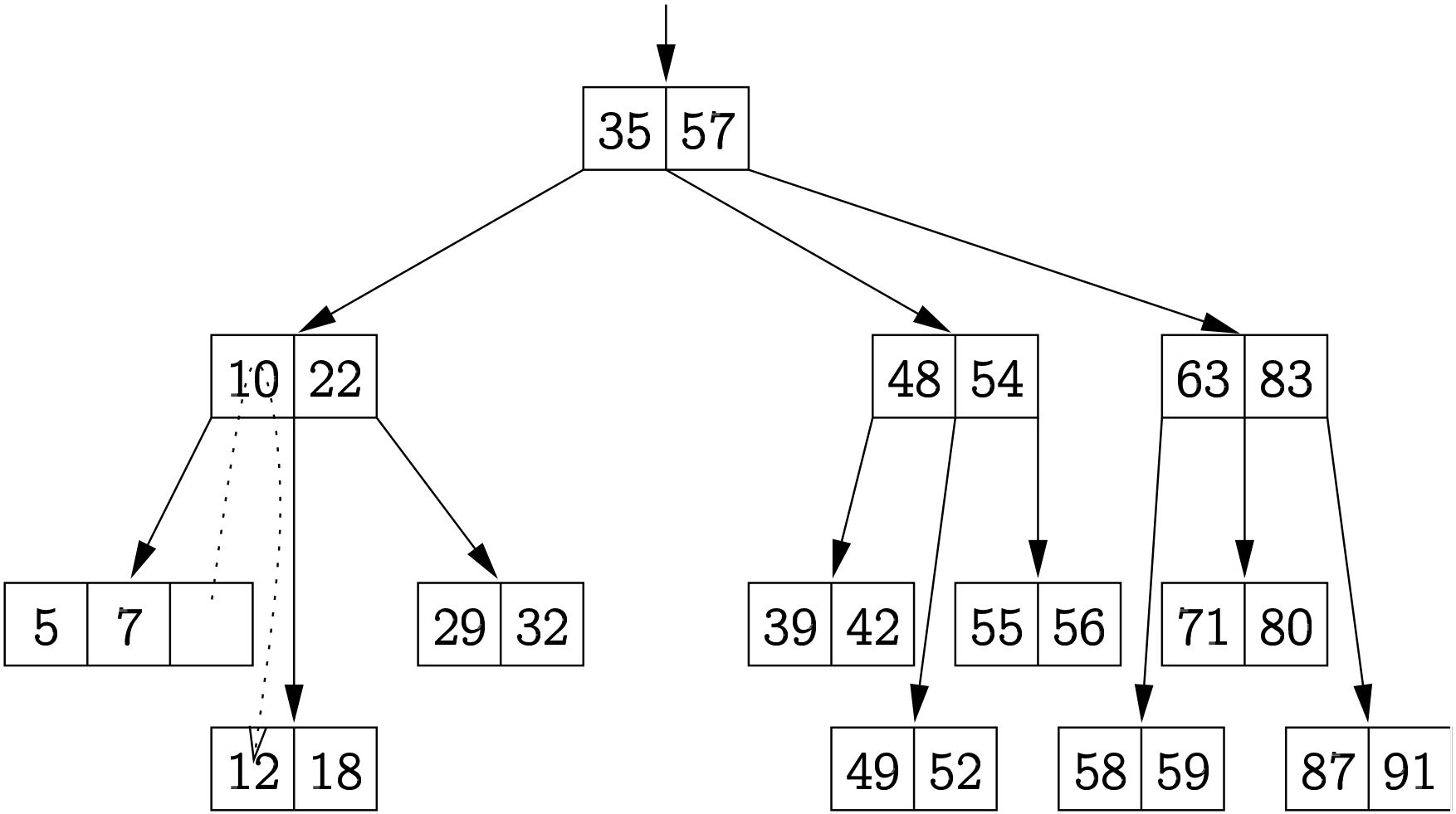
Eemaldame võtme 15



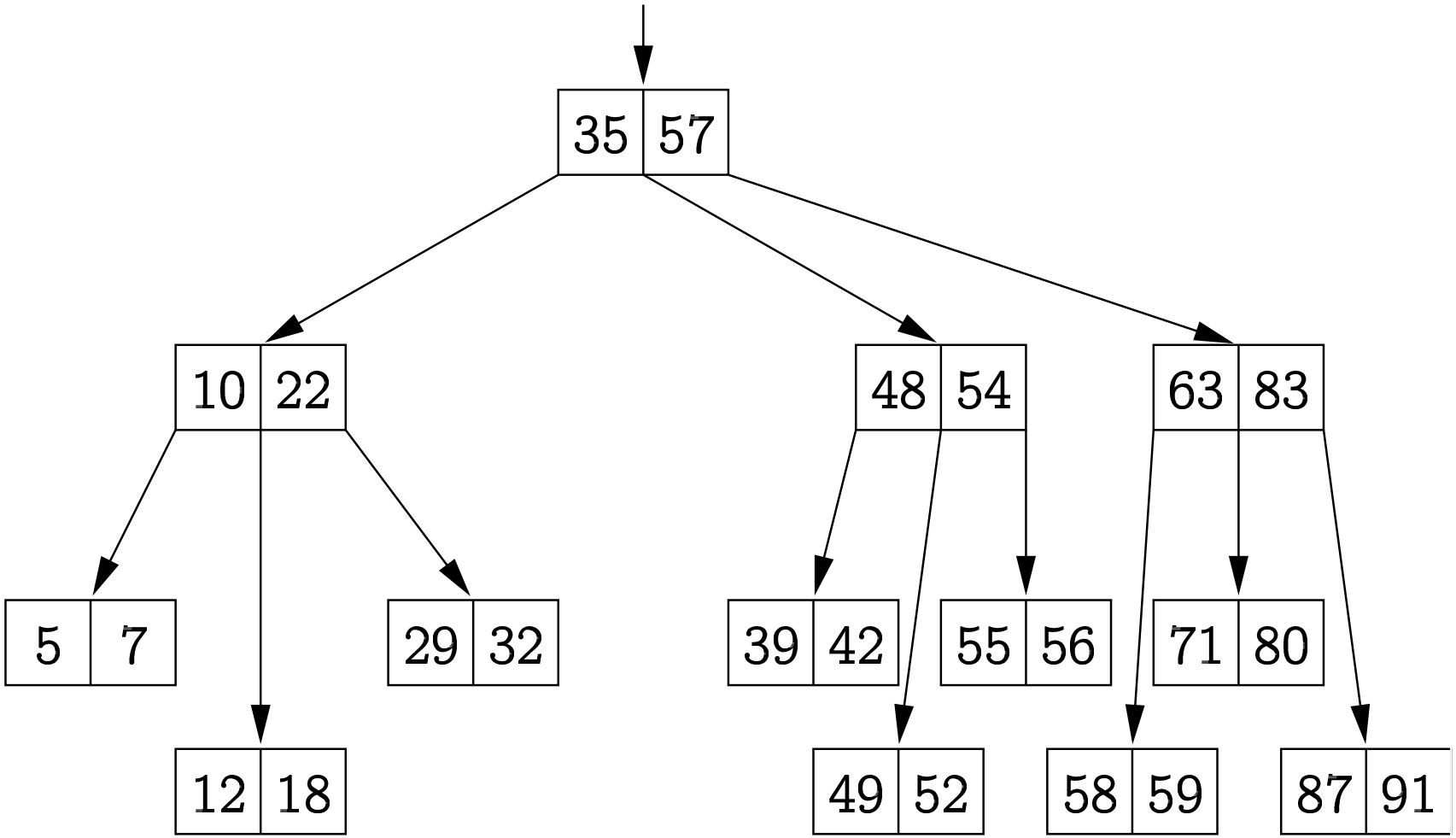
Eemaldame võtme 15



Eemaldame võtme 15



Eemaldame võtme 15



Kui kirje eemaldamisel jääb mõnda tippu vähem kui $\lfloor m/2 \rfloor$ kirjet ja selle tipu vasakus ja paremas naabris on täpselt kui $\lfloor m/2 \rfloor$ kirjet, siis ühendame tipu, tema naabertipu ja nendevahelise kirje ülemusest.

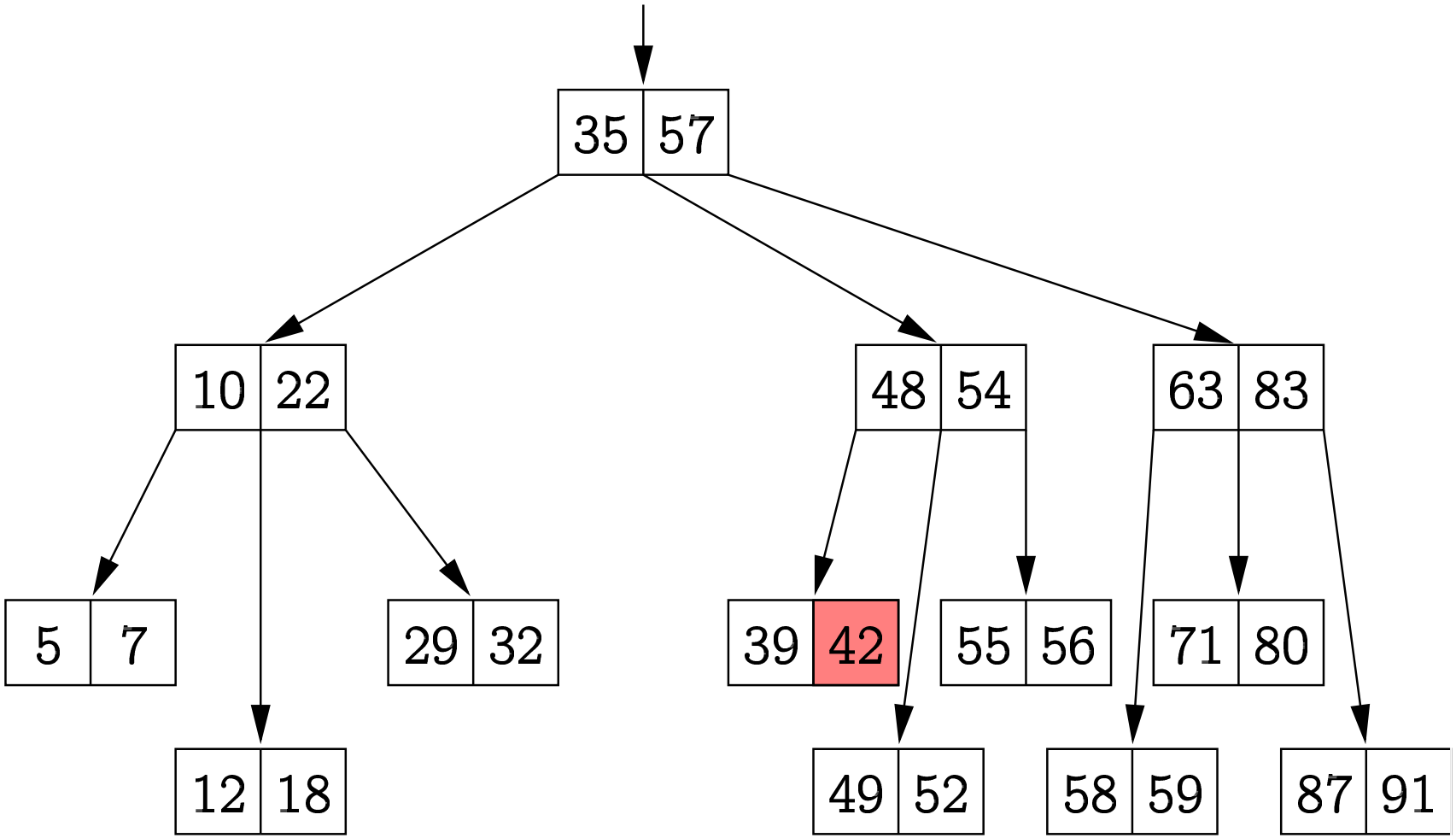
Saame tipu, kus on

$$\lfloor m/2 \rfloor - 1 + \lfloor m/2 \rfloor + 1 = 2\lfloor m/2 \rfloor = m - 1$$

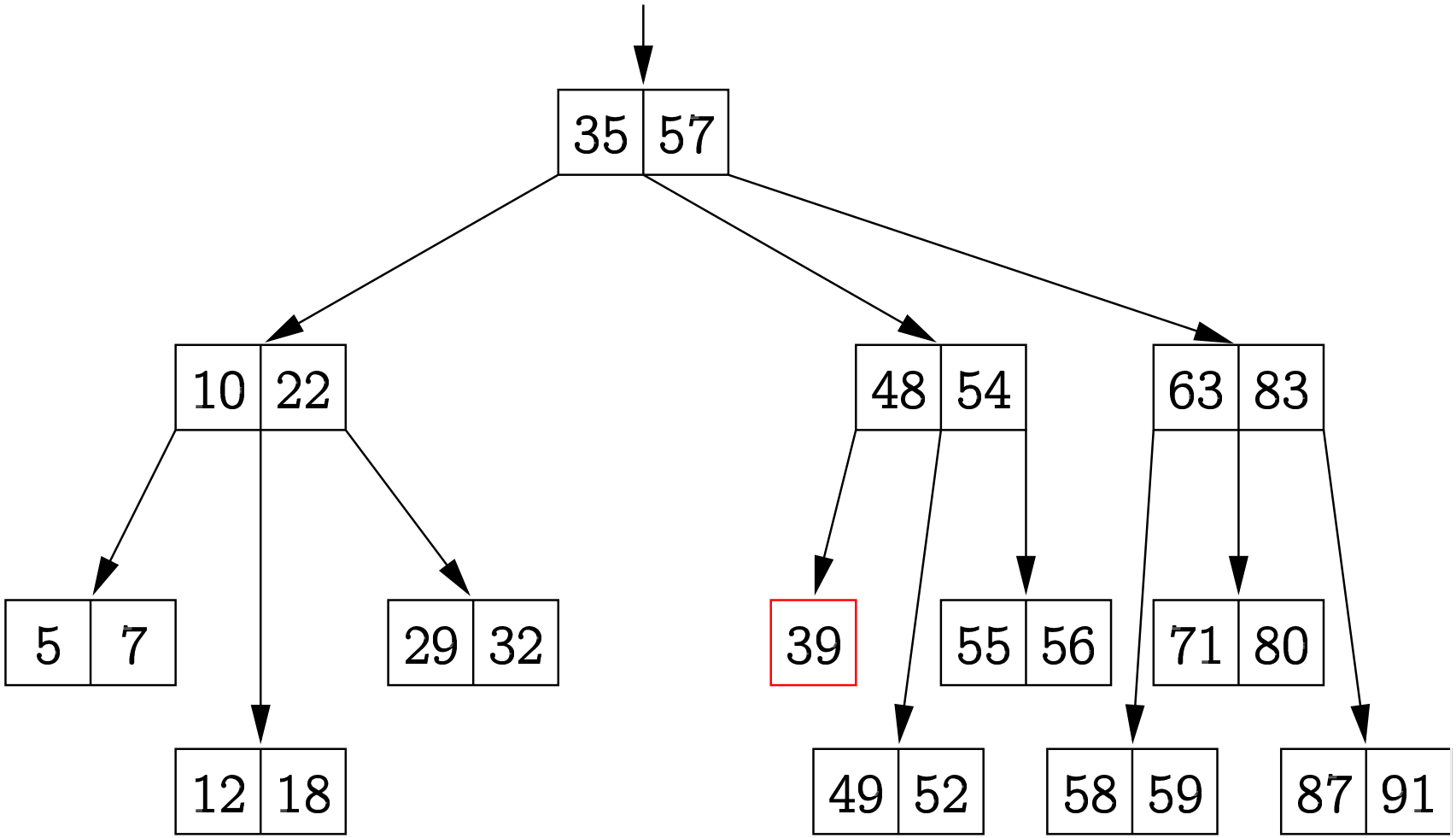
kirjet.

Seejuures väheneb kirjete arv ülemuses ühe võrra.

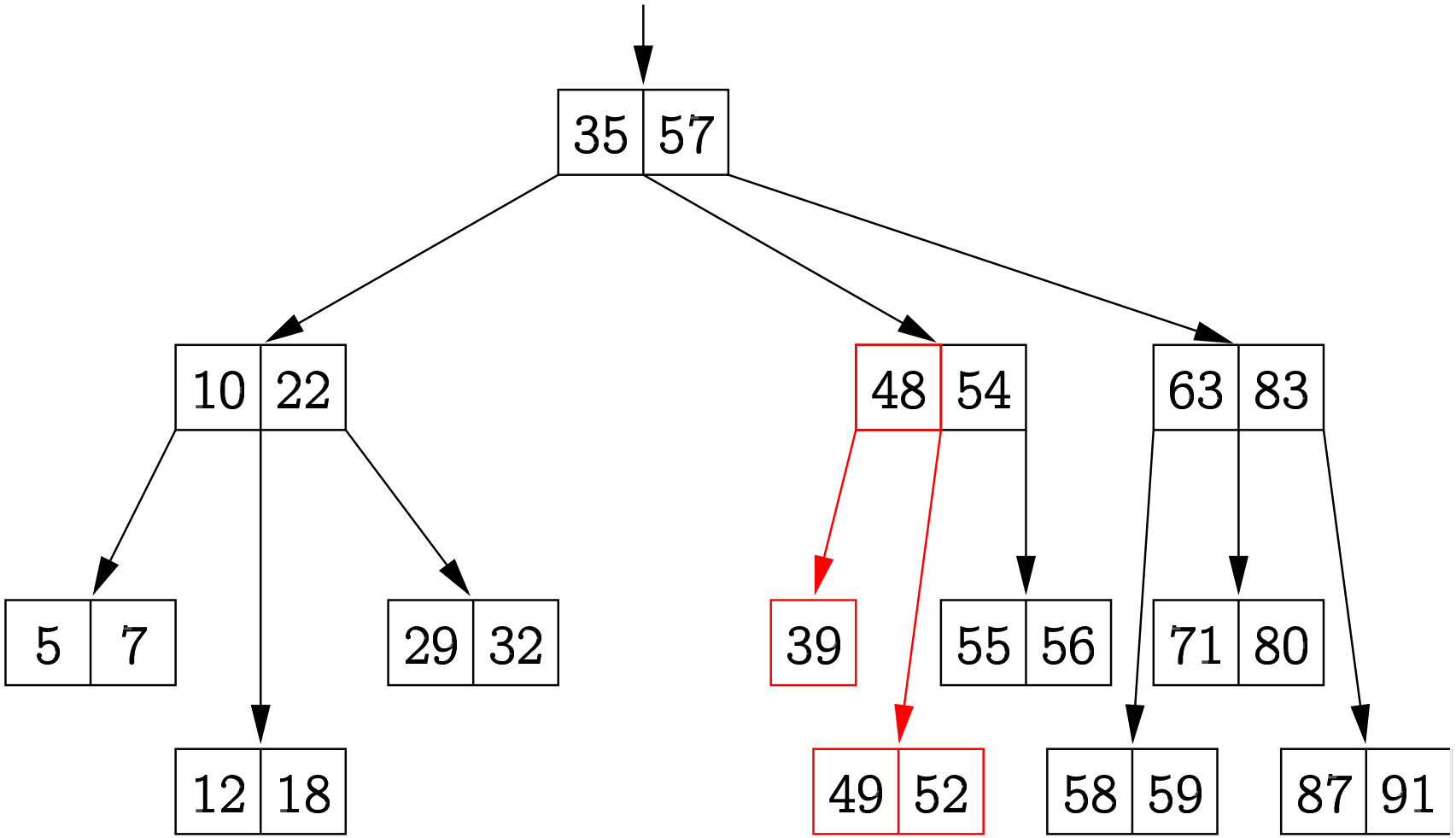
Eemaldame võtme 42



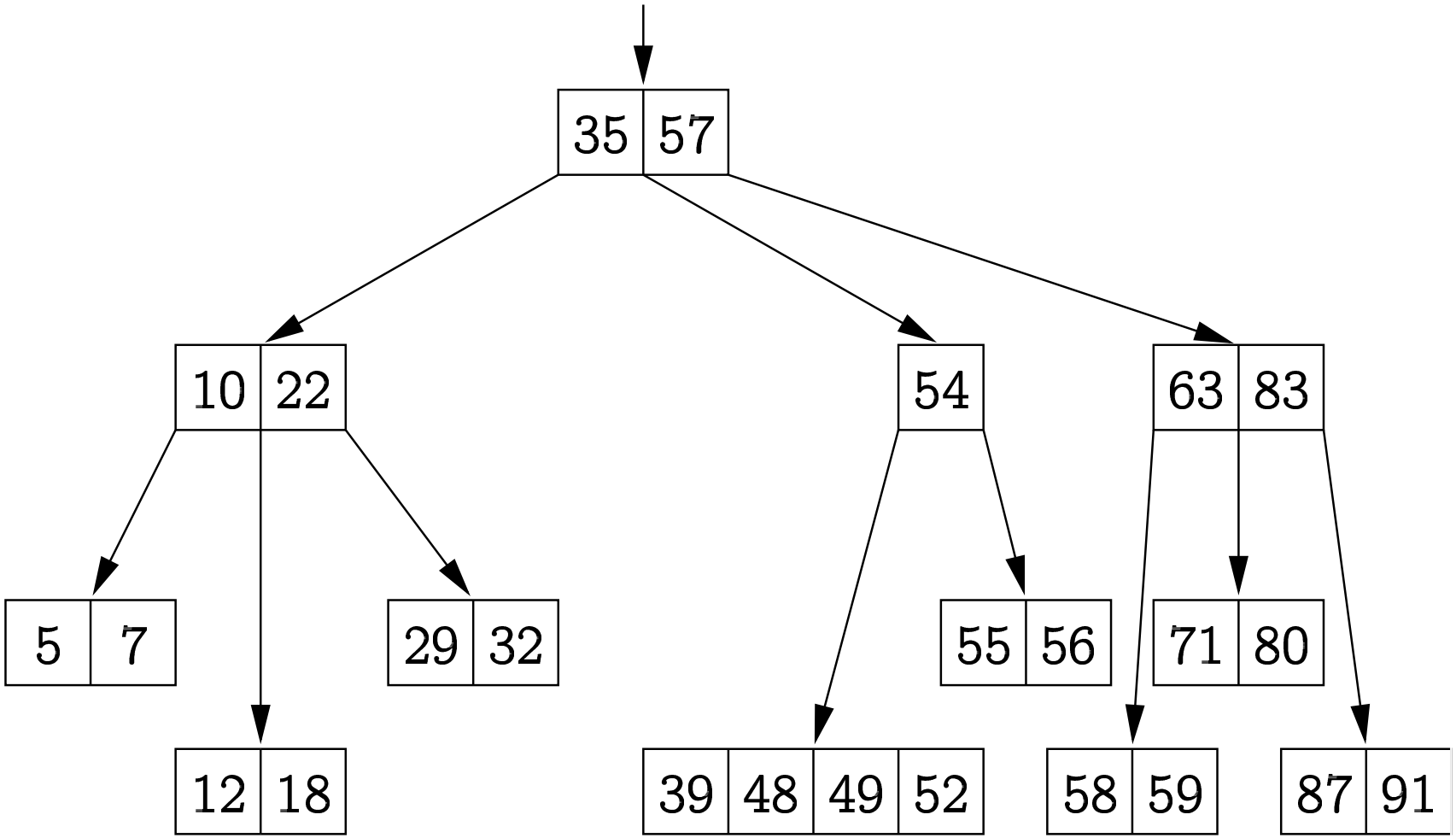
Eemaldame võtme 42



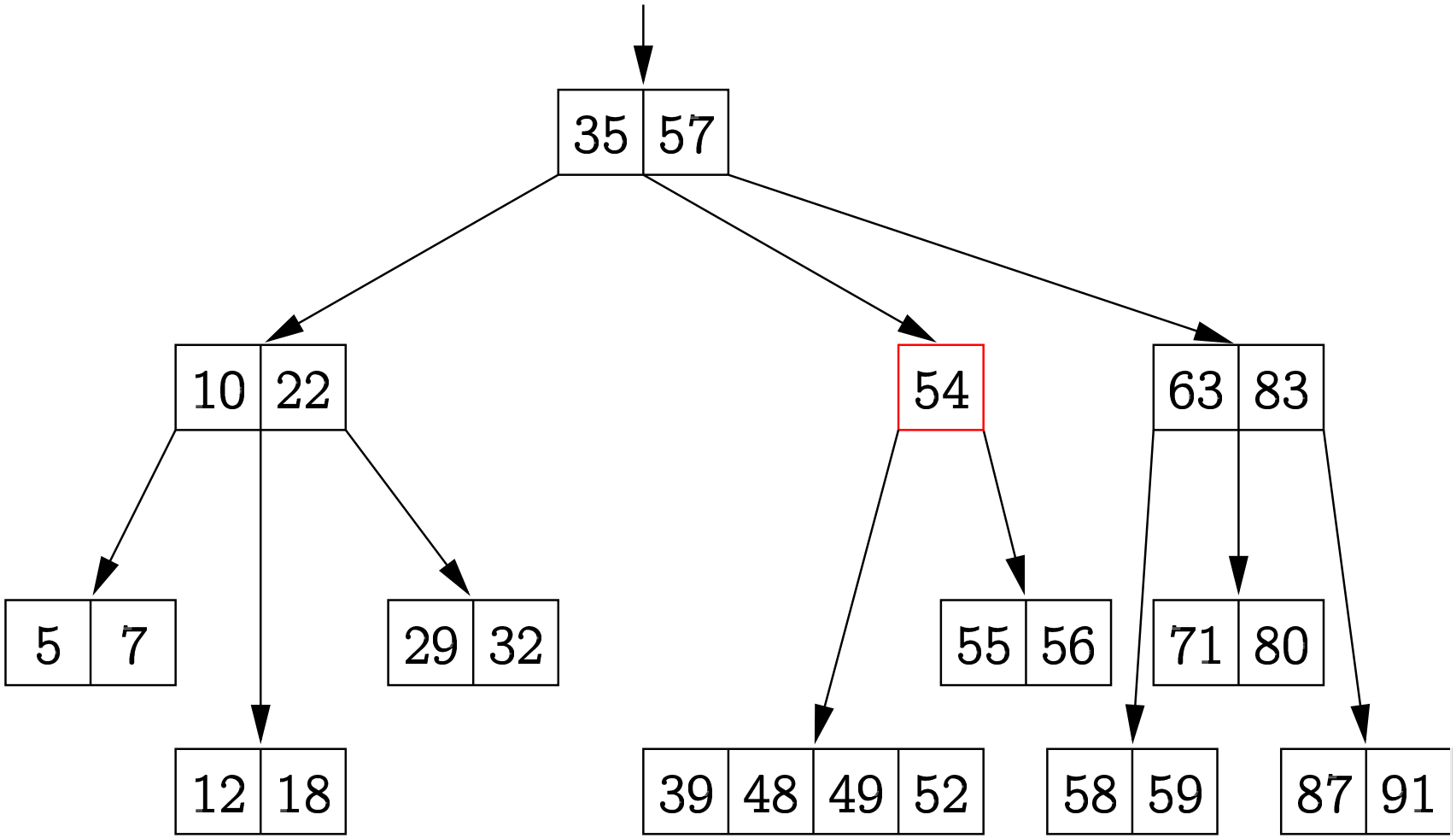
Eemaldame võtme 42



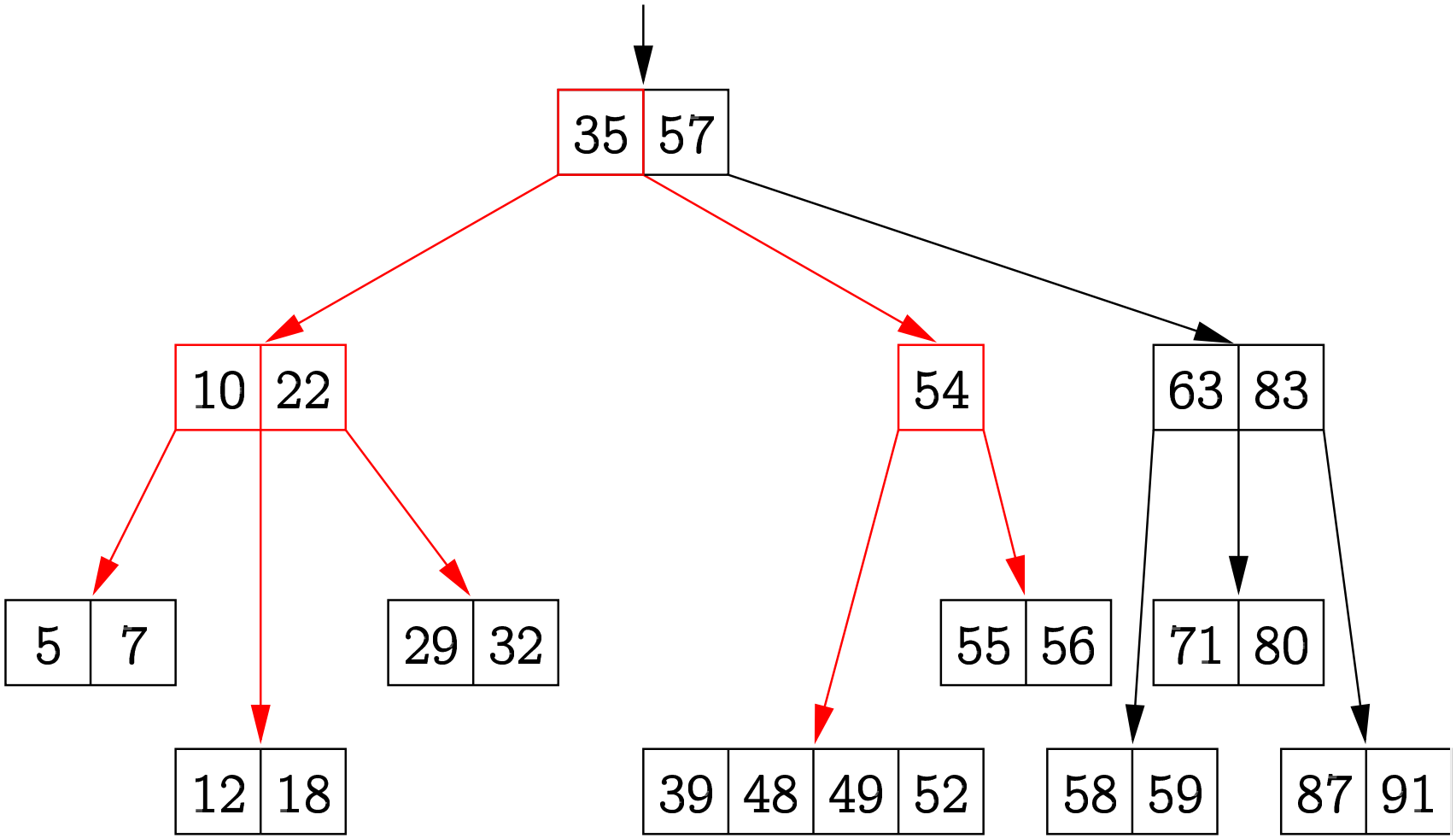
Eemaldame võtme 42



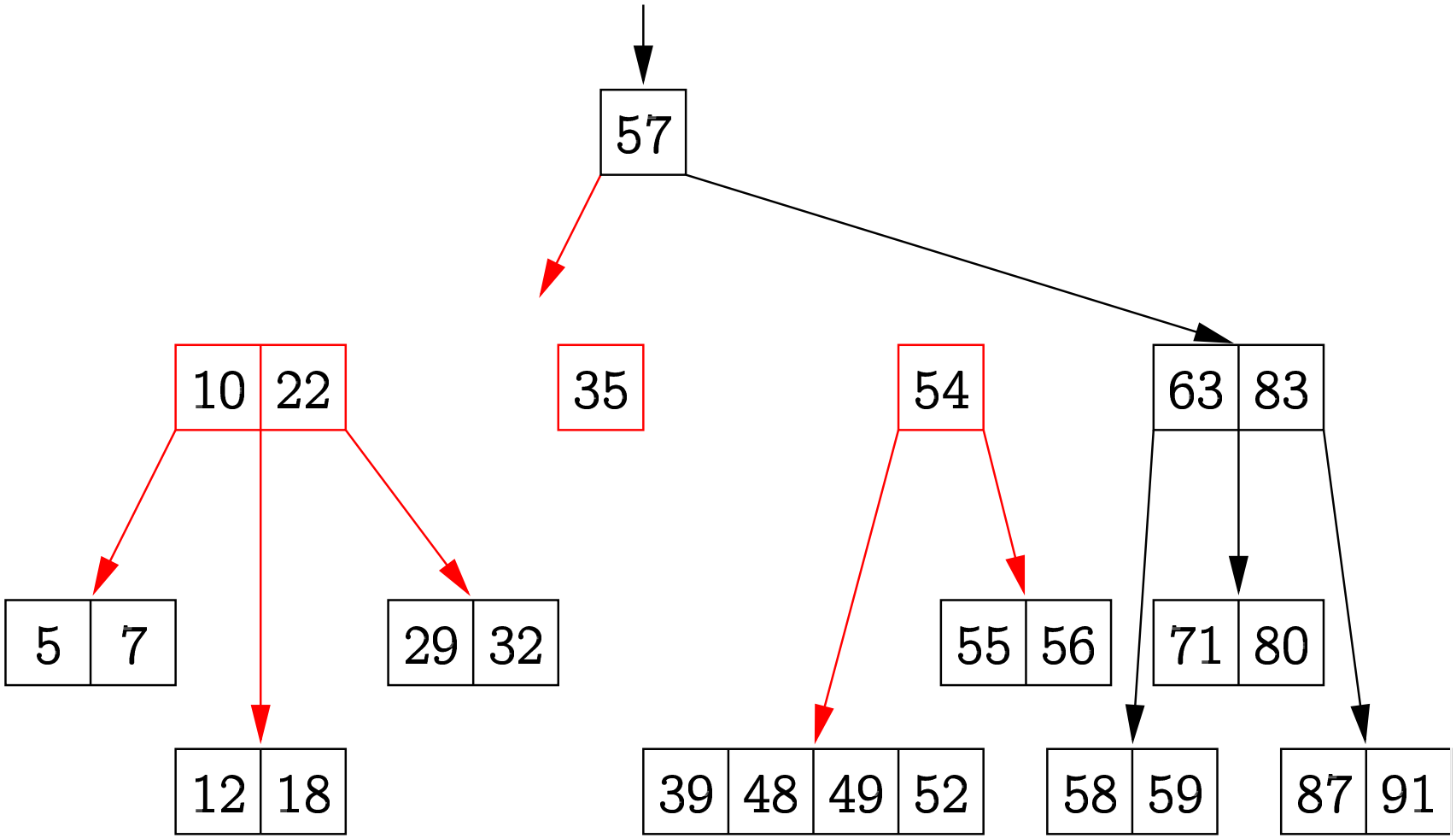
Eemaldame võtme 42



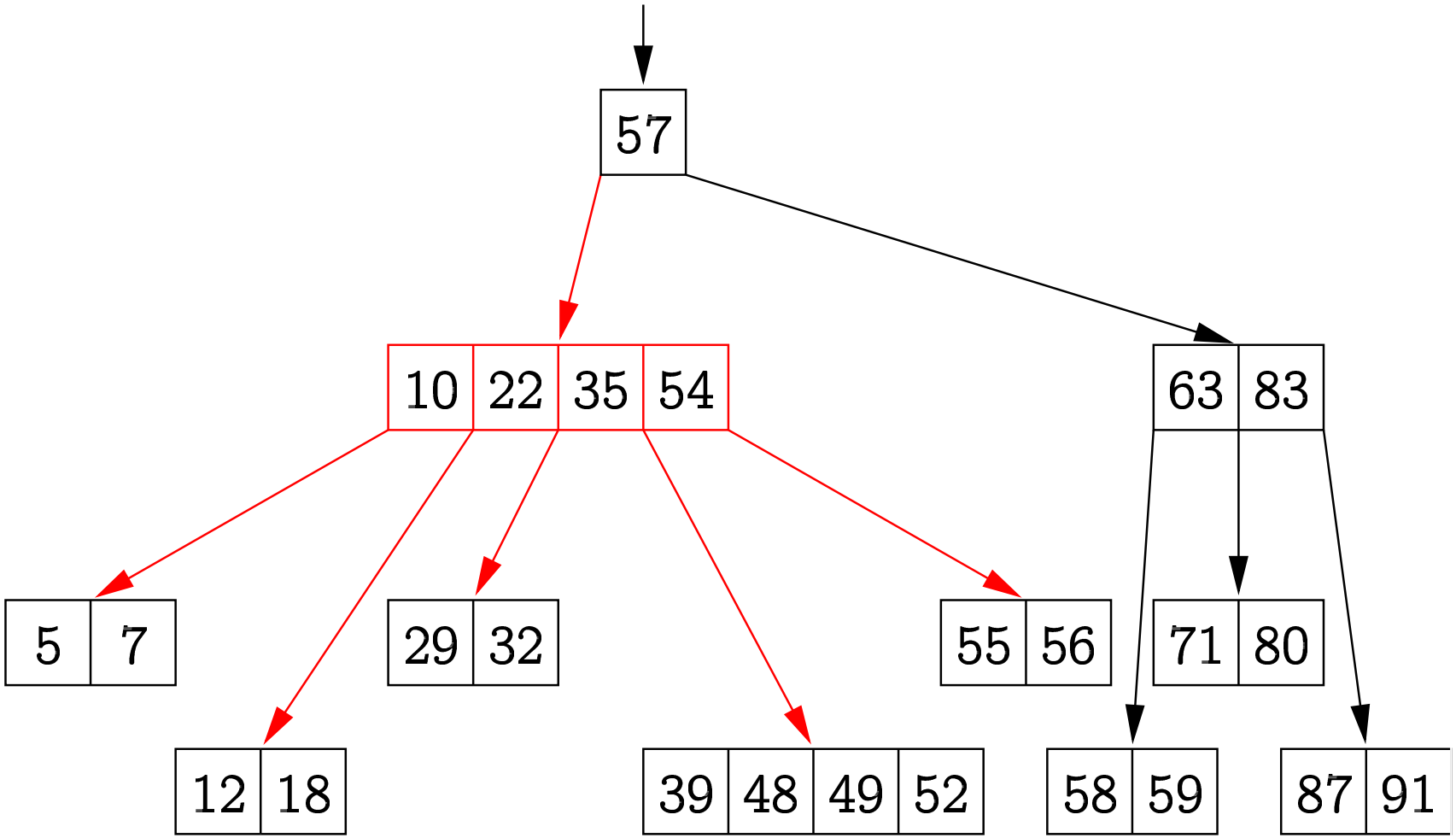
Eemaldame võtme 42



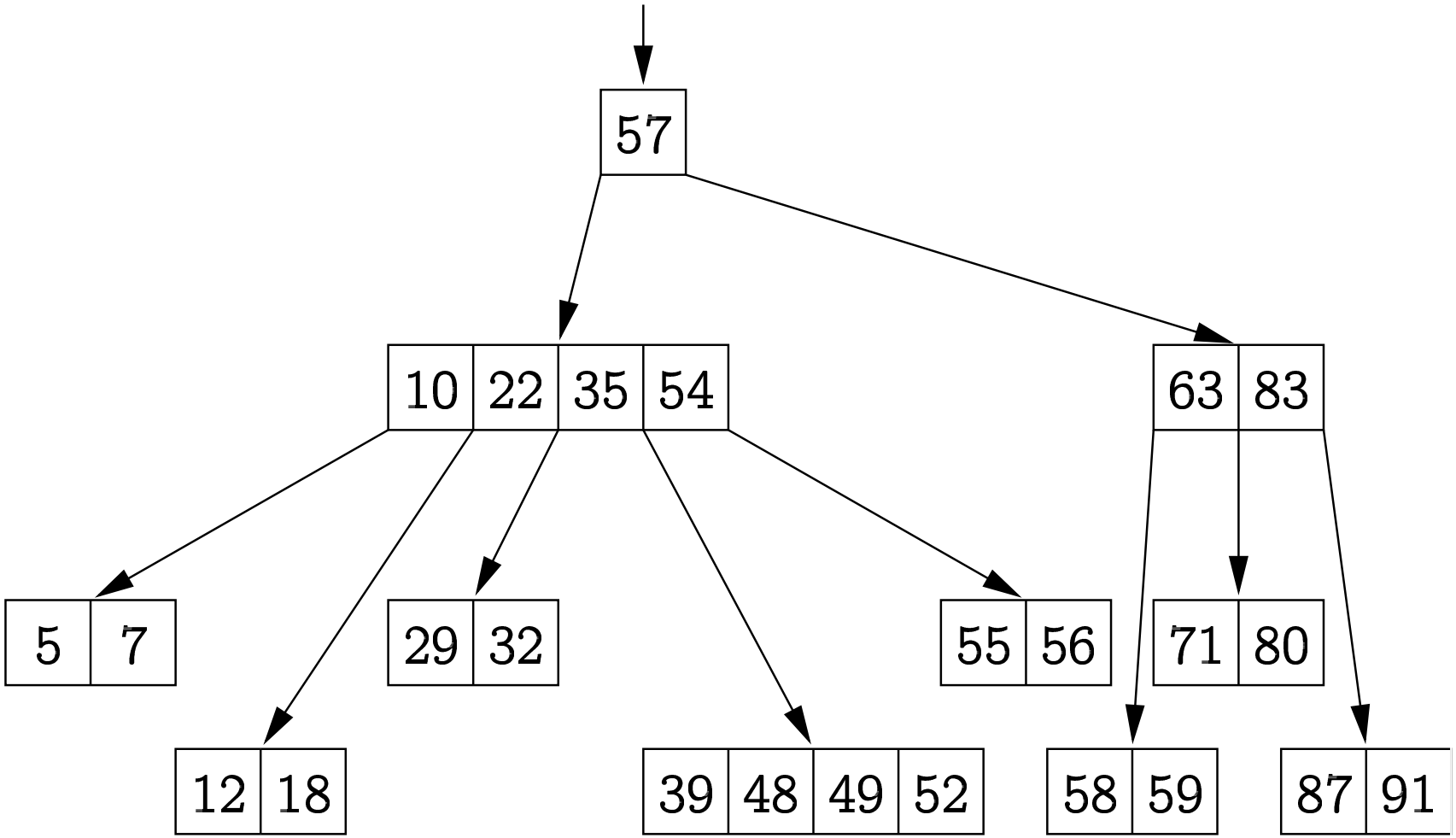
Eemaldame võtme 42



Eemaldame võtme 42

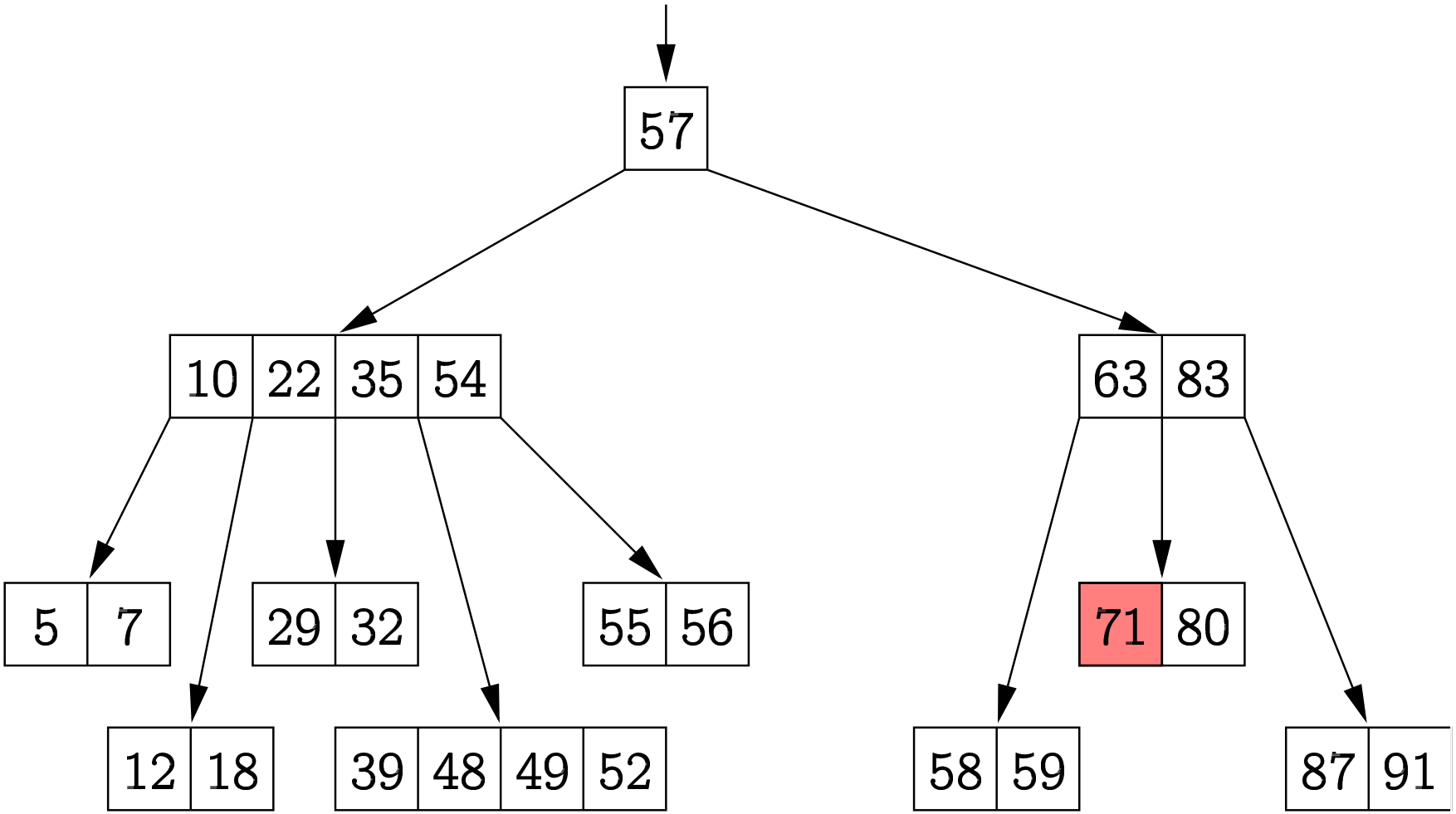


Eemaldame võtme 42

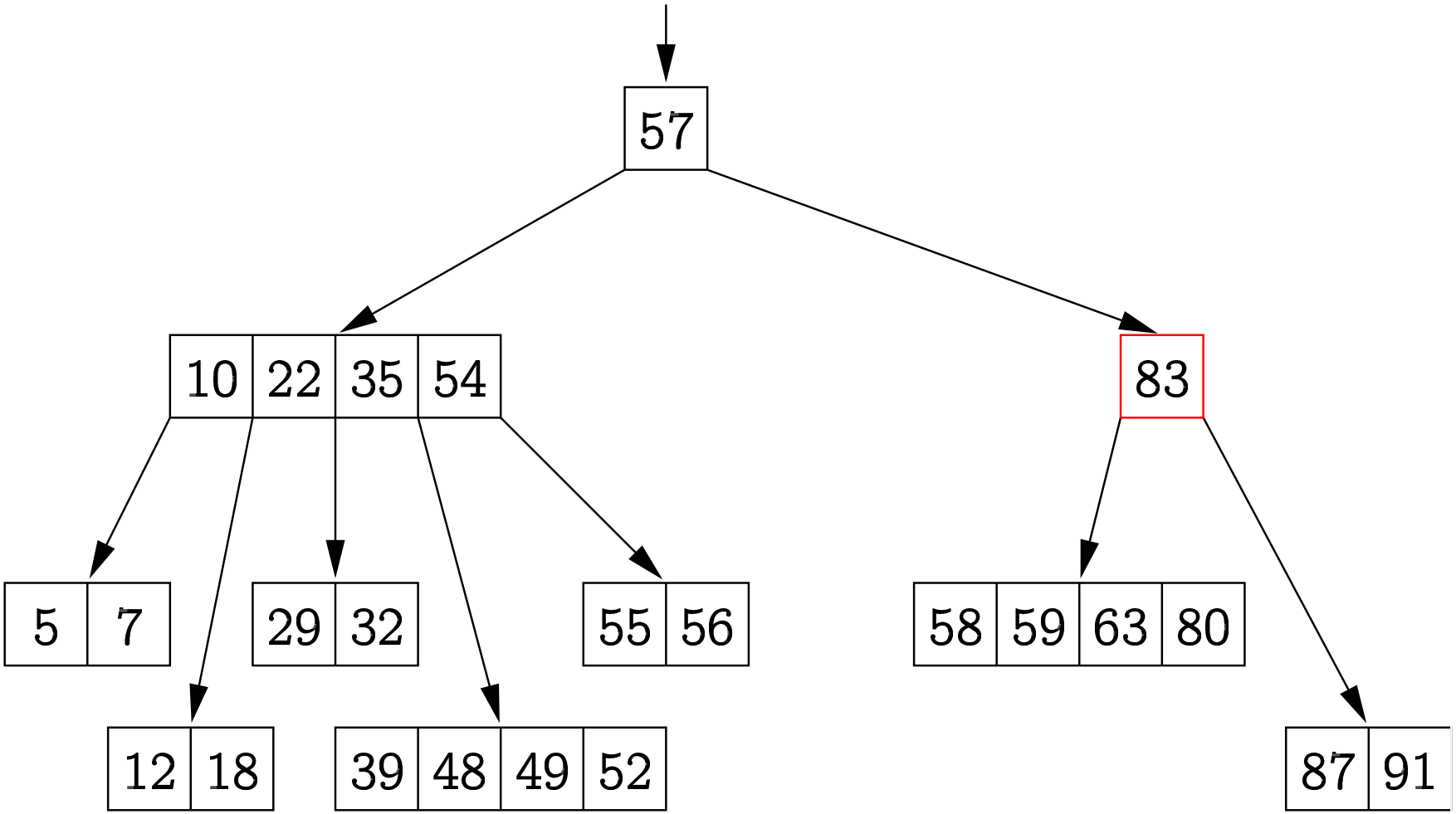


Veel üks näide naabrilt laenamisest.

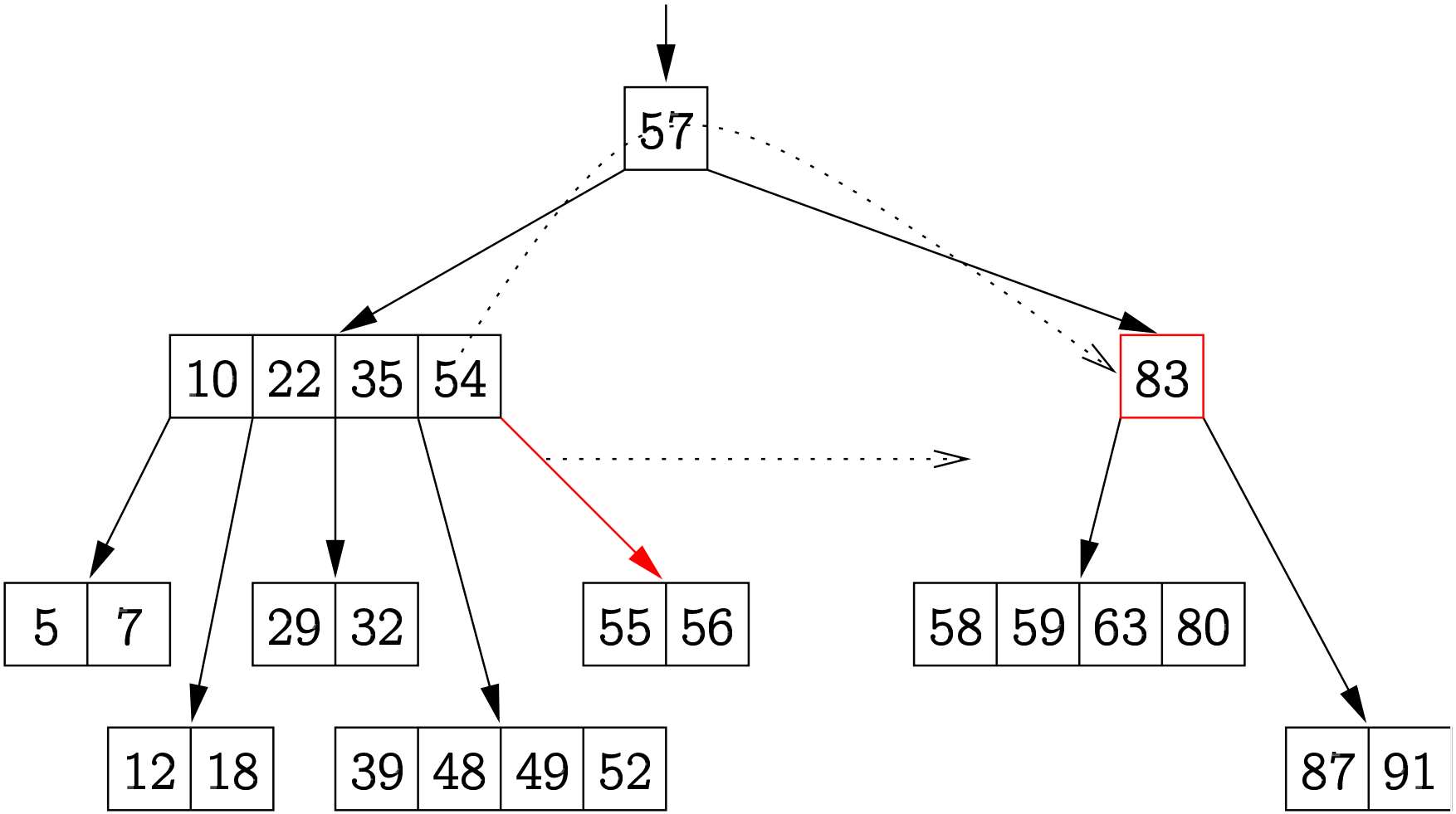
Eemaldame võtme 71



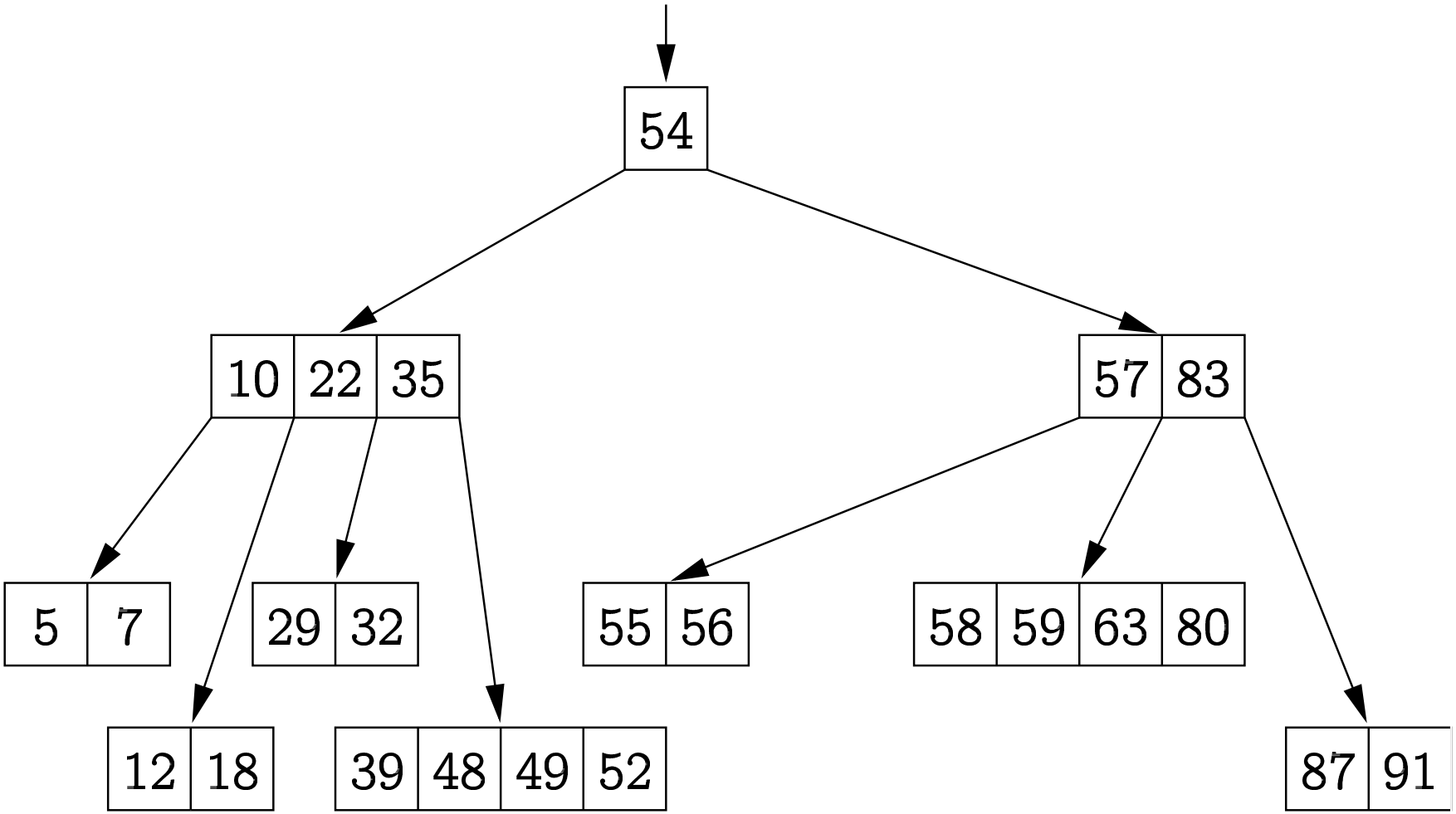
Eemaldame võtme 71



Eemaldame võtme 71



Eemaldame võtme 71



Sorteerimine:

Antud massiiv $a_1, \ldots, a_n$. Massiivi elemendiks on

$$\langle v\tilde{o}ti, \text{kirje} \rangle \ .$$

Massiivi elemendid tuleb ümber järjestada nii, et võtmed oleksid mittekahanevas järjekorras.

Kui seejuures võrdsete võtmetega elementide järjekord ei muutu, siis on sorteerimismeetod *stabiilne*.

Kõiki sorteerimismeetodeid on võimalik stabiliseerida, lisades massiivi elemendile täiendava välja:

$$\langle võti, orig, \text{kirje} \rangle,$$

omistades enne sorteerimist $a[i].orig := i$ ning lugedes sorteerides võtmeks paari $(võti, orig)$ (leksikograafiliselt järjestatuna).

Edasises tähistab

- $a_i < a_j$ massiivi i-nda ja j-nda elemendi võtmete võrdlemist;
- $a_i :=: a_j$ massiivi i-nda ja j-nda elemendi vahetamist.

Sorteerimine pistemeetodil (joonised 4.1 ja 4.2):

Kui lõik $a_1, \ldots, a_i$ on juba sorteeritud, siis leia a_{i+1} jaoks õige koht $a_1, \ldots, a_i$ seas ja pista ta sinna vahele (i muutub 1-st $(n-1)$-ni). Pistemeetod on stabiilne, kui „õige koht“ on parempoolseim võimalik.

Keerukus:

- Välimist tsüklit täidetakse $n-1$ korda.
- a_{i+1}-le koha otsimine ja ruumi tegemine võtab aega $\Theta(i)$:
  - kui a on massiiv, siis ruumi tegemine nõuab $\Theta(i)$ elemendi nihutamist (joonis 4.1);
  - kui a on list, siis koha otsimisel tuleb $\Theta(i)$ elementi läbi vaadata (joonis 4.2).

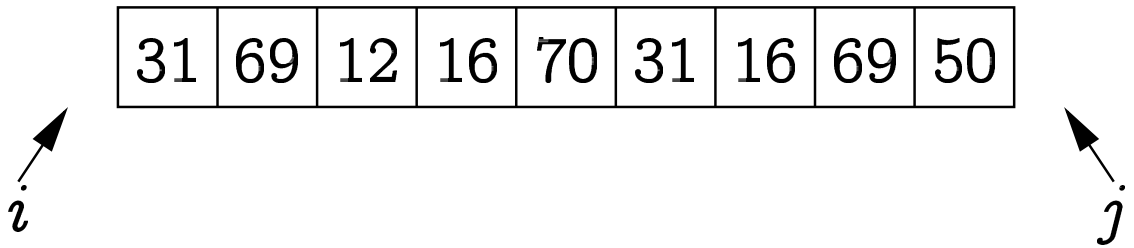
Kokku seega $\Theta(n^2)$.

Sorteerimine kiirmeetodil (joonised 4.4 ja 4.5):

- Vali mingi võtmeväärtus b (näiteks $b := a_1$).
- Kogu kõik kirjed, mis on $\leqslant b$, massiivi algusesse, ja kirjed, mis on $\geqslant b$, massiivi lõppu (prots. *jaotada*).
  - Otsi samaaegselt massiivi algusest alates suuri kirjeid ja lõpust alates väikesi kirjeid.
  - Kui on leitud algusosast suur kirje ja lõpuosast väike kirje, siis vaheta nad ära ja otsi edasi.
  - Kui otsimisjärgedega massiivi keskel kokku saadakse, siis on valmis.
- Sorteeri mõlemad massiivipooled.

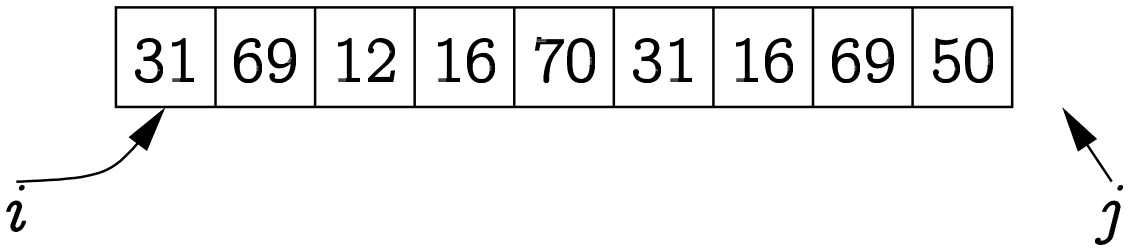
Näide jaotamisest:

$b = 31$



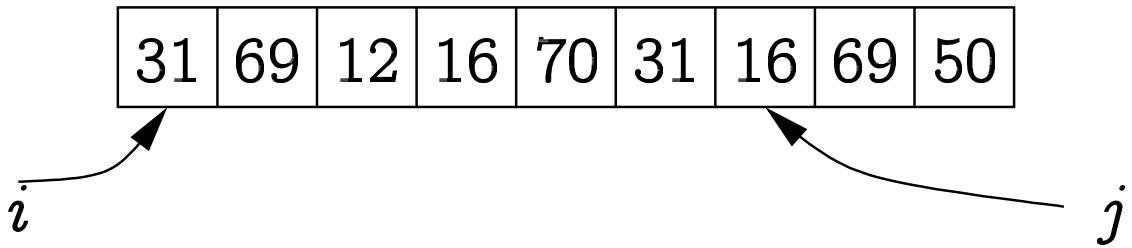
Näide jaotamisest:

$b = 31$

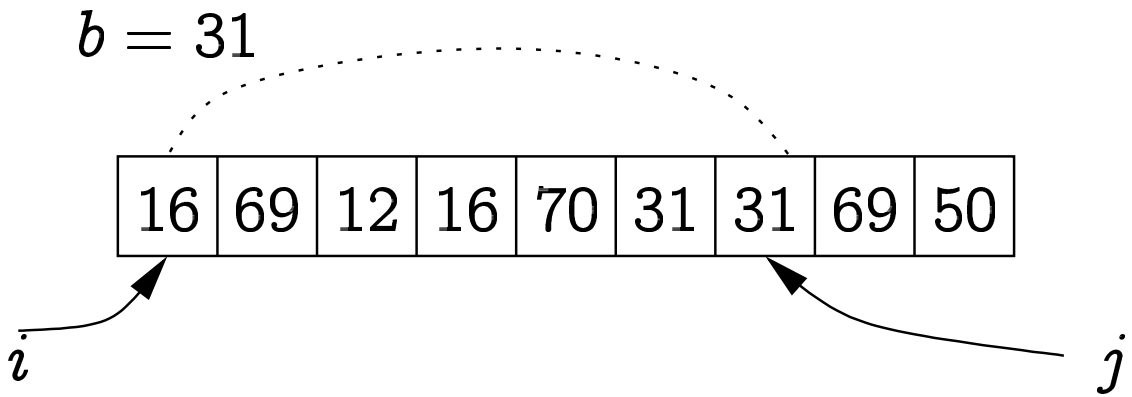


Näide jaotamisest:

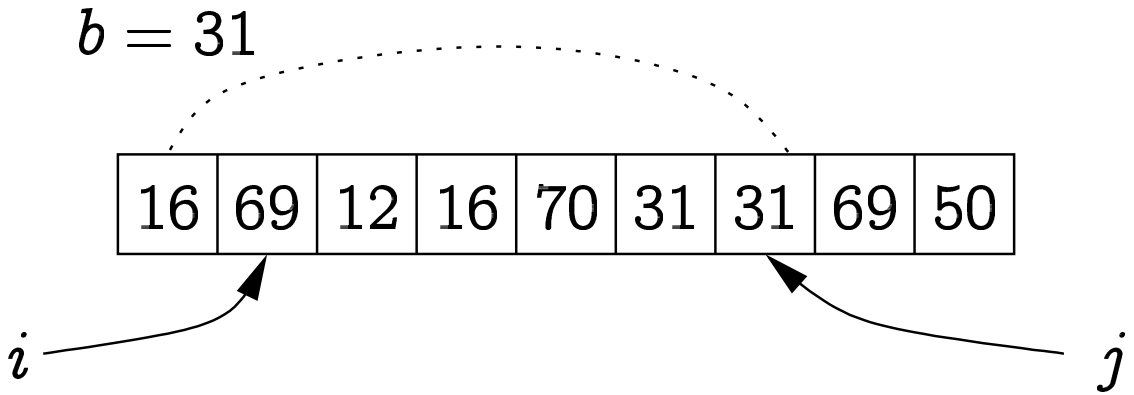
$b = 31$



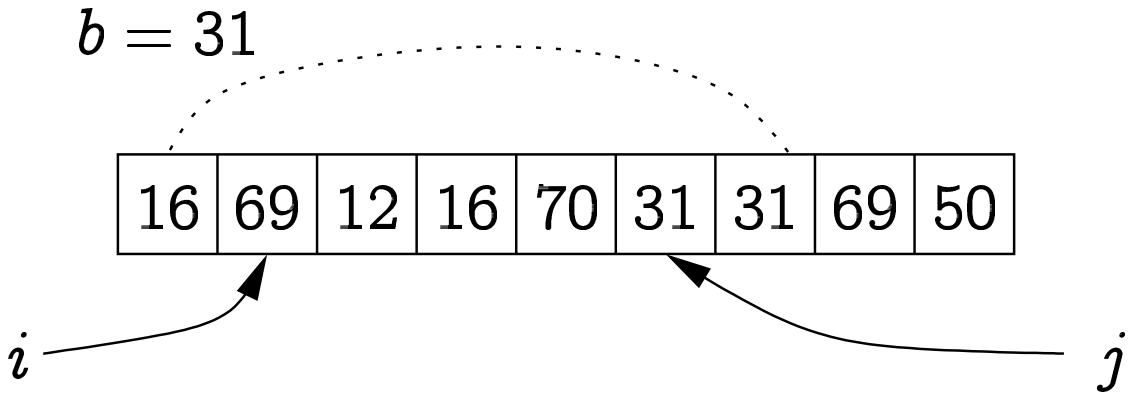
Näide jaotamisest:



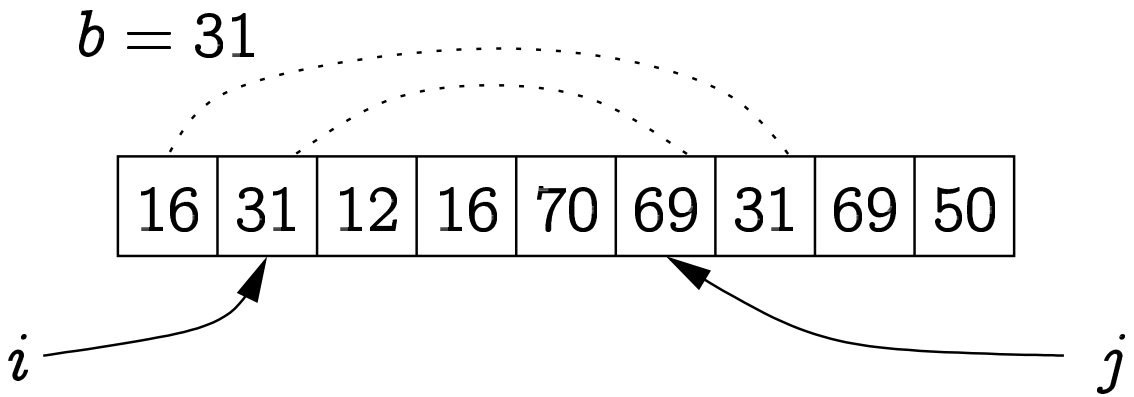
Näide jaotamisest:



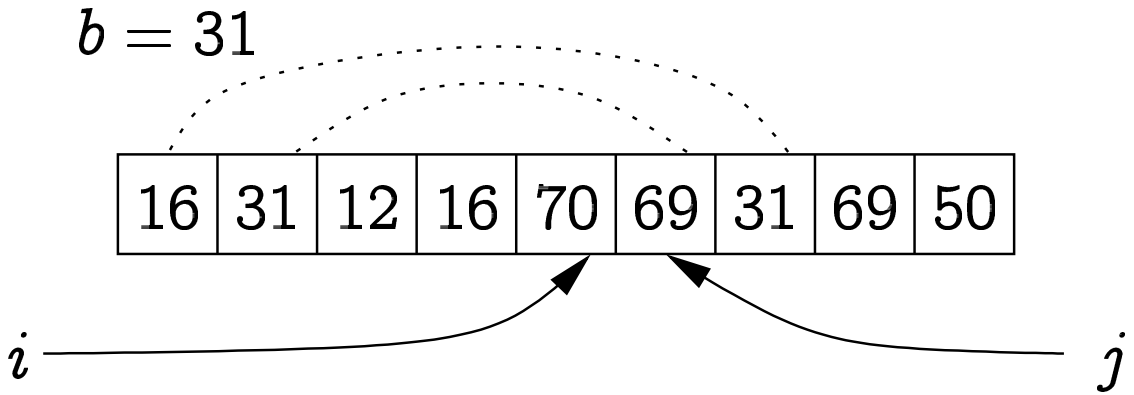
Näide jaotamisest:



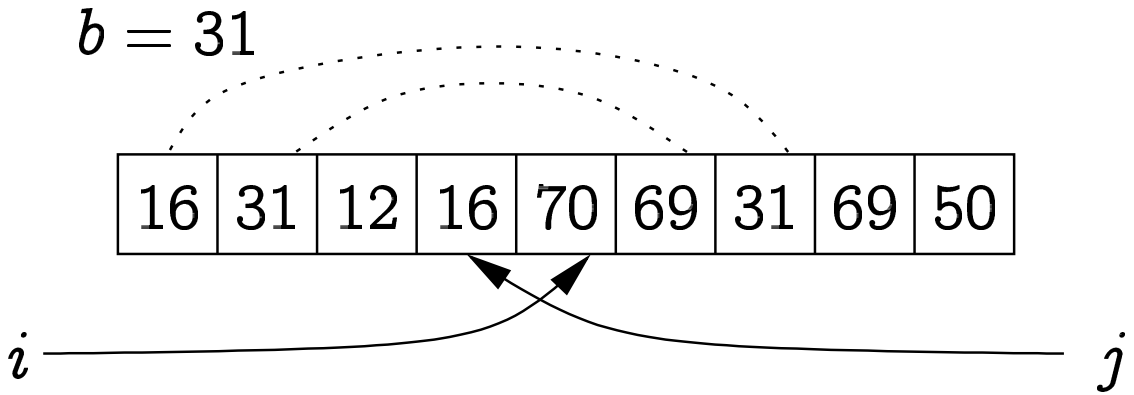
Näide jaotamisest:



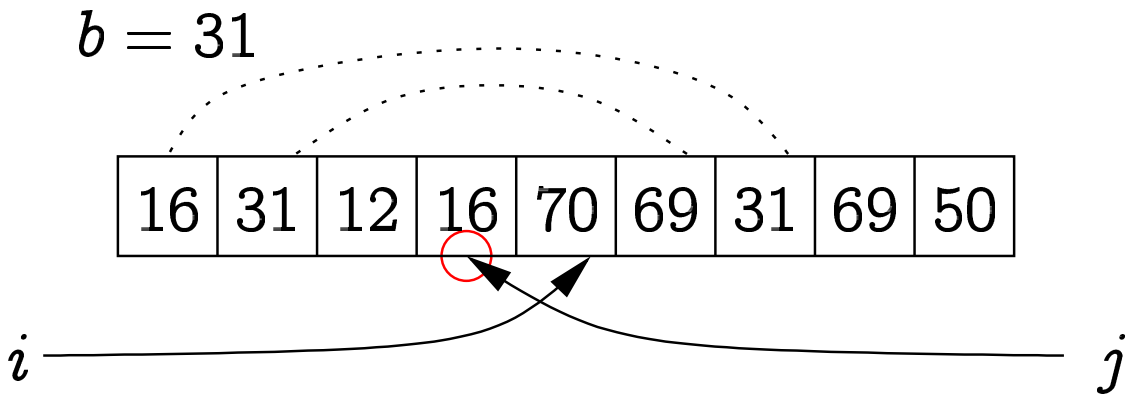
Näide jaotamisest:



Näide jaotamisest:



Näide jaotamisest:



Kiirmeetod ei sobi lihtahela sorteerimiseks, sest protseduuris *jaotada* tuleb meil ahelat läbida mõlemat pidi.

Küll aga sobib kiirmeetod topeltseotud ahela sorteerimiseks (ahela, kus igast tipust on viit järgmisele ja eelmisele tipule).

Kiirmeetod on ebastabiilne.

Keerukus halvimal juhul:

Protseduur jaotada vaatab massiivi kõik elemendid korra läbi — keerukus $\Theta(n)$.

Kuna ühte poolde võib jääda ainult üks kirje
(ja teise $(n-1)$ kirjet), siis

$$T(n) = T(1) + T(n-1) + \Theta(n) \ .$$

Kuna $T(1) = \Theta(1)$, siis

$$T(n) = T(n-1) + \Theta(n) = \sum_{i=1}^{n} \Theta(i) = \Theta(\sum_{i=1}^{n} i) = \Theta(n^2) \ .$$

Samas parimal juhul, kui jaotamine toimub alati keskelt, siis

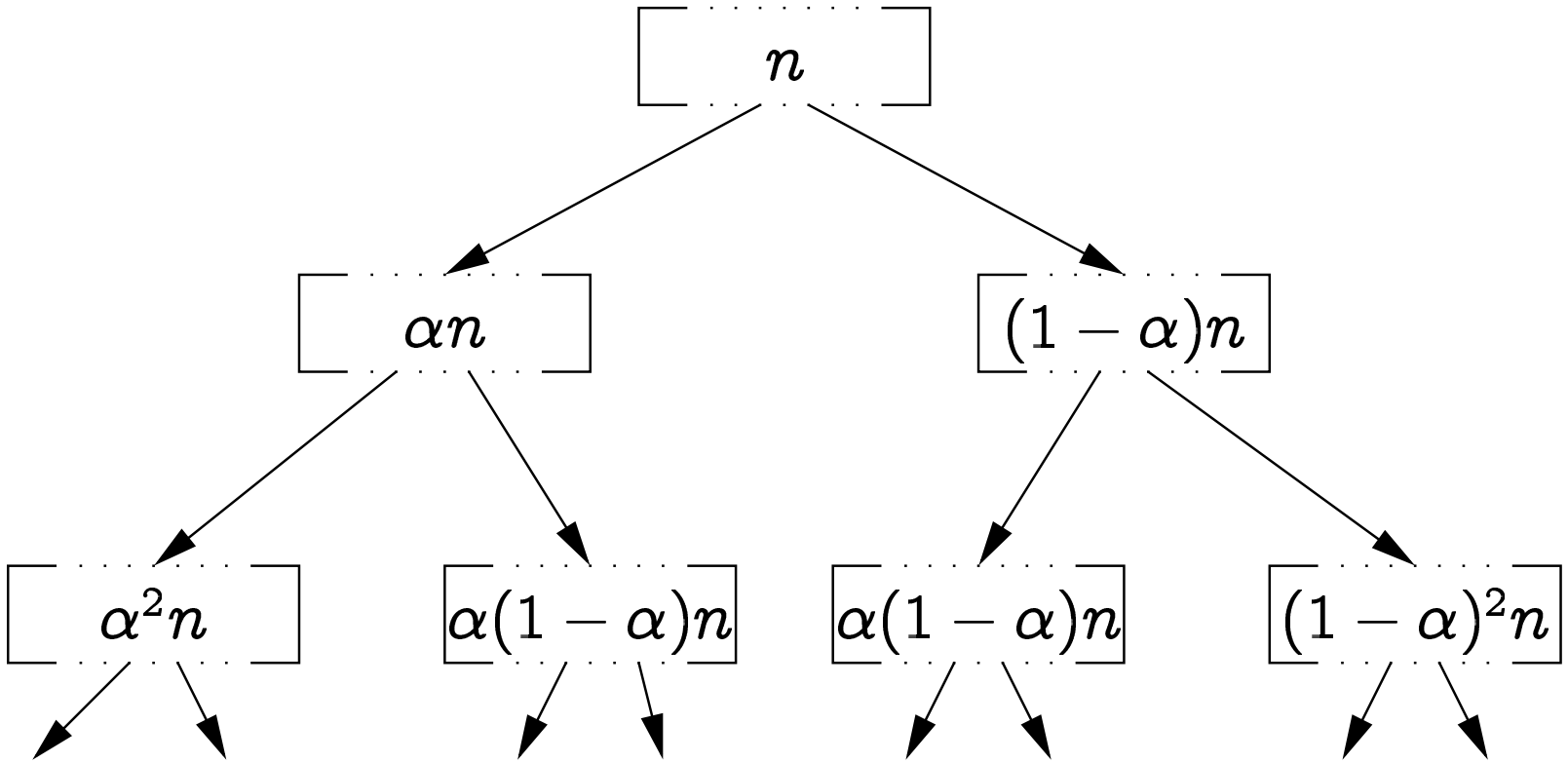
$$T(n) = 2T(n/2) + \Theta(n)$$

ja vastavalt põhiteoreemile (1. loeng), $T(n) = \Theta(n \log n)$.

Ka juhul, kui jaotamine toimub alati proportsioonis $\alpha : (1-\alpha)$, on keerukus logaritmiline. (α on suvaline reaalarv 0 ja 1 vahelt).

Tõepoolest, vaatame rekursioonipuud:

Töö hulk antud tasemel:



n

n

n

$\ldots$

$\leqslant n$

tasemete arv: $\max(\log_{1/\alpha} n, \log_{1/(1-\alpha)} n) = \Theta(\log n)$

Randomiseeritud kiirmeetod: jaotamisel vali võtmeväärtuseks juhuslikult üks sorteeritava massiivi elementidest.

randomiseeritud_jaotada(a, i, j) on

1 Vali p juhuslikult hulgast $\{i, i+1, \ldots, j\}$
2 $a_i :=: a_p$
3 **return** jaotada(a, i, j) --- *vt. joonist 4.4*

Juhuslikult valides peab igal elemendil olema sama suur tõenäosus valituks saada.

randomiseeritud_kiir_sorteerida(a, i, j) on

1 **if** $i \geqslant j$ **then return**
2 $k :=$ randomiseeritud_jaotada(a, i, j)
3 randomiseeritud_kiir_sorteerida(a, i, k)
4 randomiseeritud_kiir_sorteerida$(a, k+1, j)$

Randomiseeritud kiirmeetodi tööaeg ei sõltu sisendmassiivi kirjete järjekorrast.

Tõepoolest, protseduur jaotada võtab „veelahkmeks“ esimese elemendi, protseduur randomiseeritud_jaotada seega juhuslikult ükskõik millise elemendi.

Leiame randomiseeritud kiirmeetodi keskmise tööaja $T(n)$.

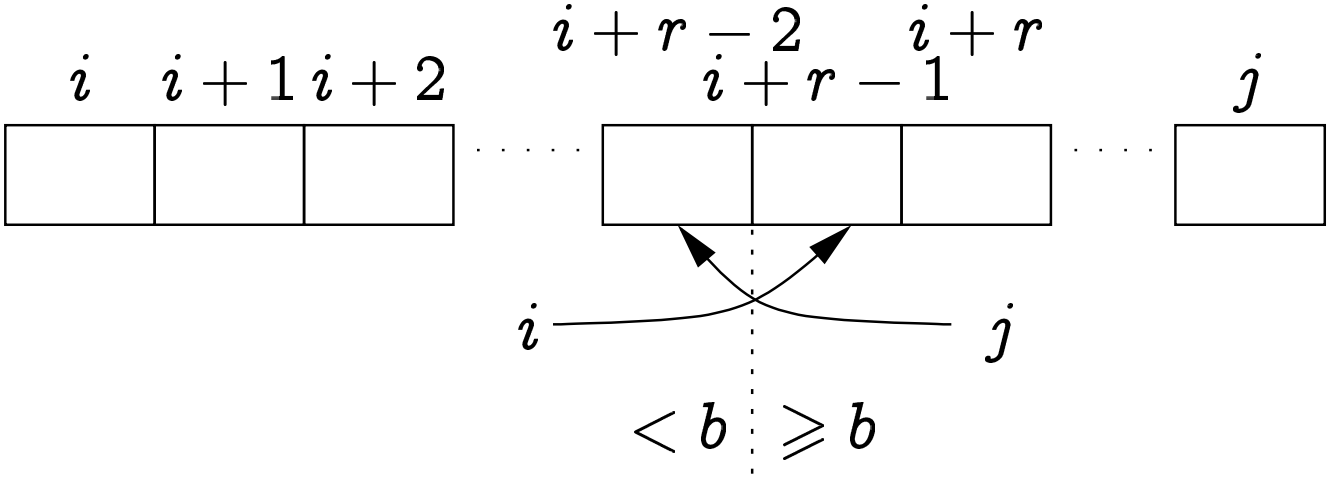
Analüüsi lihtsustamiseks loeme, et kõik massiivielemendid (s.t. kõigi elementide võtmed) on erinevad.

Kui $k :=$ randomiseeritud_jaotada(a, i, j), siis k väärtuseks võib saada $i, i+1, \ldots, j-1$. Millise tõenäosusega saab k ühe või teise väärtuse?

Kui veelahkmeks $b = a_i$ on protseduuris jaotada (joonis 4.4) vähim element lõigus $i..j$, siis on lõpuks i ja j väärtuseks esialgne i väärtus, s.t. $k = i$.

Kui veelahkmeks $b = a_i$ on suuruselt r-s element, kus $r \in \{2, \ldots, j-i+1\}$, siis kõigepealt vahetatakse omavahel a_i ja kõige kaugemal asuv b-st väiksem element.

Seis vahetult enne jaotada-st väljumist:



Tagastatakse j.

k väärtuseks saab $i + r - 2$.

Tõenäosus, et $k = i$, on $\frac{2}{j-i+1}$.

Tõenäosus, et k on võrdne mingi muu fikseeritud väärtusega lõigust $i + 1..j - 1$, on $\frac{1}{j-i+1}$.

Meil on

$$T(n) = \sum_{i=1}^{n-1} \lambda_i \cdot \big(T(i) + T(n-i)\big) + \Theta(n),$$

kus λ_i on tõenäosus, et ülesande suurusega n lahendamiseks peame me lahendama alamülesanded suurusega i ja $n-i$.

Eelneva arutelu põhjal $\lambda_1 = 2/n$ ja $\lambda_i = 1/n$, kui $i \neq 1$.

$$T(n) = \frac{1}{n}\left(T(1) + T(n-1) + \sum_{i=1}^{n-1}\big(T(i) + T(n-i)\big)\right) + \Theta(n)$$

Siin $\frac{1}{n}(T(1) + T(n-1))$ on $O(n)$, sest $T(n-1)$ on $O(n^2)$. See järeldub eespool olnud halvima juhu analüüsist.

Võime lugeda, et liige $\frac{1}{n}(T(1) + T(n-1))$ on liikme $\Theta(n)$ sees. Siis

$$T(n) = \frac{1}{n}\sum_{i=1}^{n-1}\big(T(i) + T(n-i)\big) + \Theta(n) = \frac{2}{n}\sum_{i=1}^{n-1} T(i) + \Theta(n) \ .$$

Näitame, et mingite konstantide a ja b jaoks $T(n) \leqslant an \log n + b$. Tõestus on induktsiooniga üle n.

Kui $n = 1$, siis tähendab see võrratus, et $T(1) \leqslant b$. Me võime valida b piisavalt suure selleks, et see võrratus kehtiks.

Kui $n > 1$, siis

$$T(n) = \frac{2}{n}\sum_{i=1}^{n-1} T(i) + \Theta(n) \leqslant \frac{2}{n}\sum_{i=1}^{n-1}(ai \log i + b) + \Theta(n) = \frac{2a}{n}\sum_{i=1}^{n-1} i \log i + \frac{2b}{n}(n-1) + \Theta(n) \ .$$

Hindame summat $\sum_{i=1}^{n-1} i \log i$.

Järgnev hinnang on liiga jäme:

$$\sum_{i=1}^{n-1} i \log i \leqslant \sum_{i=1}^{n-1} i \log n = \log n \frac{n(n-1)}{2} \leqslant \frac{1}{2} n^2 \log n \ .$$

Seda hinnangut kasutades saame

$$T(n) \leqslant \frac{2a}{n} \sum_{i=1}^{n-1} i \log i + \frac{2b}{n}(n-1) + \Theta(n) \leqslant$$

$$an \log n + b + \left(\frac{n-2}{n} b + \Theta(n)\right) \not\leqslant an \log n + b \ .$$

Järgnevas hindame summat $\sum_{i=1}^{n-1} i \log i$ täpsemalt.

$$\sum_{i=1}^{n-1} i \log i = \sum_{i=1}^{\lceil n/2 \rceil - 1} i \log i + \sum_{i=\lceil n/2 \rceil}^{n-1} i \log i \leqslant$$

$$\sum_{i=1}^{\lceil n/2 \rceil - 1} i \log (n/2) + \sum_{i=\lceil n/2 \rceil}^{n-1} i \log n =$$

$$(\log n - 1) \sum_{i=1}^{\lceil n/2 \rceil - 1} i + \log n \sum_{i=\lceil n/2 \rceil}^{n-1} i =$$

$$\log n \sum_{i=1}^{n-1} i - \sum_{i=1}^{\lceil n/2 \rceil - 1} i \leqslant$$

$$\log n \cdot \frac{n(n-1)}{2} - \frac{1}{2}\left(\frac{n}{2} - 1\right)\frac{n}{2} \leqslant \frac{1}{2} n^2 \log n - \frac{1}{8} n^2$$

Seda hinnangut kasutades saame

$$T(n) \leqslant \frac{2a}{n} \sum_{i=1}^{n-1} i \log i + \frac{2b}{n}(n-1) + \Theta(n) \leqslant$$

$$\frac{2a}{n}\left(\frac{1}{2}n^2 \log n - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n) \leqslant$$

$$an \log n - \frac{a}{4}n + 2b + \Theta(n) = an \log n + b + \left(\Theta(n) + b - \frac{a}{4}n\right) \quad .$$

Me võime valida a piisavalt suure selleks, et $\frac{a}{4}n \geqslant \Theta(n) + b$.
Siis $T(n) \leqslant an \log n + b$.

S.t. randomiseeritud kiirmeetodi keskmine tööaeg on $\Theta(n \log n)$.

Sorteerimine ühildusmeetodil (joonis 4.6):

- Jaga massiiv kaheks (enam-vähem) võrdse suurusega pooleks.
- Sorteeri mõlemad pooled.
- Ühilda need sorteeritud pooled.

Sobib lihtahelate sorteerimiseks, on stabiilne.

Keerukus: $T(n) = 2T(n/2) + \Theta(n)$, sest ühildamise keerukus on $\Theta(n)$. Põhiteoreemi järgi siis $T(n) = \Theta(n \log n)$.

Sorteerimine kuhjameetodil (joonis 3.12):

Olgu kuhjaomaduseks „ühegi tipu võti pole suurem kui ühegi tema järglase võti“ (s.t. vastupidine eelmise loenguga). Siis on kahendkuhja juurtipus kõige väiksema võtmeväärtusega kirje.

Meetod:

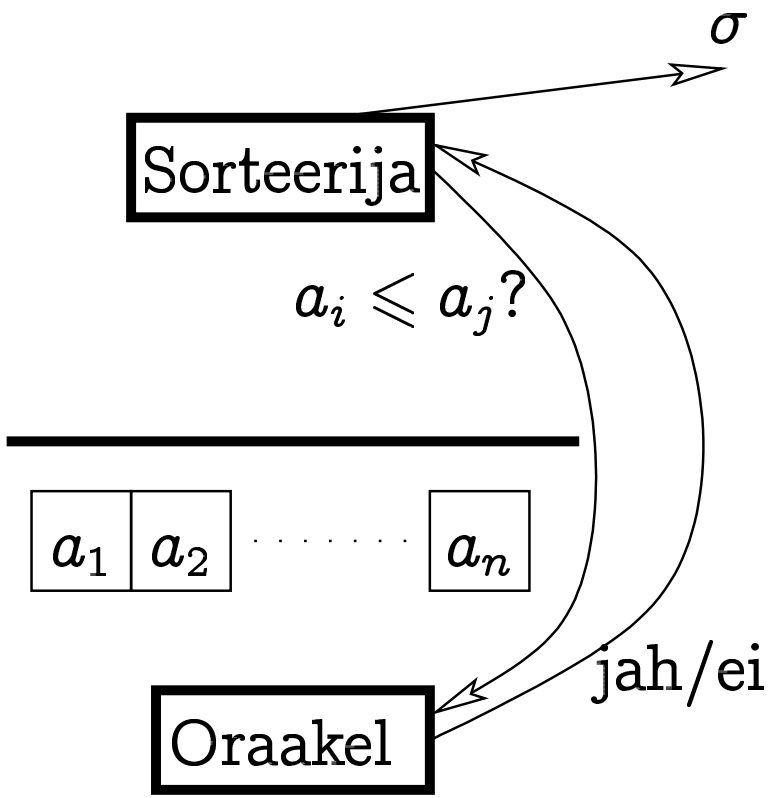
- Tee massiivist kuhi (joonis 3.11).
- Võta kuhjast vähimat elementi senikaua, kuni kuhi tühjaks saab.

Keerukus: esimene samm on $\Theta(n)$, teises sammus tuleb n korda teha võtmist, mille keerukus on $\Theta(\log n)$. Kokku $\Theta(n \log n)$.

Erinevate sorteerimismeetodite ajalises keerukuses on Θ sisse peidetud konstandid erineva suurusega, seetõttu võivad väikestel sisenditel asümptootiliselt aeglasemad algoritmid kiiremad olla (vt. joonis 4.5).

Paari loengu pärast näeme, kuidas panna kiirmeetod töö-le halvimal juhul ajaga $\Theta(n \log n)$ („veelahe“ tuleb hästi valida; see on ka deterministlikult võimalik).

Vaadeldud sorteerimismeetodid põhinesid elementide võrdlemisel:



Lõpuks peab sorteerija väljastama sellise σ, et $a_{\sigma(1)} \leqslant a_{\sigma(2)} \leqslant \cdots \leqslant a_{\sigma(n)}$.

Mitu võrdlemist on halvimal juhul vähemalt vaja?

Olgu hulgas X n elementi.

Alice on välja valinud mingi $x \in X$. Bob tahab teada, millise elemendi Alice valis.

Alice on nõus vastama selle elemendi kohta käivatele kas-küsimustele.

Mitu küsimust peab Bob halvimal juhul esitama, et teada saada, millise elemendi Alice valis?

Kui Bob pole veel midagi küsinud, siis võib elemendiks x olla (Bobi jaoks) ükskõik milline X-i element, s.t. on n võimalust.

Kui Bob on esitanud ühe küsimuse, siis halvemal juhul on veel järgi vähemalt $n/2$ võimalust.

Kui Bob on esitanud kaks küsimust, siis halvemal juhul on veel järgi vähemalt $n/4$ võimalust.

Kui Bob on esitanud k küsimust, siis halvemal juhul on veel järgi vähemalt $n/2^k$ võimalust.

Seega peab küsimusi olema vähemalt $\lceil \log n \rceil$.

Sorteerimisel on võimaluste arv $n!$, neist tuleb välja valida üks (kui kõik sorteeritavad elemendid on erinevad).

Halvemal juhul on siis võrdlemisi vähemalt

$$\log n! = \sum_{i=1}^{n} \log i \geqslant \sum_{i=\lceil n/2 \rceil}^{n} \log i \geqslant \sum_{i=\lceil n/2 \rceil}^{n} \log \lceil n/2 \rceil \geqslant$$
$$\lceil n/2 \rceil \log \lceil n/2 \rceil = \Omega(n \log n) \ .$$

(Samas muidugi ka $\log n! \leqslant \log n^n = n \log n$.)

Kui kasutada ka midagi muud peale elementide võrdlemiste, siis õnnestub sorteerida kiiremini kui keerukusega $\Theta(n \log n)$.

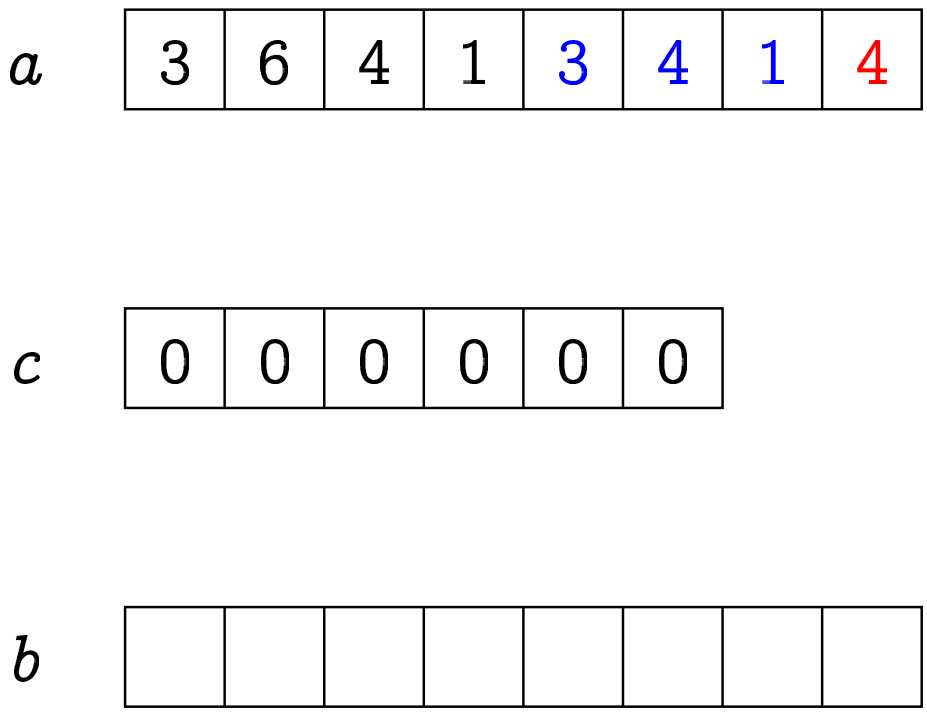
Seejuures tuleb sorteeritavate võtmete kohta midagi eeldada. Samuti kasutavad kõik meetodid täiendavalt mälu suuruses $\Theta(n)$.

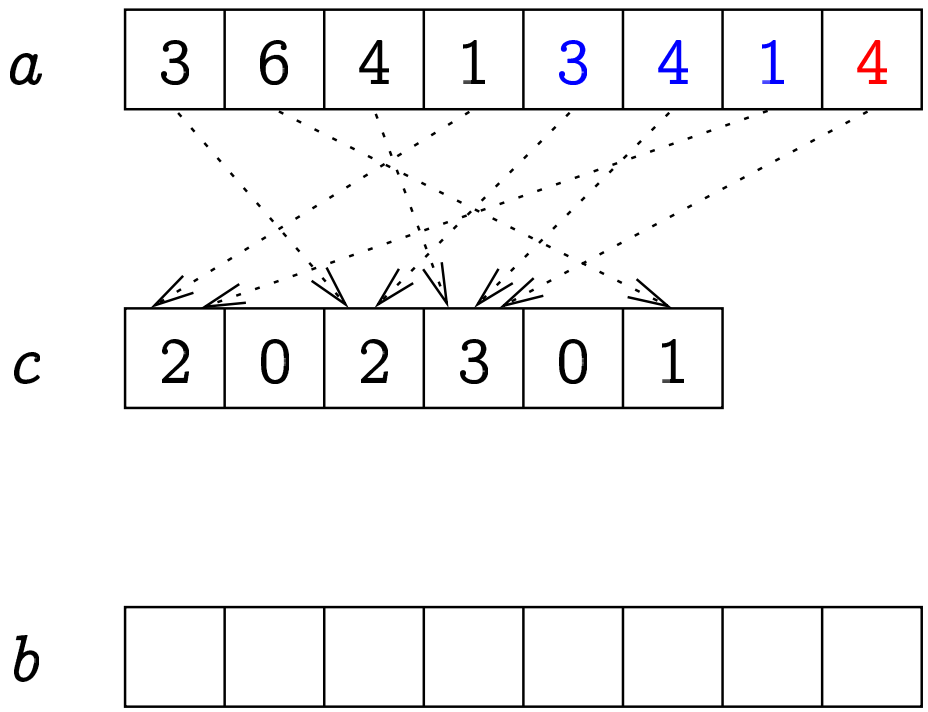
Loendamismeetodil eeldatakse, et kõik võtmed on täisarvud hulgast $\{1, \ldots, k\}$. Kirjete $a_1, \ldots, a_n$ sorteerimiseks vajatakse täiendavalt sama suurt massiivi b ja võtmemassiivi c suurusega k.

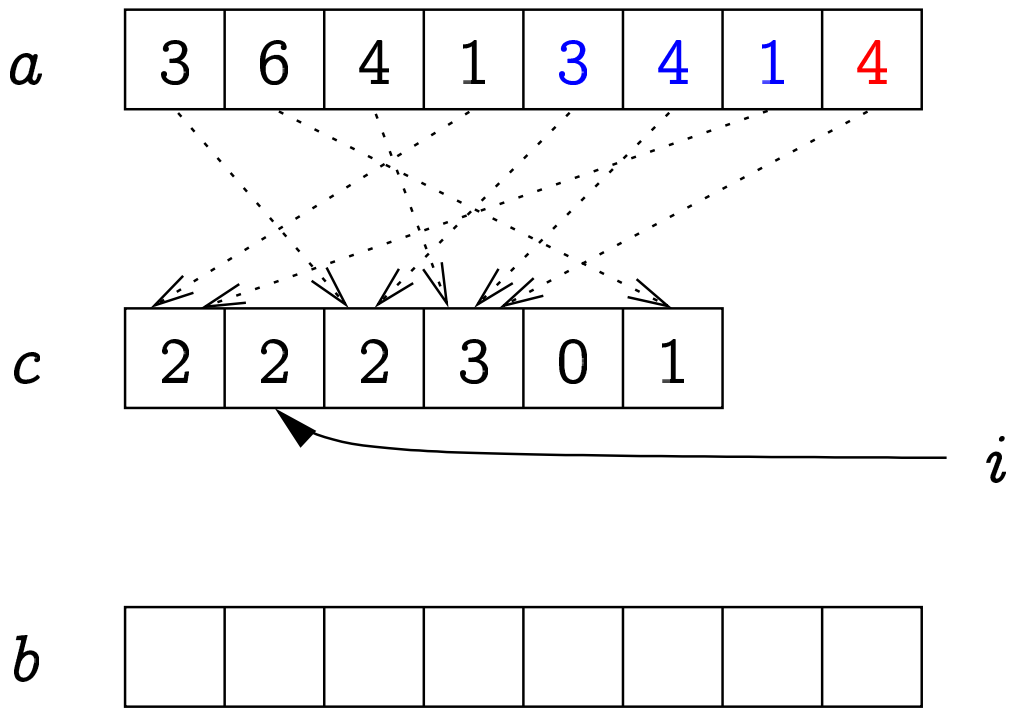
- Välja c_i väärtuseks võta võtme i esinemiste arv massiivis a.
  - Iga $i \in \{1, \ldots, n\}$ jaoks suurenda c_{a_i} väärtust ühe võrra.
- Välja c_i väärtuseks võta võtmete $\leqslant i$ esinemiste arv massiivis a.
  - Kui c_{i-1} on võtmete $\leqslant (i-1)$ esinemiste arv ja c_i võtmete i esinemiste arv, siis võta $c_i := c_i + c_{i-1}$.
  - i muutub 2-st k-ni.

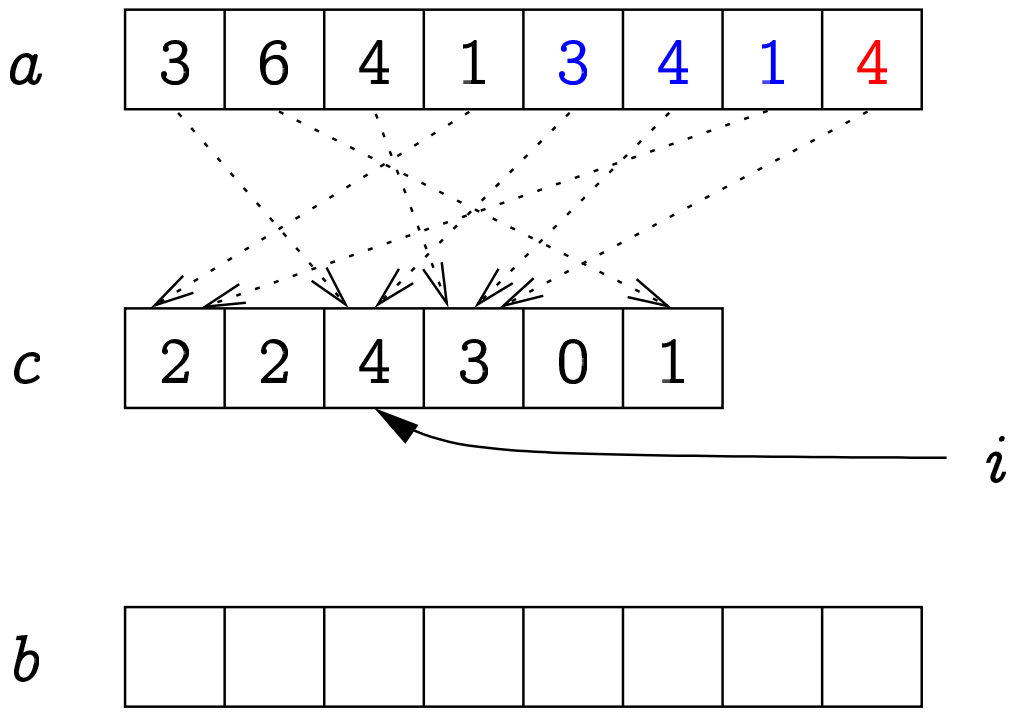
...

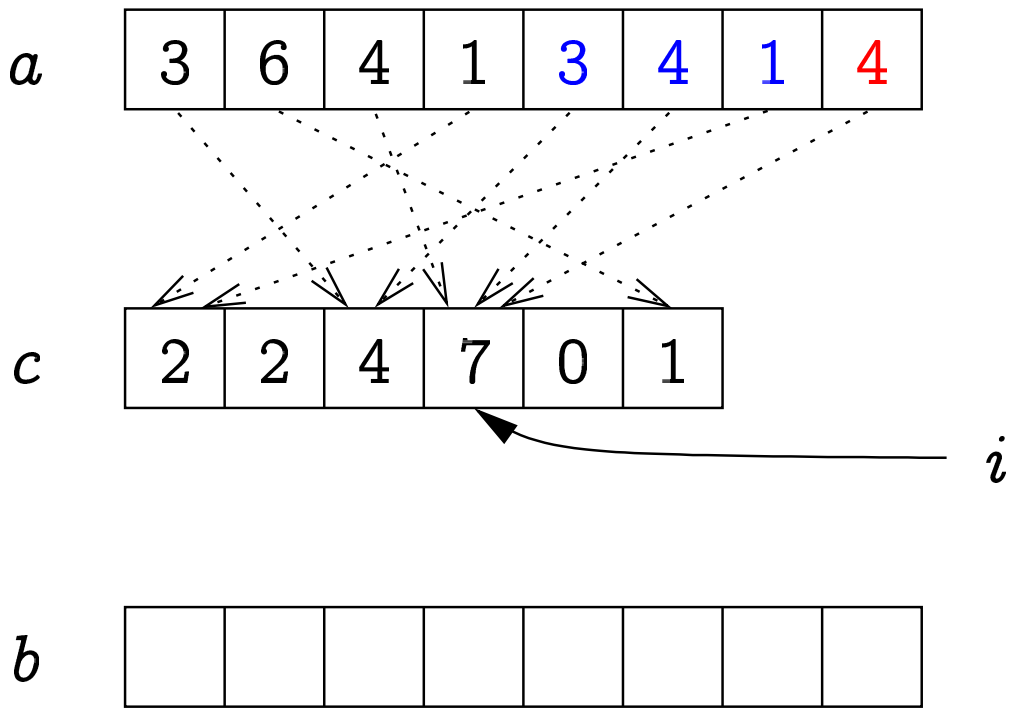
- Massiivi b väljadeks $c_{x-1}+1$ kuni c_x võta massiivi a kirjed võtmetega x.
  - Iga $i \in \{1, \ldots, n\}$ jaoks võta $b_{c_{a_i}}$ väärtuseks a_i ning vähenda c_{a_i} väärtust ühe võrra.
  - Kui seda teha i kahanemise järjekorras, siis on meetod stabiilne.
- Tagasta b.

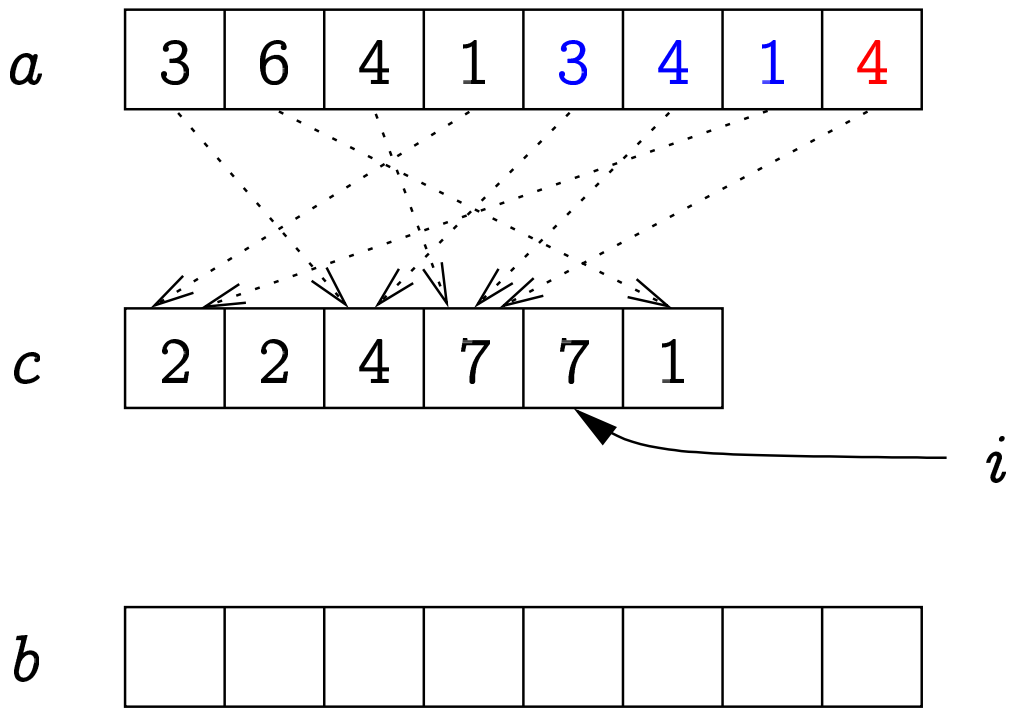

a
3 6 4 1 3 4 1 4
c
0 0 0 0 0 0
b

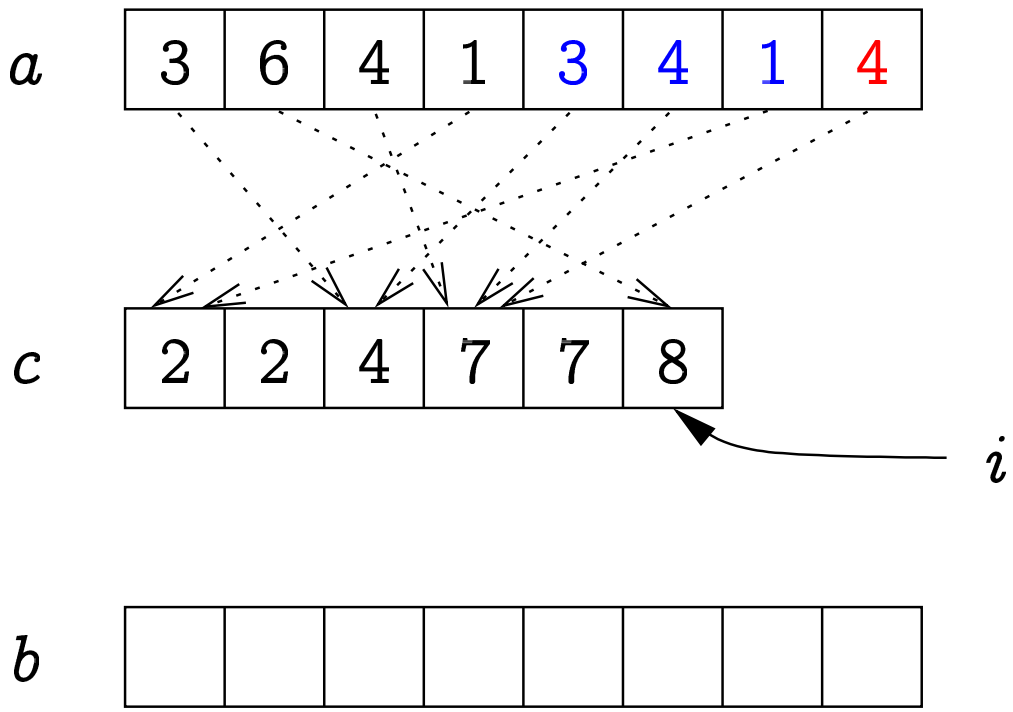

a
3 6 4 1 3 4 1 4
c
2 0 2 3 0 1
b

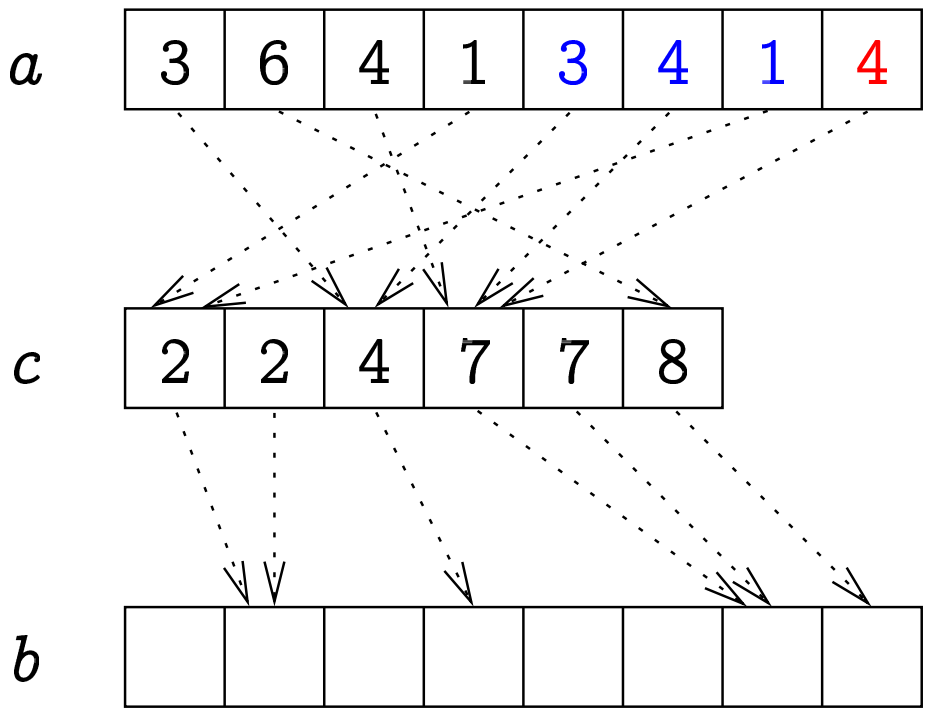

a
3
6
4
1
3
4
1
4
c
2
2
2
3
0
1
i
b

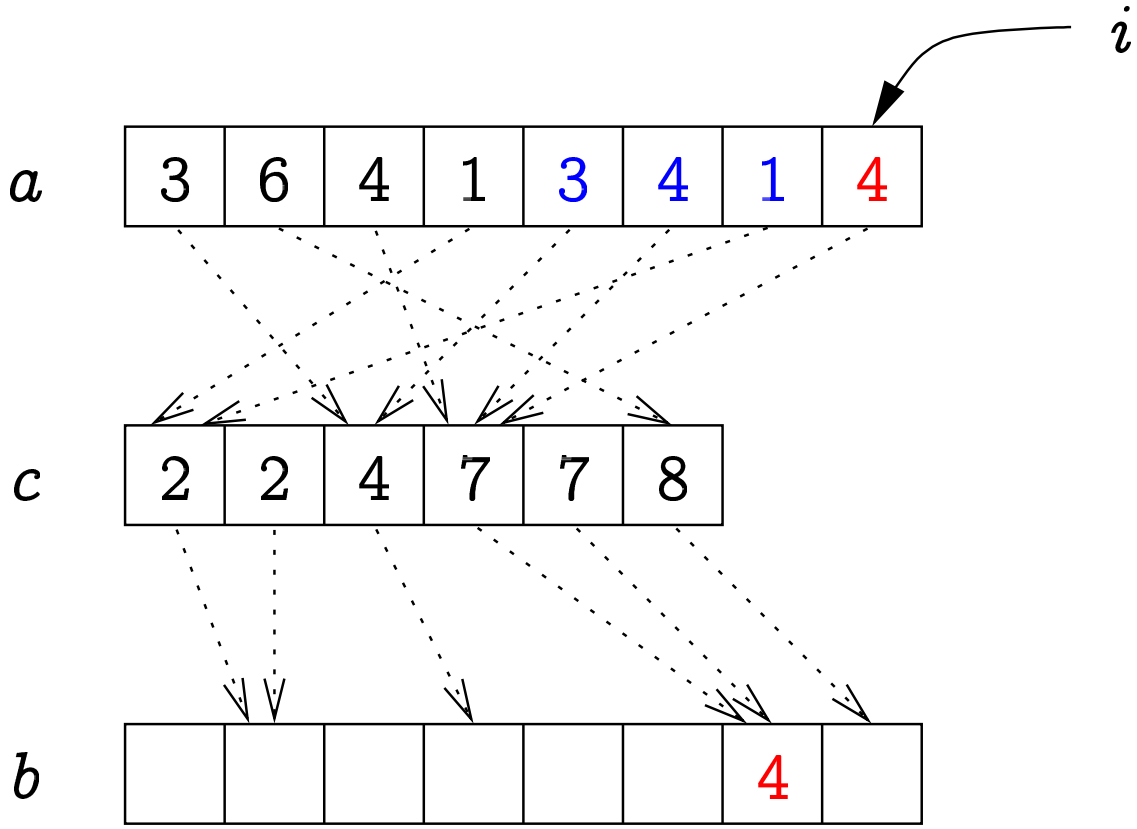

a
3
6
4
1
3
4
1
4
c
2
2
4
3
0
1
i
b

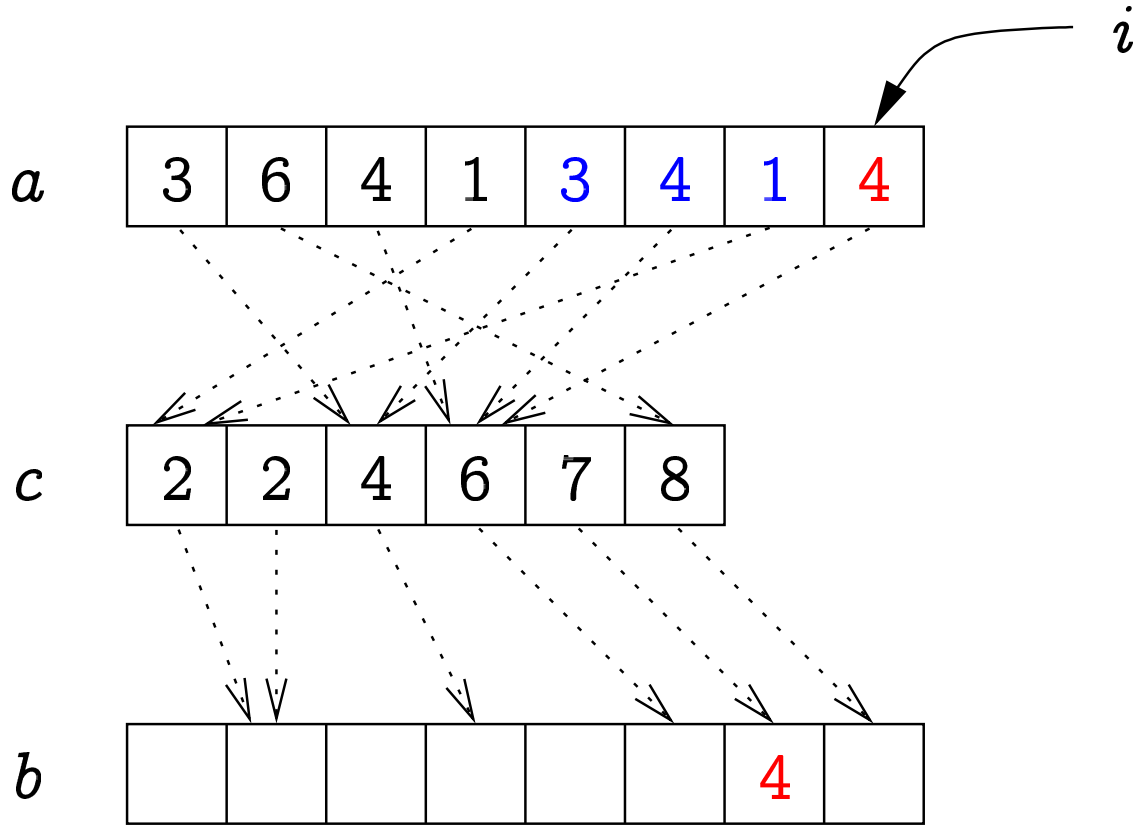

a
3
6
4
1
3
4
1
4
c
2
2
4
7
0
1
i
b

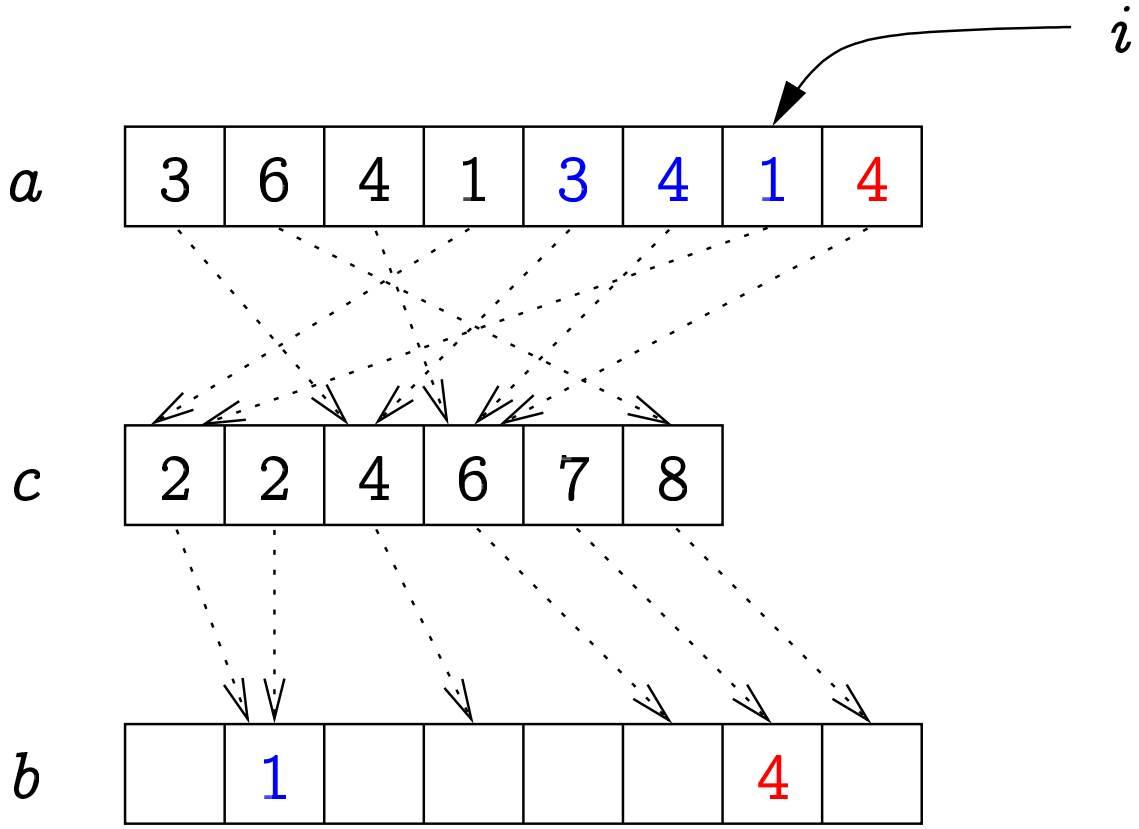

a
3
6
4
1
3
4
1
4
c
2
2
4
7
7
1
i
b

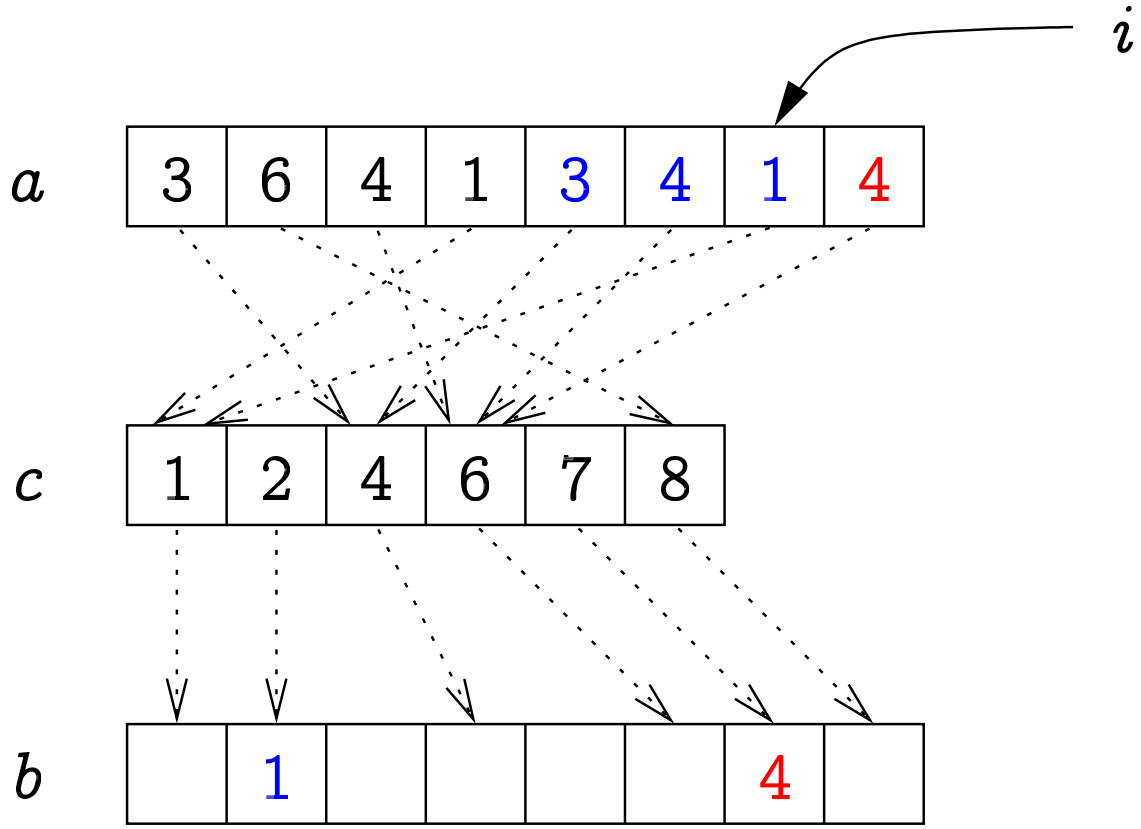

a
3
6
4
1
3
4
1
4
c
2
2
4
7
7
8
i
b

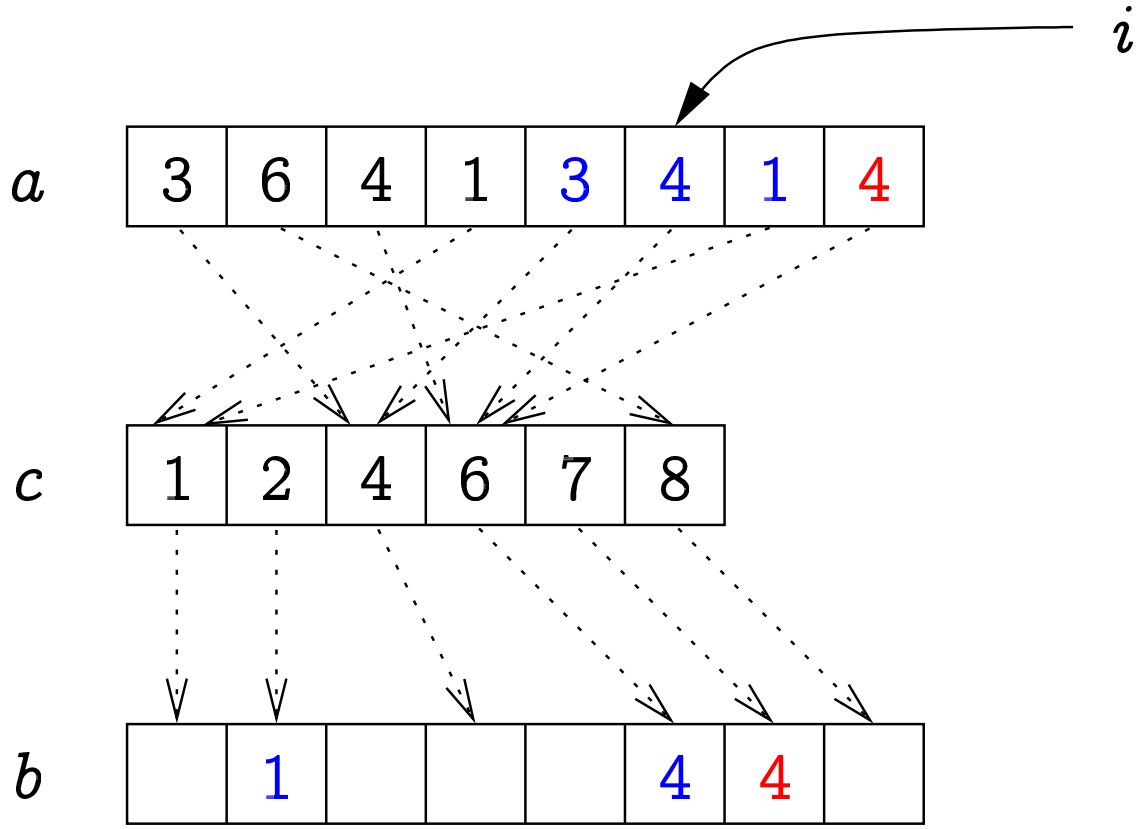

a
3
6
4
1
3
4
1
4
c
2
2
4
7
7
8
b

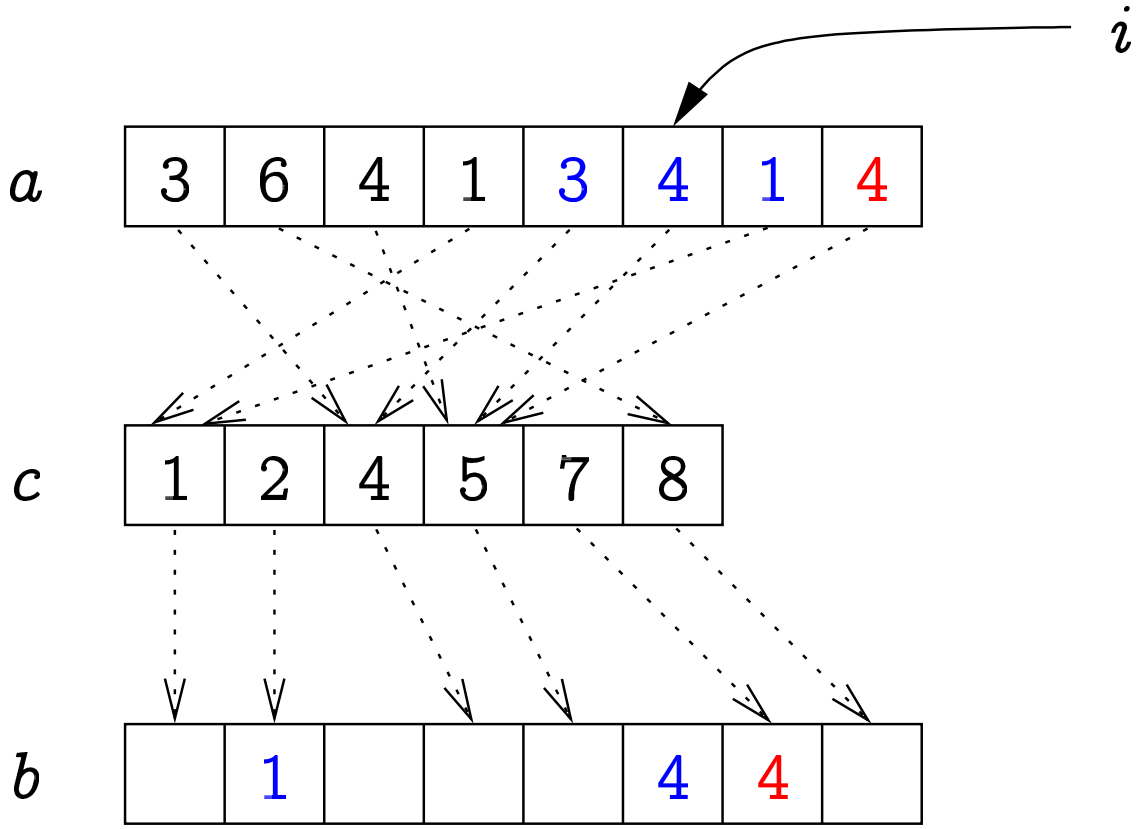

i
a
3 6 4 1 3 4 1 4
c
2 2 4 7 7 8
b
4

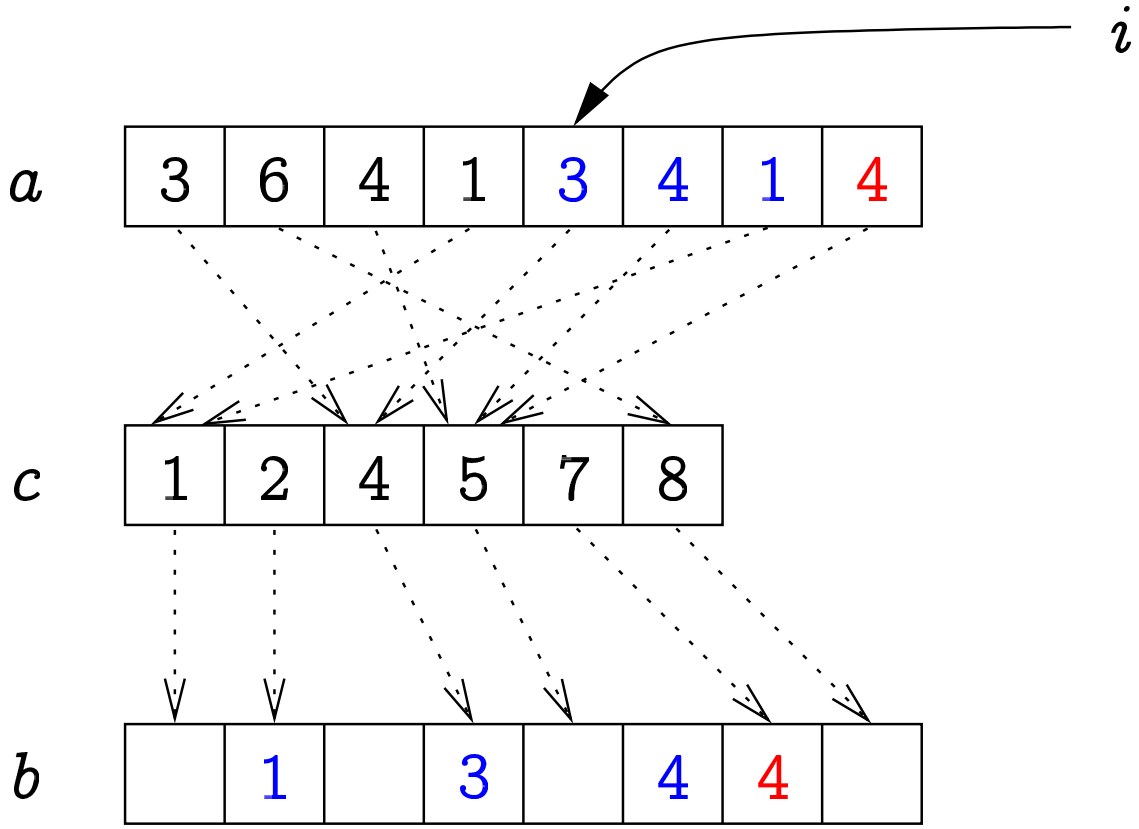

i
a
3
6
4
1
3
4
1
4
c
2
2
4
6
7
8
b
4

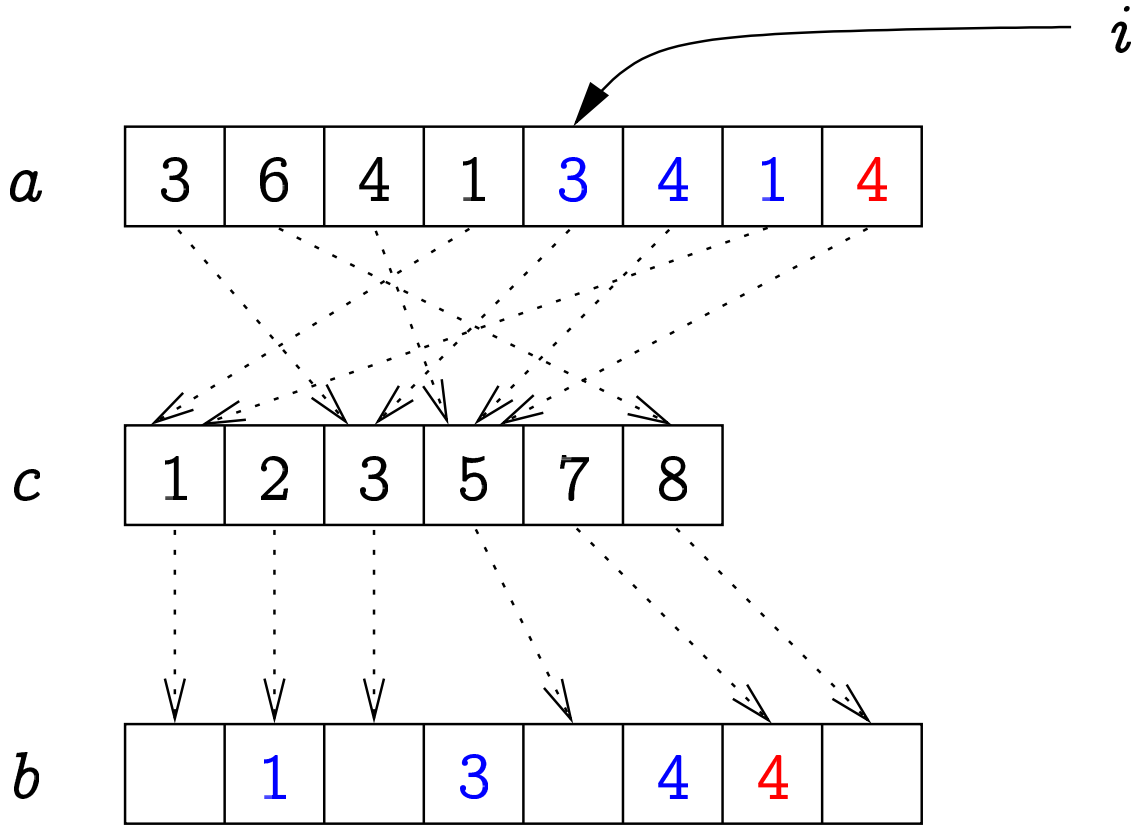

i
a
3 6 4 1 3 4 1 4
c
2 2 4 6 7 8
b
1 4

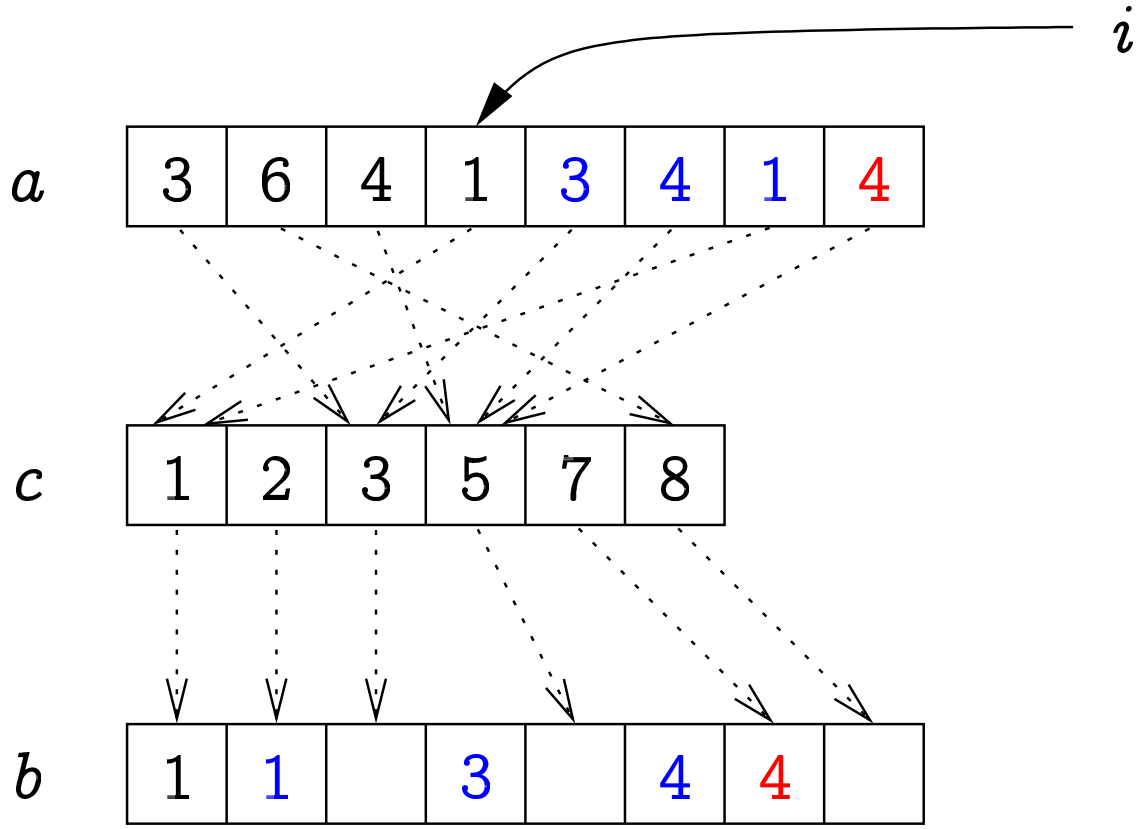

i
a
3
6
4
1
3
4
1
4
c
1
2
4
6
7
8
b
1
4

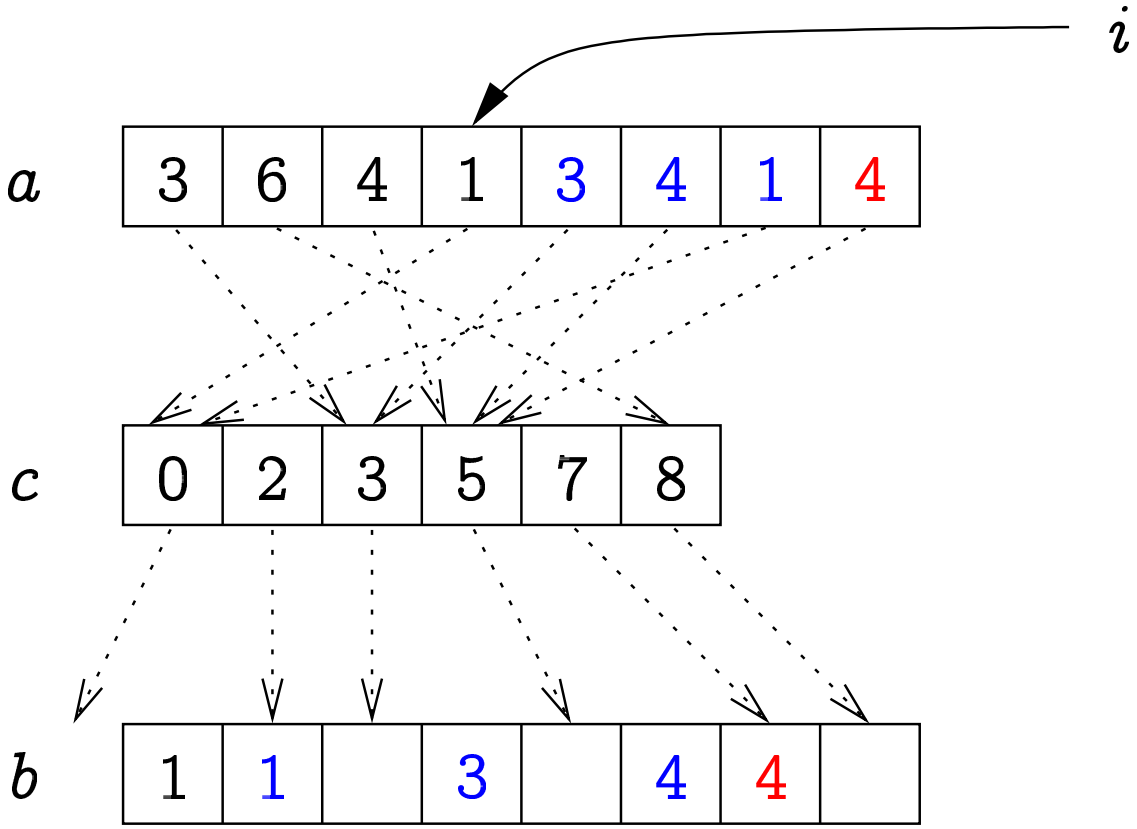

i
a
3
6
4
1
3
4
1
4
c
1
2
4
6
7
8
b
1
4
4

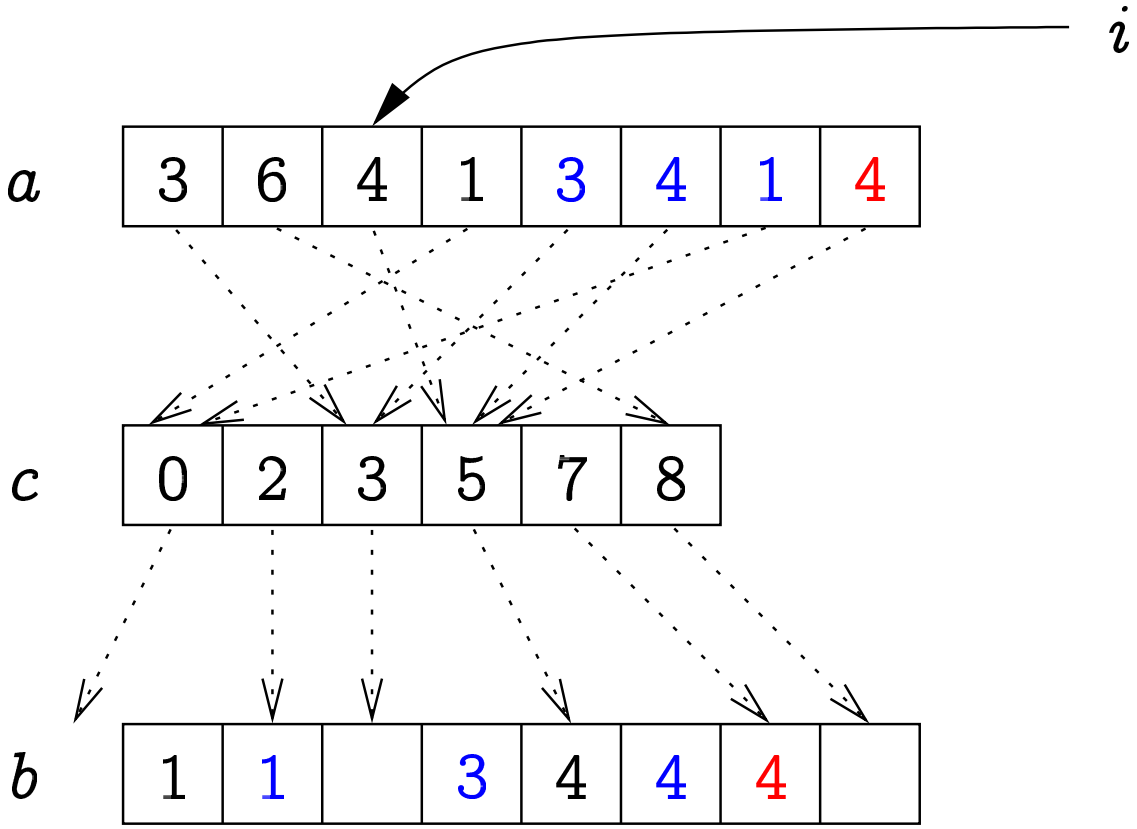

i
a
3 6 4 1 3 4 1 4
c
1 2 4 5 7 8
b
1 4 4

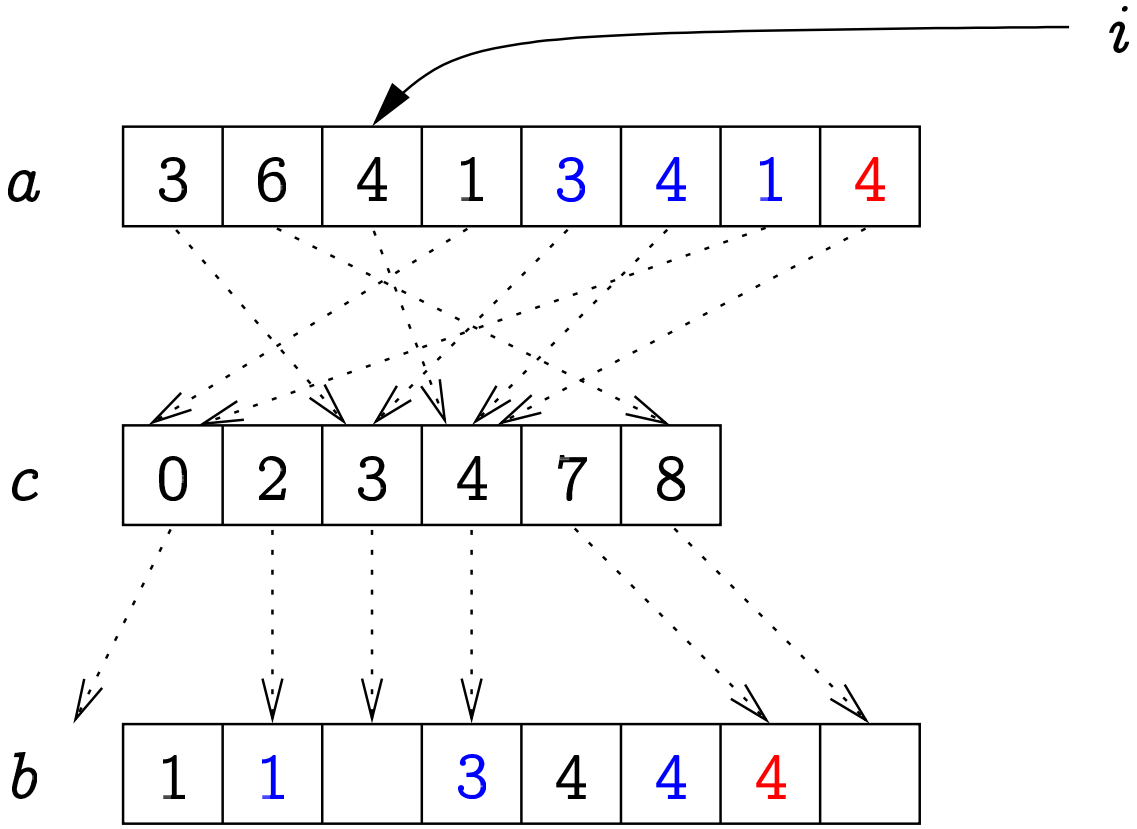

i
a
3 6 4 1 3 4 1 4
c
1 2 4 5 7 8
b
1 3 4 4

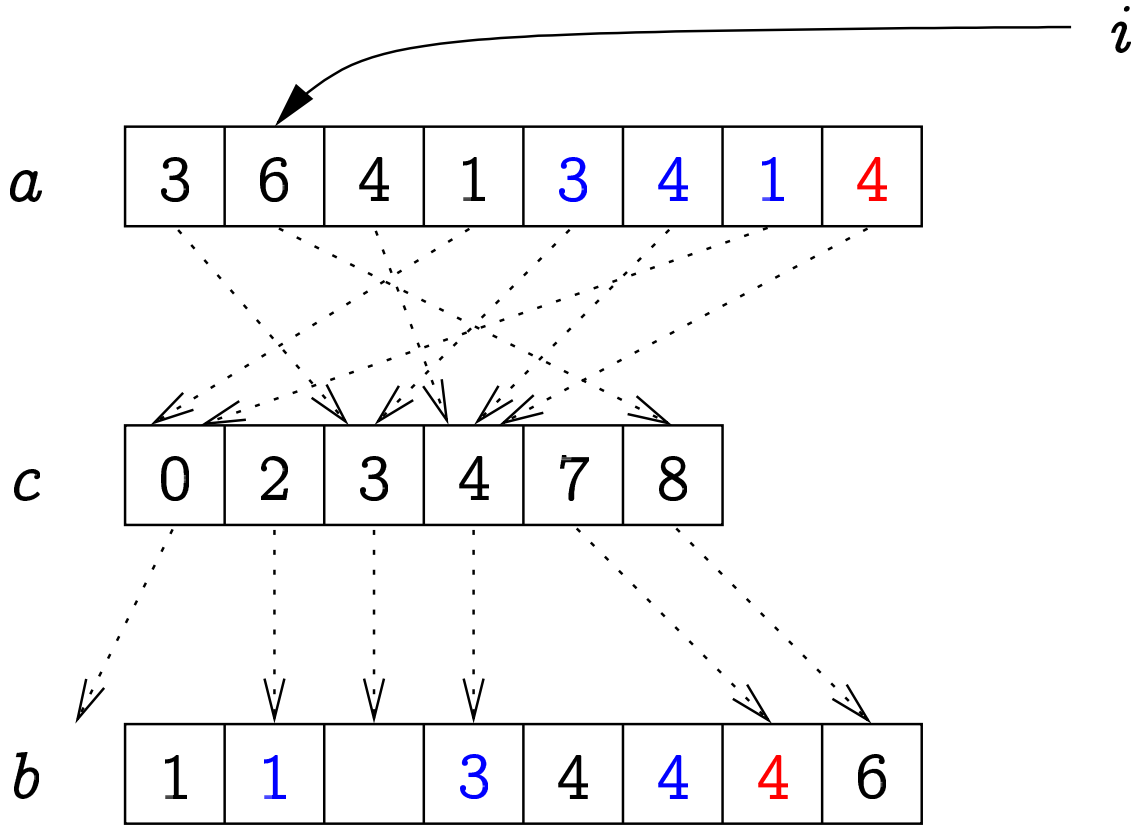

i
a
3 6 4 1 3 4 1 4
c
1 2 3 5 7 8
b
1 3 4 4

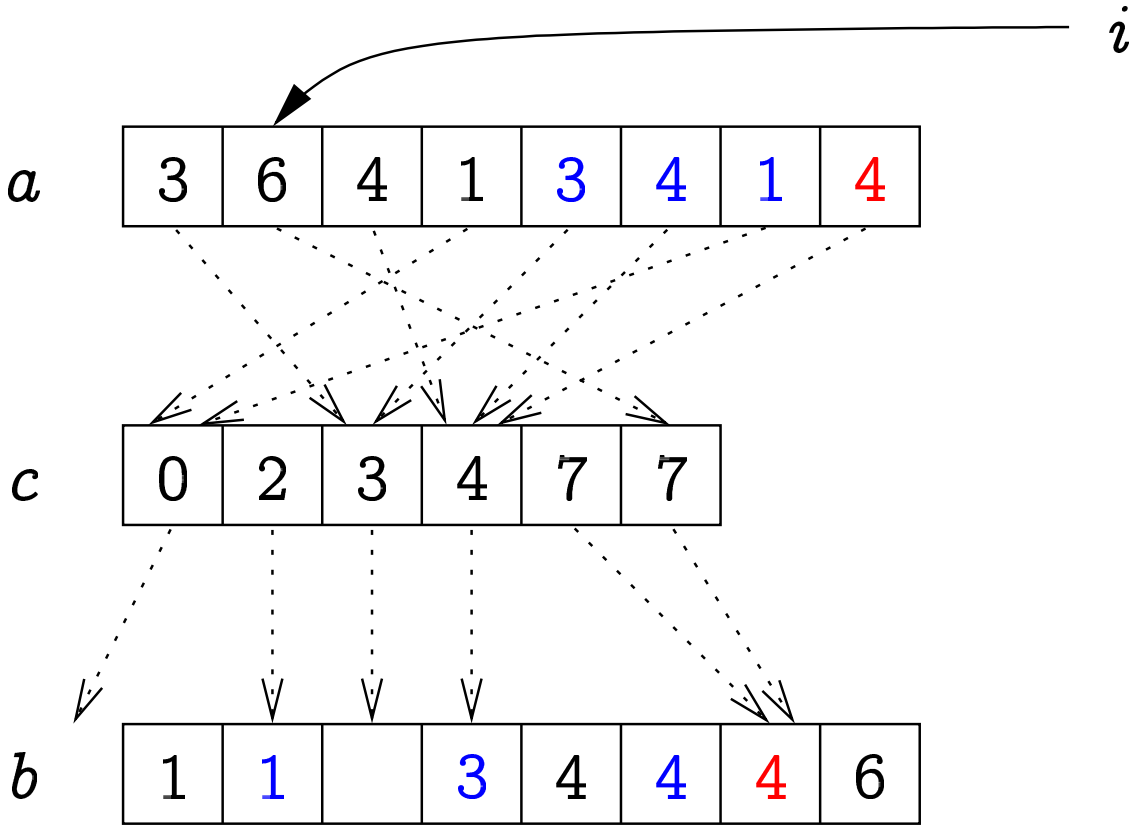

i
a
3 6 4 1 3 4 1 4
c
1 2 3 5 7 8
b
1 1 3 4 4

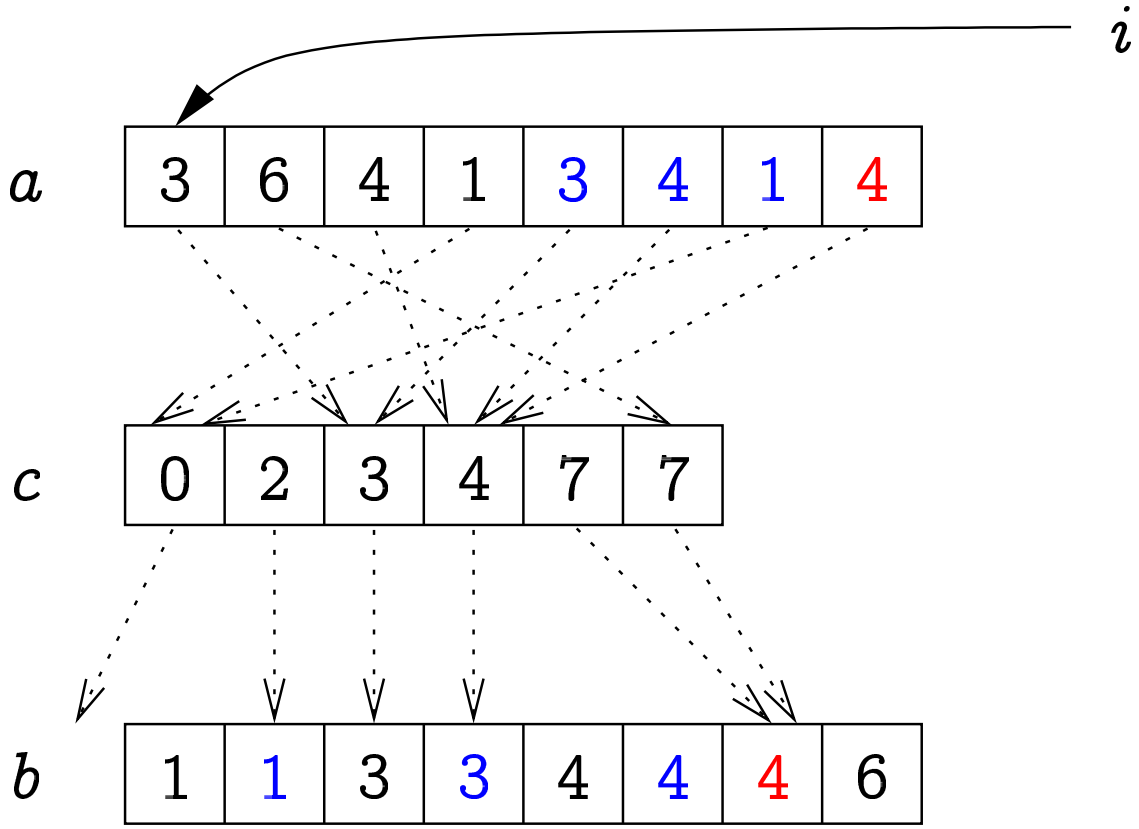

i
a
3 6 4 1 3 4 1 4
c
0 2 3 5 7 8
b
1 1 3 4 4

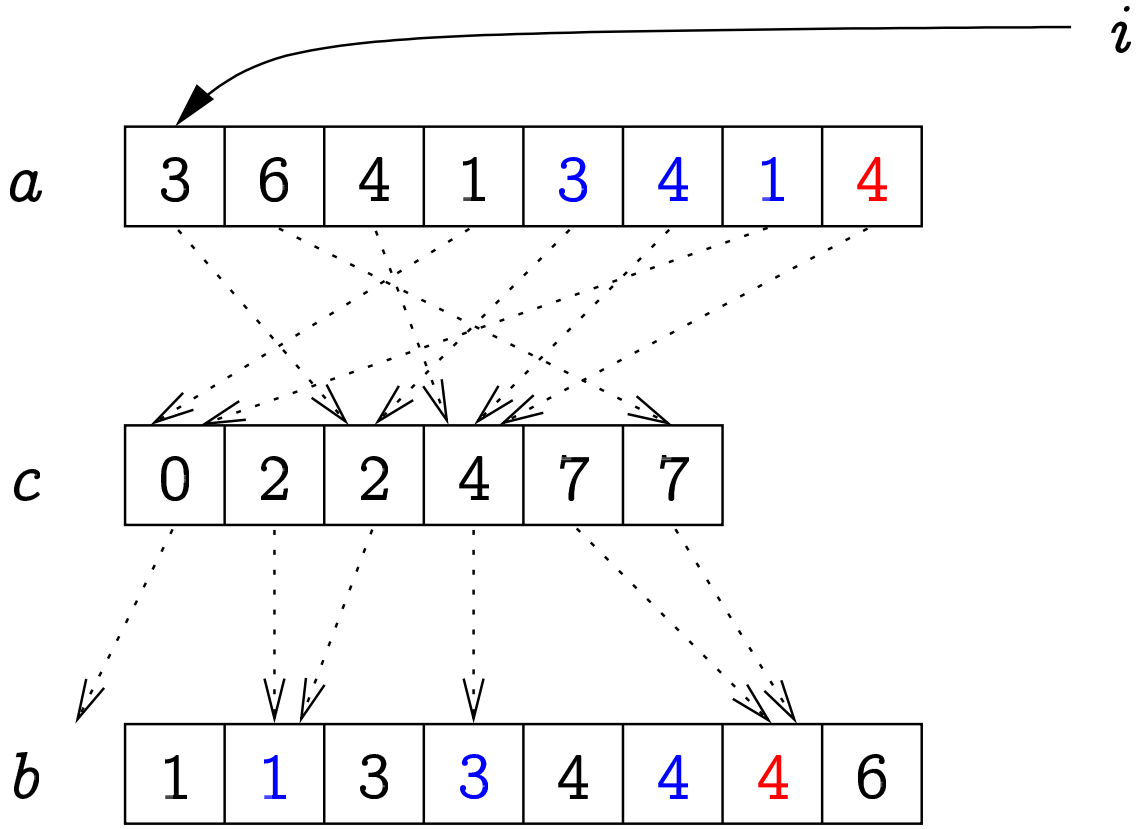

i
a
3 6 4 1 3 4 1 4
c
0 2 3 5 7 8
b
1 1 3 4 4 4


i
a
3 6 4 1 3 4 1 4
c
0 2 3 4 7 8
b
1 1 3 4 4 4


i
a
3 6 4 1 3 4 1 4
c
0 2 3 4 7 8
b
1 1 3 4 4 4 6


i
a
3 6 4 1 3 4 1 4
c
0 2 3 4 7 7
b
1 1 3 4 4 4 6


i
a
3 6 4 1 3 4 1 4
c
0 2 3 4 7 7
b
1 1 3 3 4 4 4 6


i
a
3 6 4 1 3 4 1 4
c
0 2 2 4 7 7
b
1 1 3 3 4 4 4 6

Positsioonimeetodil eeldatakse, et võti on kujul

$$(võti_1, võti_2, \ldots, võti_d)$$

ning järjestus on defineeritud leksikograafiliselt.

Algoritm:

1 **for** $j = d$ **downto** 1 **do**
2 Sorteeri a võtme $võti_j$ järgi, kasutades stabiilset meetodit.

Lause. Tsüklis, rea 2 järel on a sorteeritud

$$(võti_j, võti_{j+1}, \ldots, võti_d)$$

järgi.

Tõestus induktsiooniga. Baas, $j = d$ on ilmne.

Samm. Olgu a sorteeritud

$$(v\tilde{o}ti_{j+1}, v\tilde{o}ti_{j+2}, \ldots, v\tilde{o}ti_d)$$

järgi. Sorteerime ta stabiilselt $v\tilde{o}ti_j$ järgi.

Vaatame kirjeid a_k ja a_l. Kui $a_k.v\tilde{o}ti_j$ ja $a_l.v\tilde{o}ti_j$ on erinevad, siis pärast sorteerimist $v\tilde{o}ti_j$ järgi asuvad nad õiges järjekorras.

Kui $a_k.v\tilde{o}ti_j = a_l.v\tilde{o}ti_j$, siis pärast sorteerimist on nad samas järjekorras kui enne. Induktsiooni eelduse kohaselt olid nad enne positsioonide $j+1, \ldots, d$ järgi sorteeritud. Seega on nad õiges järjekorras.

Kimbumeetodil eeldatakse, et võtmed on reaalarvud vahemikust 0 (kaasa arvatud) kuni 1 (välja arvatud). Samuti eeldatakse, et nad on ühtlaselt ja üksteisest sõltumatult jaotunud.

Ühtlane jaotus tähendab, et kui $0 \leqslant \alpha \leqslant \beta < 1$, siis kui valime juhuslikult mingi a ja seejärel mingi $i \in \{1, \ldots, n\}$ (tõenäosus, et i saab mingi konkreetse väärtuse, on $1/n$), siis on sündmuse $\alpha \leqslant a_i \leqslant \beta$ tõenäosus $\frac{1}{\beta - \alpha}$.

Algoritm:

- Initsialiseerime n lihtahelat $p_0, \ldots, p_{n-1}$.
- Käime a läbi ning lisame elemendi a_i ahelasse $p_{\lfloor na_i \rfloor}$.
- Sorteerime kõik n lihtahelat.
- Konkateneerime nad üheks lihtahelaks.

Keerukus:

Esimene, teine ja neljas samm töötavad ajaga $\Theta(n)$.

Ühes ahelas on keskmiselt 1 element.

Loeme, et sorteerimine toimub $\Theta(n^2)$-keerukusega meetodiga. Leiame ühe ahela sorteerimise keskmise keerukuse.

Olgu n_i ahela p_i pikkus. Tõenäosus, et mingi a element on ahelas p_i on $1/n$ ja erinevate elementide ahelas p_i olemise tõenäosused üksteisest ei sõltu. (ühtlane jaotus!)

Seega on iga $1 \leqslant j_1 < j_2 < \cdots < j_k \leqslant n$ jaoks tõenäosus, et täpselt $a_{j_1}, \ldots, a_{j_k}$ ahelasse p_i satuvad $\left(\frac{1}{n}\right)^k \cdot \left(\frac{n-1}{n}\right)^{n-k}$.

Need $j_1, \ldots, j_k$ saab valida $\binom{n}{k}$ erineval viisil, s.t.

$$\Pr[n_i = k] = \left(\frac{1}{n}\right)^k \cdot \left(\frac{n-1}{n}\right)^{n-k} \cdot \binom{n}{k} = \binom{n}{k} \frac{(n-1)^{n-k}}{n^n} \quad .$$

Sorteerimise keskmine keerukus on

$$\sum_{k=0}^{n} \binom{n}{k} \frac{(n-1)^{n-k}}{n^n} \Theta(k^2) = \Theta\left(\frac{1}{n^n} \sum_{k=0}^{n} \binom{n}{k} k^2 (n-1)^{n-k}\right) \quad .$$

Kuna $k \cdot \binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{1\cdot 2\cdots(k-1)} = n \cdot \binom{n-1}{k-1}$, siis

$$k^2\binom{n}{k} = nk\binom{n-1}{k-1} = n(k-1)\binom{n-1}{k-1} + n\binom{n-1}{k-1} = n(n-1)\binom{n-2}{k-2} + n\binom{n-1}{k-1} \ .$$

Siin loeme, et kui $m < 0$, siis $\binom{n}{m} = 0$.

$$\frac{1}{n^n}\sum_{k=0}^{n}\binom{n}{k}k^2(n-1)^{n-k}=$$

$$\frac{n-1}{n^{n-1}}\sum_{k=2}^{n}\binom{n-2}{k-2}(n-1)^{n-k}+\frac{1}{n^{n-1}}\sum_{k=1}^{n}\binom{n-1}{k-1}(n-1)^{n-k}=$$

$$\frac{(n-1)n^{n-2}}{n^{n-1}}+\frac{n^{n-1}}{n^{n-1}}=2-\frac{1}{n}<2\ .$$

S.t. ühe ahela sorteerimine käib keskmiselt konstantse ajaga ning n ahela sorteerimine keskmiselt ajaga $\Theta(n)$.

Tõenäosusteooriast mõningaid teadmisi omades oleks see tuletus lihtsam olnud...