Algoritmid ja andmestruktuurid

(MTAT.03.133, 4 AP)

Loengud:	E 12:15, aud. 405
Praktikumid:	K 14:15, aud. 405

koduleht:

`http://www.ut.ee/~peeter_l/teaching/a_ja_a06s`

(sinna kogunevad loengukiled)

Hinde saamiseks: praktikumiarvestus ja eksam.

Kirjandus:

Jüri Kiho. *Algoritmid ja andmestruktuurid (kolmas trükk)*. TÜ 2003.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

Algoritm on arvutussammude tegemise eeskiri, et mingi-test sisenditest välja arvutada mingid väljundid.

Turingi masinad, osaliselt rekursiivsed funktsioonid, λ-arvutus, jne. ei defineeri, mis on algoritm.

> Nad defineerivad ainult, mis on algoritmiliselt ar-vutatav.

Käesolevas kursuses uurime algoritme erinevate ülesannete lahendamiseks.

> . . . ning neid toetavaid andmestruktuure.

- Olgu meil mingi algoritm mingi ülesande lahendamiseks. Kuidas otsustada, kas see on ka hea algoritm?
- Või, kui meil on mitu algoritmi sama ülesande lahendamiseks, siis milline neist on parim?

- Olgu meil mingi algoritm mingi ülesande lahendamiseks. Kuidas otsustada, kas see on ka hea algoritm?
- Või, kui meil on mitu algoritmi sama ülesande lahendamiseks, siis milline neist on parim?
- Algoritm on hea, kui tema aja- ning ressursikasutus on piisavalt väike.
  - Ressursid — mälu, võrguliiklus, jne.
  - Ressursivajadus ei saa olla suurem kui ajavajadus.

(Räägime algoritmidest, mitte reaalsetest programmidest.)

- Algoritmi töö sõltub ilmselt tema sisendandmetest.
- Tööaeg sõltub ka.
- Tahaks mingeid garantiisid... mis on algoritmi maksimaalne tööaeg?
- Ilmselt ∞ — mida suurem sisend, seda rohkem kulub aega.

Käesolevas loengus uurime, kuidas hinnata algoritmi tehtavate sammude arvu sõltuvalt sisendi suurusest.

Antud algoritmi jaoks huvitab meid funktsioon T, mille korral $T(n)$ on selle algoritmi tehtavate sammude maksimaalne arv sisenditel, mille suurus on n. („ajaline keerukus halvimal juhul“) Mõnikord huvitab ka

- keskmine keerukus;
- amortiseeritud keerukus.

Näide: rea summa.

Sisend: Arvumassiiv a. Suurus n on massiivi pikkus.

Algoritm:

1	$s := 0$	täidetakse 1 kord
2	**for** $i := 1$ **to** n	täidetakse n korda
3	$\quad s := s + a_i$	täidetakse n korda
4	**return** s	täidetakse 1 kord

Kokku tehakse $(c_2 + c_3)n + c_1 + c_4$ sammu, kus c_i on i-ndal real tehtavate sammude arv.

Näide: mullisort.

Sisend: Massiiv a. Suurus n on massiivi pikkus.

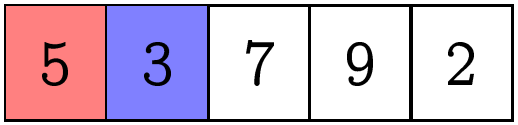
Algoritm:

```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```

Näide algoritmi tööst

Algoritm:

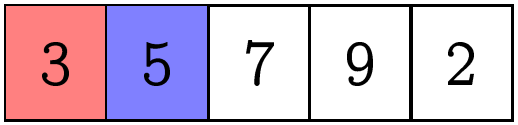
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

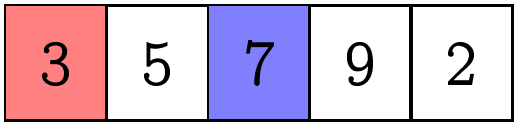
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

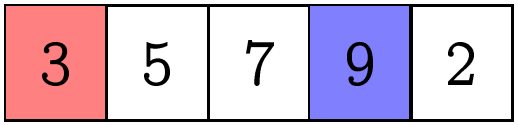
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

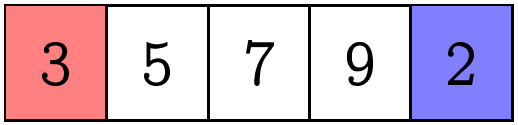
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

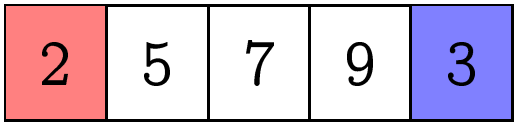
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

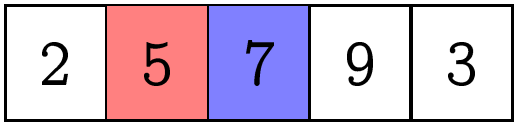
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

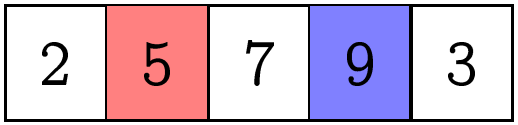
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

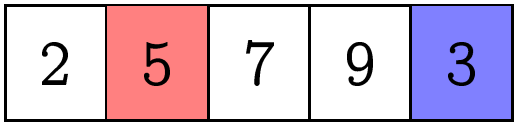
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

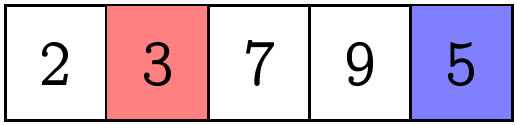
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

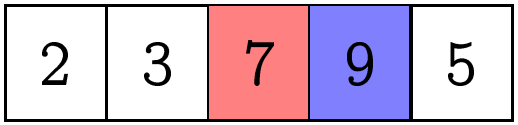
```
1   for i := 1 to n − 1
2       for j := i + 1 to n
3           if a_i > a_j then
4               a_i :=: a_j
5   return a
```



Näide algoritmi tööst

Algoritm:

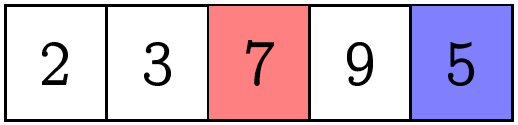
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

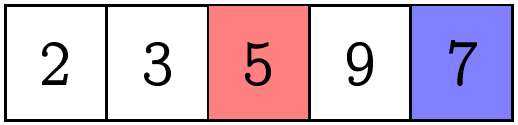
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

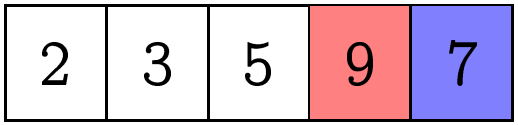
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

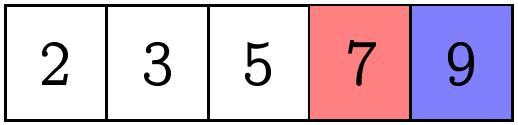
```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```



Näide algoritmi tööst

Algoritm:

```
1  for i := 1 to n − 1
2      for j := i + 1 to n
3          if a_i > a_j then
4              a_i :=: a_j
5  return a
```

2	3	5	7	9

Näide: mullisort.

Sisend: Massiiv a. Suurus n on massiivi pikkus.

Algoritm:

1	**for** $i := 1$ **to** $n-1$	$n-1$ korda
2	**for** $j := i+1$ **to** n	$n(n-1)/2$ korda
3	**if** $a_i > a_j$ **then**	$n(n-1)/2$ korda
4	$a_i :=: a_j$	ülimalt $n(n-1)/2$ korda
5	**return** a	1 kord

Kokku tehakse ülimalt $\frac{c_2+c_3+c_4}{2}n^2 + \frac{2c_1-c_2-c_3-c_4}{2}n + (c_5 - c_1)$ sammu ning vähemalt $\frac{c_2+c_3}{2}n^2 + \frac{2c_1-c_2-c_3}{2}n + (c_5 - c_1)$ sammu.

Rea summeerimisel tehakse ülimalt $pn+q$ sammu mingite konstantide p ja q korral ning mullisordil ülimalt $p'n^2 + q'n + r'$ sammu mingite konstantide p', q' ja r' korral.

Konstantide täpsed väärtused sõltuvad realisatsioonist, me ei püüa neid leida.

Kui n on küllalt suur, siis pn on enam-vähem sama suur kui $pn+q$ ning $p'n^2$ on enam-vähem sama suur kui $p'n^2 + q'n + r'$, s.t.

$$\lim_{n\to\infty} \frac{pn}{pn+q} = 1 \qquad \text{ja} \qquad \lim_{n\to\infty} \frac{p'n^2}{p'n^2+q'n+r'} = 1 \ .$$

Seega rea summeerimisel tehakse ülimalt umbes pn sammu ja mullisordil umbes $p'n^2$ sammu.

See, kui palju üks samm aega võtab, sõltub arvuti kiirusest.

> **Moore'i „seadus".** Transistoride tihedus kiibil (ja seega ka arvuti kiirus) kahekordistub iga 18 kuu järel.

Seega pole ka p ja p' konkreetsed väärtused eriti huvitavad.

Huvitav osa on rea summeerimise sammude arvu juures „n" ja mullisordi sammude arvu juures „n^2".

Olgu f ja g kaks naturaalarvuliste argumentidega ja positiivsete väärtustega funktsiooni.

f on $O(g)$, kui leiduvad $c > 0$ ja $n_0 \in \mathbb{N}$ nii, et iga $n \geqslant n_0$ korral $f(n) \leqslant cg(n)$.

$pn+q$ on $O(n)$ ja pn^2+qn+r on $O(n^2)$. Kordajaks c võib võtta suvalise arvu, mis on suurem kui p.

Lause. f on $O(g)$ parajasti siis, kui leidub $c > 0$ nii, et iga $n \in \mathbb{N}$ korral $f(n) \leqslant cg(n)$.

Tõestus. Vastavalt „on $O(\ldots)$“ definitsioonile leiduvad $c > 0$ ja $n_0 \in \mathbb{N}$ nii, et iga $n \geqslant n_0$ jaoks $f(n) \leqslant cg(n)$.

Iga $i \in \{1, 2, \ldots, n_0 - 1\}$ jaoks olgu $c_i = f(i)/g(i)$.

> Nõudsime, et f ja g oleksid positiivsed. Seega nulliga jagamist siin ei toimu.

Olgu $c' = \max\{c, c_1, c_2, \ldots, c_{n_0-1}\}$.

Siis iga $n \in \mathbb{N}$ jaoks $f(n)/g(n) \leqslant c'$, s.t. $f(n) \leqslant c'g(n)$. $\square$

f *on* $\Theta(g)$, kui f on $O(g)$ ja g on $O(f)$, s.t. leidub $c > 0$ ja $n_0 \in \mathbb{N}$ nii, et iga $n \geqslant n_0$ korral $\frac{g(n)}{c} \leqslant f(n) \leqslant cg(n)$.

f *on* $\Omega(g)$, kui g on $O(f)$.

Kui sama ülesande lahendamiseks on teada kaks erinevat algoritmi, millest ühe keerukushinnang on parem (väiksem) kui teise oma, siis on tõenäoline, et vähegi suuremate sisendite korral töötab esimene algoritm kiiremini.

Rea summeerimise algoritmi tööaeg halvimal juhul on $\Theta(n)$.

Mullisorteerimisalgoritmi tööaeg halvimal juhul on $\Theta(n^2)$.

n — ülesande suurus (massiivi pikkus).

Tööaeg — funktsioon $T(n)$, mida me alguses mainisime.

Üldiselt: kui mitterekursiivses algoritmis on k **for**-tsüklit üksteise sees ja neist kõigi iteratsioonide arv sõltub n-st lineaarselt, siis algoritmi keerukus on $\Theta(n^k)$.

Näide: kahe $n \times n$-ruutmaatriksi korrutamine.

Sisendid: maatriksid a ja b. Väljund: maatriks c.

Algoritm:

```
1  for i := 1 to n
2      for j := 1 to n
3          c_ij := 0
4          for k := 1 to n
5              c_ij := c_ij + a_ik · b_kj
```

Keerukus $\Theta(n^3)$.

Näide: kahendotsimine.

Sisend: Sorteeritud massiiv a ja element x. Suurus n on massiivi pikkus.

Algoritm: otsi$(a, x, 1, n)$, kus otsi(a, x, l, r) on

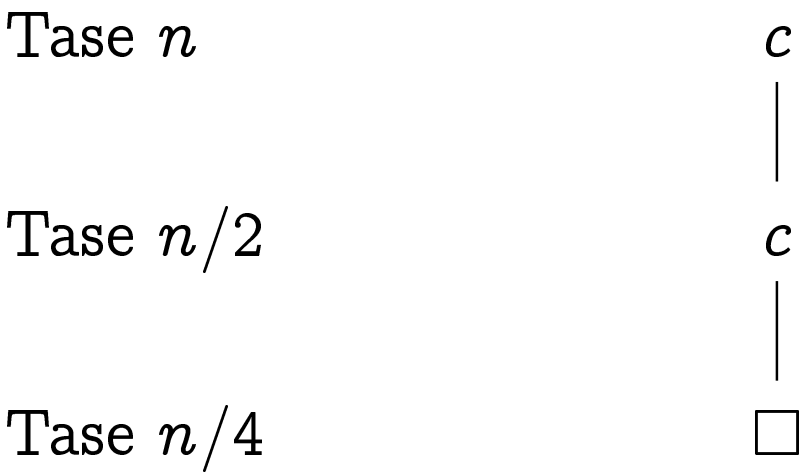
```
1  if l > r then
2      return FALSE
3  k := ⌊(l + r)/2⌋
4  if a_k = x then
5      return TRUE
6  else if x < a_k then
7      return otsi(a, x, l, k − 1)
8  else
9      return otsi(a, x, k + 1, r)
```

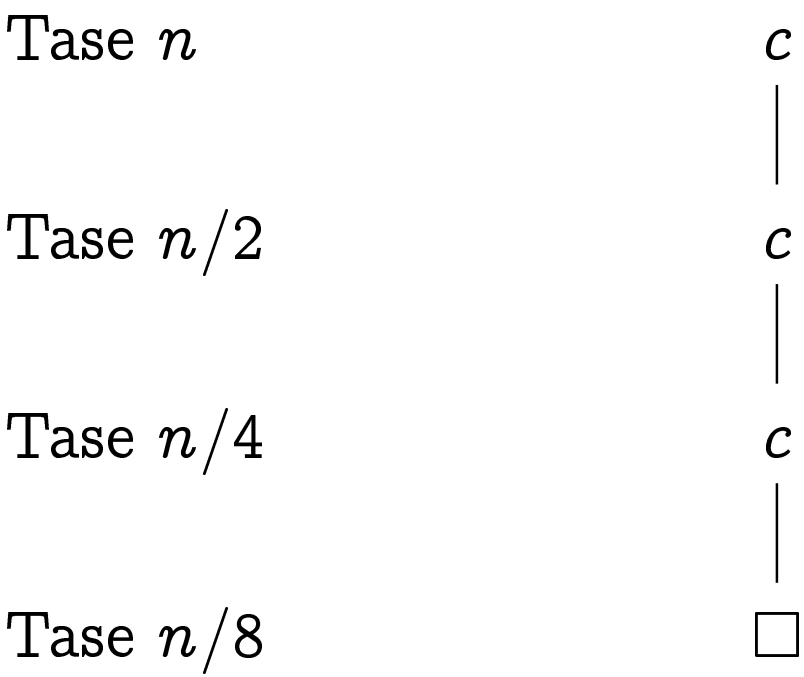
Rekursiivsest väljakutsest ülejääva osa tööaeg on piiratud mingi konstandiga c.

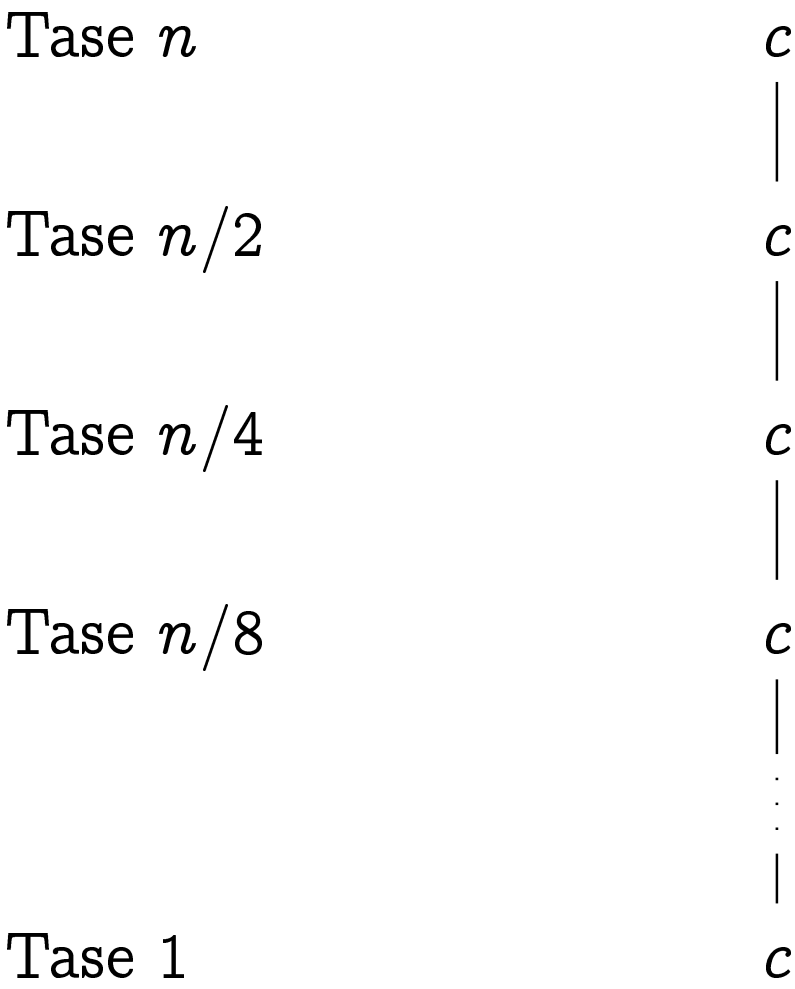
Rekursiivne väljakutse lahendab ülesande suurusega $\approx n/2$.

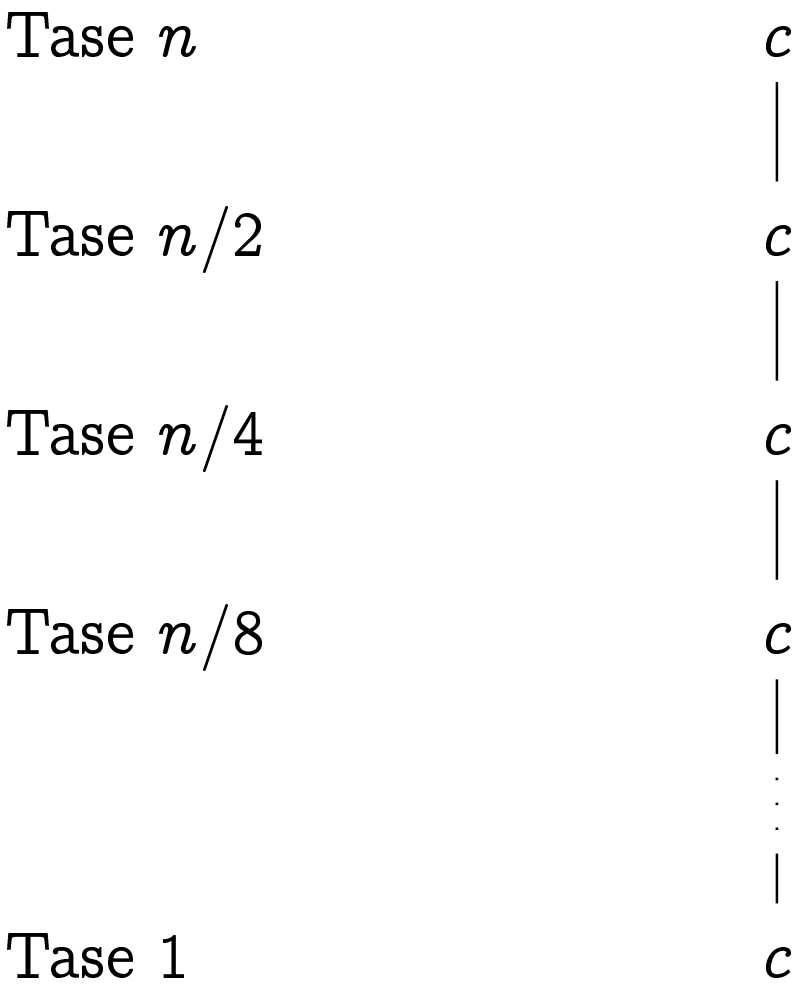
Tase n $\square$


Tase n
c
Tase $n/2$

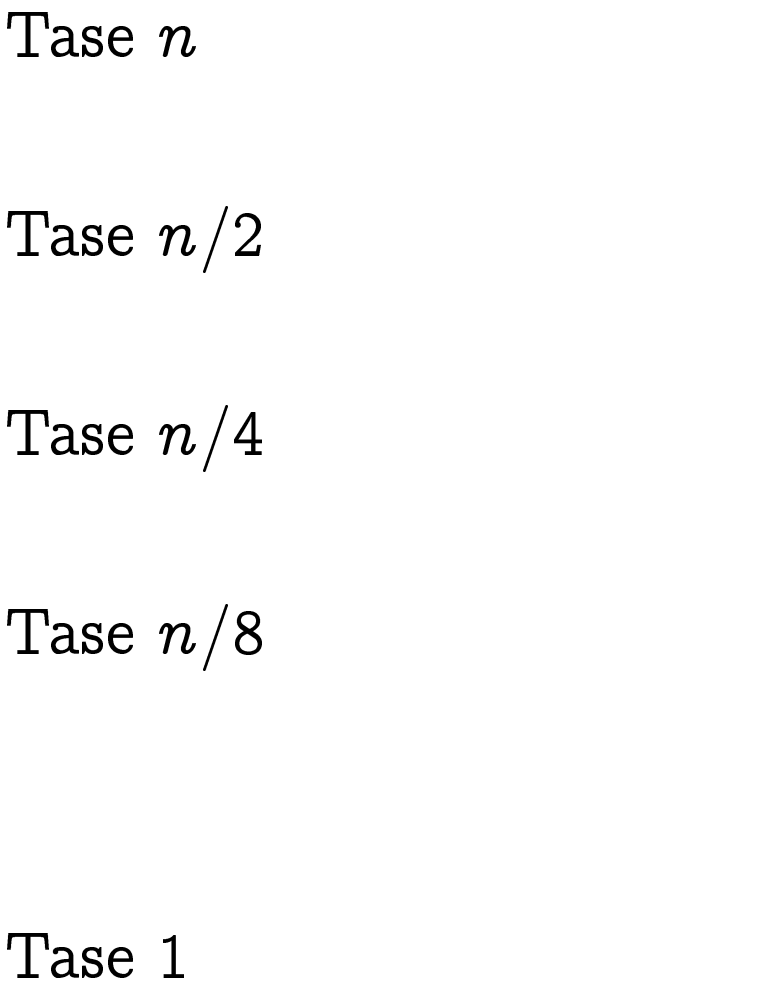

Tase n
c
Tase $n/2$
c
Tase $n/4$
$\square$


Tase n
c
Tase $n/2$
c
Tase $n/4$
c
Tase $n/8$
$\square$


Tase n
c
Tase $n/2$
c
Tase $n/4$
c
Tase $n/8$
c
Tase 1
c



Mitu taset on kokku?



Mitu taset on kokku? $\lceil \log_2 n \rceil$

Tööaeg: $\leqslant c \cdot \log_2 n$, mis on $\Theta(\log n)$

1. $a \cdot f$ on $O(f)$.
2. Kui f on $O(h)$ ja g on $O(h)$, siis $f + g$ on $O(h)$.
3. Kui f on $O(g)$ ja g on $O(h)$, siis f on $O(h)$.
4. Kui $r \leqslant s$, siis n^r on $O(n^s)$.
5. k-nda astme polünoom on $O(n^k)$.
6. Kui f_1 on $O(g_1)$ ja f_2 on $O(g_2)$, siis $f_1 f_2$ on $O(g_1 g_2)$.
7. n^k on $O(a^n)$, kui $a > 1$.
8. $\log_b n$ on $O(n^k)$, kui $b > 1$.
9. $\log_b n$ on $\Theta(\log_d n)$ iga $b, d > 1$ korral.
10. $\sum_{i=1}^{n} i^r$ on $\Theta(n^{r+1})$.

Tõestus. Eelmise lause kohaselt:

$$f \text{ on } O(g) \iff \exists c > 0\, \forall n \in \mathbb{N} : f(n) \leqslant cg(n)$$
$$\iff \exists c > 0\, \exists n_0 \in \mathbb{N}\, \forall n \geqslant n_0 : f(n) \leqslant cg(n) \ .$$

1. väide: $a \cdot f$ on $O(f)$. Konstandiks c võtame a.

2. väide: f on $O(h)$ ja g on $O(h) \Rightarrow f + g$ on $O(h)$. Olgu c_f ja c_g sellised konstandid, et $f(n) \leqslant c_f h(n)$ ja $g(n) \leqslant c_g h(n)$ iga $n \in \mathbb{N}$ jaoks. Siis $f(n) + g(n) \leqslant c_f h(n) + c_g h(n)$. Konstandiks c võtame $c_f + c_g$.

3. väide: f on $O(g)$ ja g on $O(h) \Rightarrow f$ on $O(h)$. Olgu c_f ja c_g sellised konstandid, et $f(n) \leqslant c_f g(n)$ ja $g(n) \leqslant c_g h(n)$. Siis $f(n) \leqslant c_f c_g h(n)$. Konstandiks c võtame $c_f c_g$.

4. väide: Kui $r \leqslant s$ siis n^r on $O(n^s)$. Olgu $c = 1$. Siis $n^s/n^r = n^{s-r}$. Kuna $n \geqslant 1$ ja $s - r \geqslant 0$, siis $n^{s-r} \geqslant 1$.

5. väide: k-nda astme polünoom on $O(n^k)$. Järeldub 1., 2. ja 4. väitest.

6. väide: f_1 on $O(g_1)$ ja f_2 on $O(g_2) \Rightarrow f_1 f_2$ on $O(g_1 g_2)$. Olgu c_1 ja c_2 sellised konstandid, et $f_1(n) \leqslant c_1 g_1(n)$ ja $f_2(n) \leqslant c_2 g_2(n)$ iga $n \in \mathbb{N}$ jaoks. Siis $f_1(n) f_2(n) \leqslant c_1 c_2 g_1(n) g_2(n)$. Konstandiks c võtame $c_1 c_2$.

Lemma. Kui f ja g on naturaalarvuliste argumentide ja positiivsete väärtustega funktsioonid ning $\lim\limits_{n\to\infty} \frac{f(n)}{g(n)}$ eksisteerib ja on väiksem kui ∞, siis f on $O(g)$.

Tõestus. Olgu $c' = \lim_{n\to\infty} \frac{f(n)}{g(n)}$ ja $c = c' + 1$. Vastavalt piirväärtuse definitsioonile leidub selline $n_0 \in \mathbb{N}$, nii et iga $n \geqslant n_0$ jaoks erineb $\frac{f(n)}{g(n)}$ suurusest c' vähem kui 1 võrra. Need c ja n_0 rahuldavadki „on $O(\ldots)$“ definitsiooni.

7. ja 8. väide: n^k on $O(a^n)$ ja $\log_b n$ on $O(n^k)$, kui $a, b > 1$. Järeldub sellest, et $\lim_{n\to\infty} \frac{n^k}{a^n} = \lim_{n\to\infty} \frac{\log_b n}{n^k} = 0$.

9. väide: $\log_b n$ on $\Theta(\log_d n)$ iga $b, d > 1$ korral. Kuna kehtib $\log_b n = \log_b d \cdot \log_d n$, siis võttes $c = \log_b d$ näeme, et $\log_b n$ on $O(\log_d n)$. Analoogiliselt $\log_d n$ on $O(\log_b n)$.

10. väide: $\sum_{i=1}^{n} i^r$ on $\Theta(n^{r+1})$.

$$\sum_{i=1}^{n} i^r \leqslant \sum_{i=1}^{n} n^r = n \cdot n^r = n^{r+1} \ .$$

Seega võttes $c = 1$ näeme, et $\sum_{i=1}^{n} i^r$ on $O(n^{r+1})$.

$$\sum_{i=1}^{n} i^r \geqslant \sum_{i=\lfloor n/2 \rfloor}^{n} i^r \geqslant \sum_{i=\lfloor n/2 \rfloor}^{n} \lfloor n/2 \rfloor^r =$$

$$\lceil n/2 \rceil \cdot \lfloor n/2 \rfloor^r \geqslant^{\text{kui } n \geqslant 2} (n/3)^{r+1} \ .$$

Seega võttes $n_0 = 2$ ja $c = 1/3^{r+1}$ näeme, et $\sum_{i=1}^{n} i^r$ on $\Omega(n^{r+1})$.

Kui f on funktsioon, siis kirjutis $O(f)$ tähistab ka kõigi selliste funktsioonide g hulka, mis on $O(f)$. $\Theta(f)$ ja $\Omega(f)$ omavad sarnaseid tähendusi.

Kirjutis $A(O(f_1), \ldots, O(f_k)) = A'(O(f'_1), \ldots, O(f'_l))$ tähendab, et

- iga $g_1 \in O(f_1), \ldots, g_k \in O(f_k)$ jaoks
- leiduvad $g'_1 \in O(f'_1), \ldots, g'_l \in O(f'_l)$ nii, et
- $A(g_1, \ldots, g_k) = A'(g'_1, \ldots, g'_l)$.

Näiteks kirjutis $f(n) = O(n^2)$ tähendab, et $f(n)$ on $O(n^2)$.

Kirjutis $f(n) + O(\log n) = O(n^2)$ tähendab, et $f + g$ on $O(n^2)$ iga g jaoks, mis on $O(\log n)$.

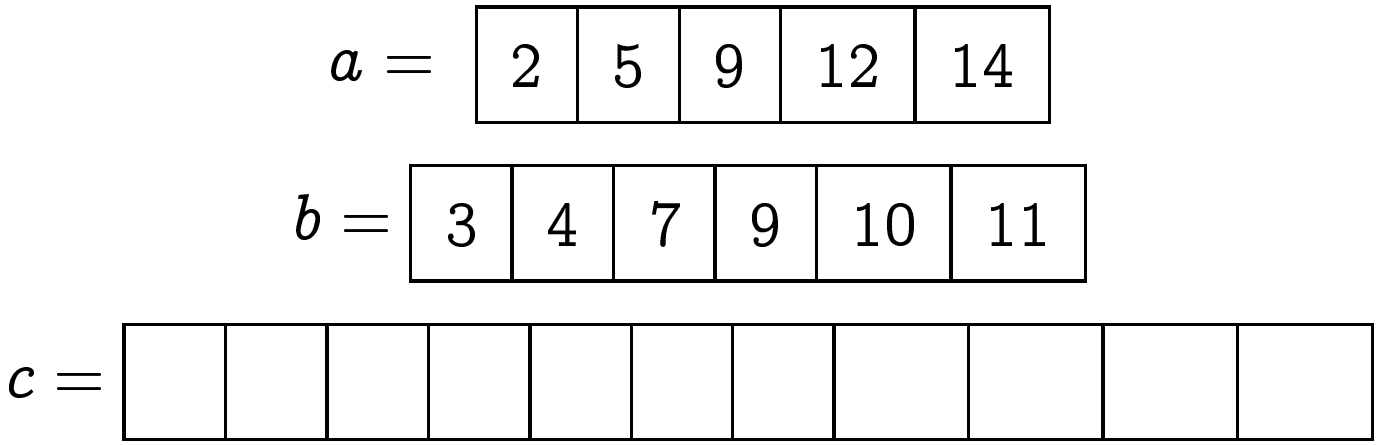
Kui f on $O(g)$, siis $f(n) = O(1)g(n)$.

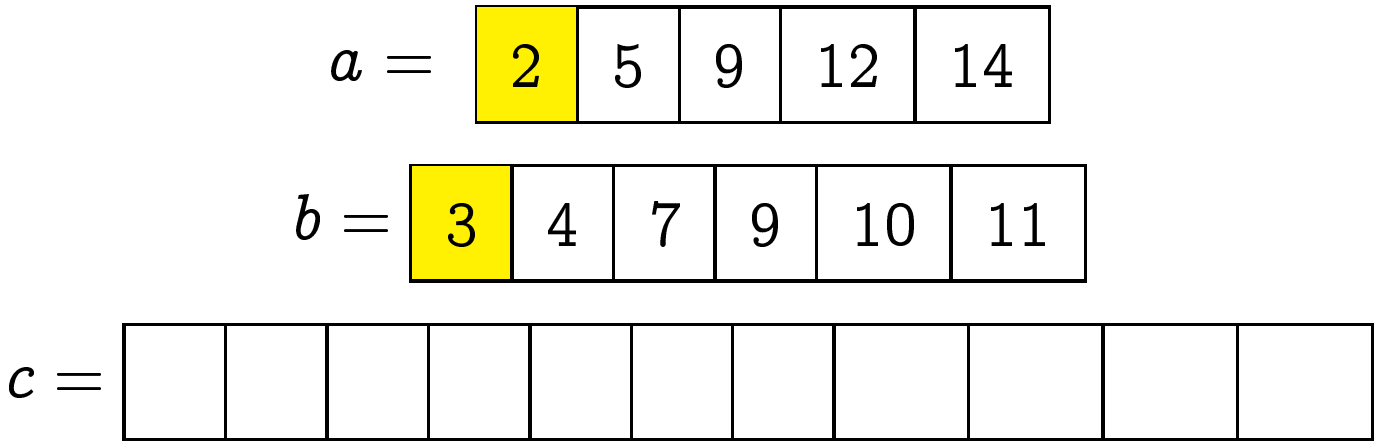
Näide: sorteerimine ühildusmeetodil

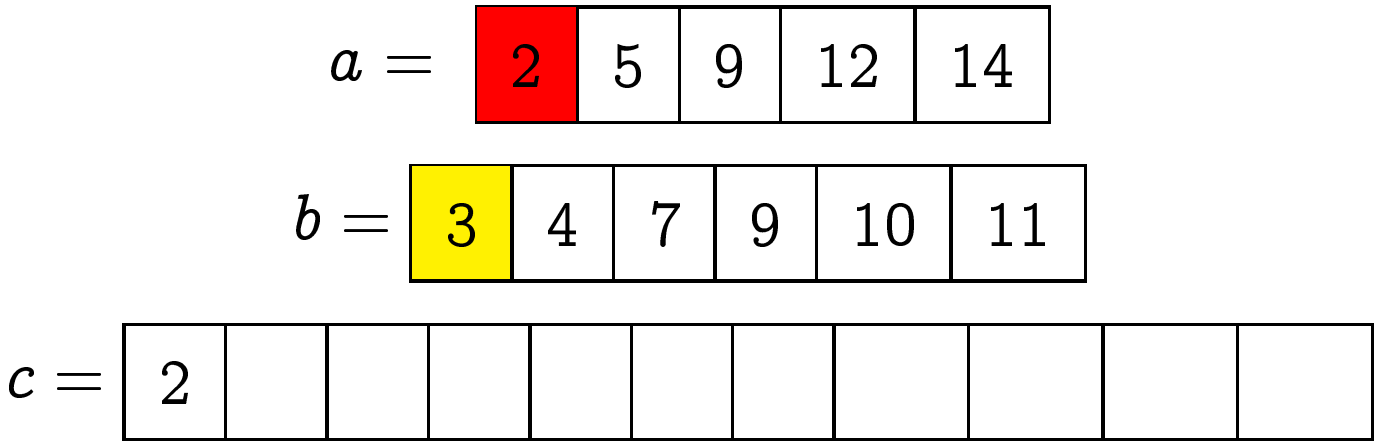
Algoritm massiivi a pikkusega n sorteerimiseks:

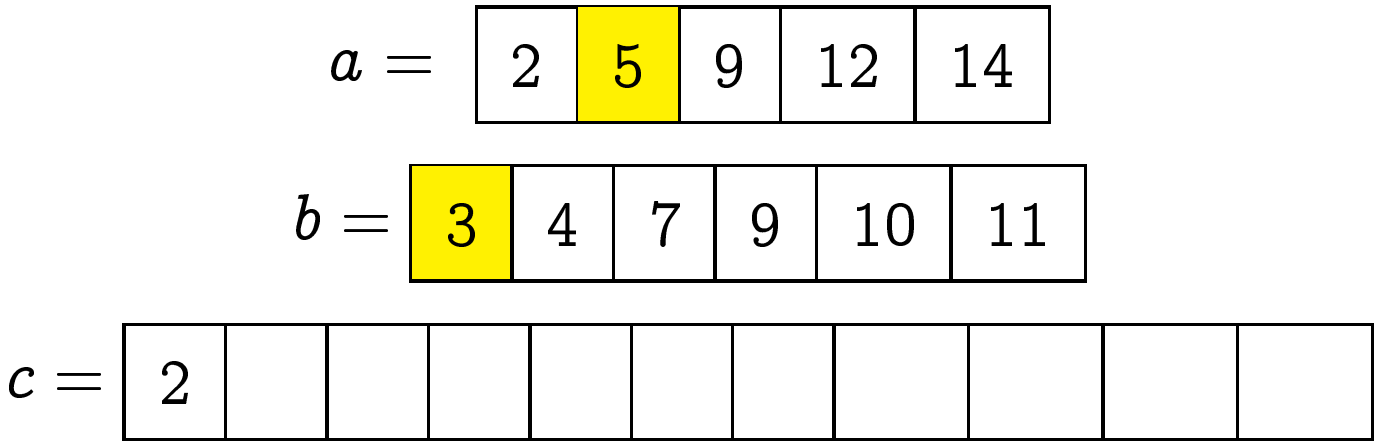
- Sorteeri massiivi a esimene pool $a_{1..\lfloor n/2 \rfloor}$.
- Sorteeri massiivi a teine pool $a_{\lfloor n/2 \rfloor + 1..n}$.
- *Ühilda* kaks sorteeritud poolt.

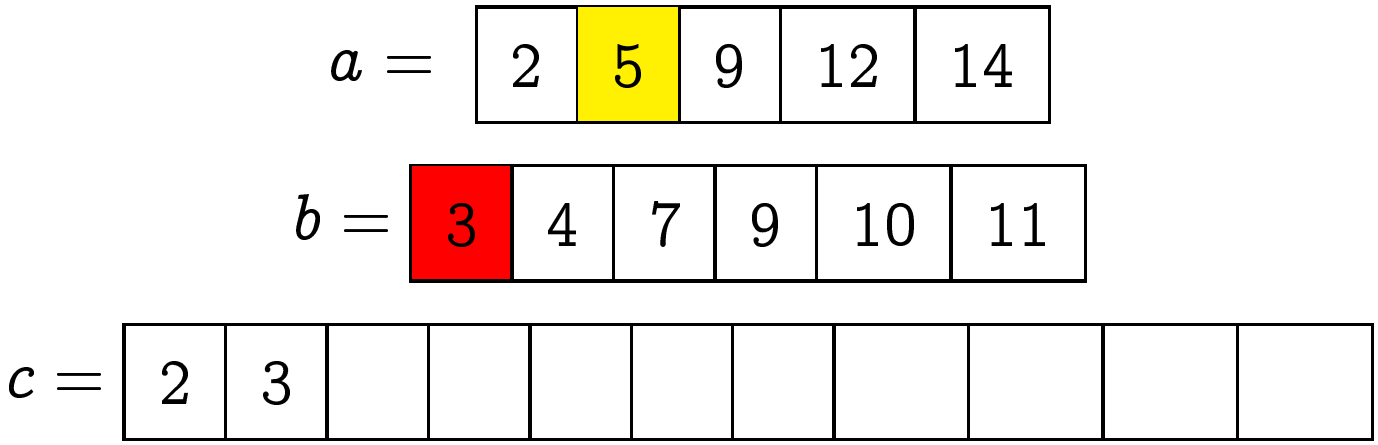
Alamülesanne: olgu antud sorteeritud massiivid a ja b, kumbki pikkusega $\approx n$. Kuidas neid ühildada? S.t., kuidas koostada nende massiivide sisust sorteeritud massiiv c pikkusega $2n$?

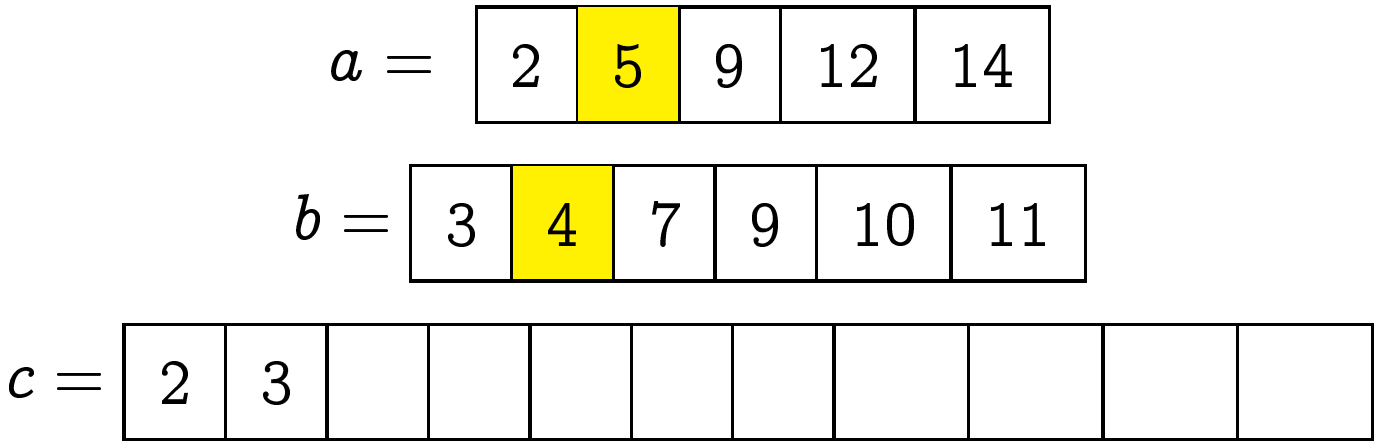

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =

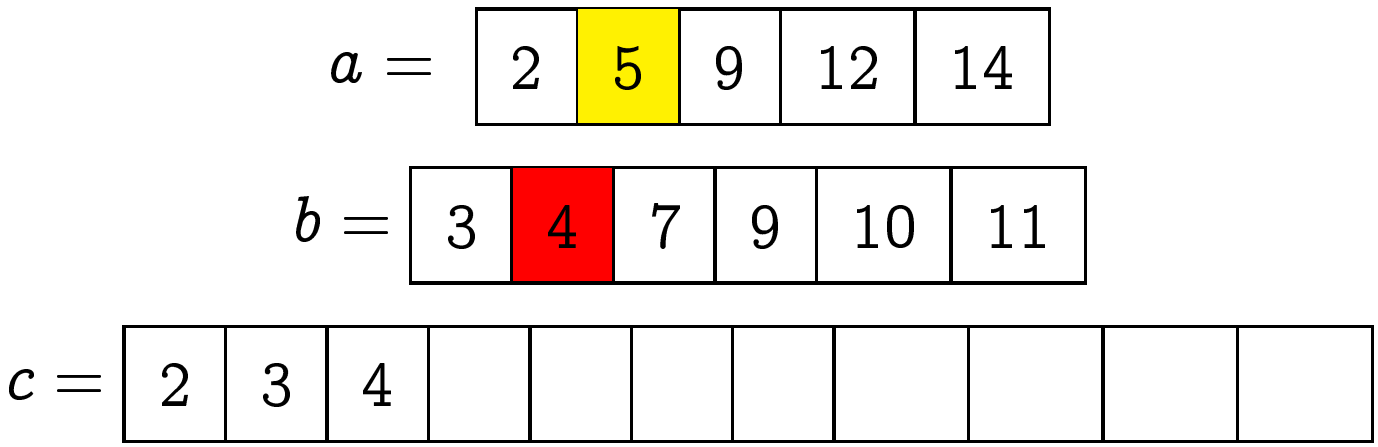

a = 2 5 9 12 14
b = 3 4 7 9 10 11
c =

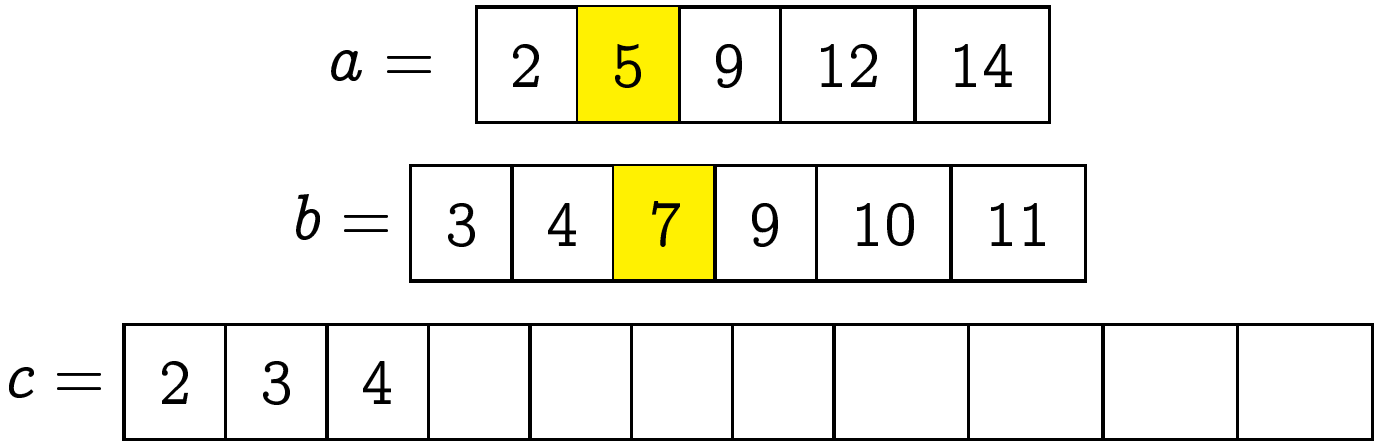

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2

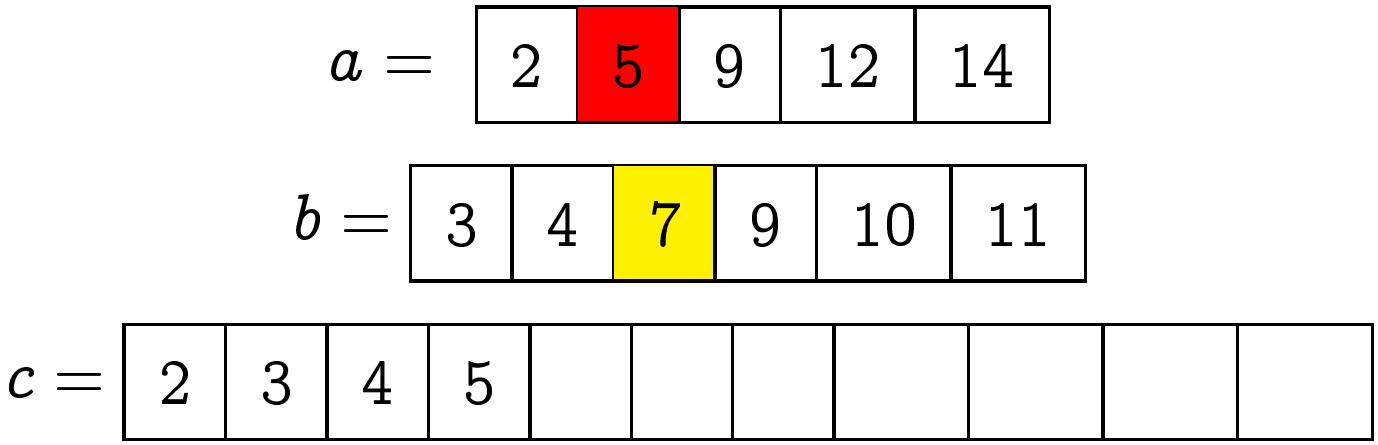

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2

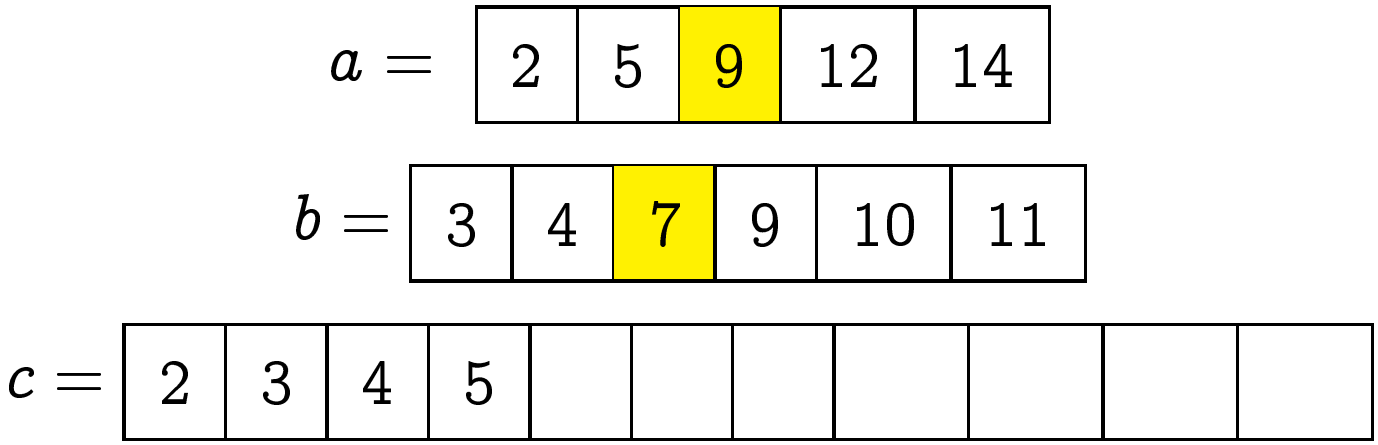

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3

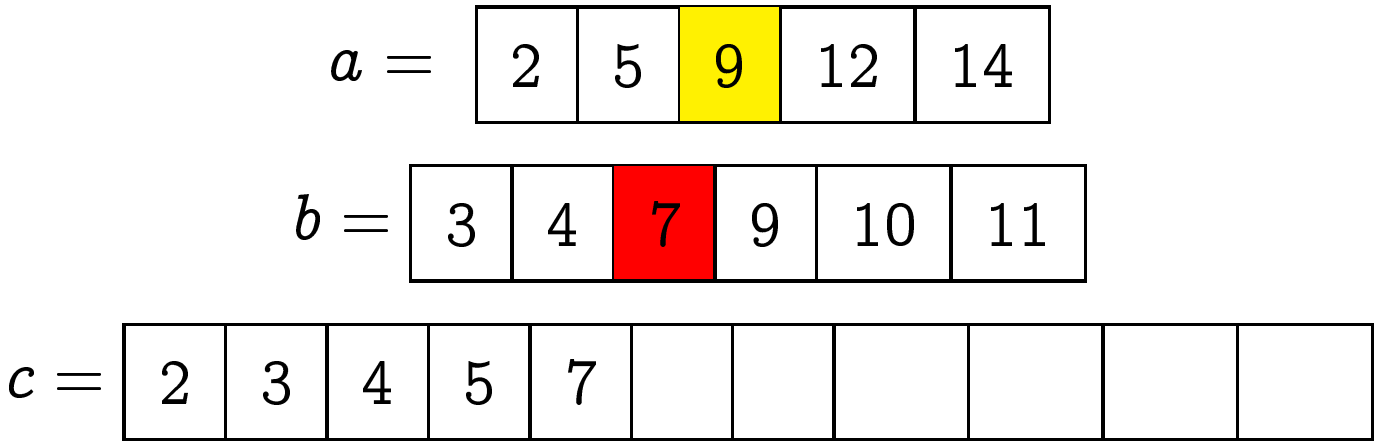

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3

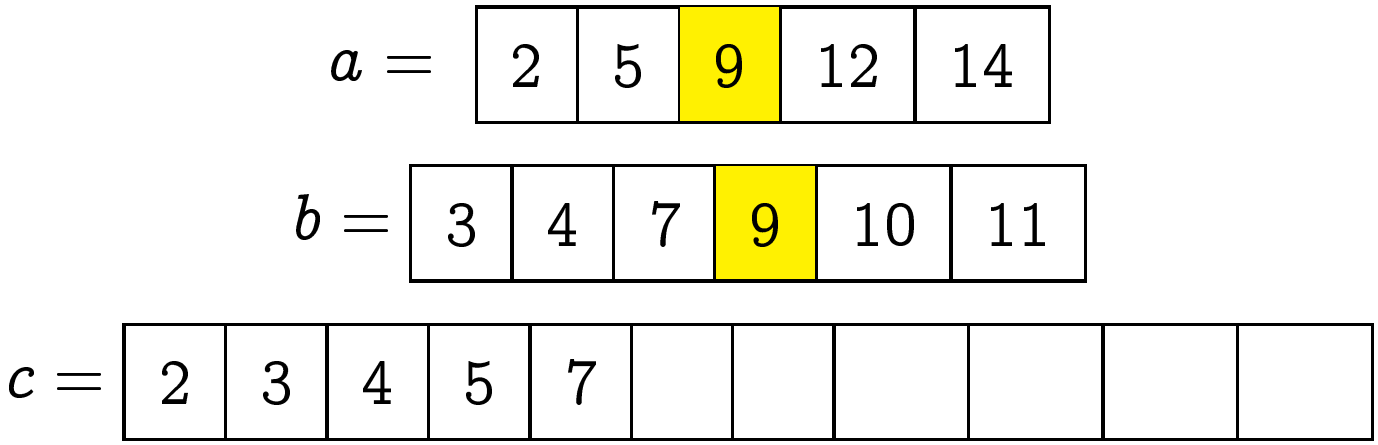

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4

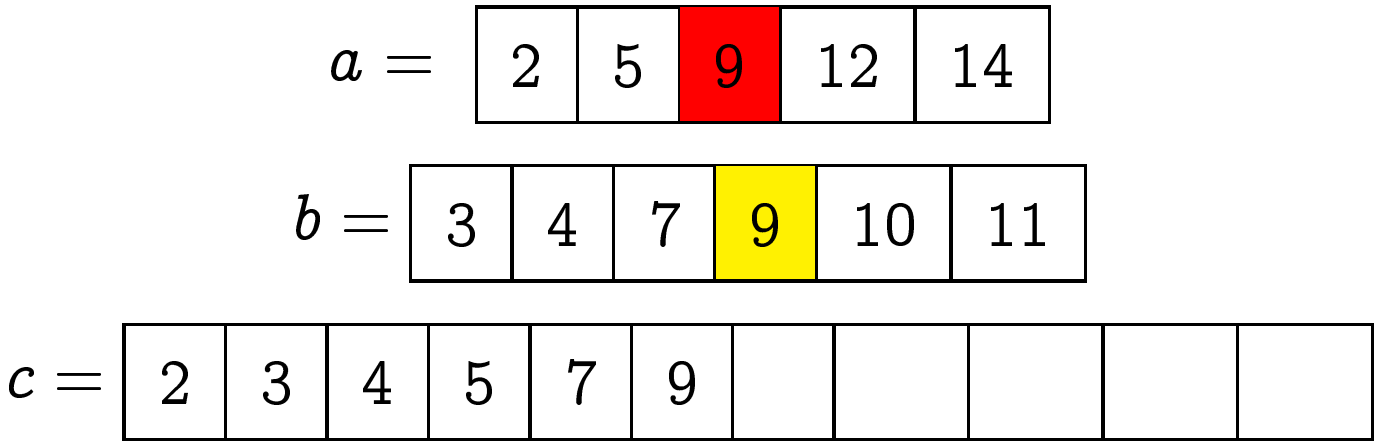

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4

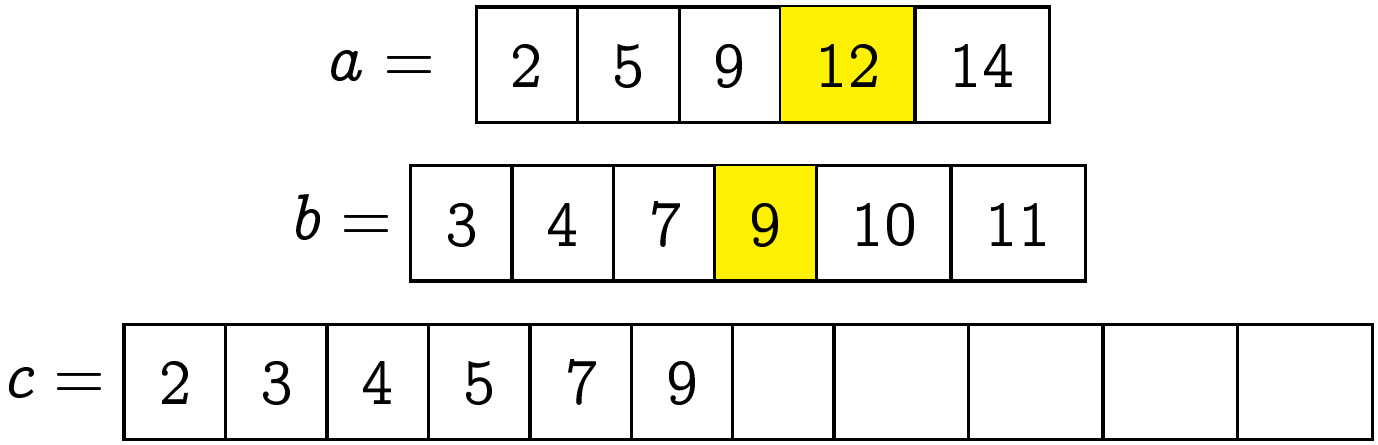

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5

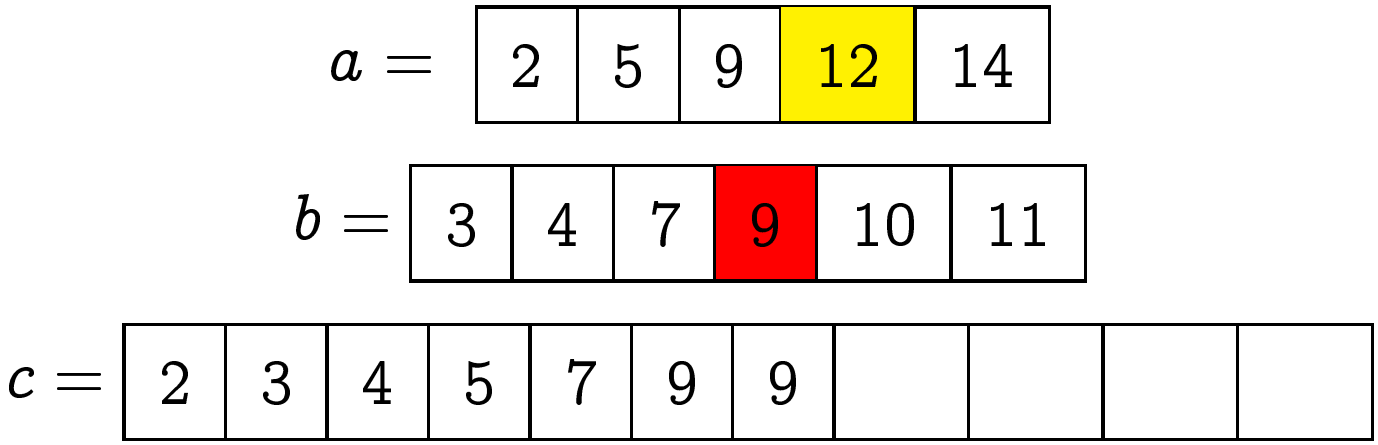

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5

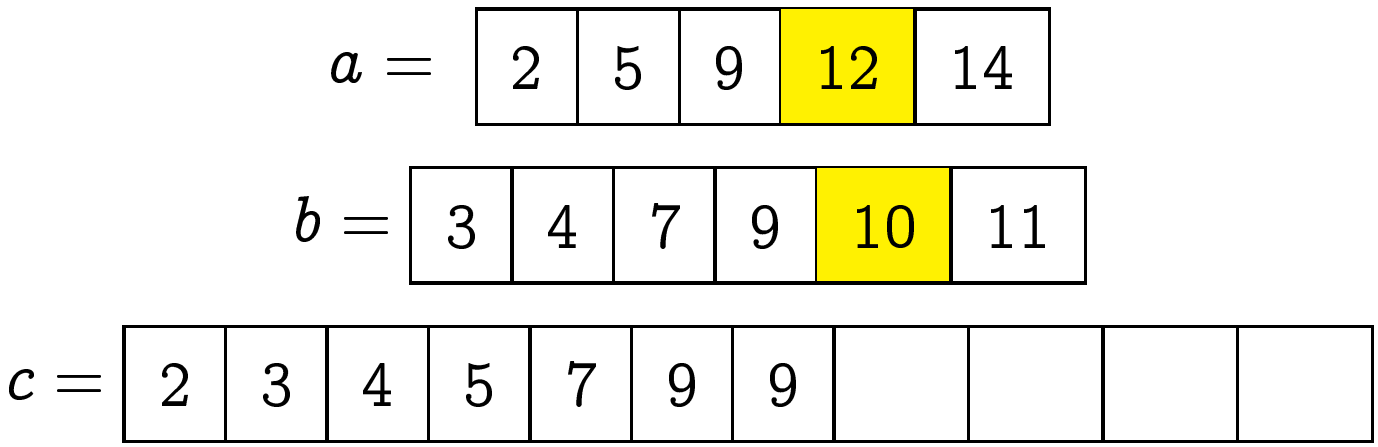

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7

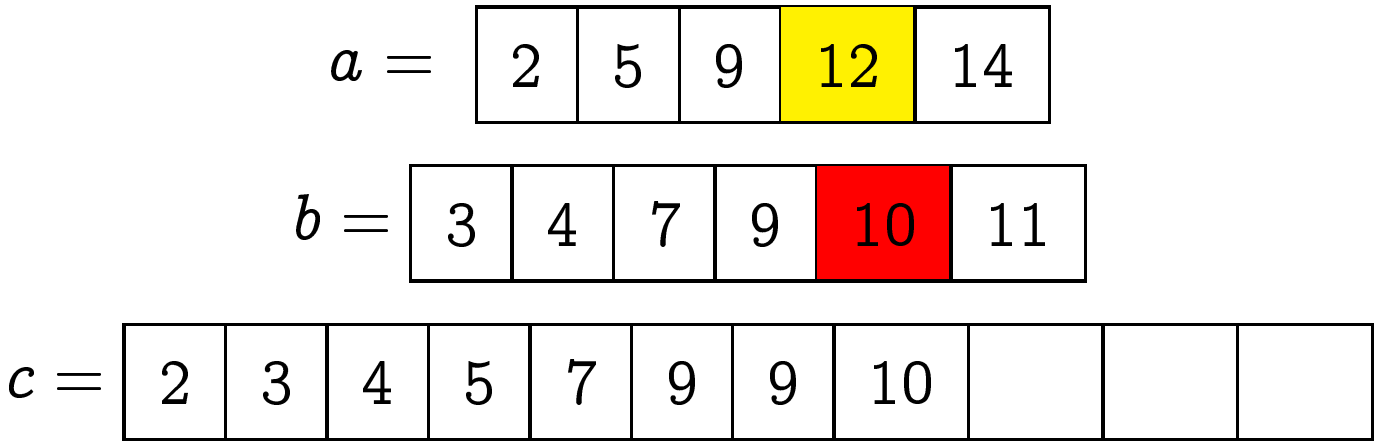

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7

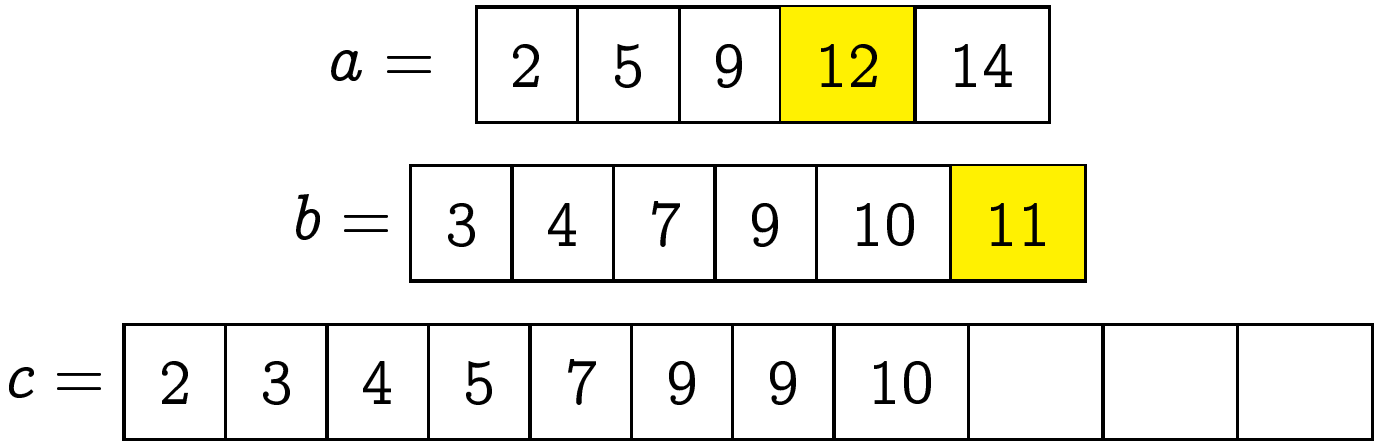

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7
9

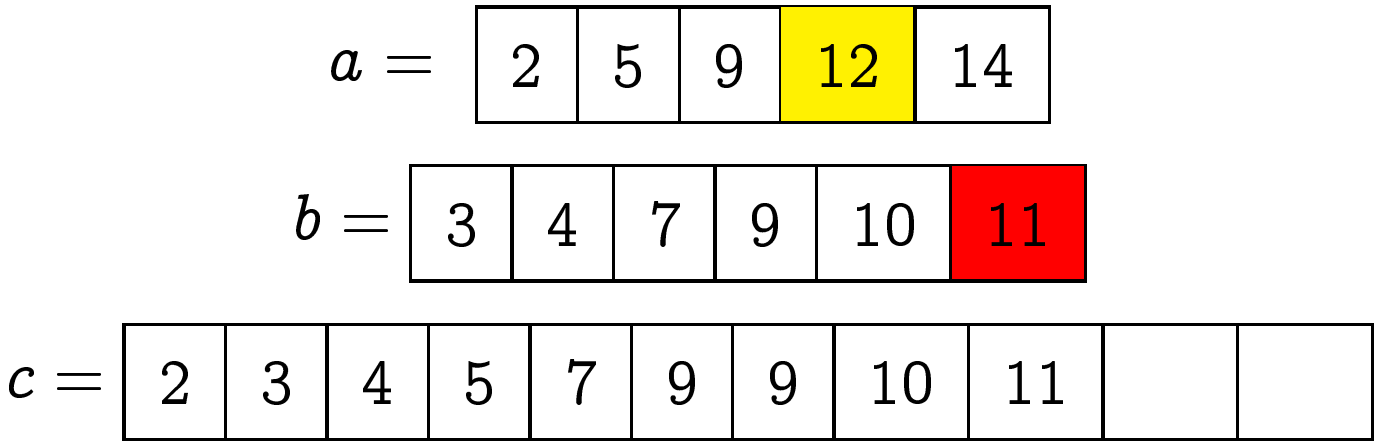

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7
9

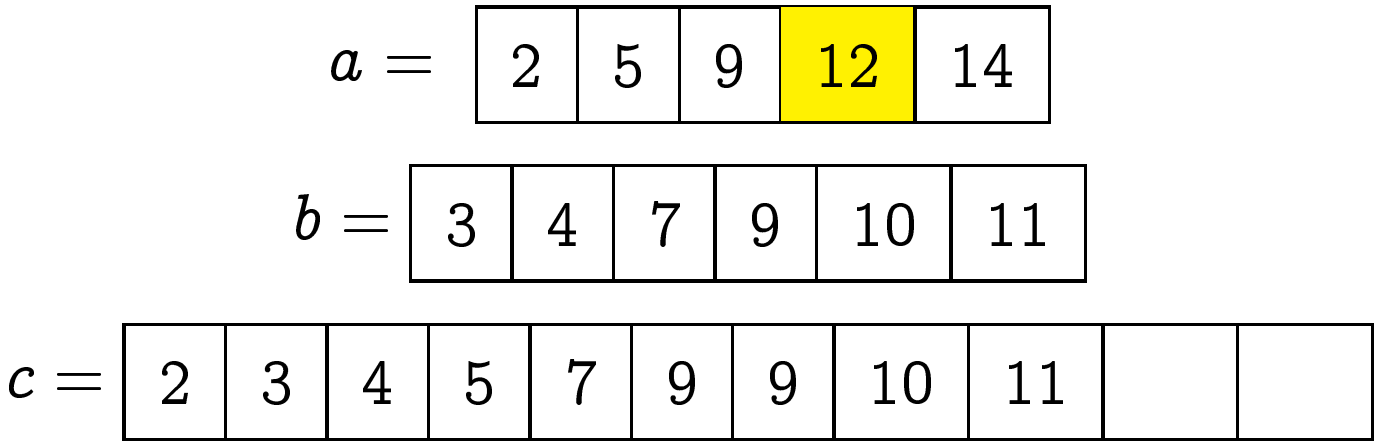

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7
9
9

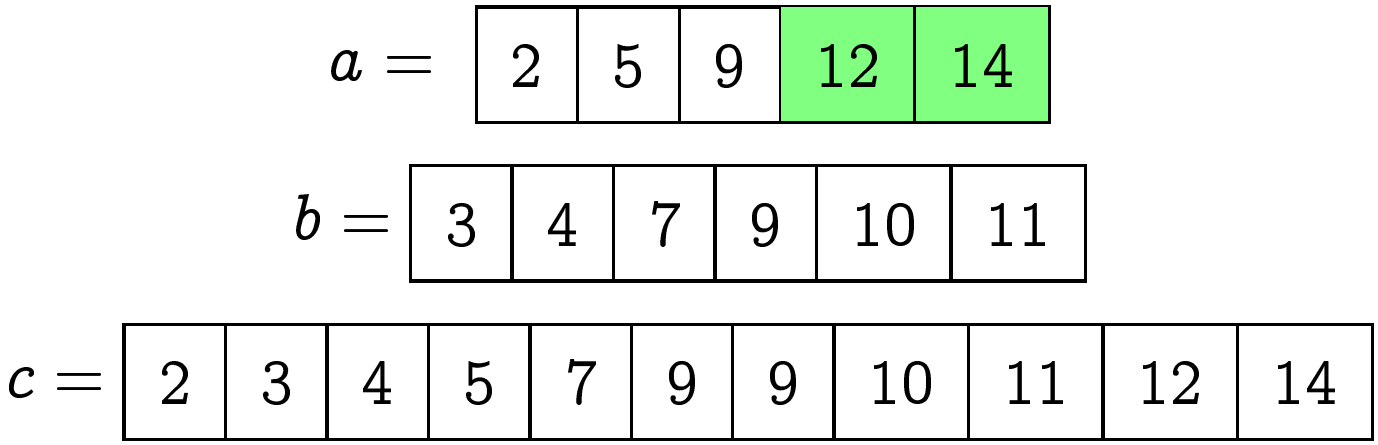

a =
2
5
9
12
14
b =
3
4
7
9
10
11
c =
2
3
4
5
7
9
9


a = 2 5 9 12 14
b = 3 4 7 9 10 11
c = 2 3 4 5 7 9 9 10


a =
2 5 9 12 14
b =
3 4 7 9 10 11
c =
2 3 4 5 7 9 9 10


a = 2 5 9 12 14
b = 3 4 7 9 10 11
c = 2 3 4 5 7 9 9 10 11


a = 2 5 9 12 14
b = 3 4 7 9 10 11
c = 2 3 4 5 7 9 9 10 11


a = 2 5 9 12 14
b = 3 4 7 9 10 11
c = 2 3 4 5 7 9 9 10 11 12 14

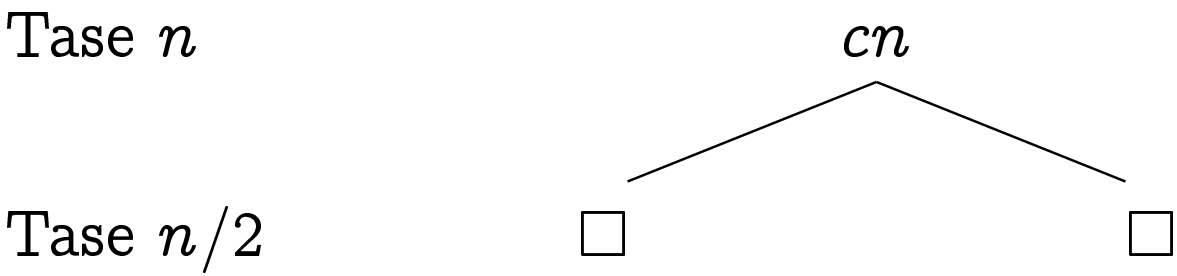
Näide: sorteerimine ühildusmeetodil

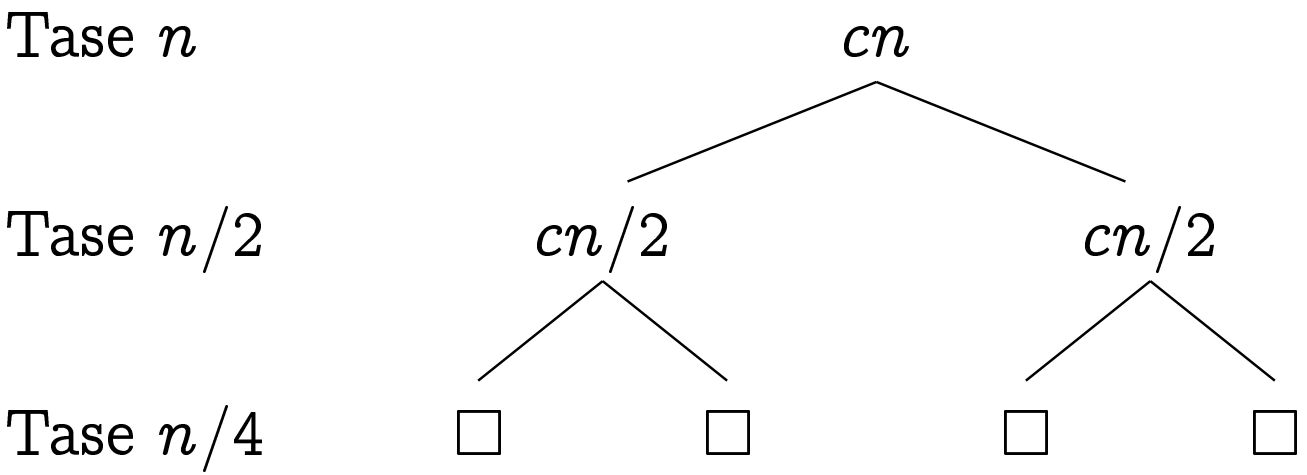
Algoritm massiivi a pikkusega n sorteerimiseks:

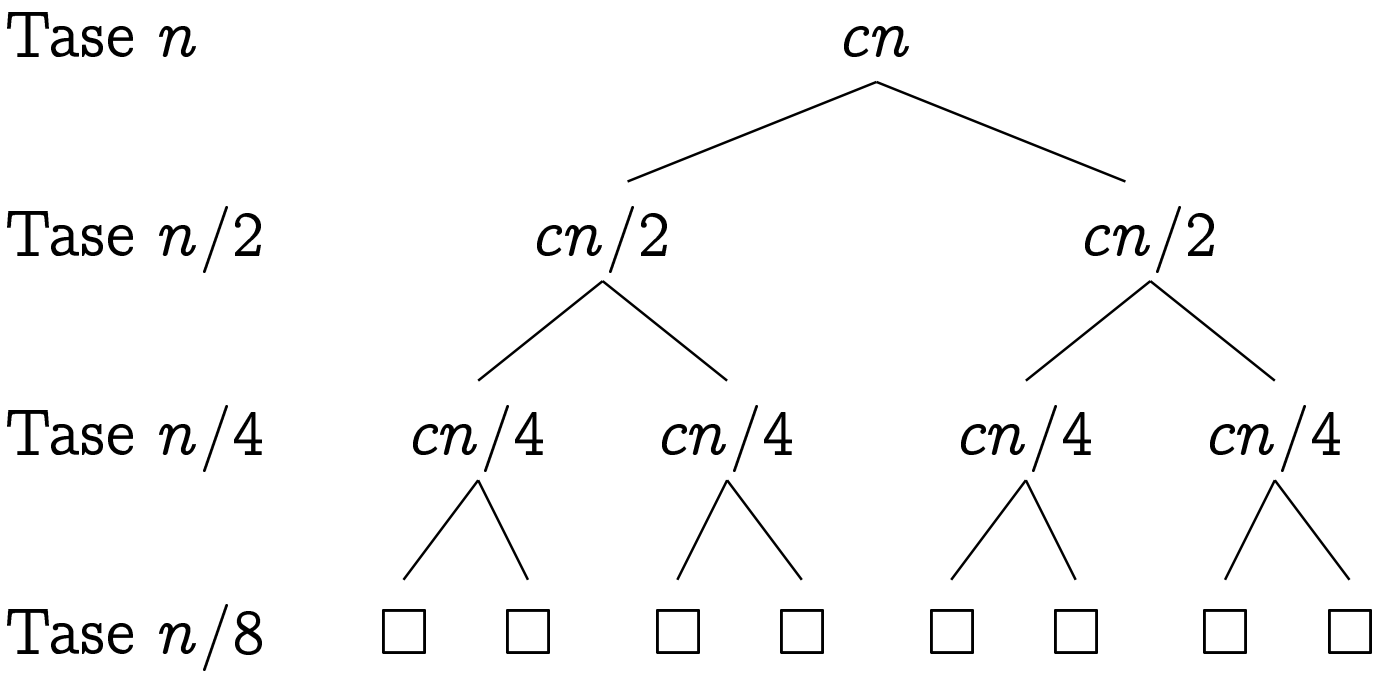
- Sorteeri massiivi a esimene pool $a_{1..\lfloor n/2 \rfloor}$.
- Sorteeri massiivi a teine pool $a_{\lfloor n/2 \rfloor + 1..n}$.
- *Ühilda* kaks sorteeritud poolt.
  - loo ajutine massiiv b pikkusega n;
  - ühilda massiivi a kaks poolt, salvesta tulemus massiivi b;
  - kopeeri massiiv b massiivi a.
  - Keerukus: $\Theta(n)$.

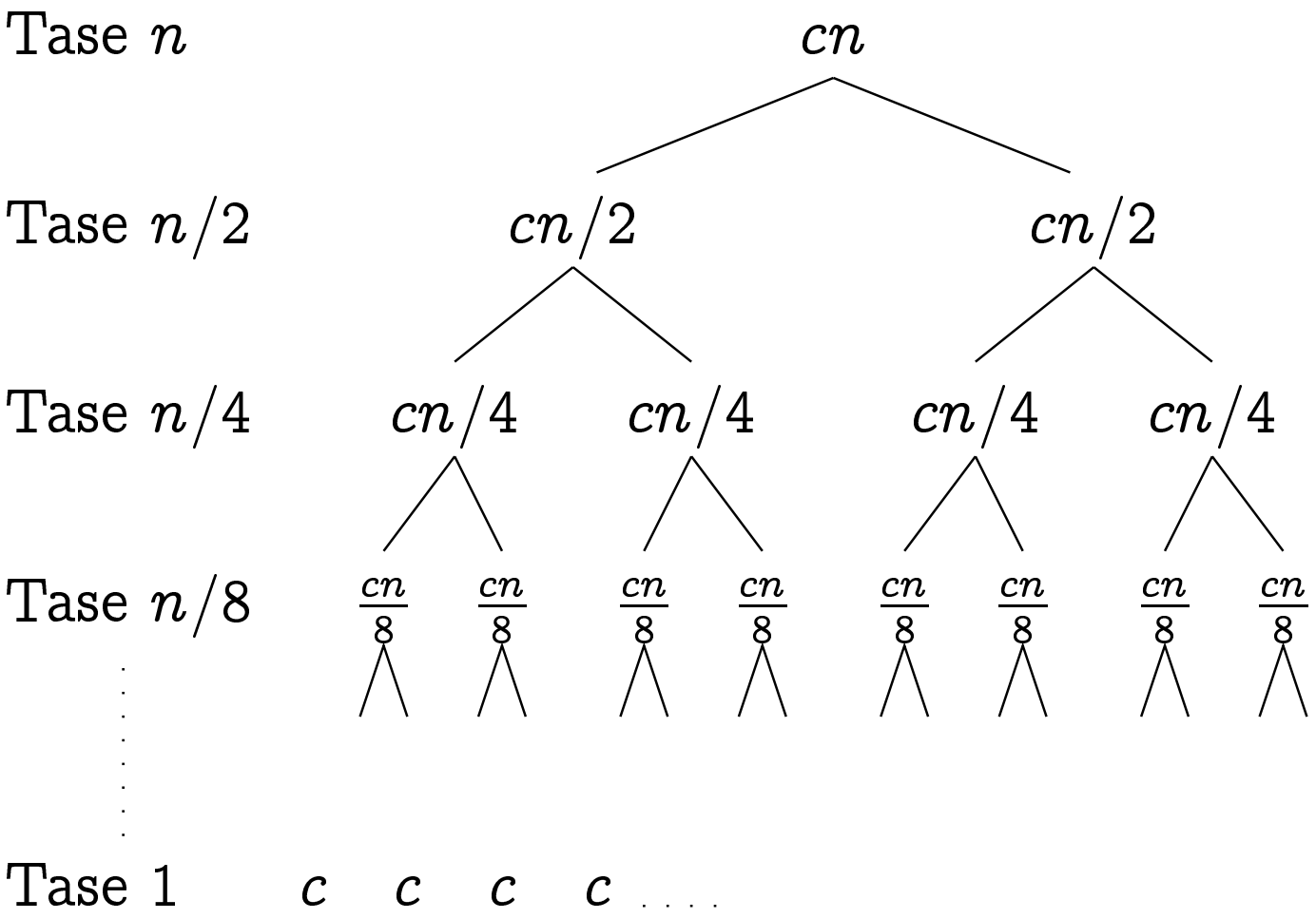
Mitterekursiivne osa vajab seega $\leqslant c \cdot n$ sammu.

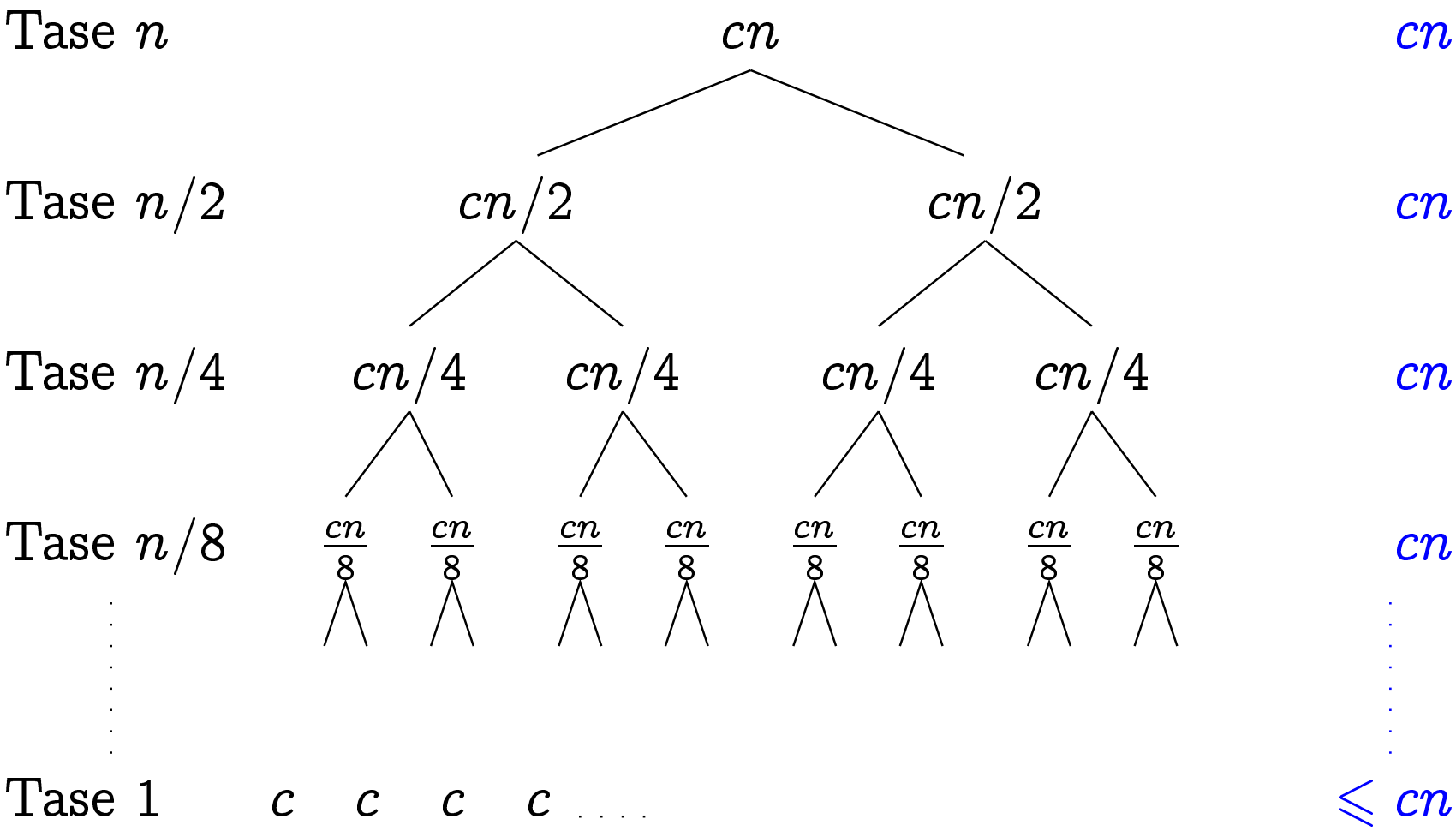
Tase n □


Tase n
cn
Tase $n/2$


Tase n
cn
Tase $n/2$
$cn/2$
$cn/2$
Tase $n/4$


Tase n
cn
Tase $n/2$
$cn/2$
$cn/2$
Tase $n/4$
$cn/4$
$cn/4$
$cn/4$
$cn/4$
Tase $n/8$


Tase n
cn
Tase $n/2$
$cn/2$
$cn/2$
Tase $n/4$
$cn/4$
$cn/4$
$cn/4$
$cn/4$
Tase $n/8$
$\frac{cn}{8}$
Tase 1
c c c c . . . .



Tasemete arv: $\log_2 n$

Tööaeg: $\leqslant cn \log_2 n = \Theta(n \log n)$

Näide: lausearvutusvalemi rahuldatavus.

Sisend: Lausearvutusvalem $\mathcal{A}$ (koosneb muutujatest ja loogilistest tehetest). Suuruseks n loeme muutujate arvu.

Algoritm: rahuldatav?($\mathcal{A}$), kus rahuldatav?($\mathcal{A}$) on

```
1  „lihtsusta“ A
2  if A ≡ true then
3      return TRUE
4  else if A ≡ false then
5      return FALSE
6  Olgu x üks A muutujatest
7  A_t := A_{x←true}; A_f := A_{x←false}
8  return rahuldatav?(A_t) ∨ rahuldatav?(A_f)
```

Olgu $T(n)$ programmi tööaeg (halvimal juhul) sisendil suurusega n. Siis $T(1) = c$ ja $T(n) \leqslant 2T(n-1) + p(n)$ mingi konstandi c ja polünoomi p jaoks.

[Pole päris korrektne, aga loeme siin, et valemi pikkus on polünomiaalne muutujate arvu suhtes.]

Seega $T(n) \leqslant 2^{n-1}c + \sum_{i=2}^{n} 2^{n-i}p(i) = O(np(n)2^n) = 2^{O(n)}$.

Teoreem. („põhiteoreem“, “*Master Theorem*”) Olgu $a \geqslant 1$, $b > 1$, olgu $f(n)$ mingi kasvav funktsioon. Tähistame $k = \log_b a$. Olgu $T(n)$ defineeritud järgmiselt:

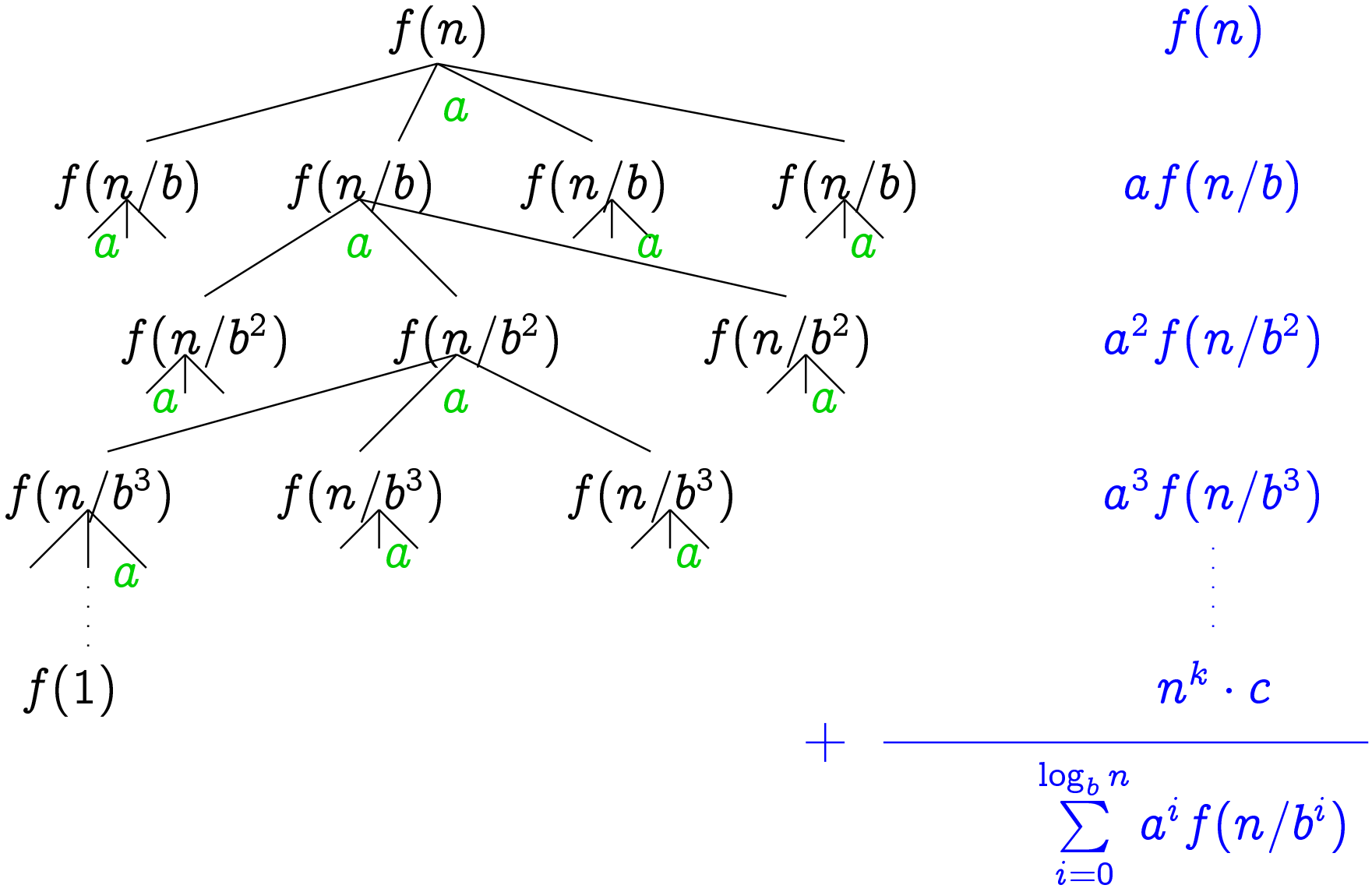
$$\begin{aligned} T(1) &= t \\ T(n) &= a \cdot T(n/b) + f(n) \ . \end{aligned}$$

Siin n/b võib olla kas $\lfloor n/b \rfloor$ või $\lceil n/b \rceil$.

- Kui $f(n)$ on $O(n^{k-\varepsilon})$, siis $T(n)$ on $\Theta(n^k)$.
- Kui $f(n)$ on $\Theta(n^k)$, siis $T(n)$ on $\Theta(n^k \log n)$.
- Kui $f(n)$ on $\Omega(n^{k+\varepsilon})$ ning $af(n/b) \leqslant cf(n)$ mingi $c < 1$ ja kõigi küllalt suurte arvude n jaoks, siis $T(n)$ on $\Theta(f(n))$.

Kahendotsimisel oli $T(n) = T(n/2) + f(n)$, kus $f(n)$ on $\Theta(1)$. Siin $a = 1$, $b = 2$ ja $k = \log_b a = 0$. Seega $f(n)$ on $\Theta(n^k)$. Teoreemi teise variandi järgi siis $T(n)$ on $\Theta(n^k \log n)$ ehk $\Theta(\log n)$.

Ühildusmeetodil sorteerimisel oli $T(n) = 2T(n/2) + f(n)$, kus $f(n)$ on $\Theta(n)$. Siin $a = 2$, $b = 2$ ja $k = \log_b a = 1$. Seega $f(n)$ on $\Theta(n^k)$. Teoreemi teise variandi järgi siis $T(n)$ on $\Theta(n \log n)$.


$f(n)$
a
$f(n/b)$ $f(n/b)$ $f(n/b)$ $f(n/b)$
a a a a
$f(n/b^2)$ $f(n/b^2)$ $f(n/b^2)$
a a a
$f(n/b^3)$ $f(n/b^3)$ $f(n/b^3)$
a a a
$f(1)$
$f(n)$
$af(n/b)$
$a^2 f(n/b^2)$
$a^3 f(n/b^3)$
$n^k \cdot c$
$+$
$\sum_{i=0}^{\log_b n} a^i f(n/b^i)$

Tõestus. Kui $n = b^i$, siis

$$\begin{aligned} T(n) &= f(n) + aT(n/b) = \\ &= f(n) + af(n/b) + a^2T(n/b^2) = \cdots = \\ &= f(n) + af(n/b) + \cdots + a^{i-1}f(n/b^{i-1}) + a^iT(1) = \\ &= T(1)n^k + \sum_{j=0}^{i-1} a^j f(n/b^j) = \Theta(n^k) + \sum_{j=0}^{i-1} a^j f(n/b^j), \end{aligned}$$

sest $a^i = a^{\log_b n} = n^{\log_b a} = n^k$.

Kirjutades $\Theta(n^k)$ oleme eeldanud, et T on ainult b astmetel defineeritud.

Kui $n = b^i$ ja $f(n) = O(n^{k-\varepsilon})$, siis

$$\sum_{j=0}^{i-1} a^j f(n/b^j) = O\left(\sum_{j=0}^{i-1} a^j((n/b^j)^{k-\varepsilon})\right)$$

$$\sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j}\right)^{k-\varepsilon} = n^{k-\varepsilon} \sum_{j=0}^{i-1} \left(\frac{ab^\epsilon}{b^k}\right)^j =$$

$$= n^{k-\varepsilon} \sum_{j=0}^{i-1} b^{\varepsilon j} = n^{k-\varepsilon} \frac{b^{\varepsilon i} - 1}{b^\varepsilon - 1} =$$

$$= n^{k-\varepsilon} \frac{n^\varepsilon - 1}{b^\varepsilon - 1} = O(n^k) \ .$$

Põhiosa „raskusest“ on puu lehtedes.

Kui $n = b^i$ ja $f(n) = \Theta(n^k)$, siis

$$\sum_{j=0}^{i-1} a^j f(n/b^j) = \Theta\left(\sum_{j=0}^{i-1} a^j((n/b^j)^k)\right)$$

$$\sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j}\right)^k = n^k \sum_{j=0}^{i-1} \left(\frac{a}{b^k}\right)^j =$$

$$= n^k \sum_{j=0}^{i-1} 1 =$$

$$= n^k \cdot i = n^k \log_b n \ .$$

Puu kõik tasemed on sama „rasked“.

Kui $n = b^i$ ja $af(n/b) \leqslant cf(n)$, kus $c < 1$, siis

$$\sum_{j=0}^{i-1} a^j f(n/b^j) \leqslant \sum_{j=0}^{i-1} c^j f(n) \leqslant$$

$$\leqslant f(n) \sum_{j=0}^{\infty} c^j = f(n) \frac{1}{1-c} = O(f(n)) \ .$$

Kuna see summa sisaldab liiget $f(n)$ ning kõik liikmed on mittenegatiivsed, siis on ta ka $\Omega(f(n))$, kokku seega $\Theta(f(n))$.

Põhiosa „raskusest“ on puu juures.

Oleme tõestanud teoreemi juhuks, kui $T(n)$ on defineeritud ainult b astmetel.

Olgu n suvaline naturaalarv. Kui $T(n) = aT(\lceil n/b \rceil) + f(n)$, siis on lihtne näidata, et $T(n)$ on $\Omega(\dots)$ (tingimustel, mille kohta teoreem väidab, et $T(n)$ on $\Theta(\dots)$). Meil tuleb veel näidata, et $T(n)$ on $O(\dots)$.

Olgu $n_0, n_1, \dots$ defineeritud järgmiselt:

$$n_0 = n$$

$$n_{j+1} = \lceil n_j / b \rceil \ .$$

Kuna $n_{j+1} \leqslant \frac{n_j}{b} + 1$, siis $n_j \leqslant \frac{n}{b^j} + \sum_{s=0}^{j-1} \frac{1}{b^s} \leqslant \frac{n}{b^j} + \frac{b}{b-1}$.

Olgu $i = \lfloor \log_b n \rfloor$, siis $n_i = O(1)$.

$$T(n) = a^i T(n_i) + \sum_{j=0}^{i-1} a^j f(n_j) \leqslant \Theta(n^k) + \sum_{j=0}^{i-1} a^j f\left(\frac{n}{b^j} + \frac{b}{b-1}\right)$$

Kui $f(n) = O(n^{k-\varepsilon})$, siis

$$\sum_{j=0}^{i-1} a^j f\left(\frac{n}{b^j} + \frac{b}{b-1}\right) = O\left(\sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j} + \frac{b}{b-1}\right)^{k-\varepsilon}\right)$$

$$\begin{aligned}\sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j} + \frac{b}{b-1}\right)^{k-\varepsilon} &= \sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j}\right)^{k-\varepsilon} \left(1 + \frac{b^j}{n} \cdot \frac{b}{b-1}\right)^{k-\varepsilon} \leqslant \\ &\leqslant \sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j}\right)^{k-\varepsilon} \left(1 + \frac{b}{b-1}\right)^{k-\varepsilon} = \\ &= O(1) \sum_{j=0}^{i-1} a^j \left(\frac{n}{b^j}\right)^{k-\varepsilon} = O(n^k)\end{aligned}$$

Kui $f(n) = \Theta(n^k)$, siis analoogiliselt

$$\sum_{j=0}^{i-1} a^j f\left(\frac{n}{b_j} + \frac{b}{b-1}\right) = O(1) \sum_{j=0}^{i-1} a^j \left(\frac{n}{b_j}\right)^k = O(n^k \log n) \ .$$

Kui $af(\lceil n/b \rceil) \leqslant cf(n)$, kus $c < 1$, siis $a^j f(n_j) \leqslant c^j f(n)$ ning

$$\sum_{j=0}^{i-1} a^j f(n_j) \leqslant \sum_{j=0}^{i-1} c^j f(n) = O(f(n)) \ .$$

- Asümptootikuid kasutades tehakse eeldus, et probleemi suurusega võib lõpmatusse minna.
- Arvutiarhitektuuri ebaregulaarsustest tingituna võib mõne algoritmi reaalne tööaeg (arvutile jõukohastel probleemisuurustel) olla sarnasem mõne teise funktsiooniga kui leitud O-hinnang.
- Käesolevas kursuses uurime ikkagi asümptootikuid.