Programmeerimine

15. loeng

Täna loengus

- Algoritmid ja keerukus
- Lineaarne vs. kahendotsimine
- Sorteerimine
  - Valikumeetodil sorteerimine
  - Ühildusmeetodil sorteerimine

Algoritmid ja keerukus

- Konkreetse programmi efektiivsus sõltub:
  - programmis kasutatud algoritmidest;
  - sisendandmete suurusest;
  - aga näiteks ka konkreetse masina riistvarast;
  - ...
- Põhiidee: ignoreerime masinast sõltuvaid konstante ja uurime asümptootilist keerukust:
  - millise funktsiooni $T(n)$ järgi kasvab algoritmis tehtavate sammude arv, kui sisendi suurus n kasvab lõpmatuks.
- Asümtootilist keerukust tähistame $\mathcal{O}(T(n))$.
- Keerukuse tähistamisel piisab kui näidata ainult kõige "kõrgemat järku" term.
  - Näiteks: $2n^3 + 5n^2 + 12n + 4 = \mathcal{O}(n^3)$.

Algoritmid ja keerukus

Ülesande lahendamise aeg

Kui palju aega kulutab kaasaegne arvuti aega ülesande lahendamiseks sõltuvalt algandmete hulgast?

	10	20	30	40	50	60
n	0,00001	0,00002	0,00003	0,00004	0,00005	0,00006
n^2	0,0001	0,0004	0,0009	0,0016	0,0025	0,0036
n^3	0,001	0,008	0,027	0,064	0,125	0,216
n^5	0,1	3,2	24,3	$1{,}7m$	$5{,}2m$	$13m$
2^n	0,001	1,0	$17{,}9m$	$12{,}7p$	$35{,}3a$	$366saj$
3^n	0,059	$58m$	$6{,}5a$	$3855saj$	$2 * 10^8 saj$	$10^{13} saj$

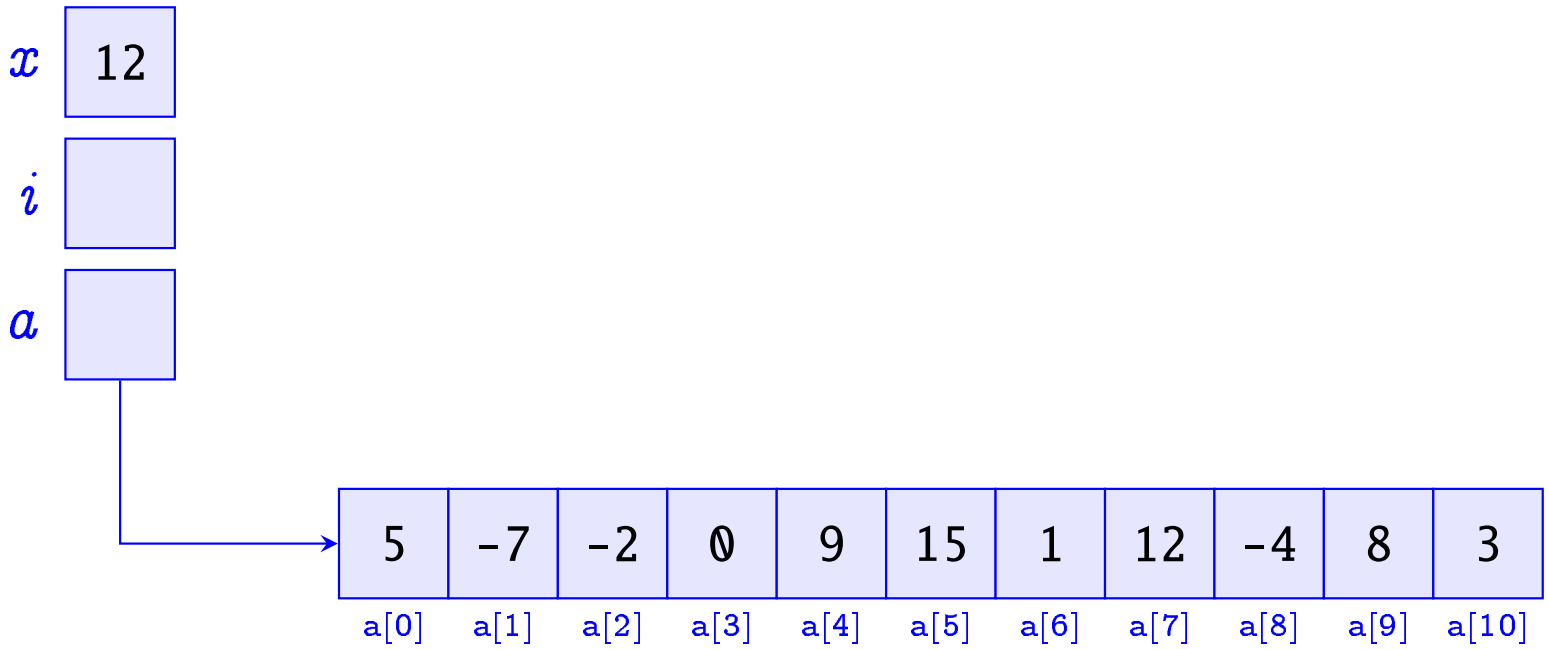
Massiivist elemendi otsimine

- Lihtsaim meetod massiivist elemendi otsimiseks vaatab massiivi elemente ükshaaval järjestikku läbi.
- Kui jõutakse otsitava elemendini, siis väljastatakse vastava elemendi indeks.
- Kui otsitavat elementi ei leidu, siis väljastatakse -1.

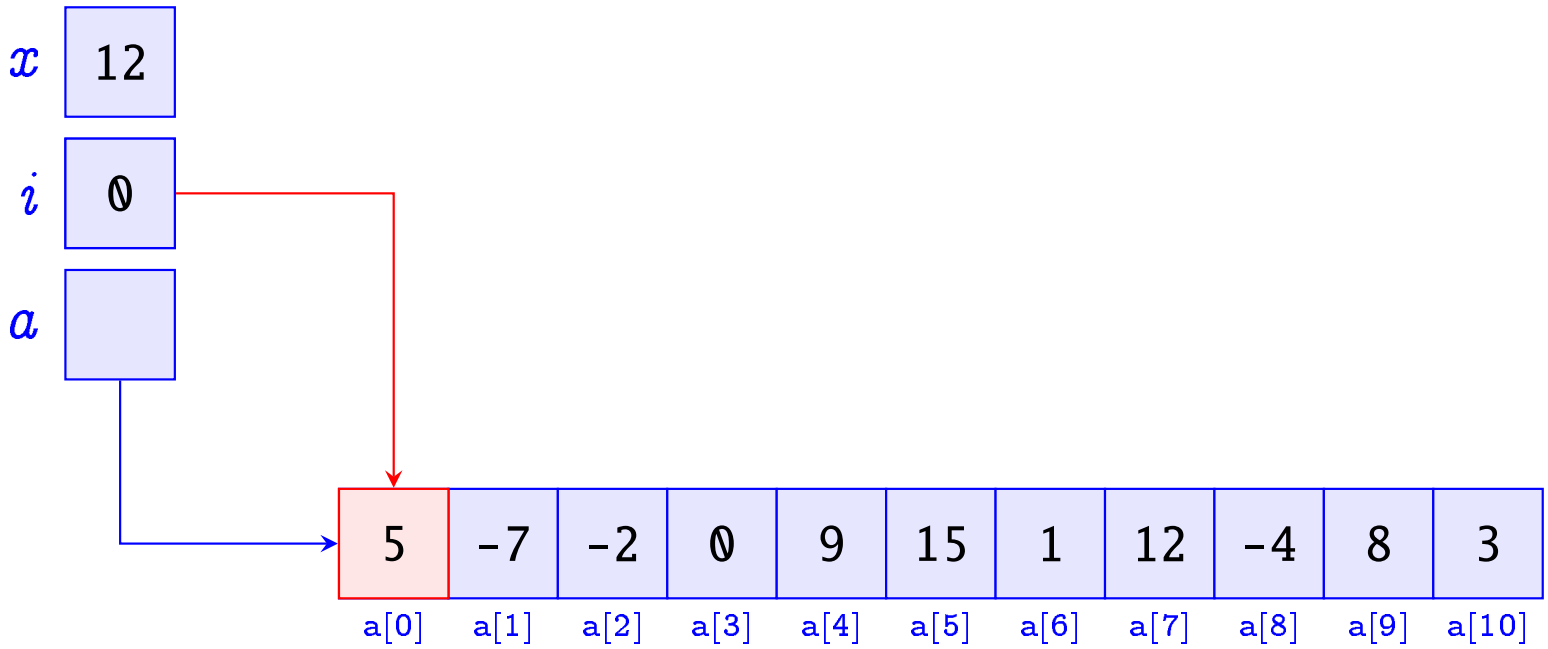
Lineaarne otsimine

```
def linearSearch(x, a):
    for i in range(len(a)):
        if a[i] == x: return i
    return -1
```

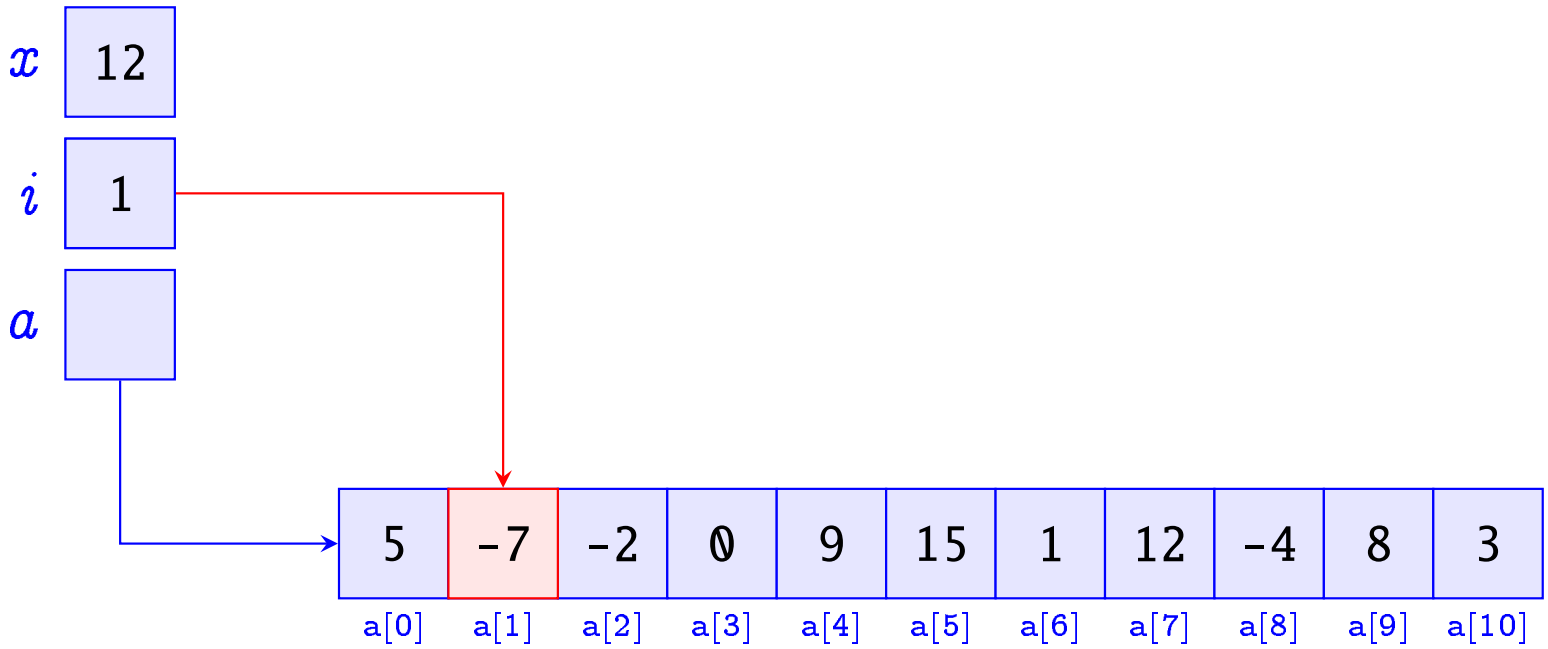
Massiivist elemendi otsimine



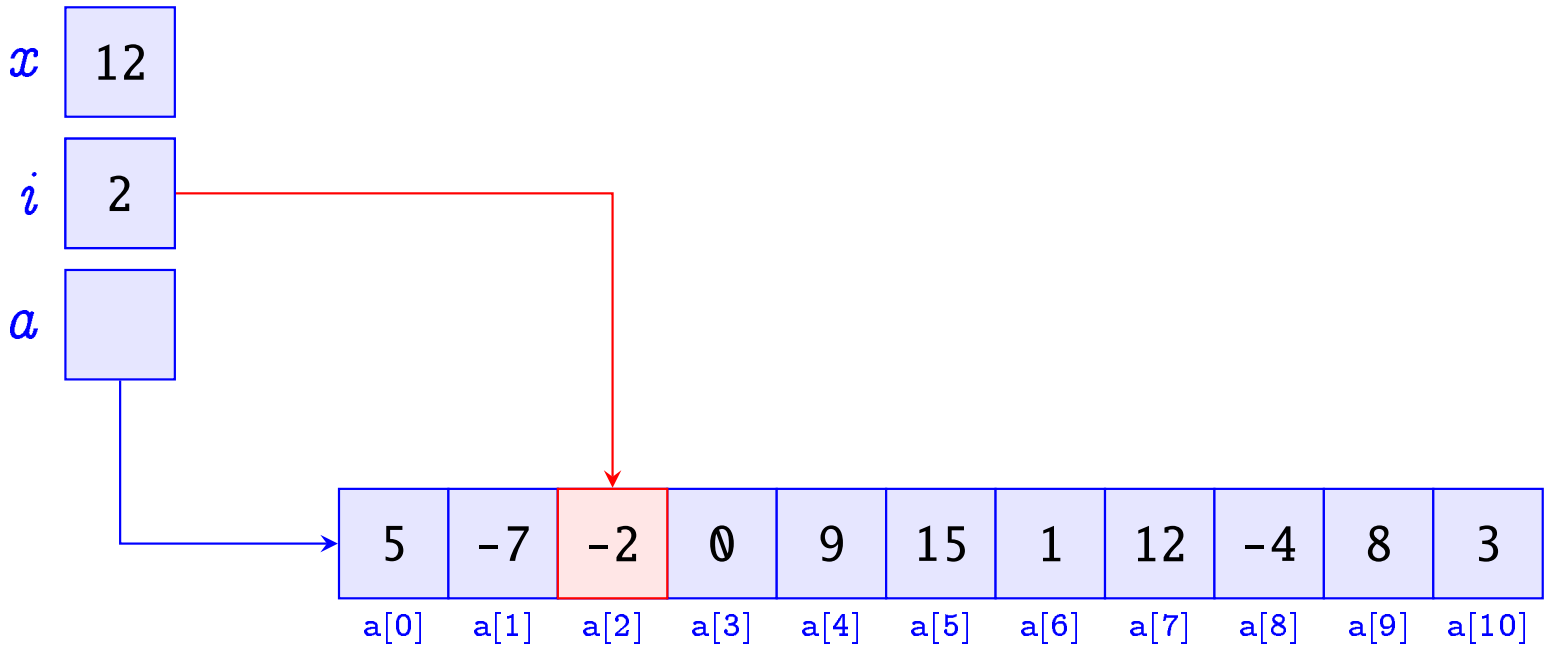
Massiivist elemendi otsimine



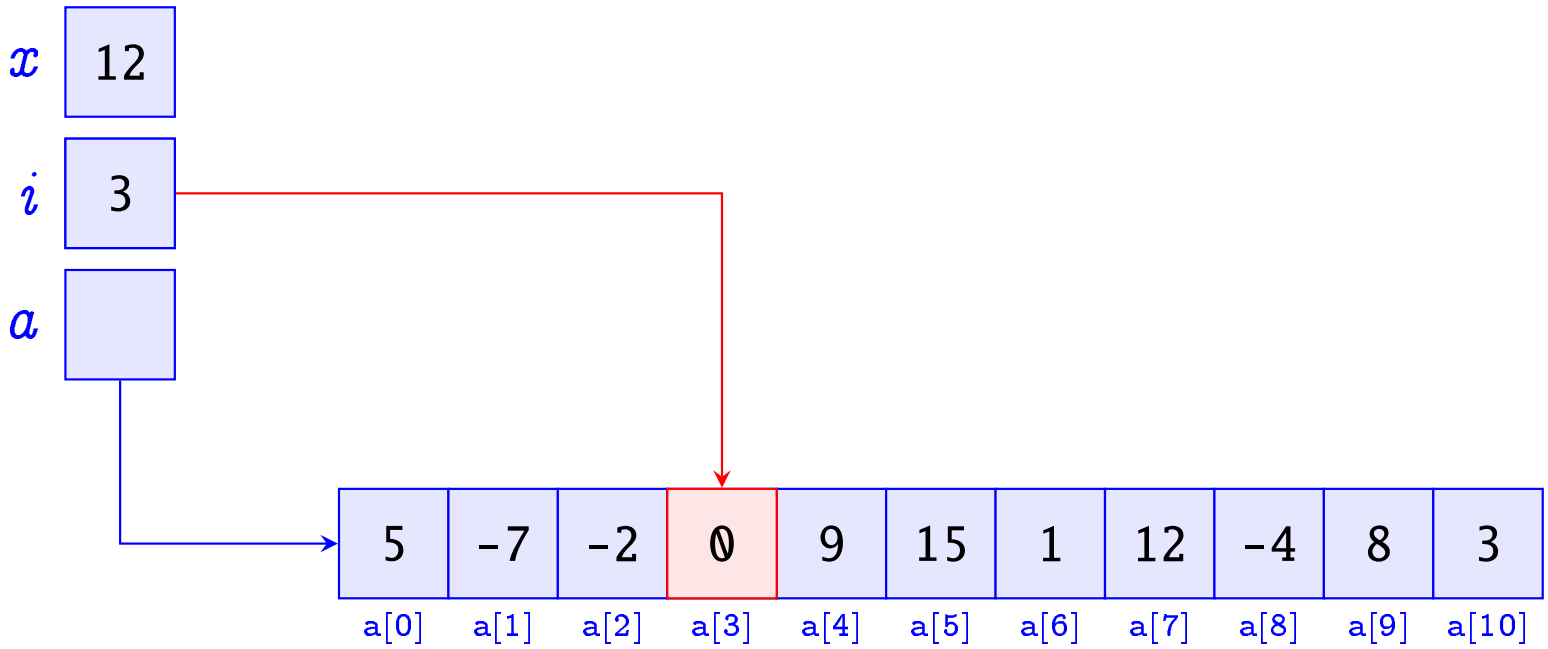
Massiivist elemendi otsimine



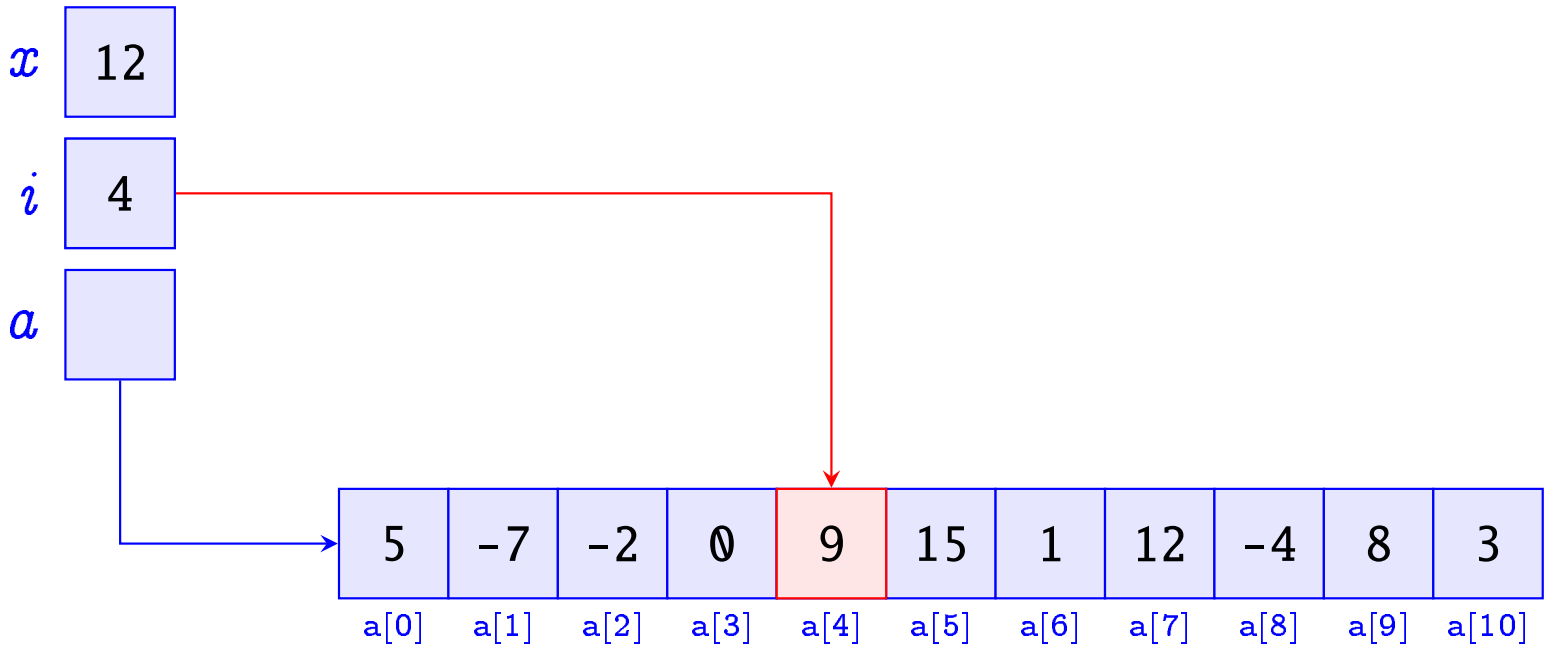
Massiivist elemendi otsimine



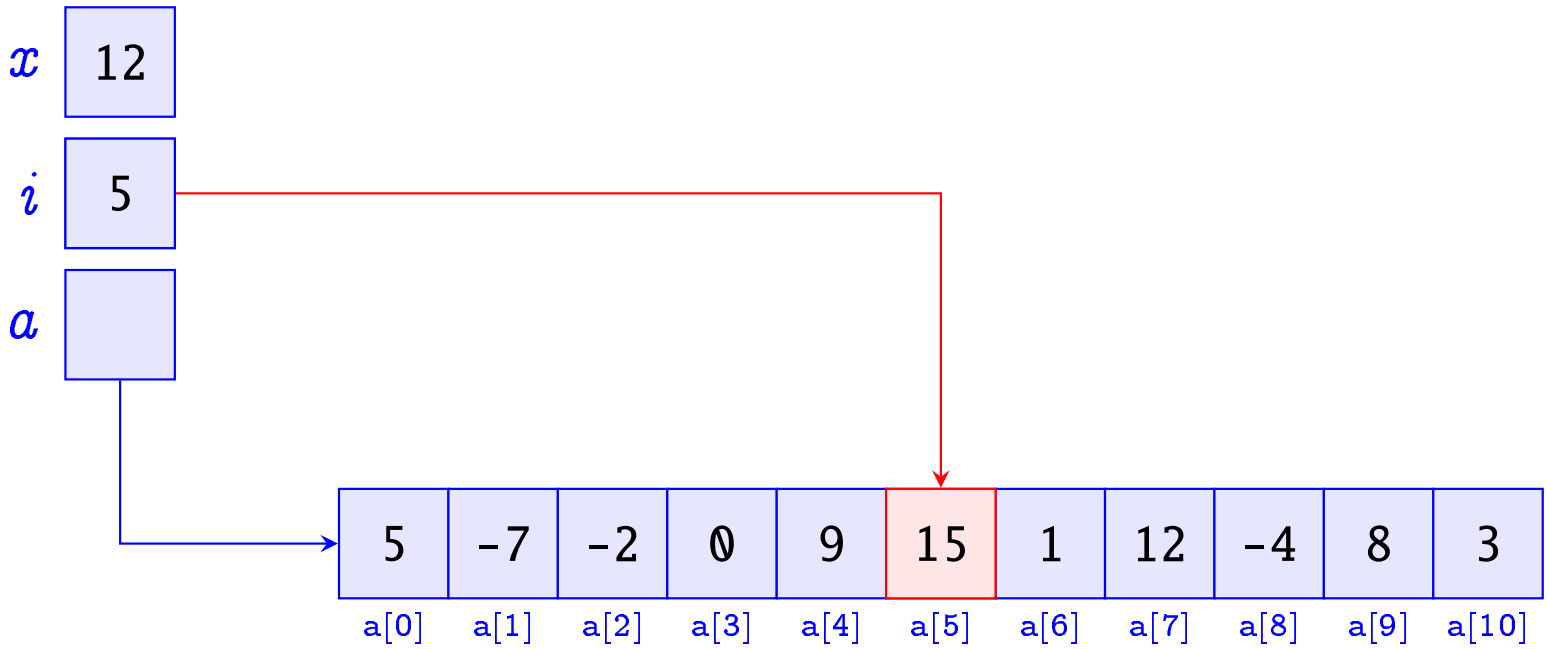
Massiivist elemendi otsimine



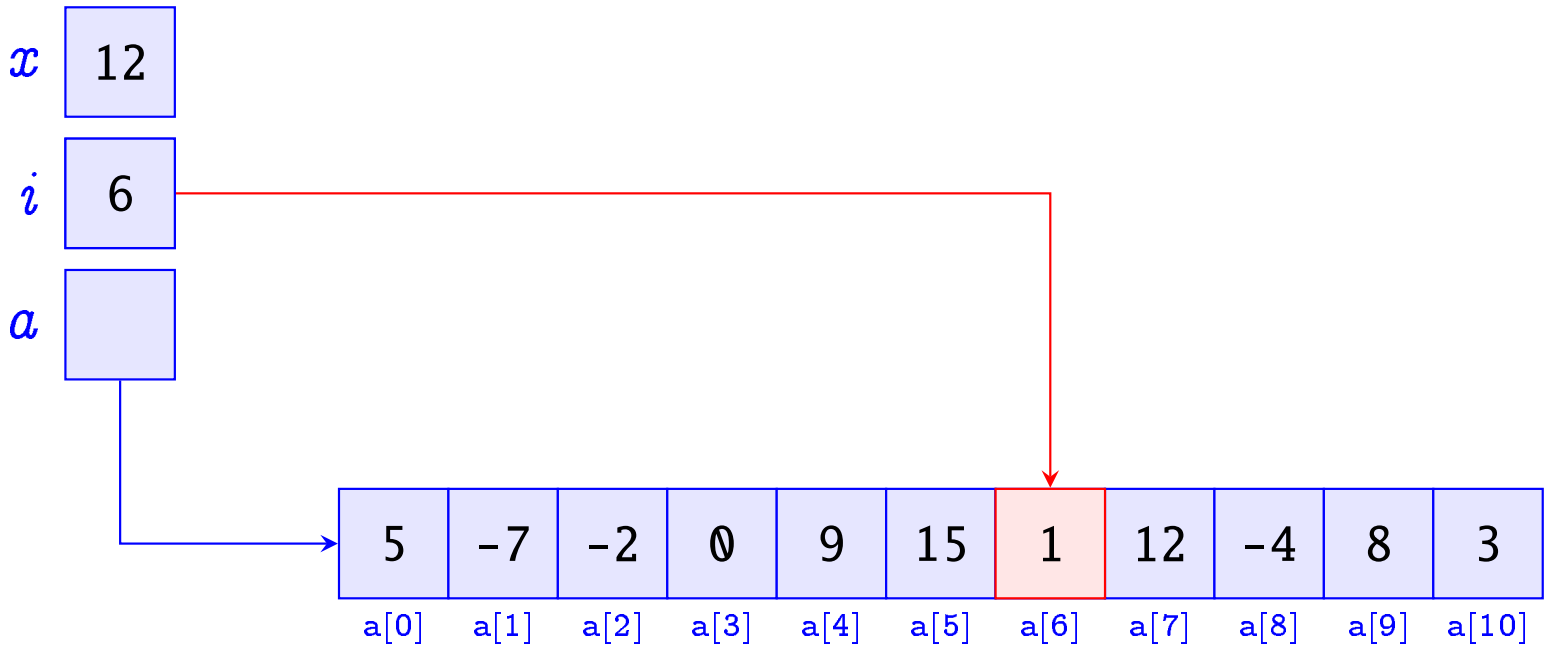
Massiivist elemendi otsimine



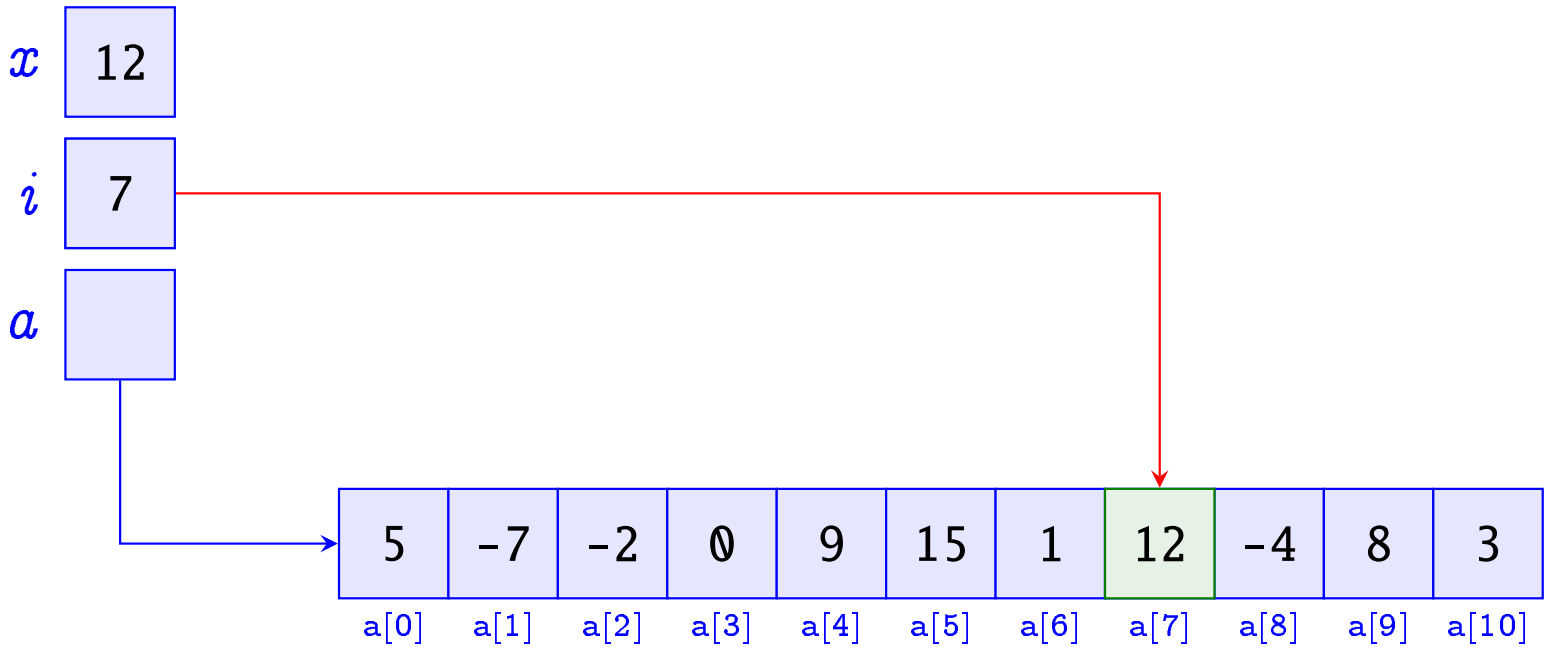
Massiivist elemendi otsimine



Massiivist elemendi otsimine



Massiivist elemendi otsimine



Massiivist elemendi otsimine

- Antud meetod on linaarse keerukusega $\mathcal{O}(n)$, kuna halvimal juhul läbitakse kogu massiiv.
- Kui sisendmassiiv on *järjestatud*, siis on võimalik otsida tunduvalt efektiivsemalt kasutades kahendotsimist.
- Kahendotsimise idee:
  - Kontrollime massiivi keskmist elementi.
  - Kui see on otsitav, siis olemegi elemendi leidnud.
  - Kui see on otsitavast väiksem, siis otsitav saab olla ainult järgmiste elementide hulgas.
  - Vastasel korral saab otsitav olla ainult eelnevate elementide hulgas.
  - Kordame kogu protsessi vastaval alammassiivil.

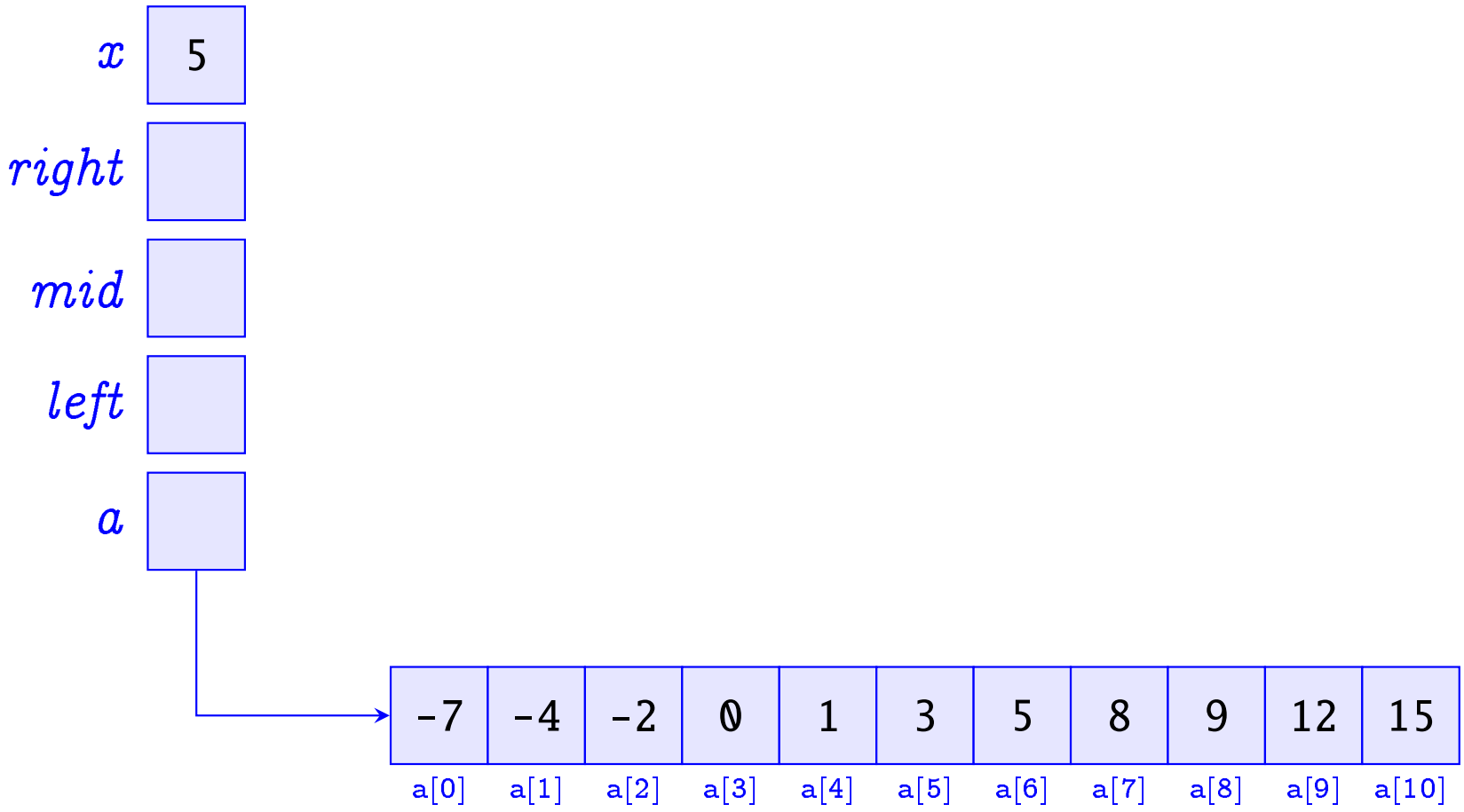
Kahendotsimine

Kahendotsimine

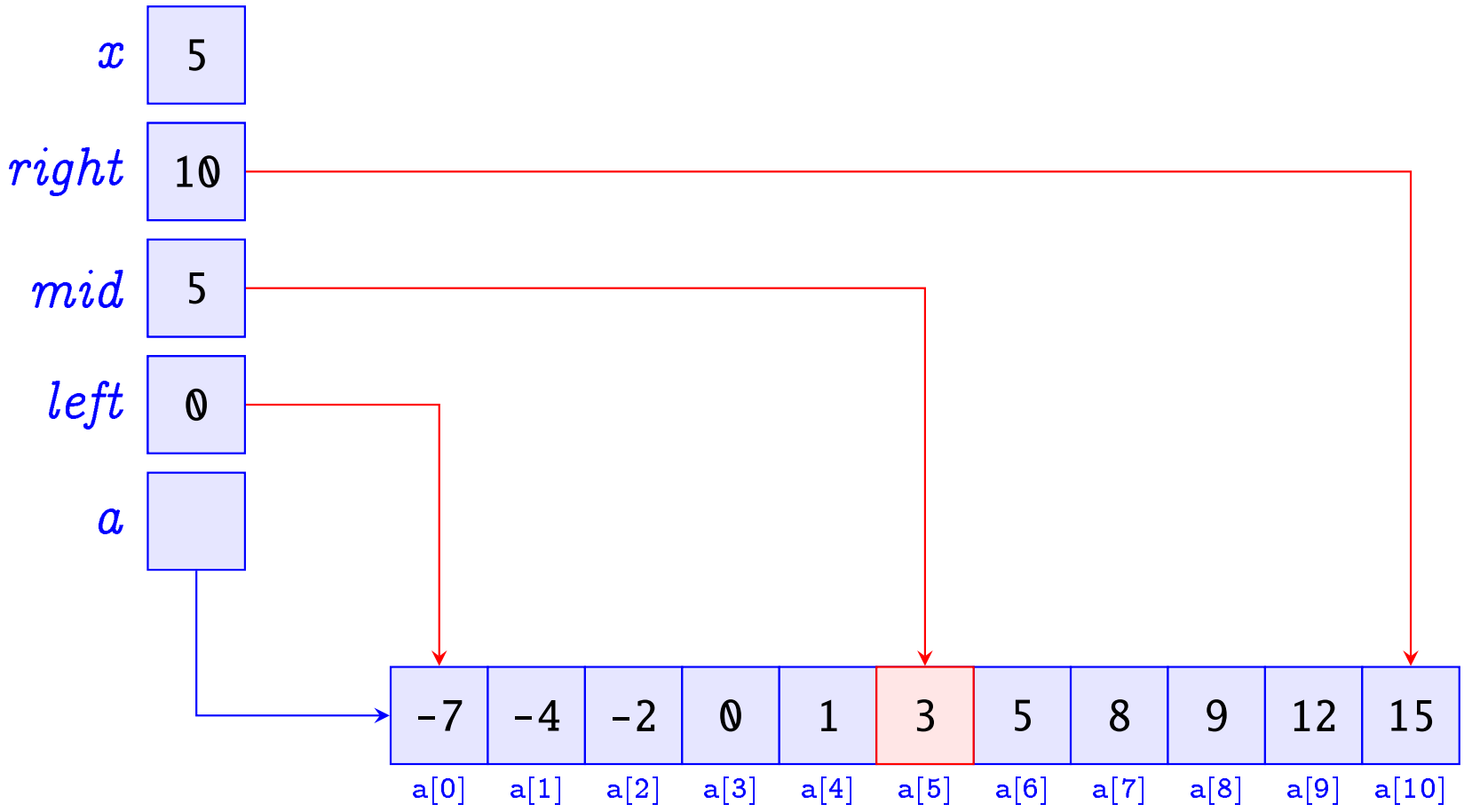
```
def recBinSearch(x, a, left, right):
    if left > right:
        return -1
    mid = (left+right) // 2
    if a[mid] == x:
        return mid
    elif a[mid] > x:
        return recBinSearch(x, a, left, mid-1)
    else:
        return recBinSearch(x, a, mid+1, right)

def binarySearch(x, a):
    return recBinSearch(x, a, 0, len(a)-1)
```

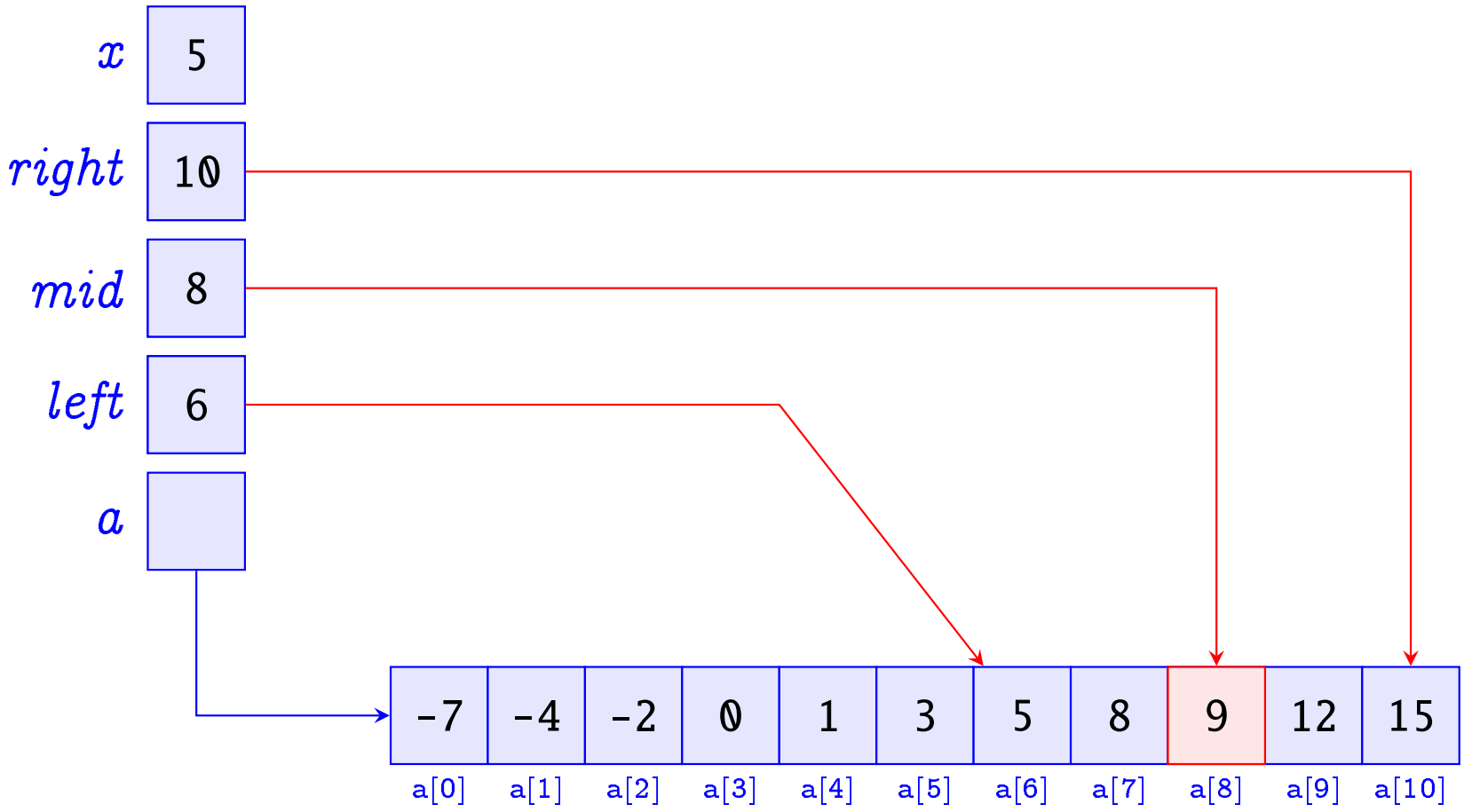
Kahendotsimine



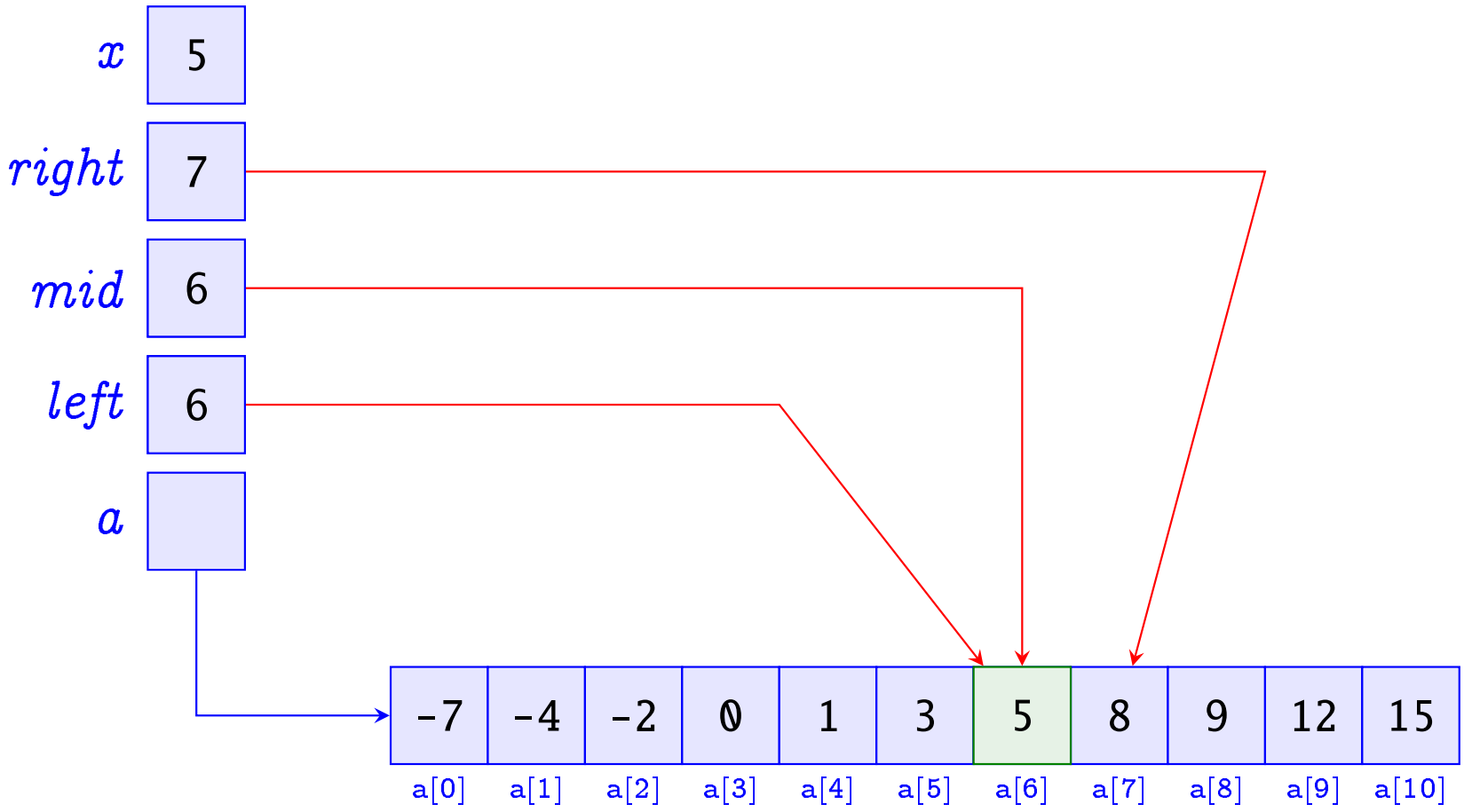
Kahendotsimine



Kahendotsimine



Kahendotsimine



Kahendotsimine

- Igal iteratsioonisammul välistatakse pool massiivist.
- Seega on kahendotsimine logaritmilise keerukusega $\mathcal{O}(\log_2(n))$.
- Suurte massiivide korral on erinevus efektiivsuses vägagi märgatav.
  - Näiteks, kui massiivis on $2^{20} = 1048576$ elementi, siis lineaarse otsimisega tuleb halvimal juhul kogu massiiv läbida.
  - Kahendotsimise korral tuleb halvimal juhul teha vaid 20 võrdlust.
- NB! Kahendotsimine eeldab, et sisendmassiiv on järjestatud; lineaarne otsimine seda ei eelda.

Sorteerimine

- Sorteerimine on andmete järjestamine nende väärtuste järgi.
- Kõigis meie näidetes vaatleme massiivielementide järjestamist kasvavasse järjekorda.
- Sorteerimiseks on mitmeid erinevaid meetode, milledest alljärgnevalt vaatleme kahte:
  - valikumeetod,
  - ühildusmeetod.

Valikumeetodil sorteerimine

Valikumeetodi idee

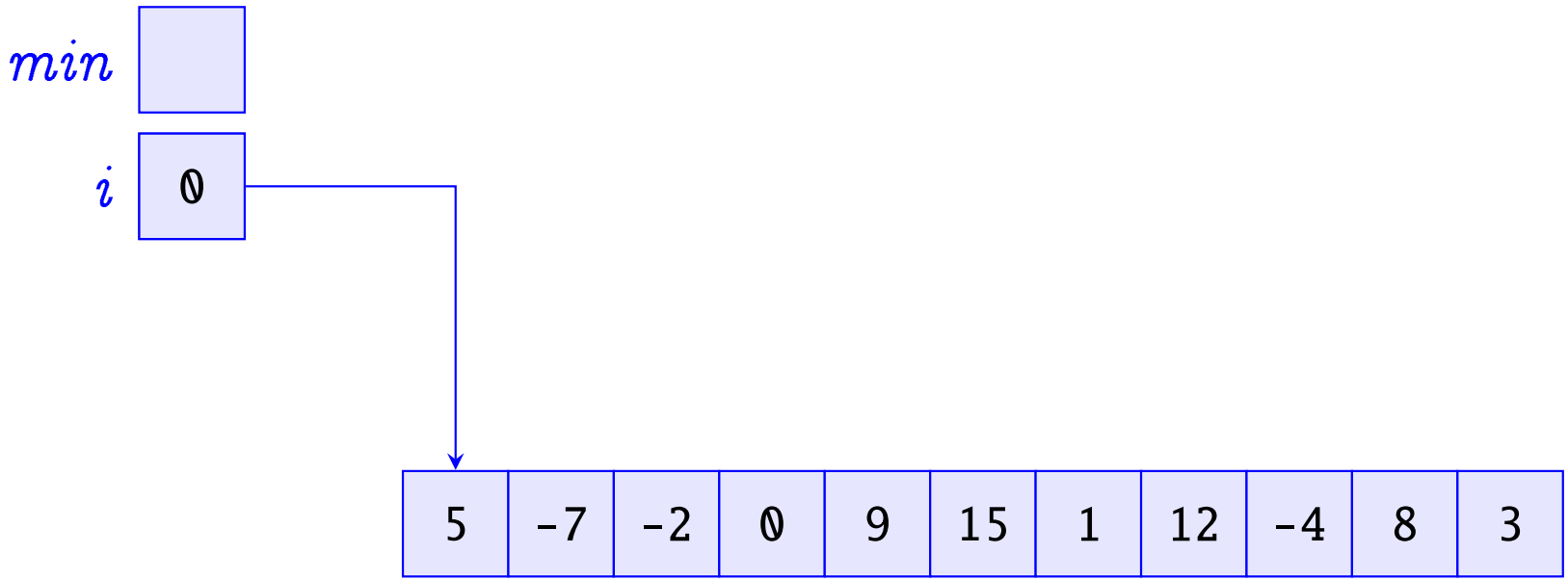
- Leiame elemendi, mis peaks olema kõige esimene.
  - Selleks otsime massiivist minimaalse elemendi.
- Vahetame selle elemendi massiivi tegeliku esimese elemendiga.
- Massiivi algus (so. esimene element) on nüüd sorteeritud ja edaspidi seda enam ei muuda.
- Ülejäänud massiiv on aga veel sorteerimata.
- Kordame kogu protsessi sorteerimata massiiviosal, kuni kogu massiiv saab sorteeritud.

Valikumeetodil sorteerimine

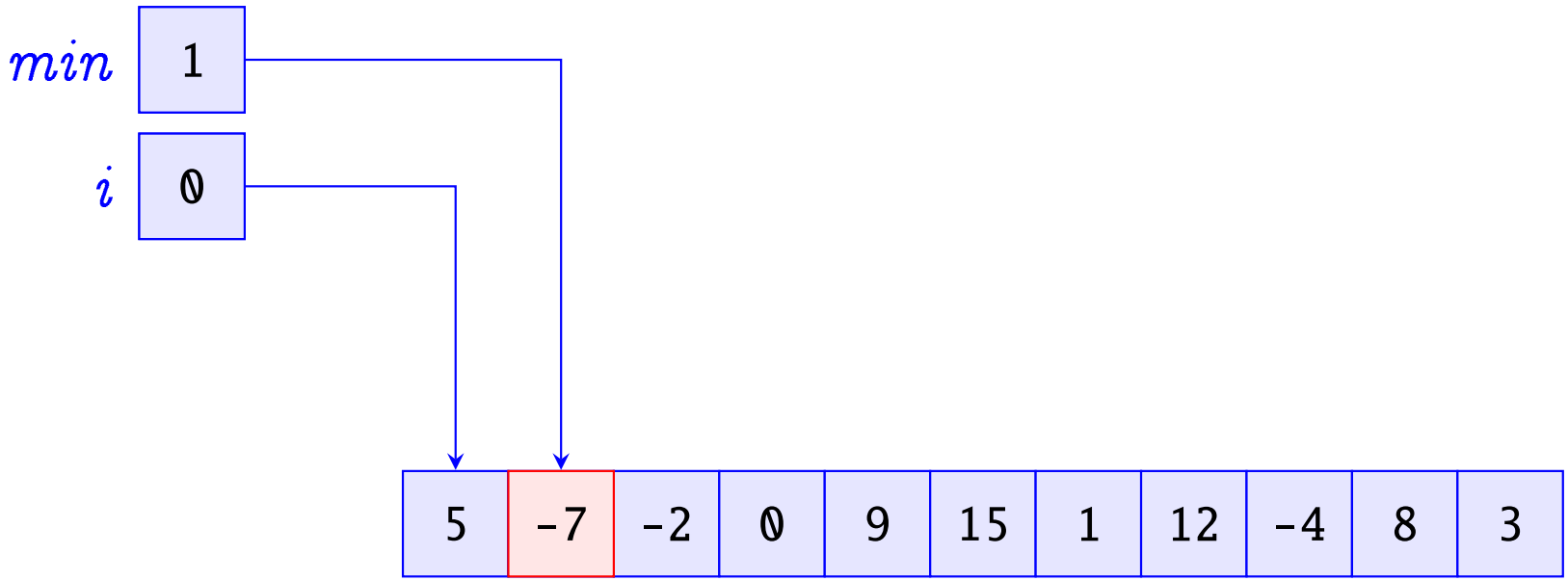
Valikumeetodil sorteerimine

```
def selectionSort(a):
    for i in range(len(a)):
        min = i
        for j in range(i+1,len(a)):      # otsi minimaalne
            if a[j] < a[min]:  min = j
        if i != min:                     # vaheta elemendid
            tmp    = a[i]
            a[i]   = a[min]
            a[min] = tmp
```

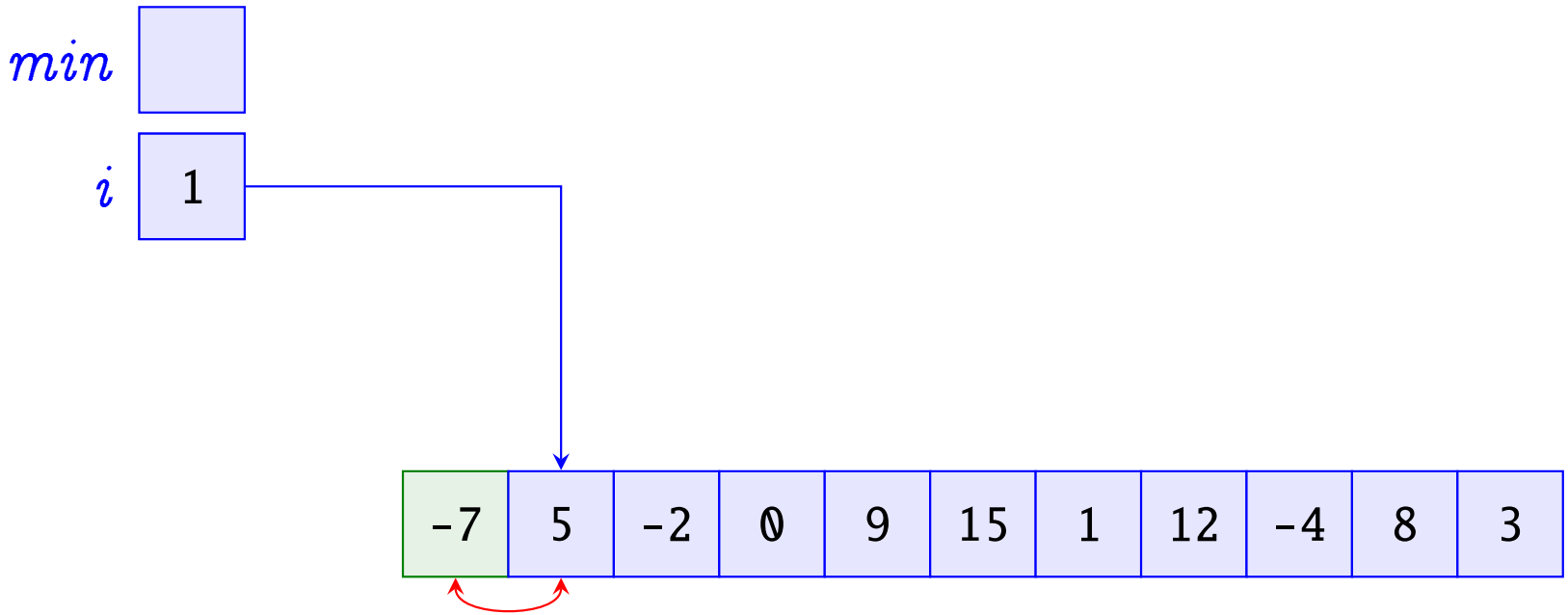
Valikumeetodil sorteerimine



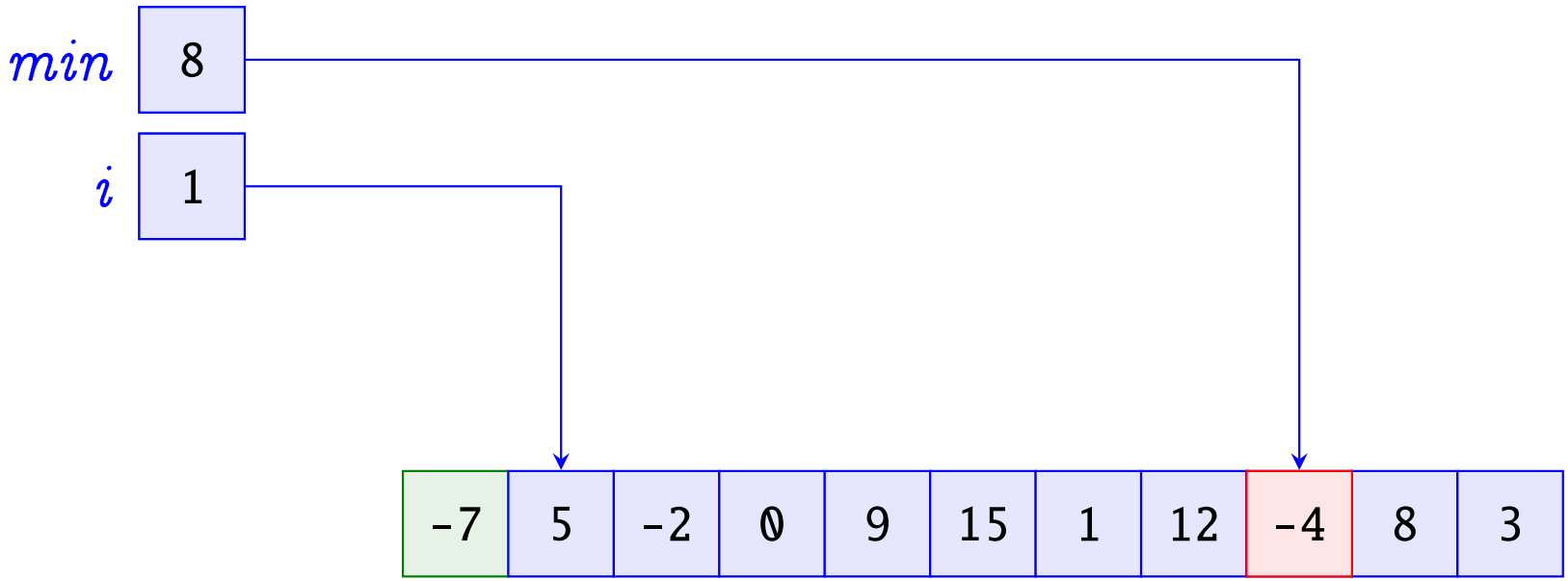
Valikumeetodil sorteerimine



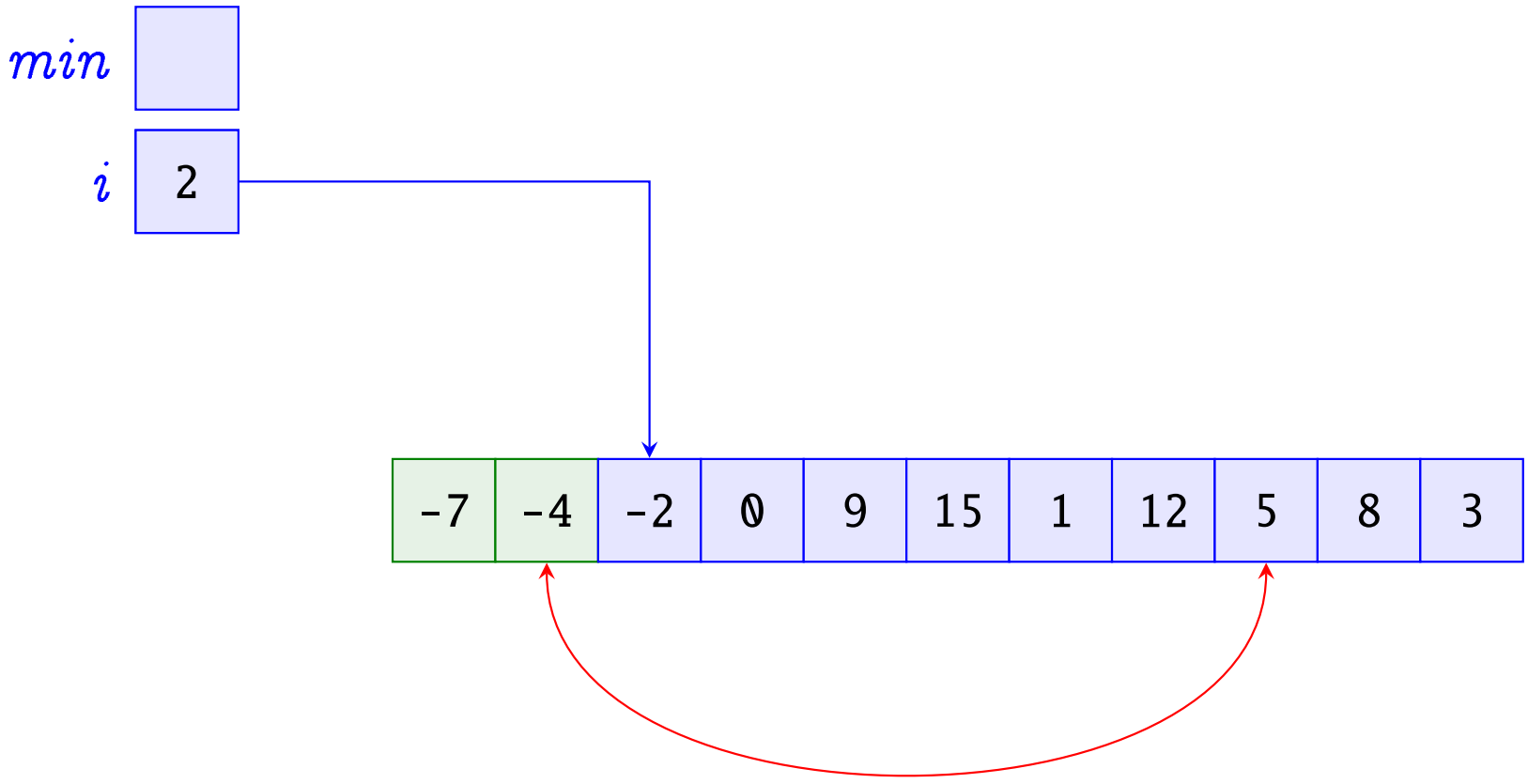
Valikumeetodil sorteerimine



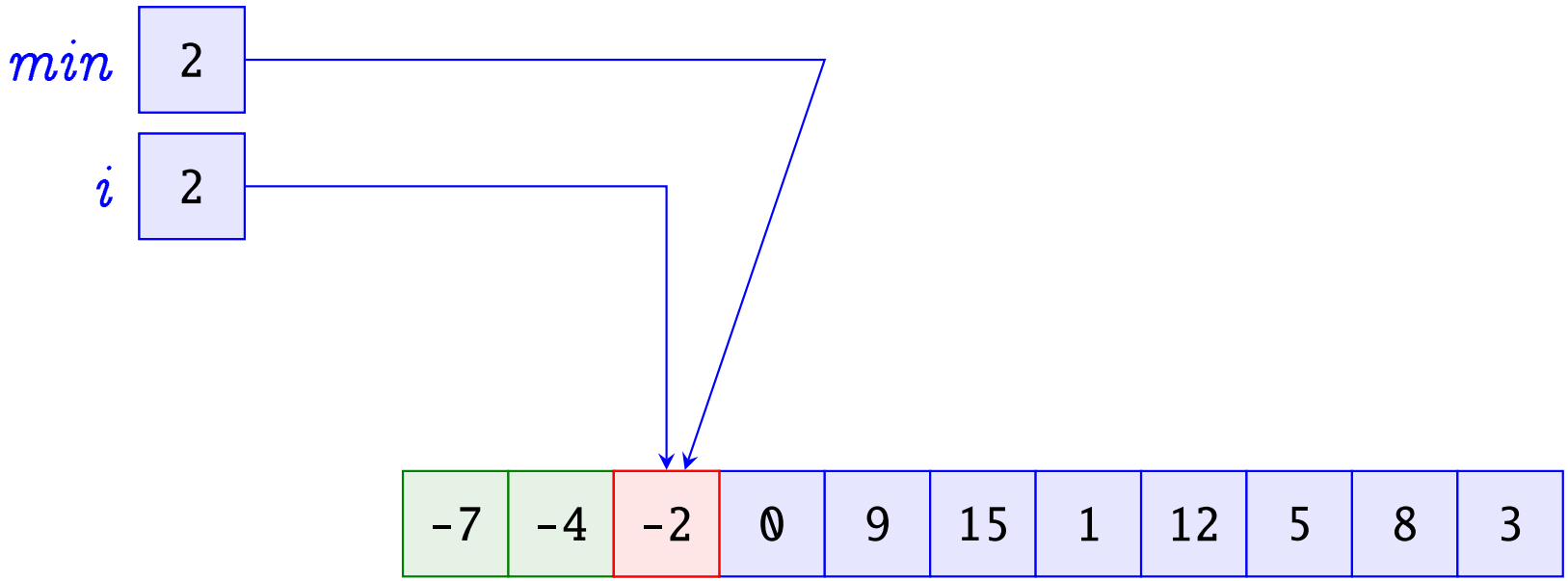
Valikumeetodil sorteerimine



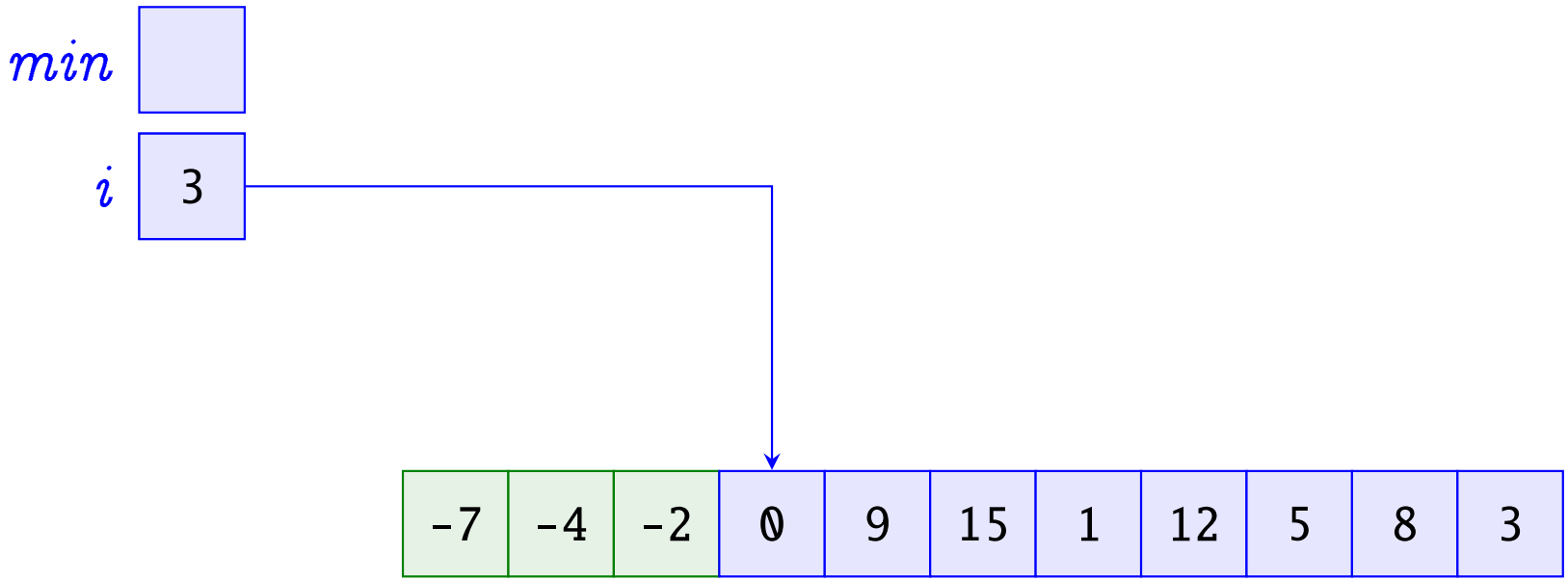
Valikumeetodil sorteerimine



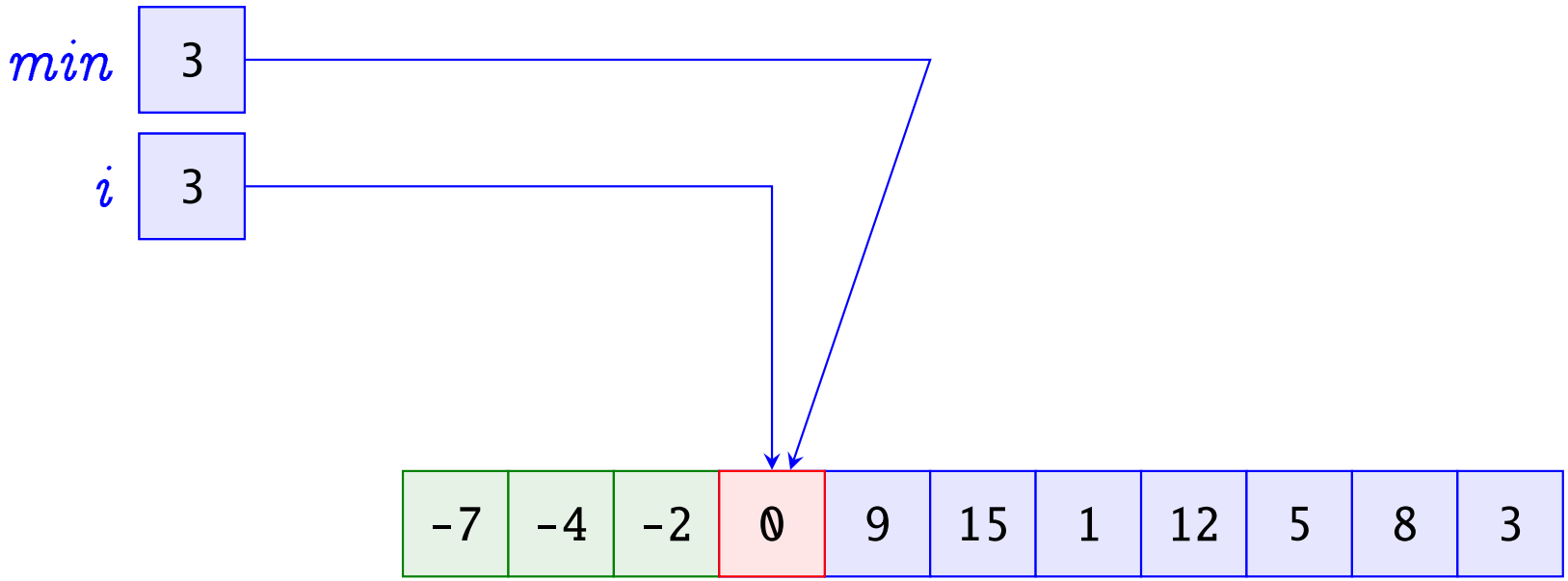
Valikumeetodil sorteerimine



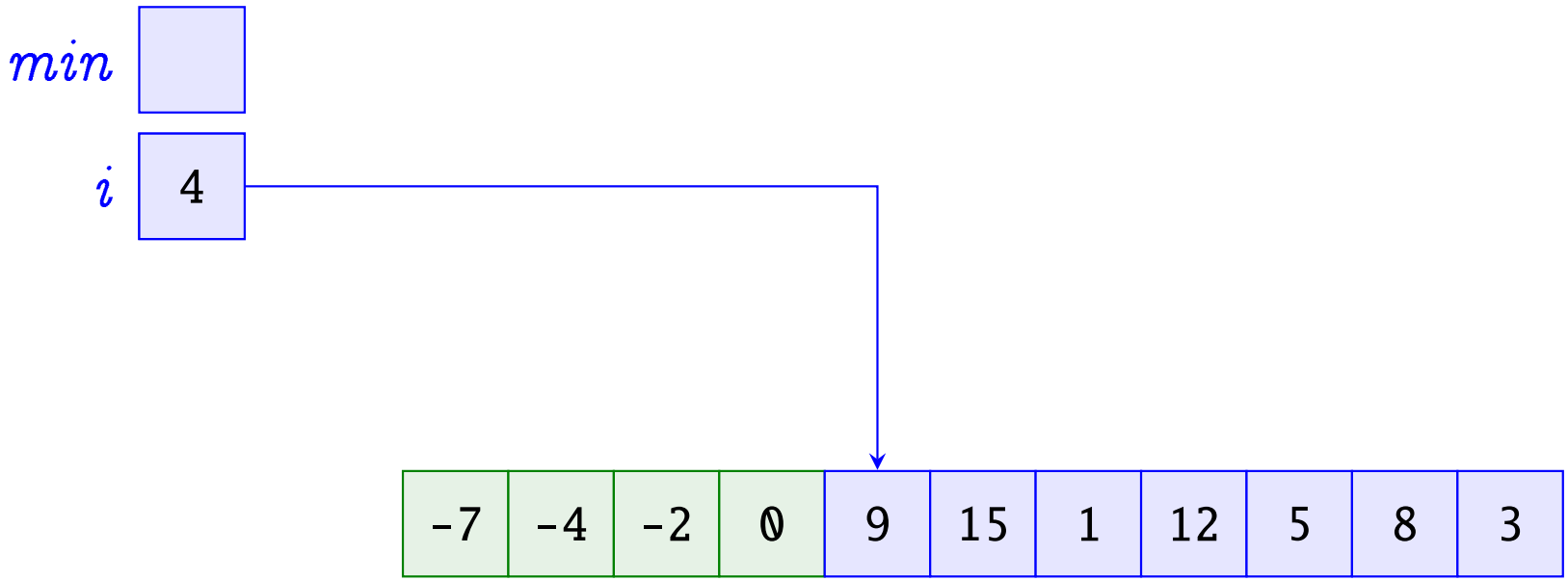
Valikumeetodil sorteerimine



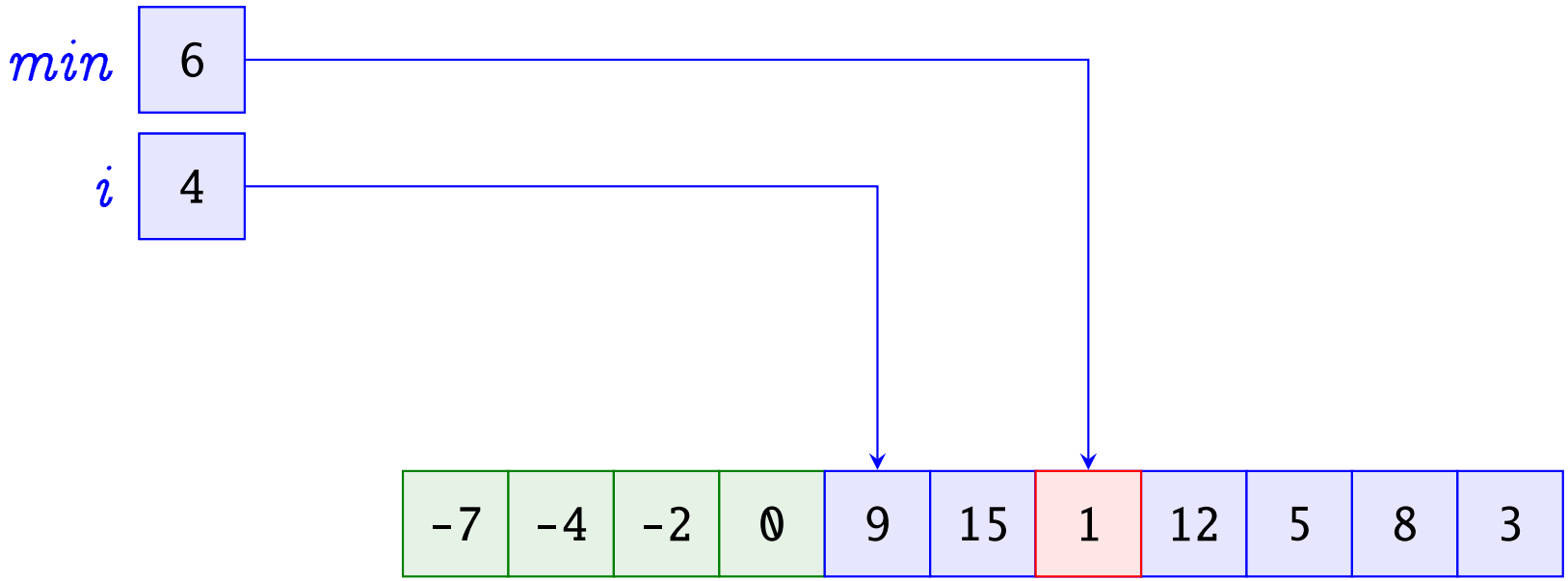
Valikumeetodil sorteerimine



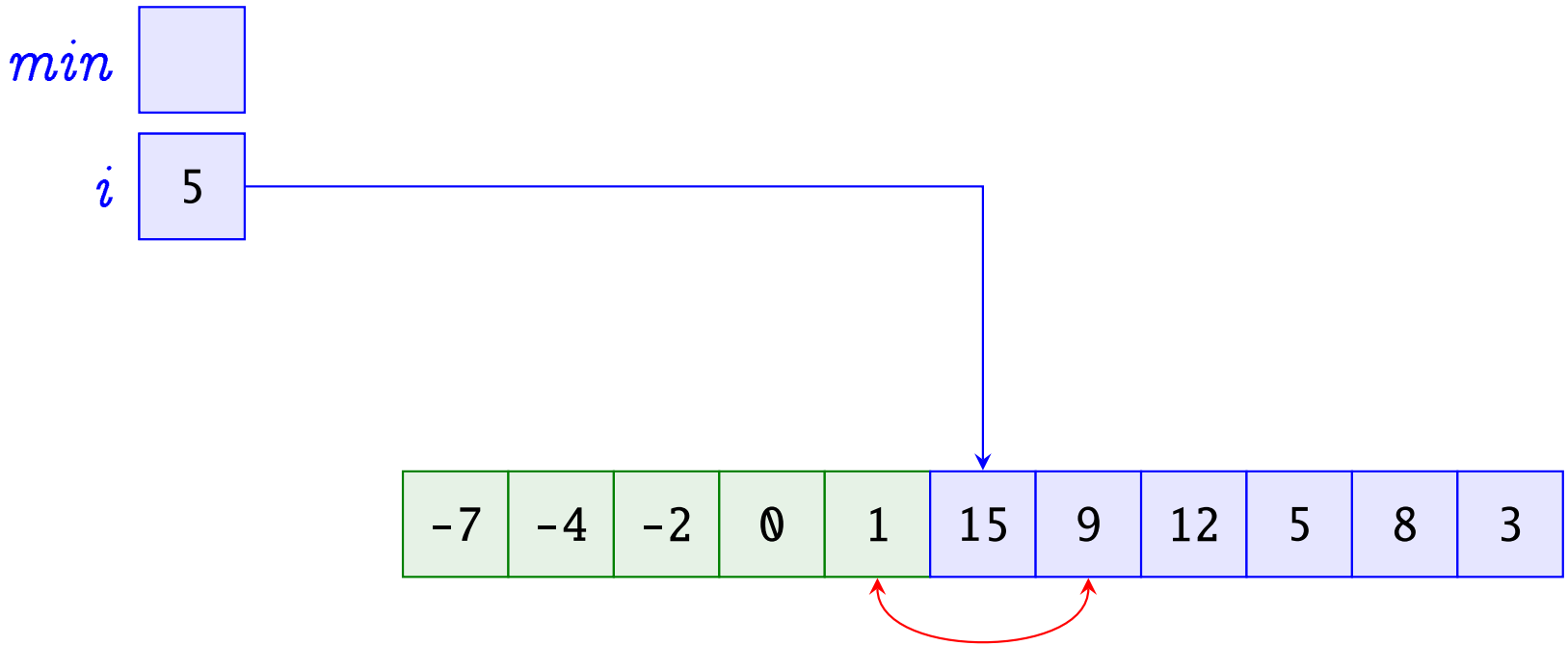
Valikumeetodil sorteerimine



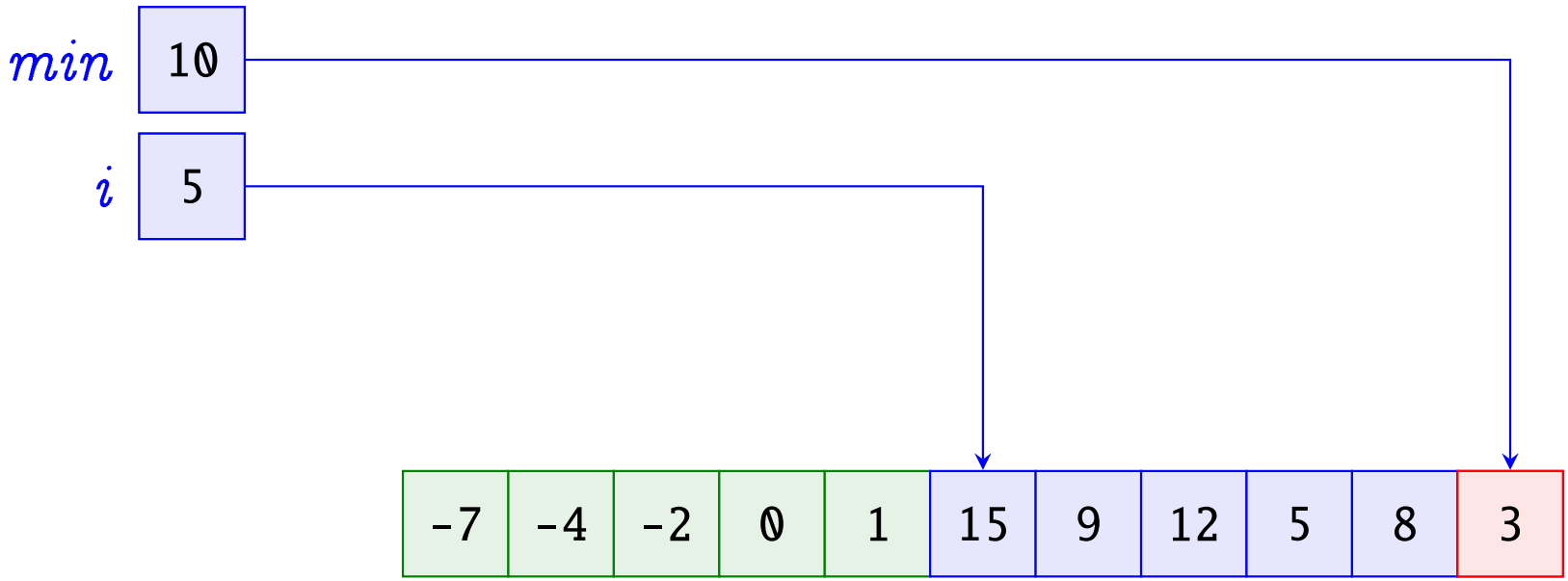
Valikumeetodil sorteerimine



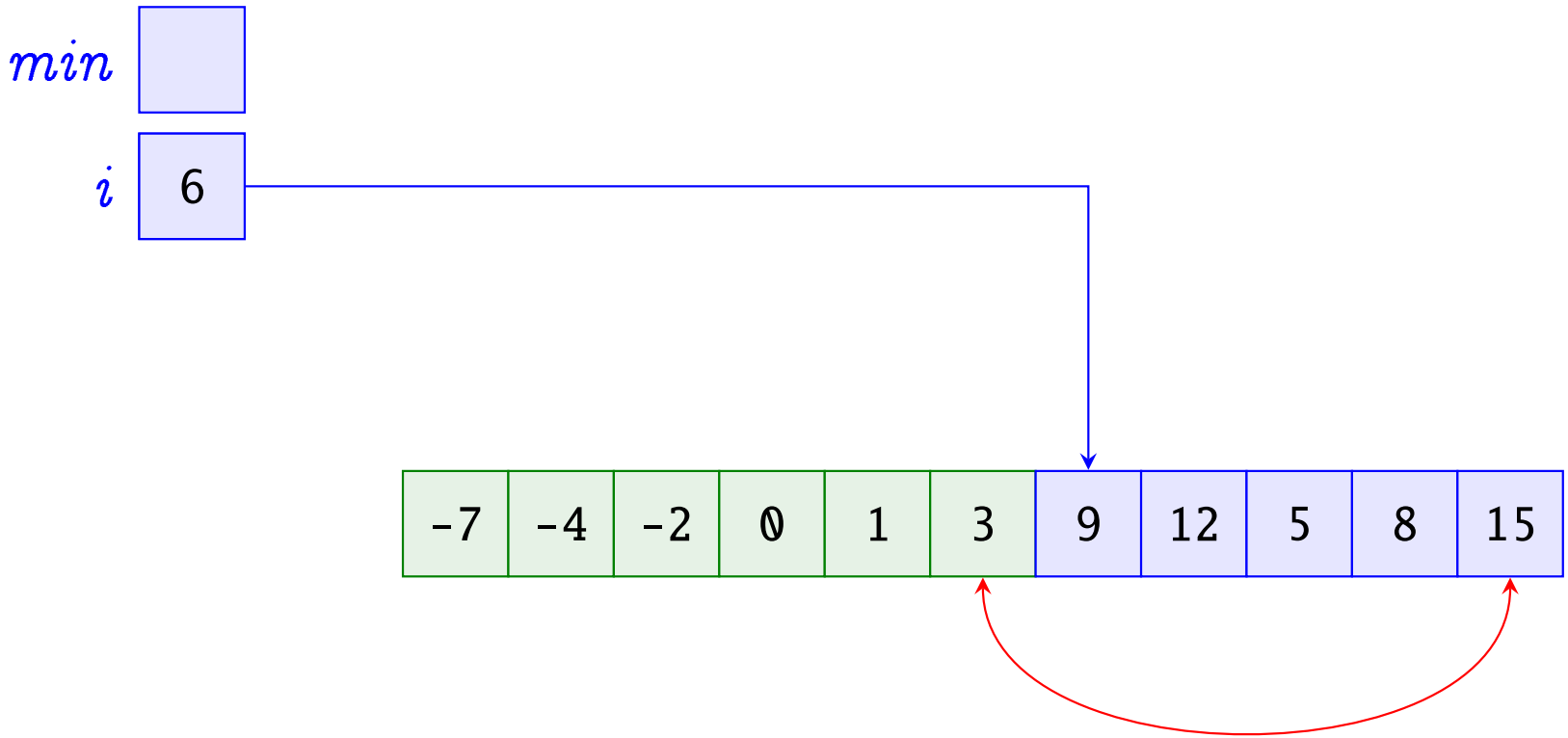
Valikumeetodil sorteerimine



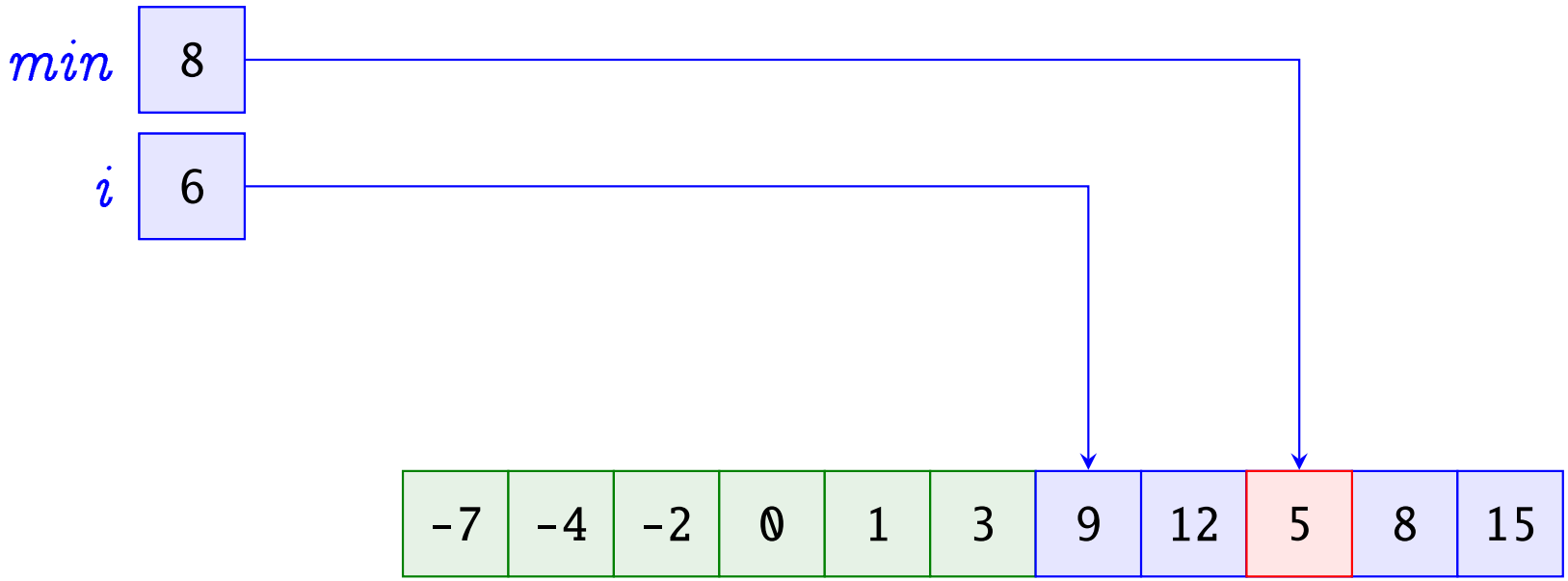
Valikumeetodil sorteerimine



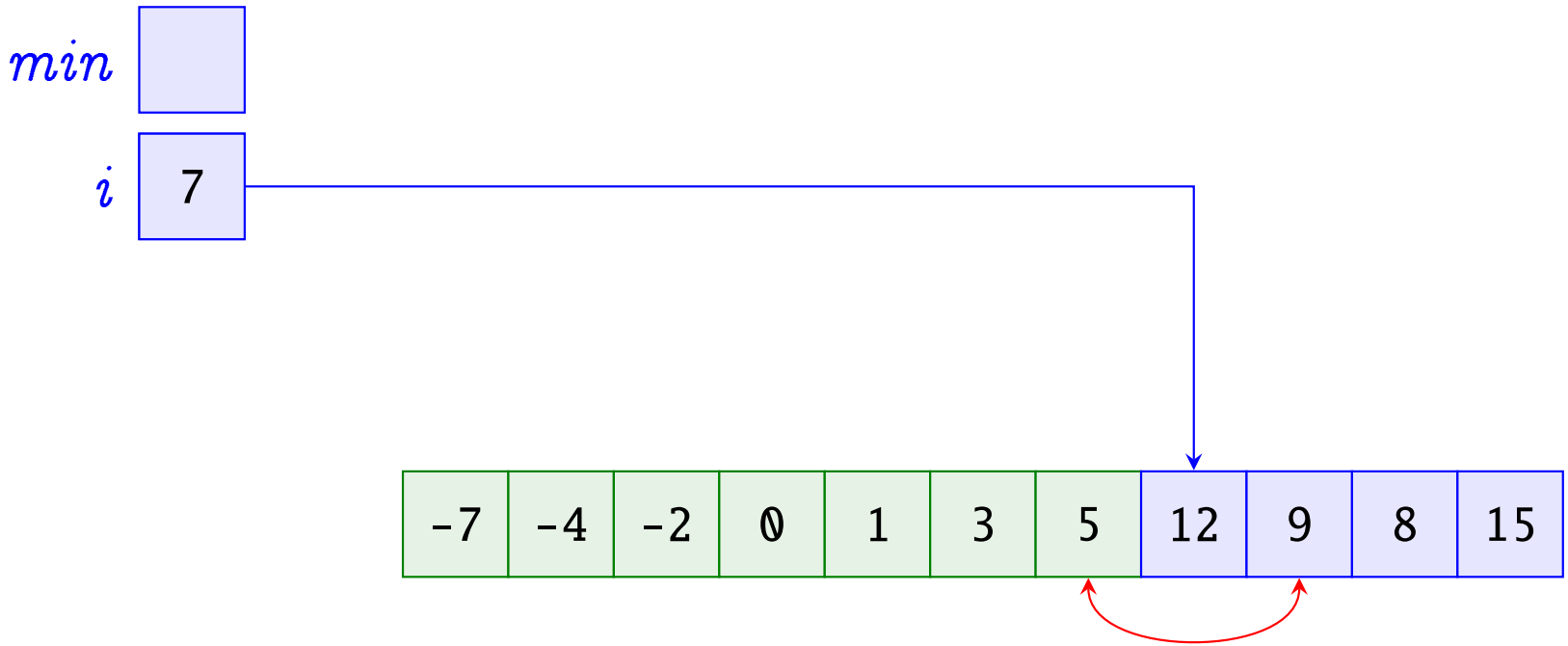
Valikumeetodil sorteerimine



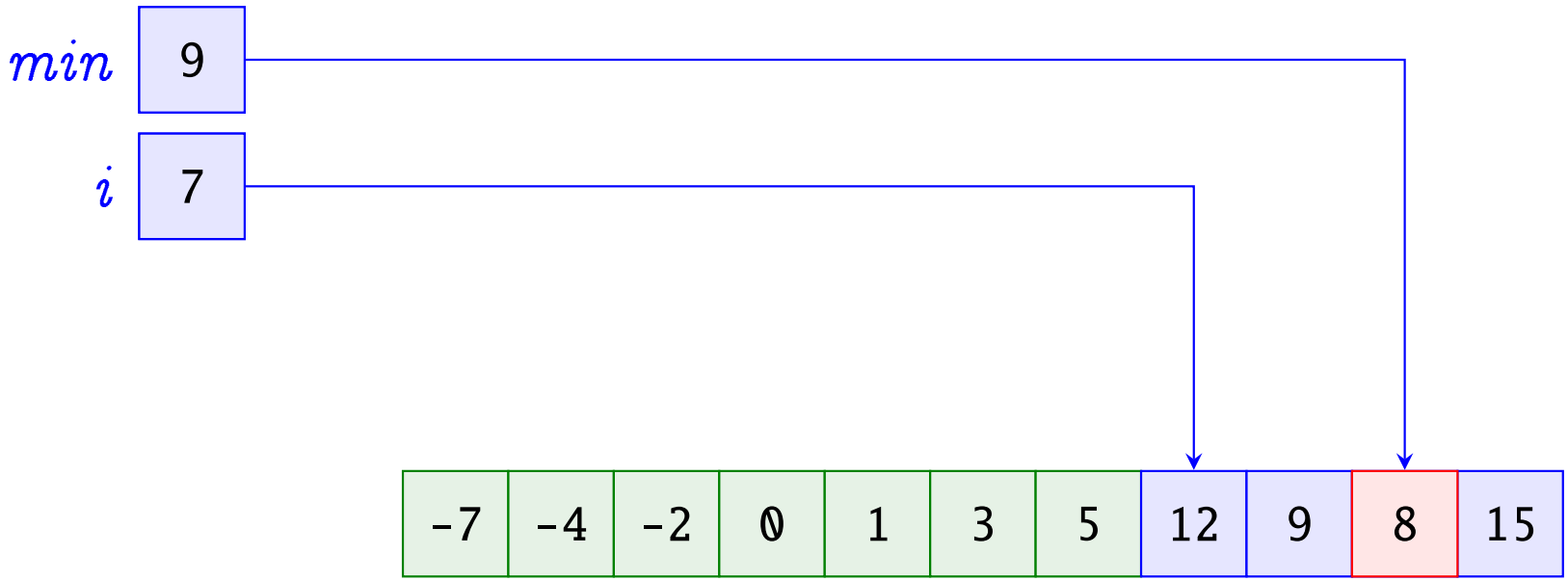
Valikumeetodil sorteerimine



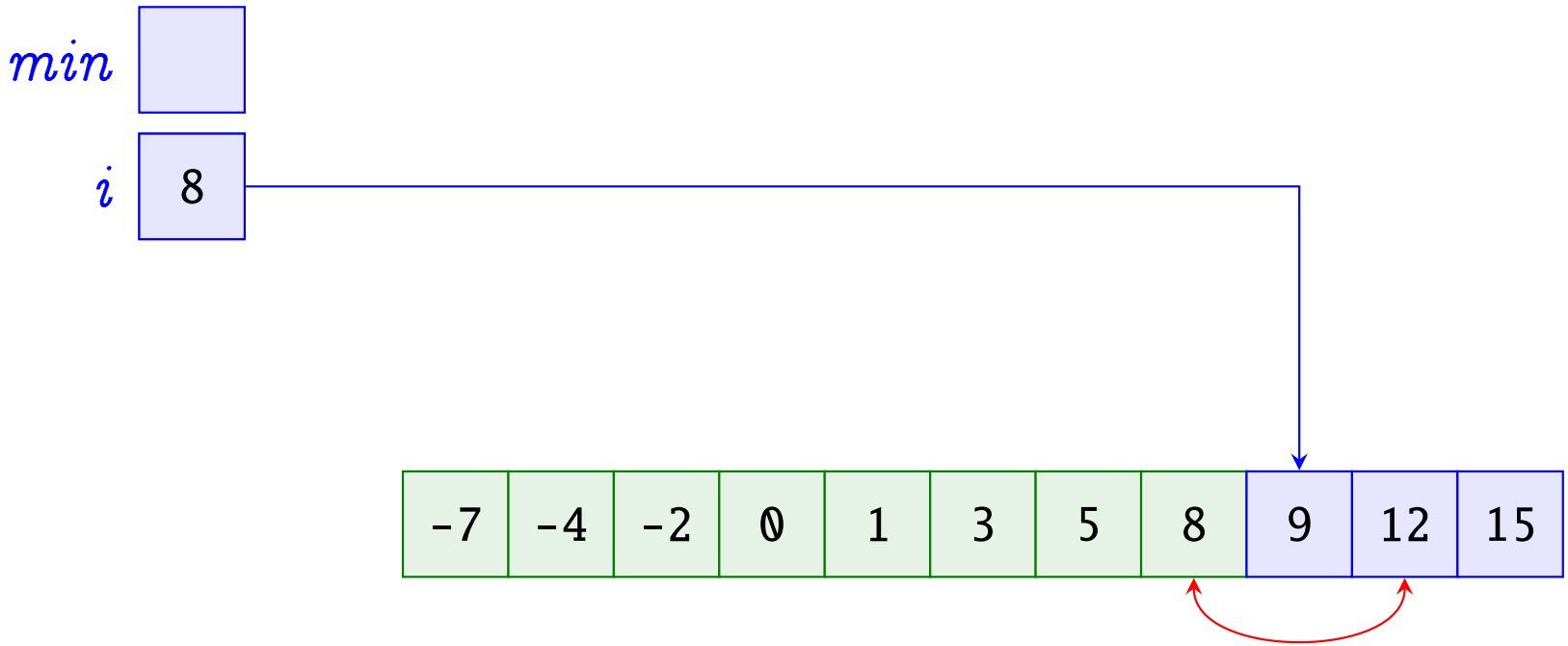
Valikumeetodil sorteerimine



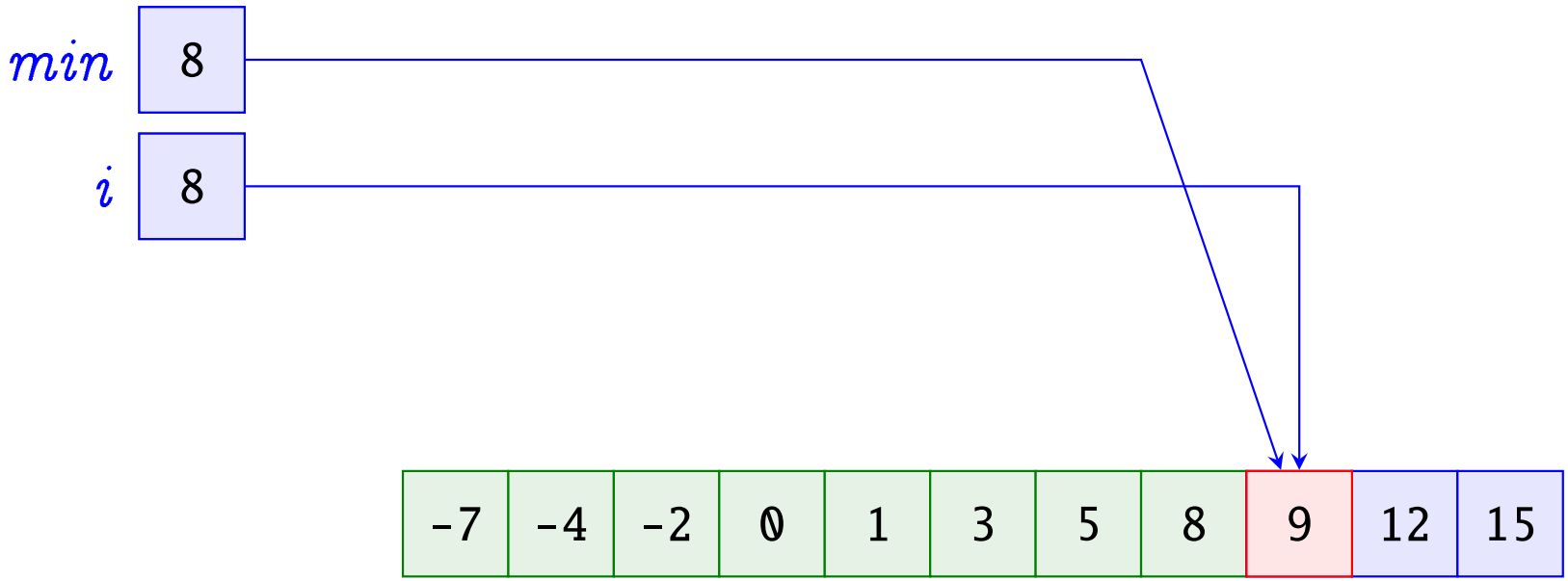
Valikumeetodil sorteerimine



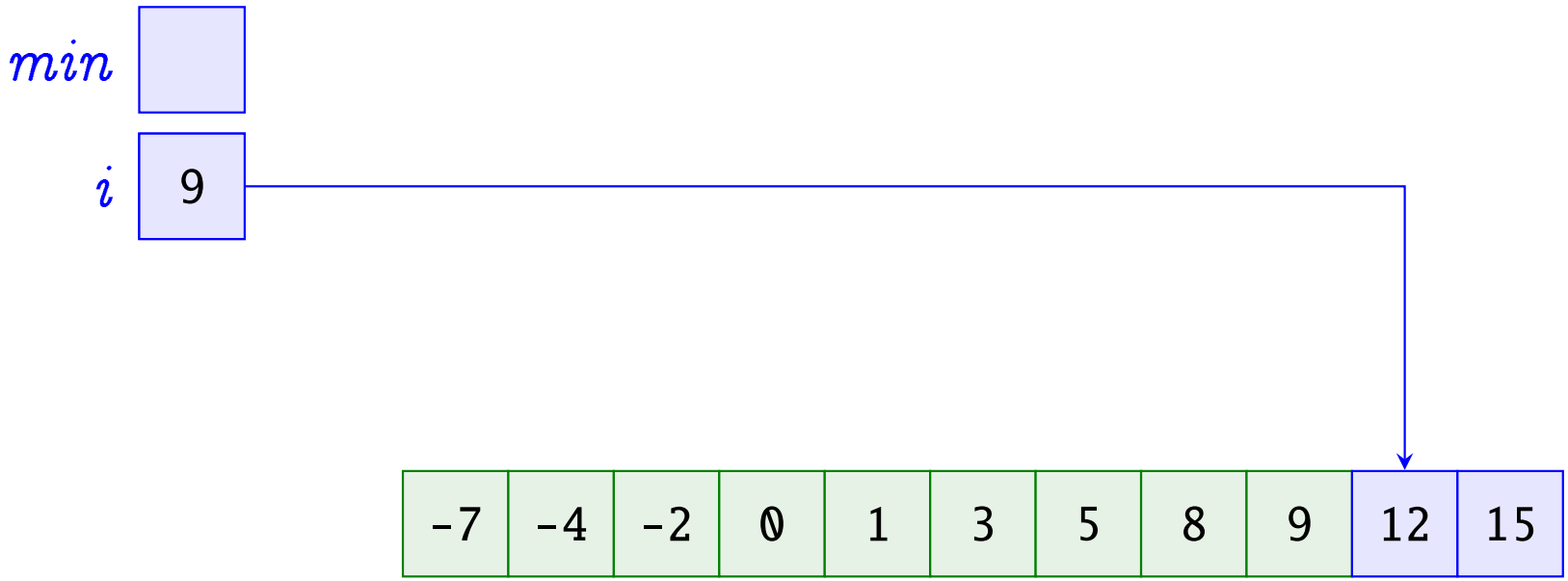
Valikumeetodil sorteerimine



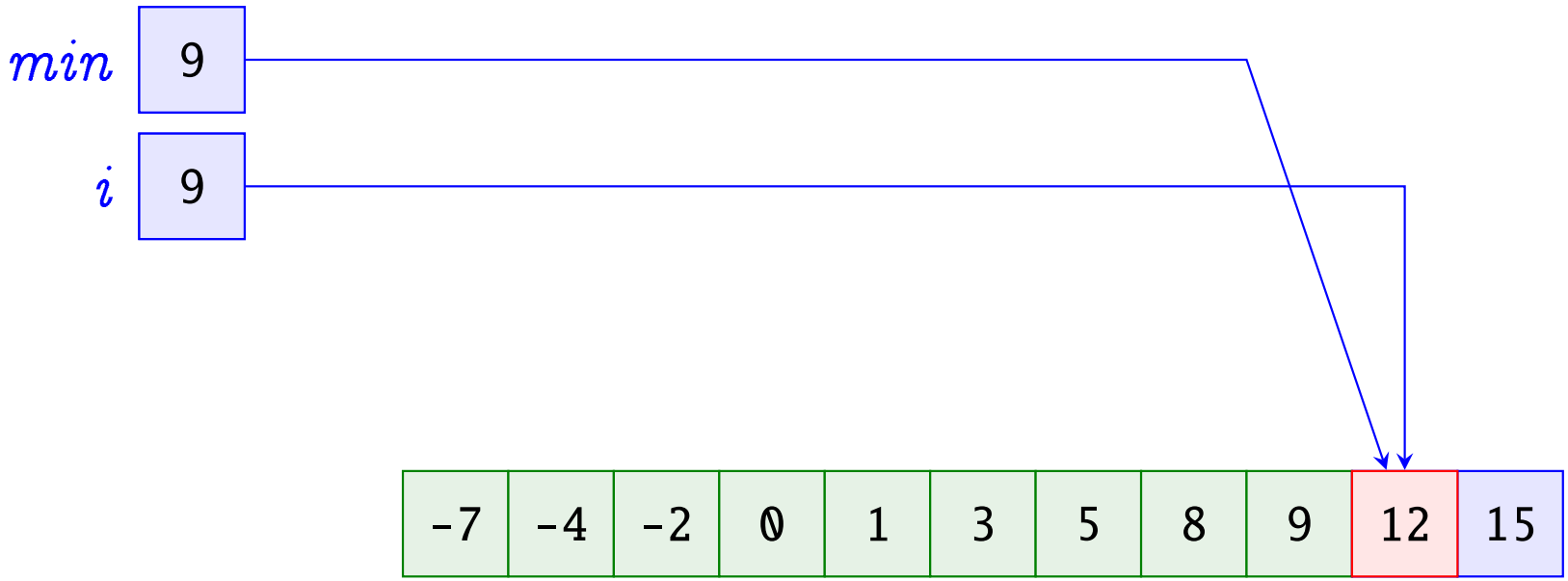
Valikumeetodil sorteerimine



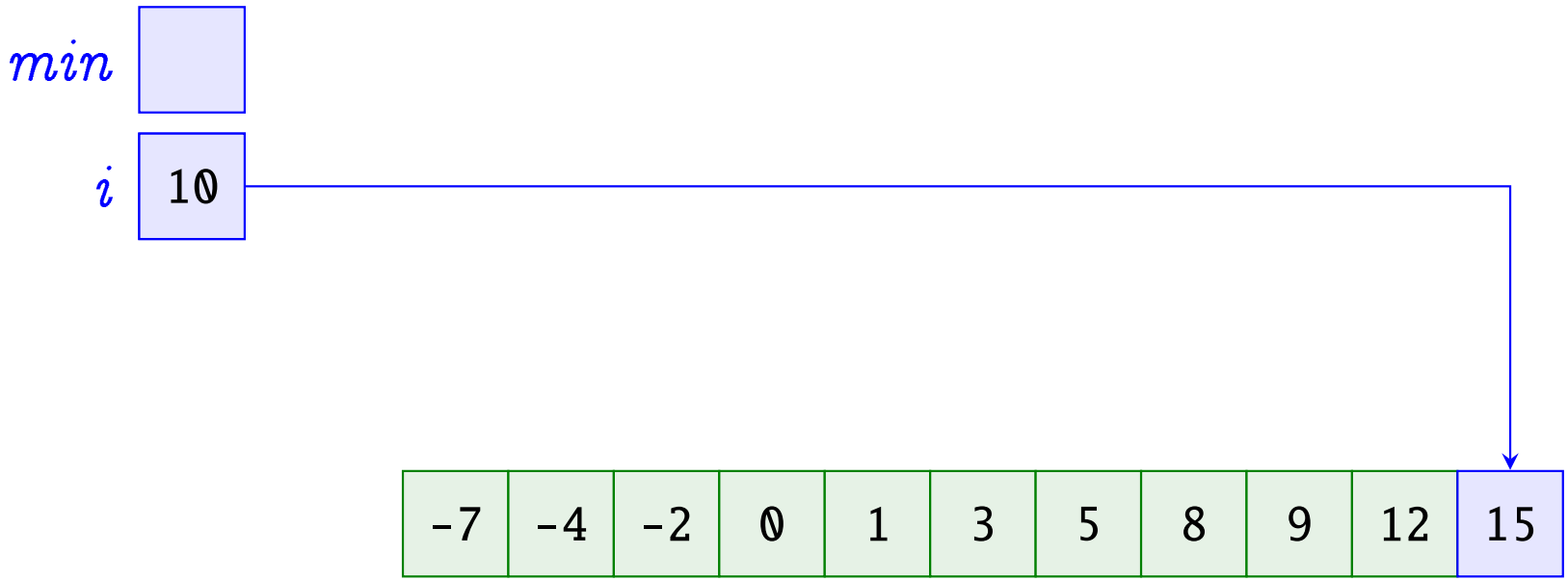
Valikumeetodil sorteerimine



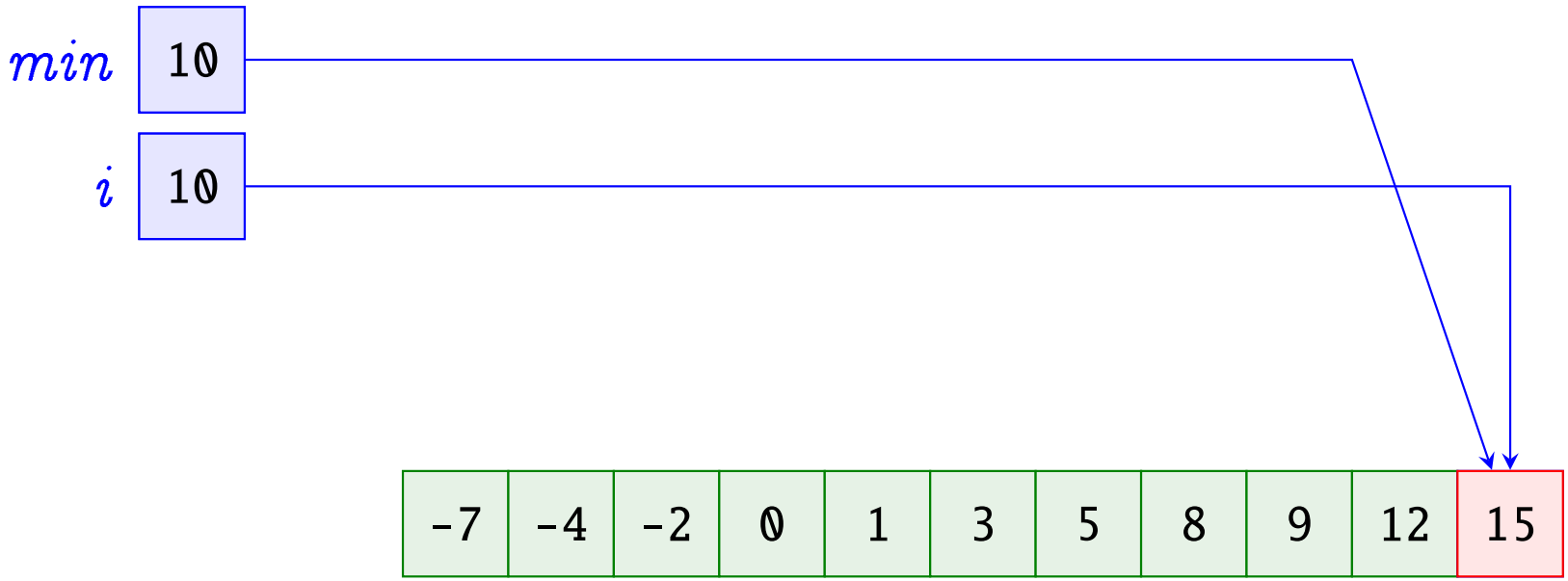
Valikumeetodil sorteerimine



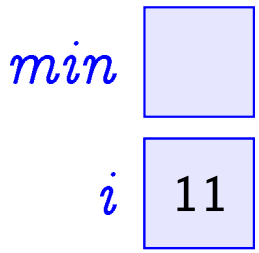
Valikumeetodil sorteerimine



Valikumeetodil sorteerimine



Valikumeetodil sorteerimine



-7	-4	-2	0	1	3	5	8	9	12	15

Valikumeetodil sorteerimine

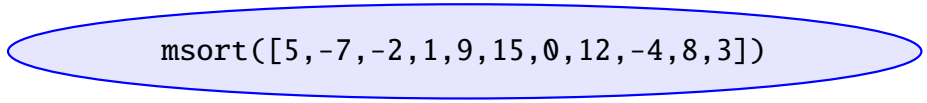
- Valikumeetodil otsitakse igal välisel iteratsioonisammul ülejäänud massiivist vähimat elementi.
- See nõuab kogu ülejäänud massiivi läbimist.
- Seega on valikumeetod ruutkeerukusega $\mathcal{O}(n^2)$ massiivi pikkuse n suhtes.

Ühildusmeetodil sorteerimine

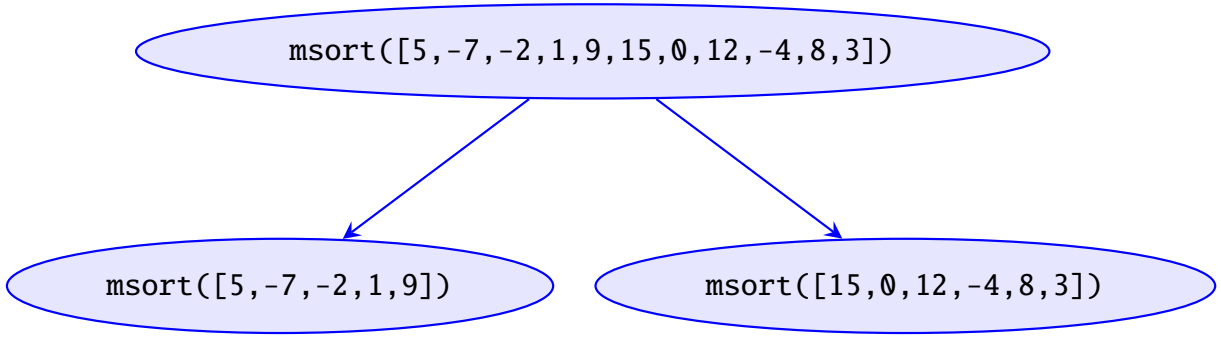
Ühildusmeetodil idee

- Jaotame massiivi kaheks võrdse pikkusega alammassiiviks.
  - Kui massiivis on paaritu arv elemente, siis esimene on ühe võrra lühem.
- Sorteerime alammassiivid rekursiivselt.
  - Rekursiooni baasiks on massiivid mille pikkus on väiksem kui 2.
- Ühildame alammassivid (mis nüüd on juba sorteeritud) üheks terveks sorteeritud massiiviks.

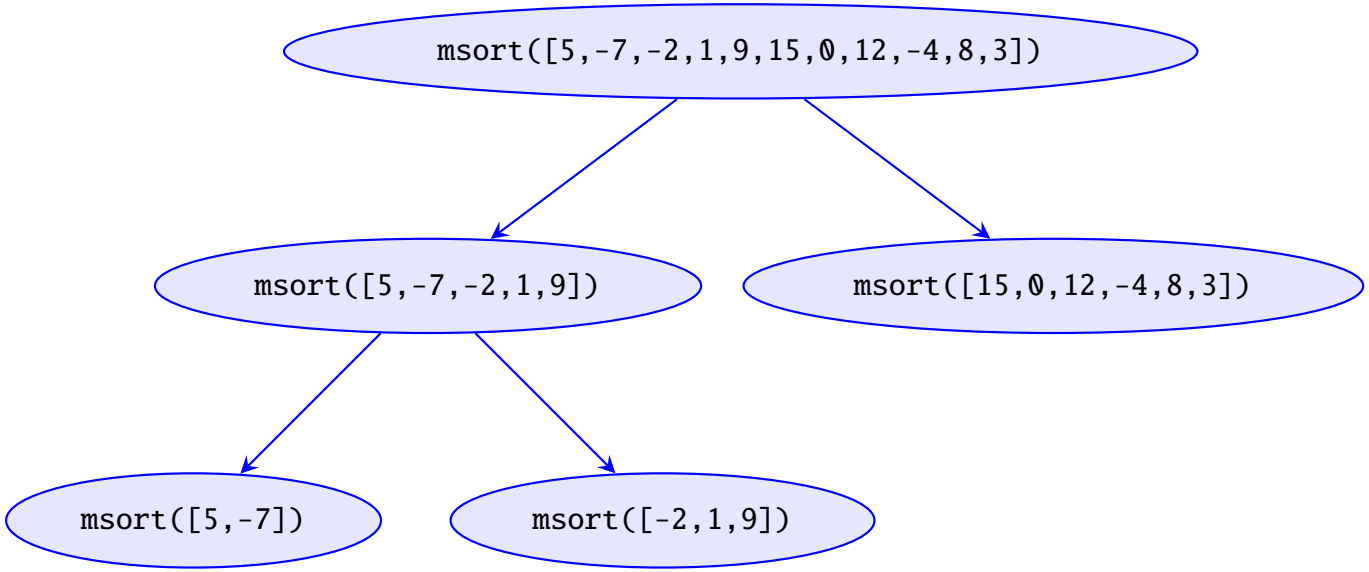
Ühildusmeetodil sorteerimine



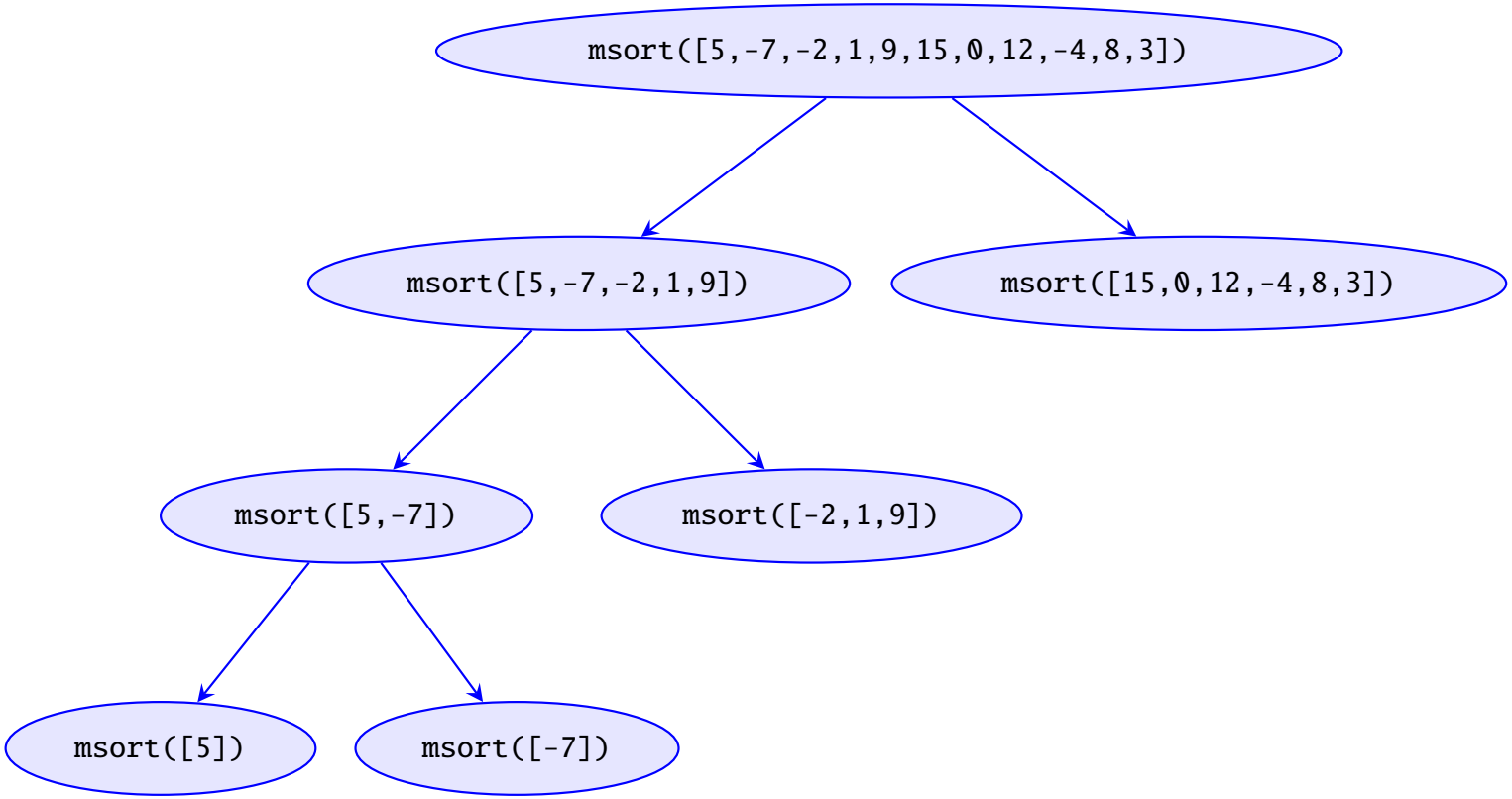
Ühildusmeetodil sorteerimine



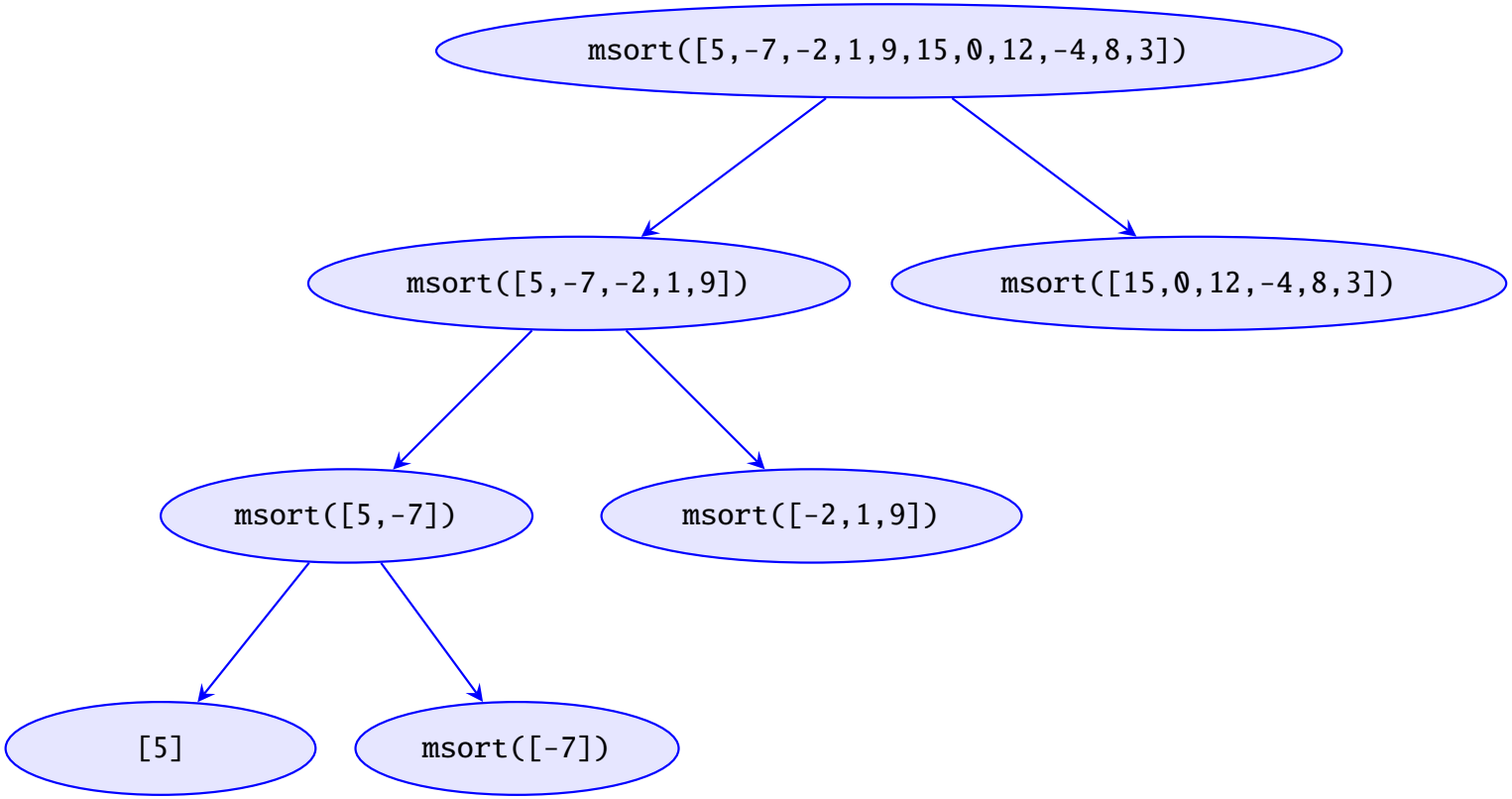
Ühildusmeetodil sorteerimine



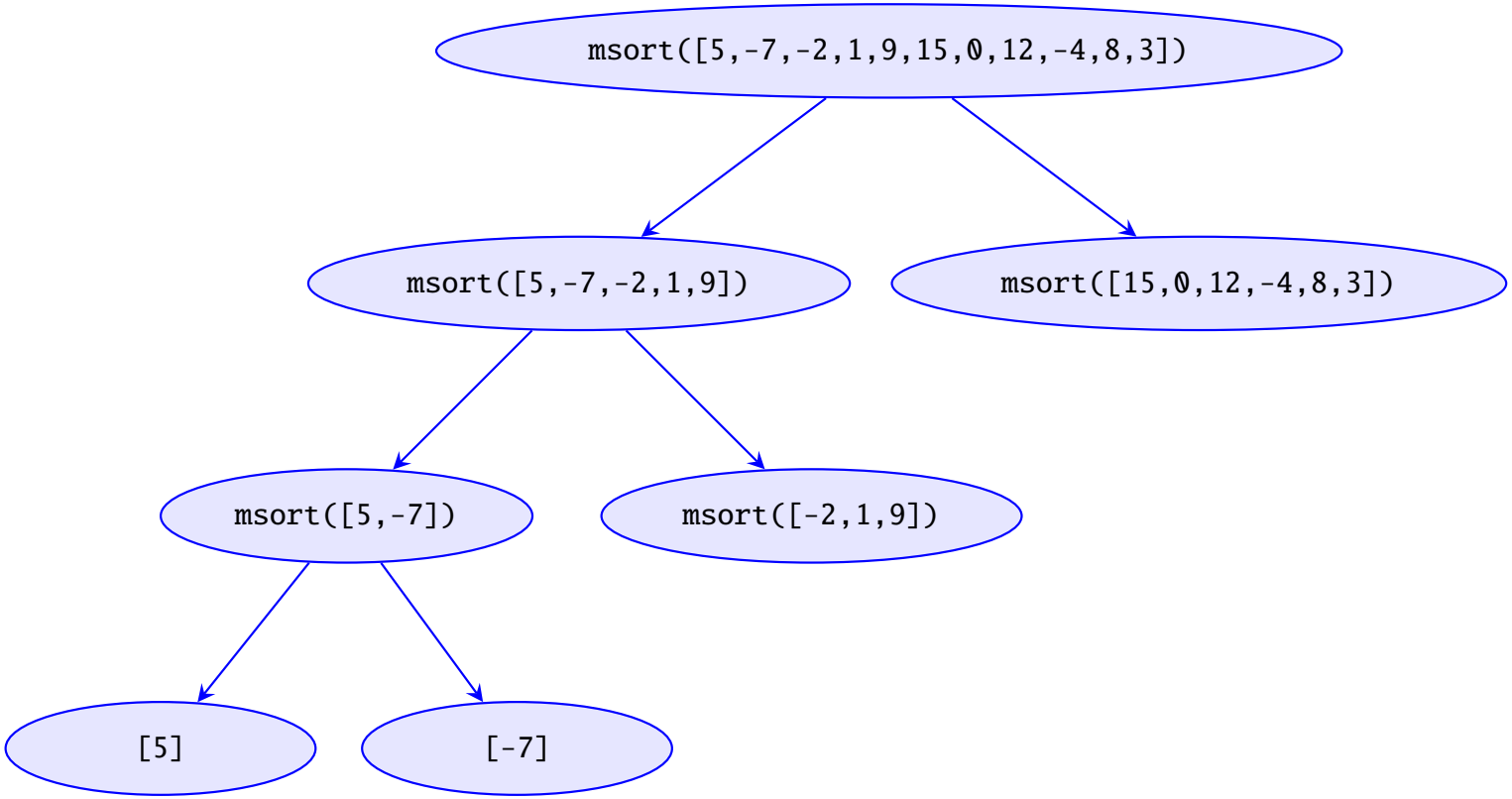
Ühildusmeetodil sorteerimine



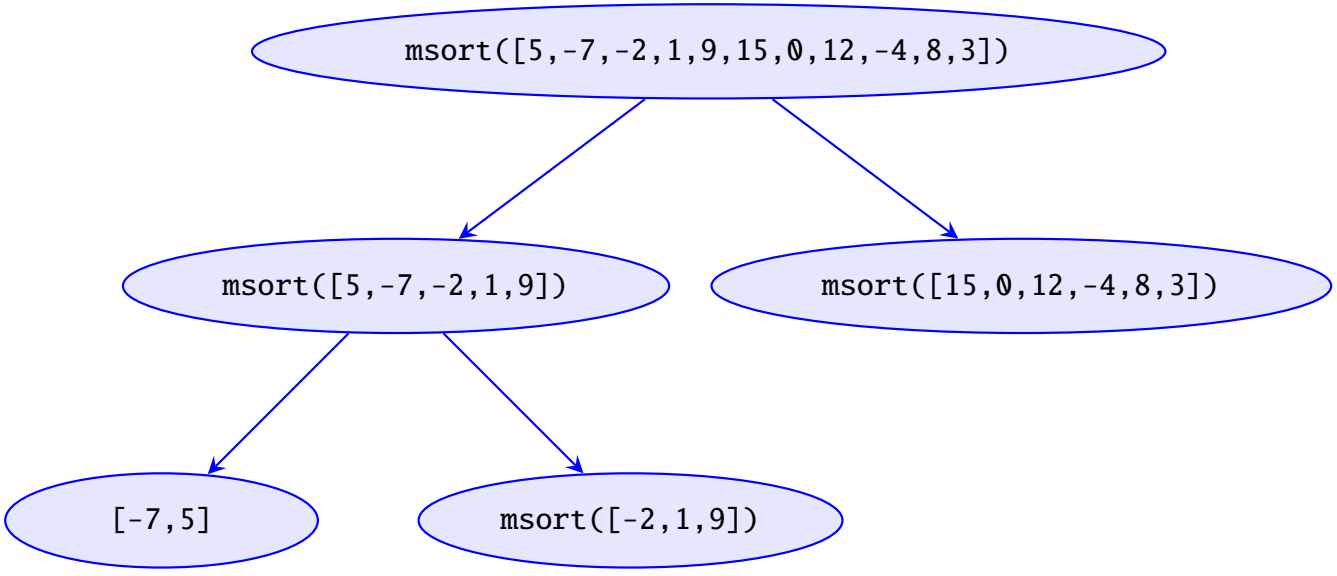
Ühildusmeetodil sorteerimine



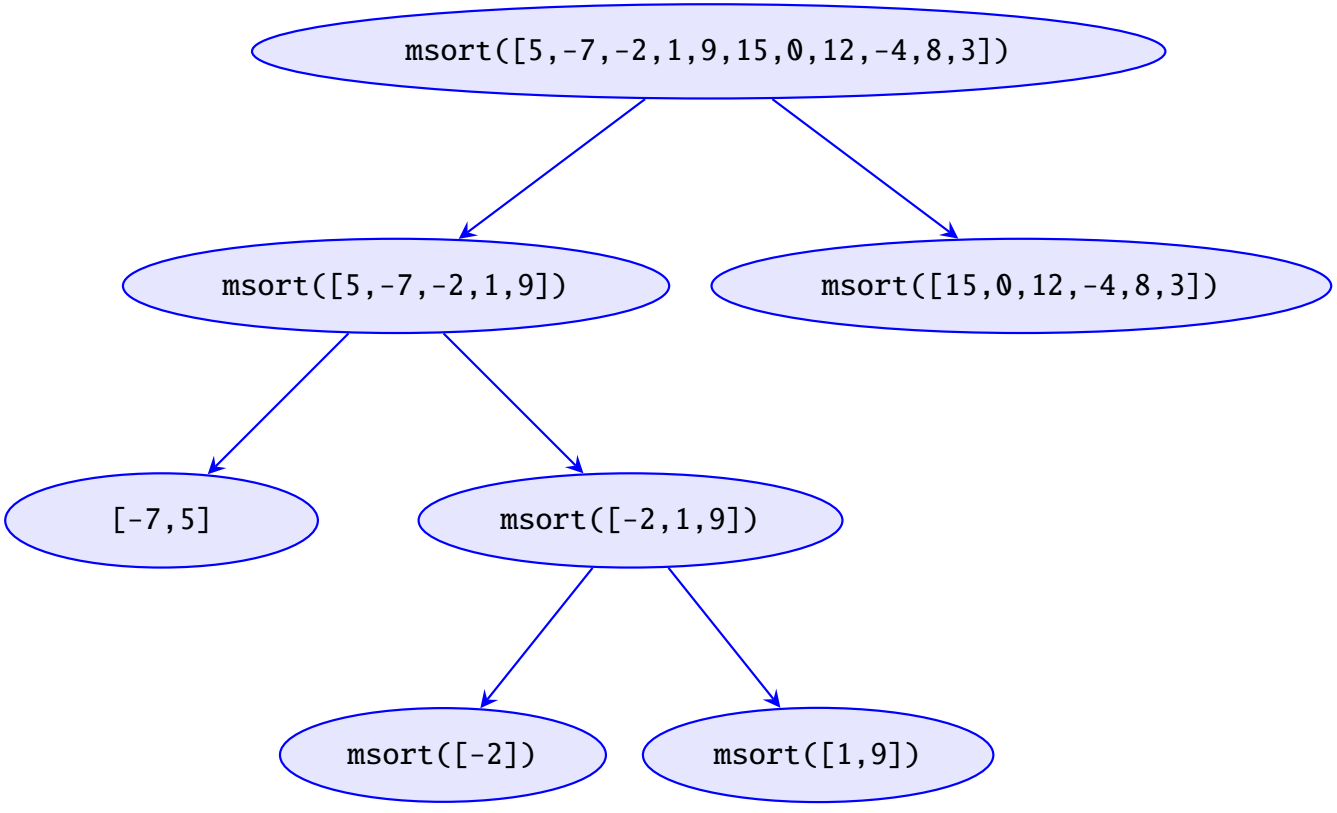
Ühildusmeetodil sorteerimine



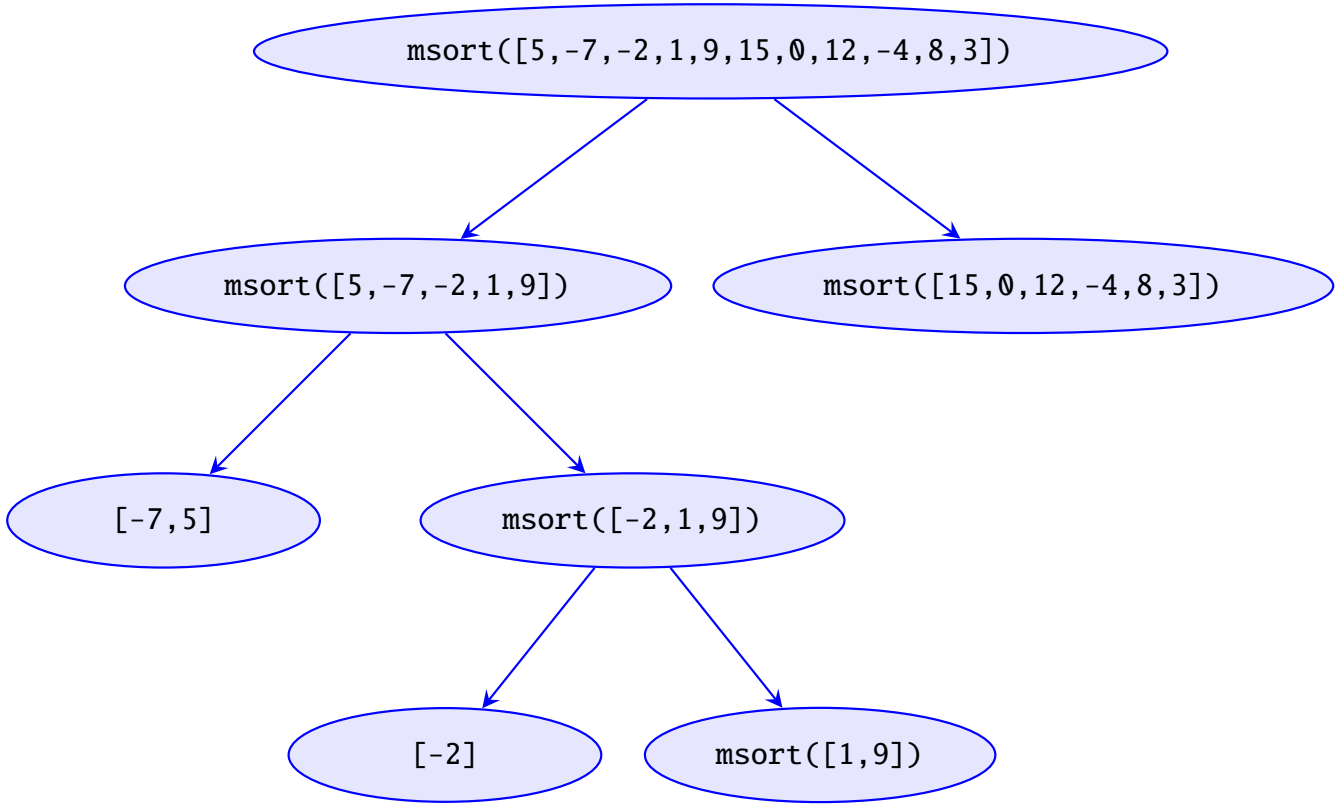
Ühildusmeetodil sorteerimine



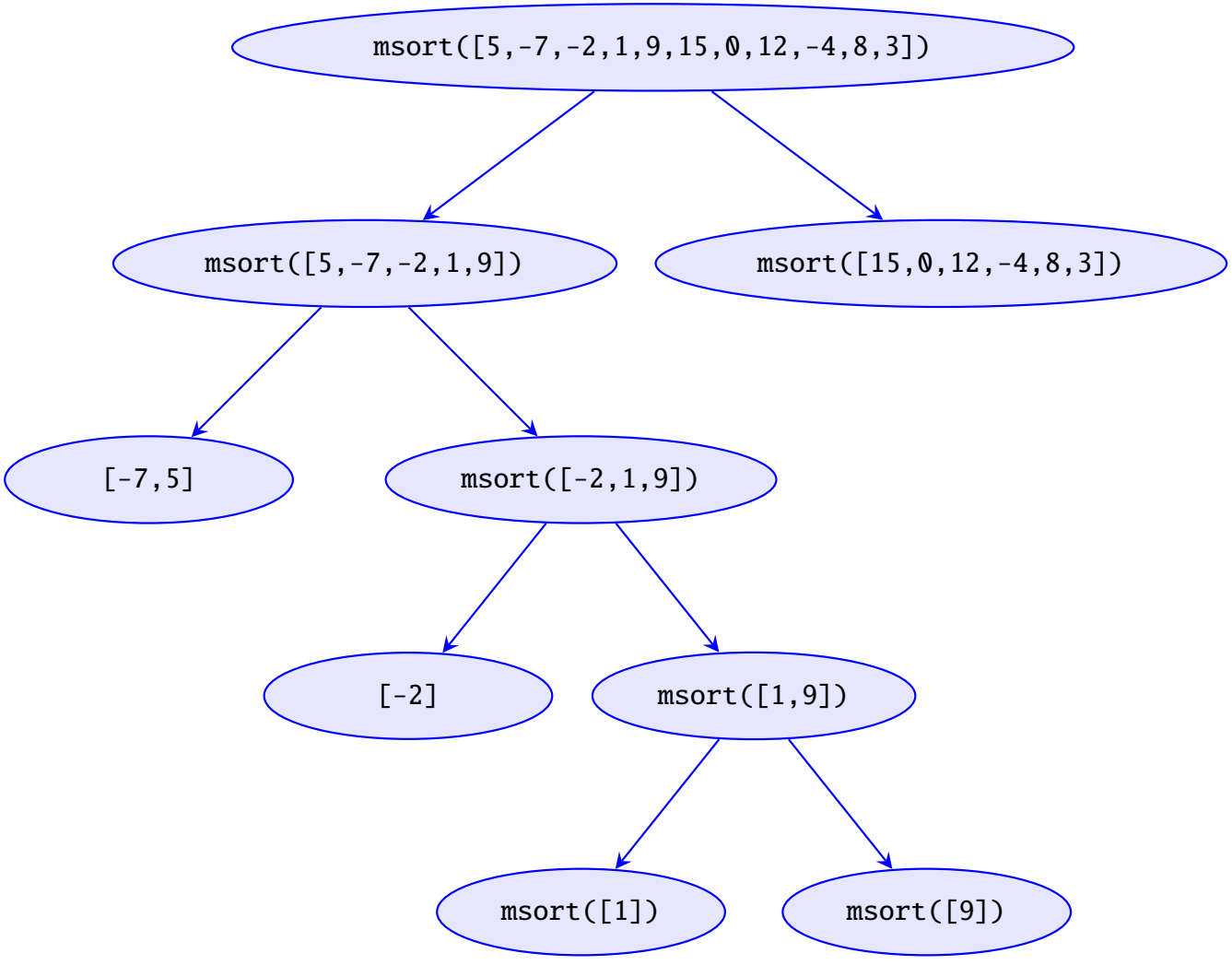
Ühildusmeetodil sorteerimine



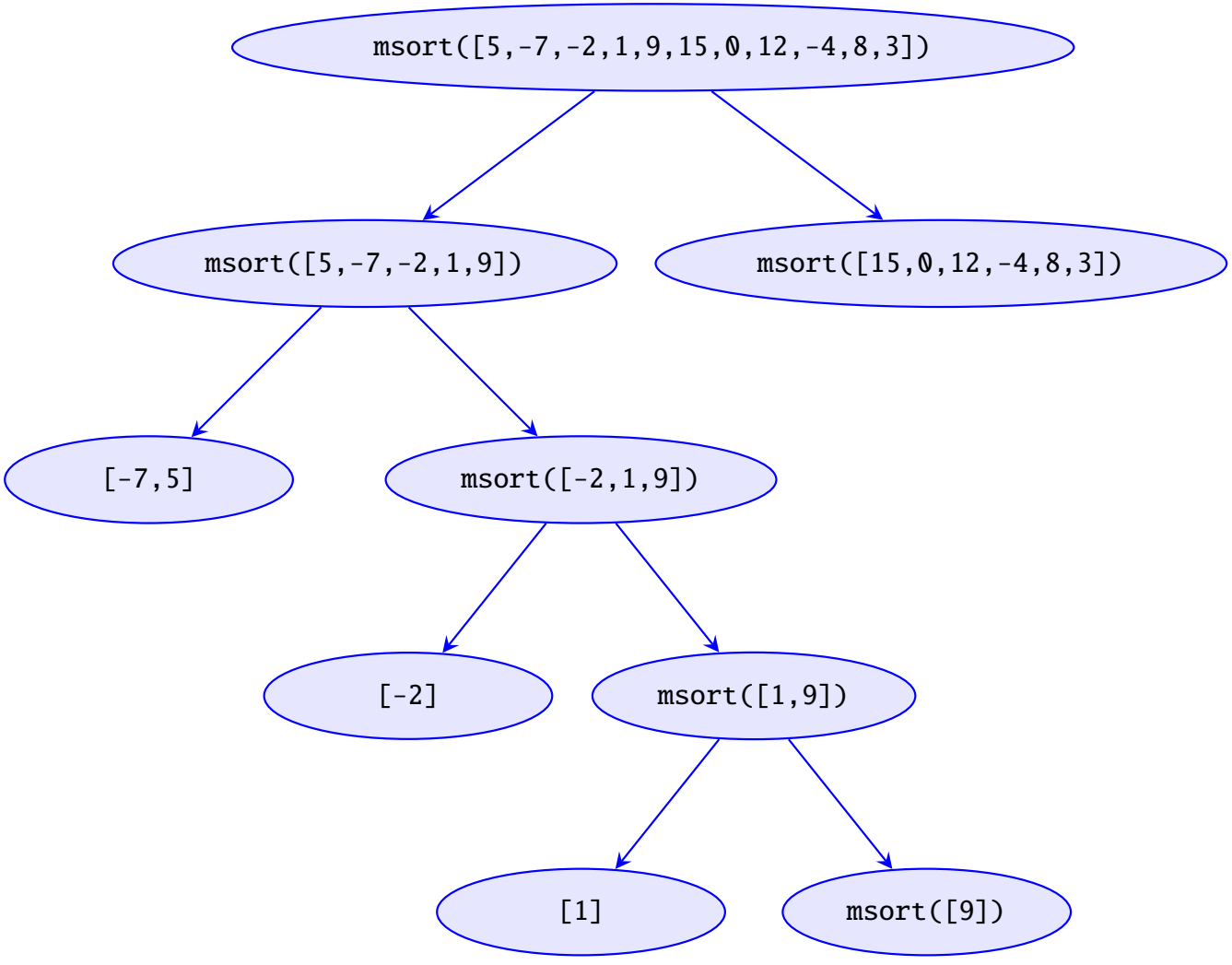
Ühildusmeetodil sorteerimine



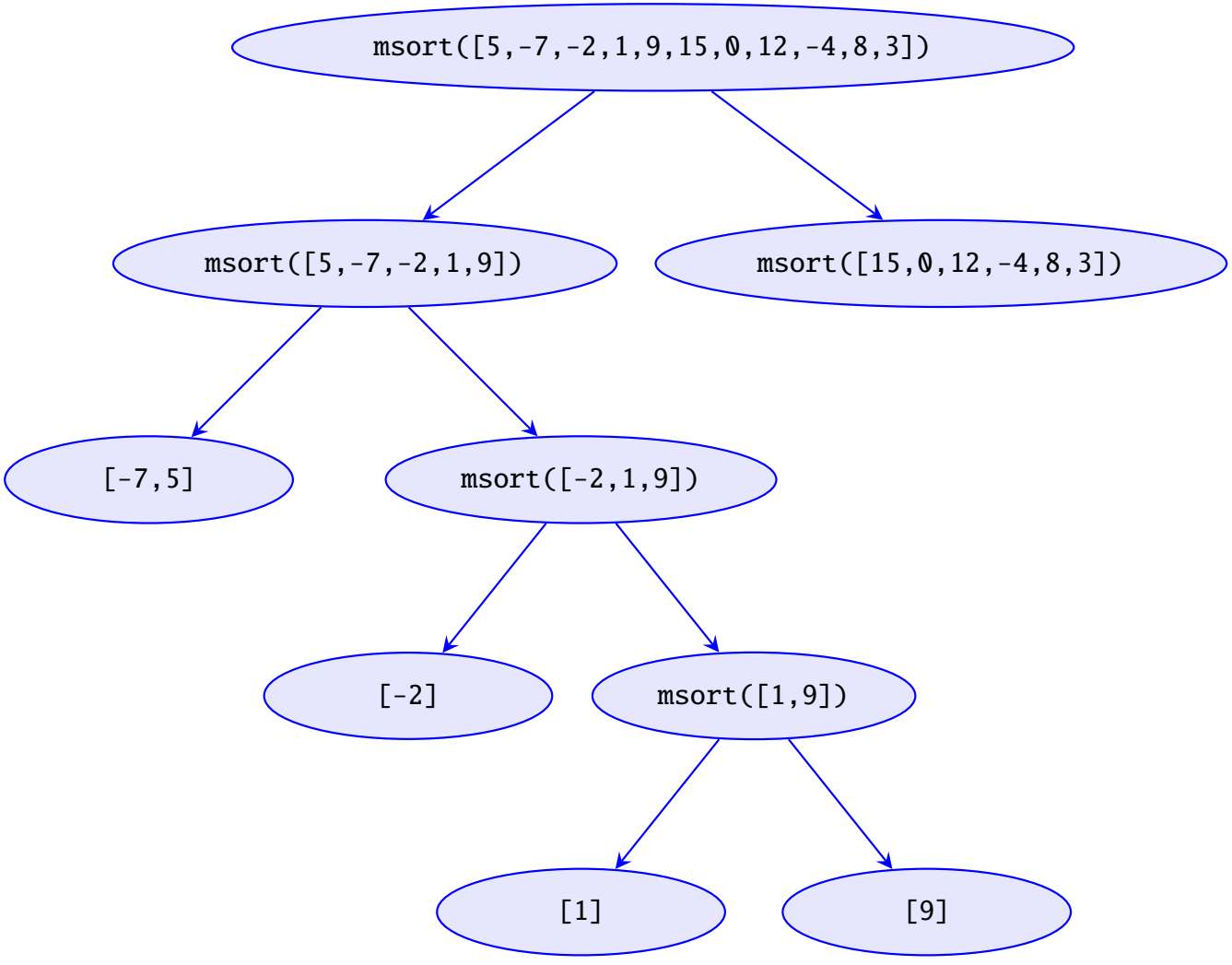
Ühildusmeetodil sorteerimine



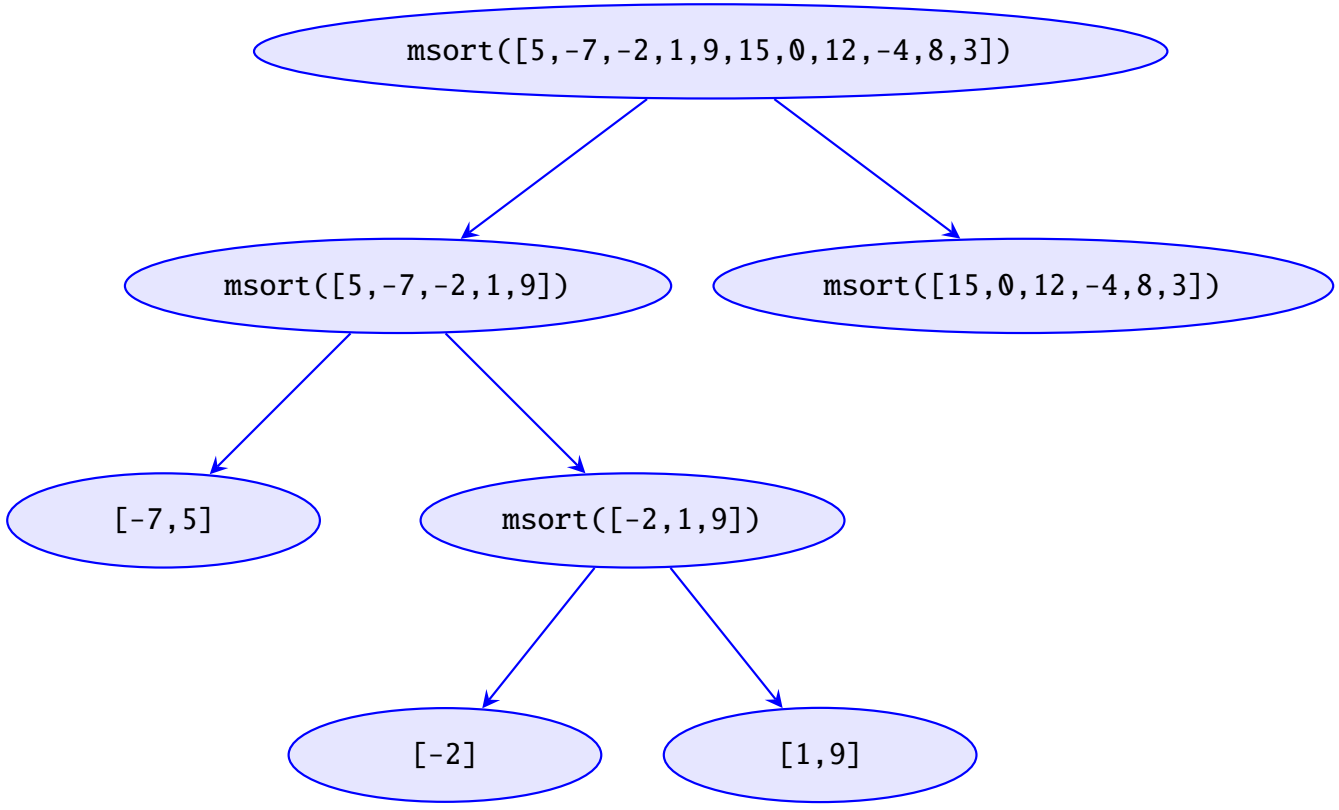
Ühildusmeetodil sorteerimine



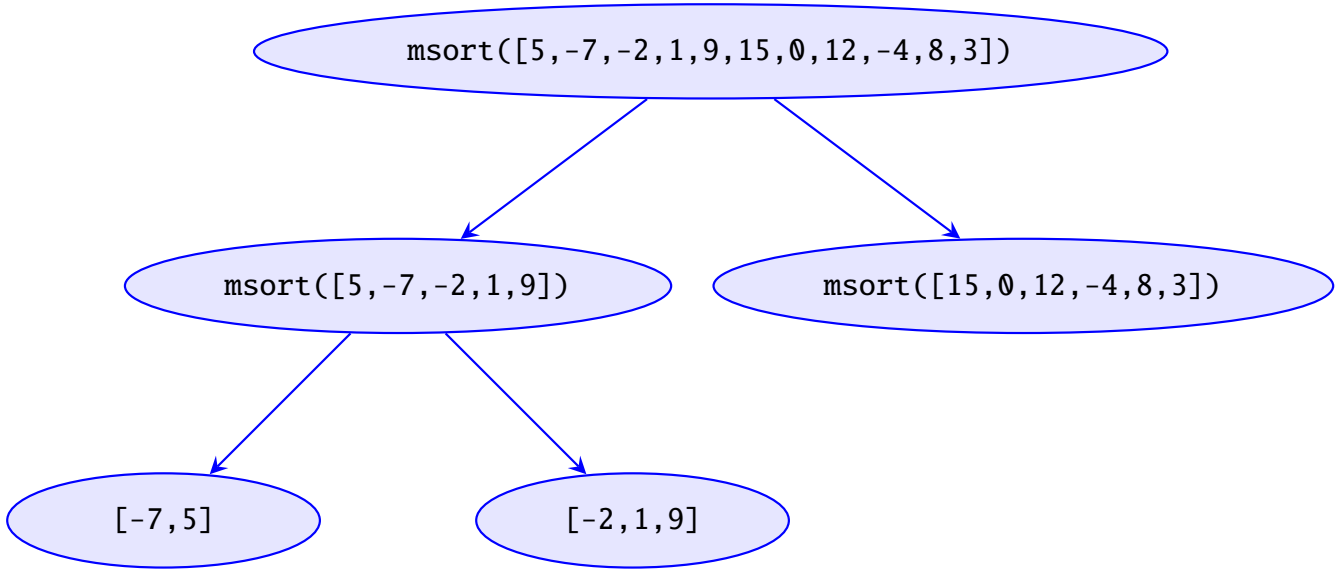
Ühildusmeetodil sorteerimine



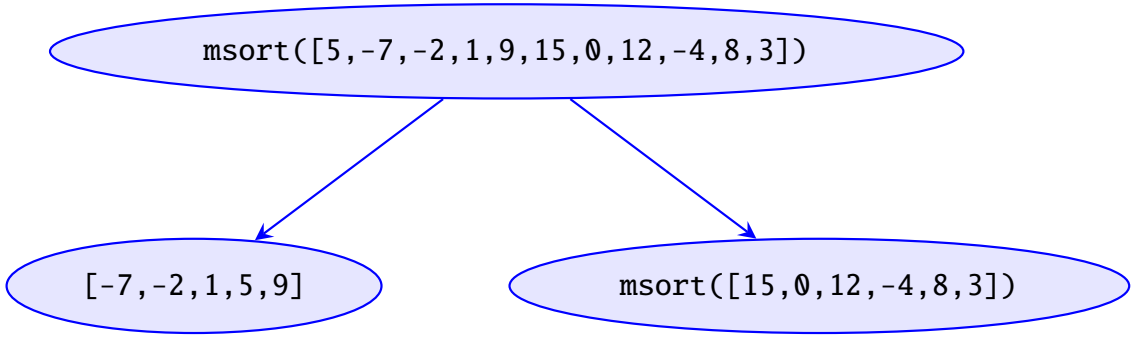
Ühildusmeetodil sorteerimine



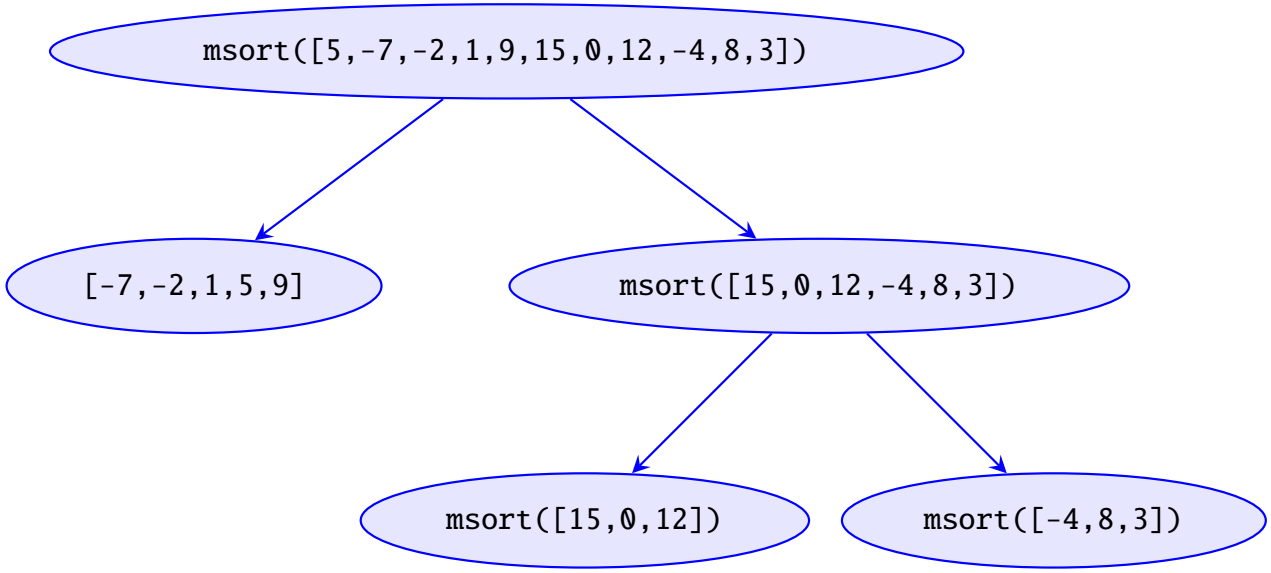
Ühildusmeetodil sorteerimine



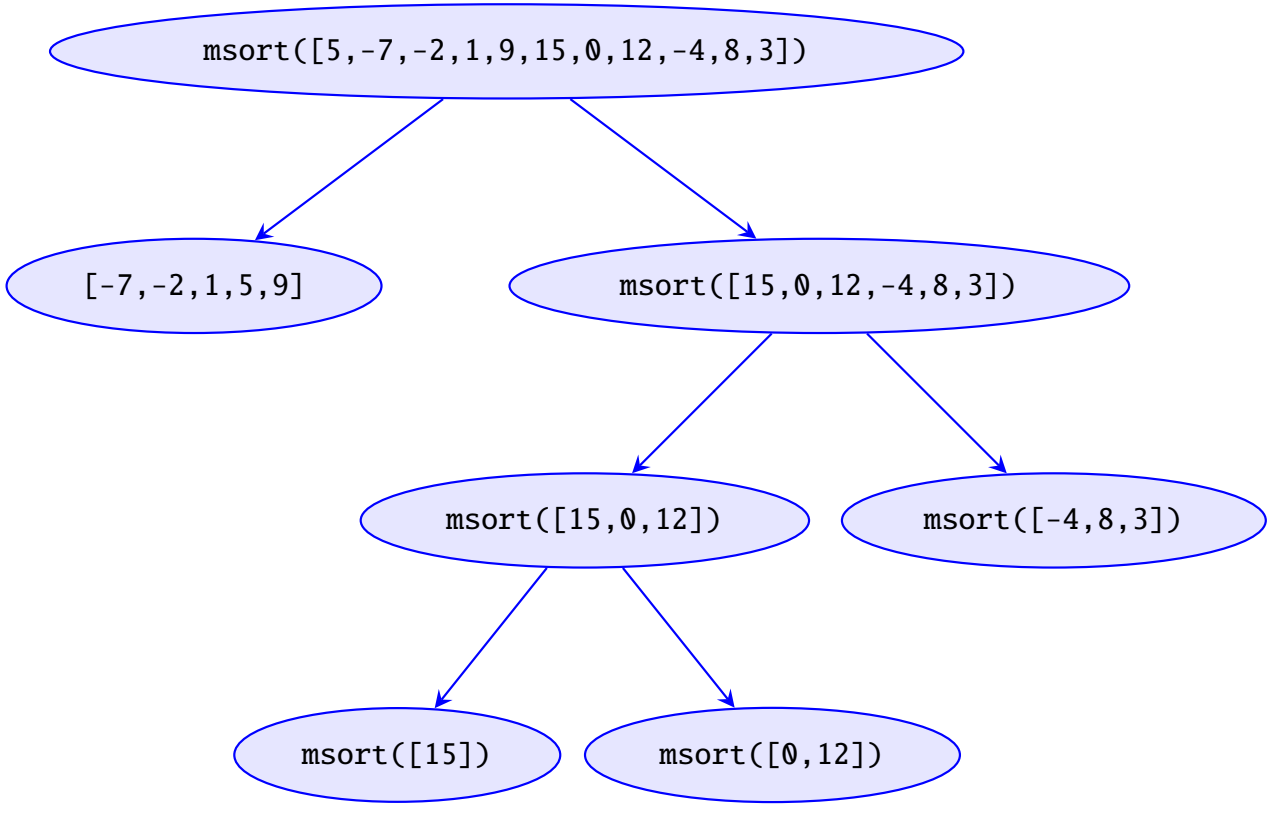
Ühildusmeetodil sorteerimine



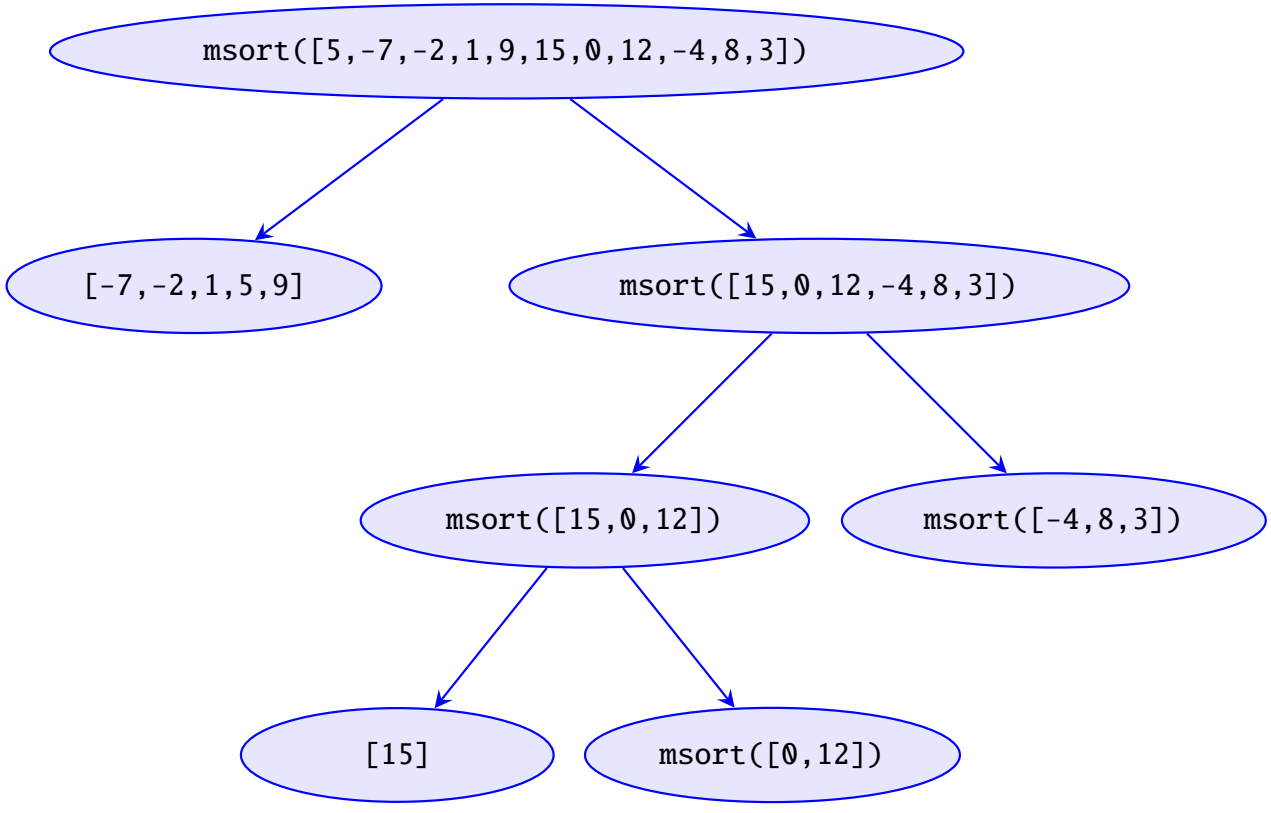
Ühildusmeetodil sorteerimine



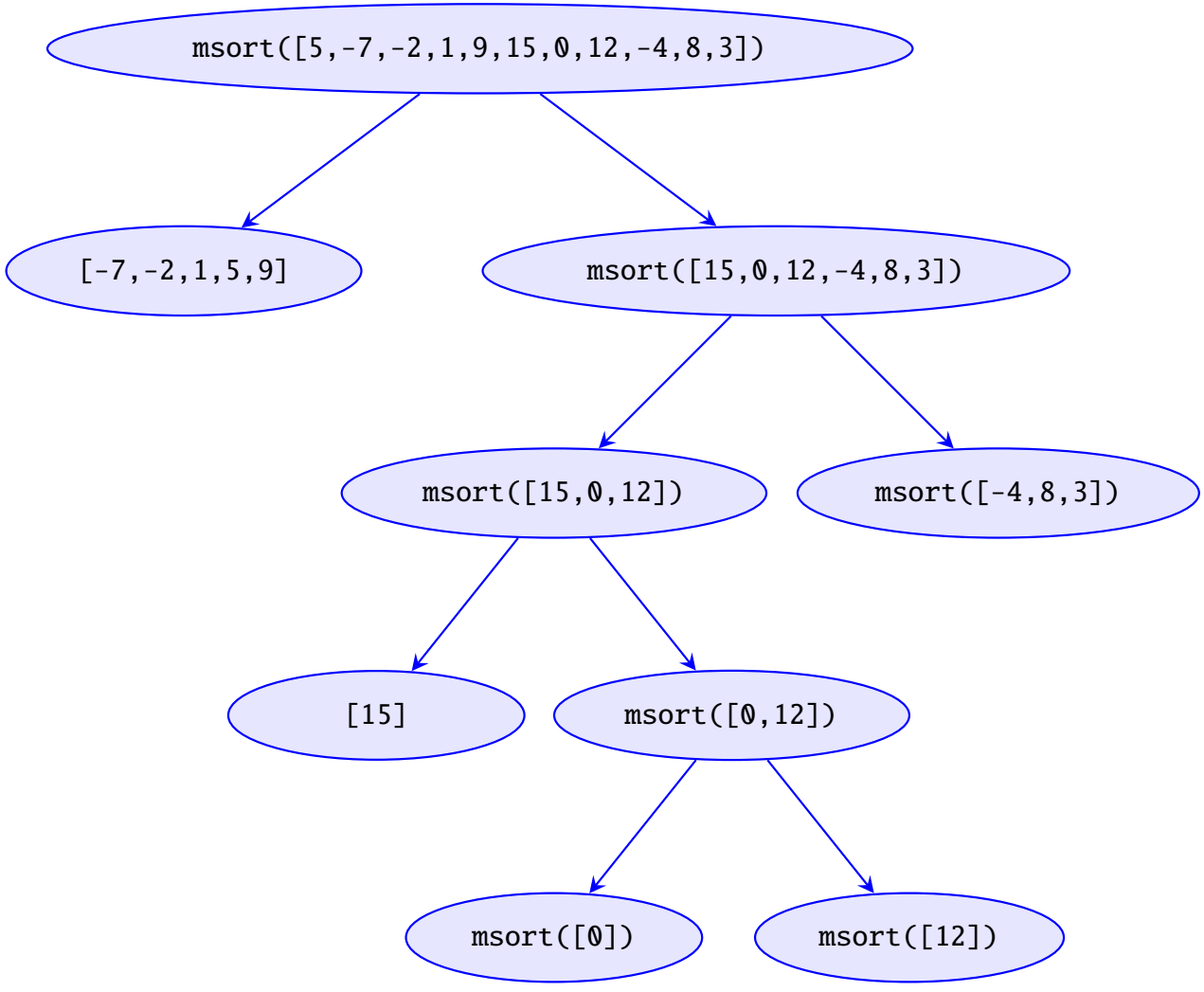
Ühildusmeetodil sorteerimine



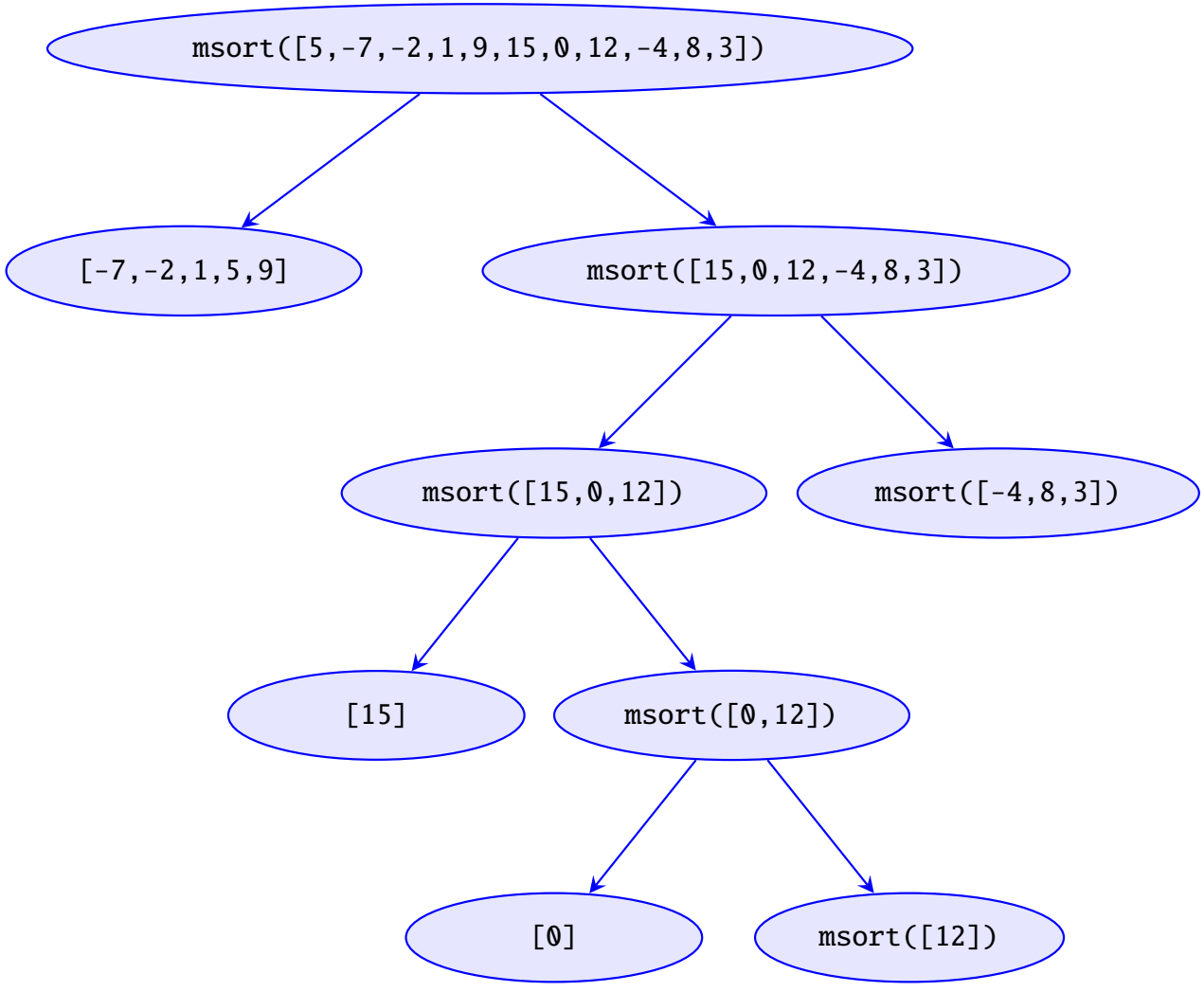
Ühildusmeetodil sorteerimine



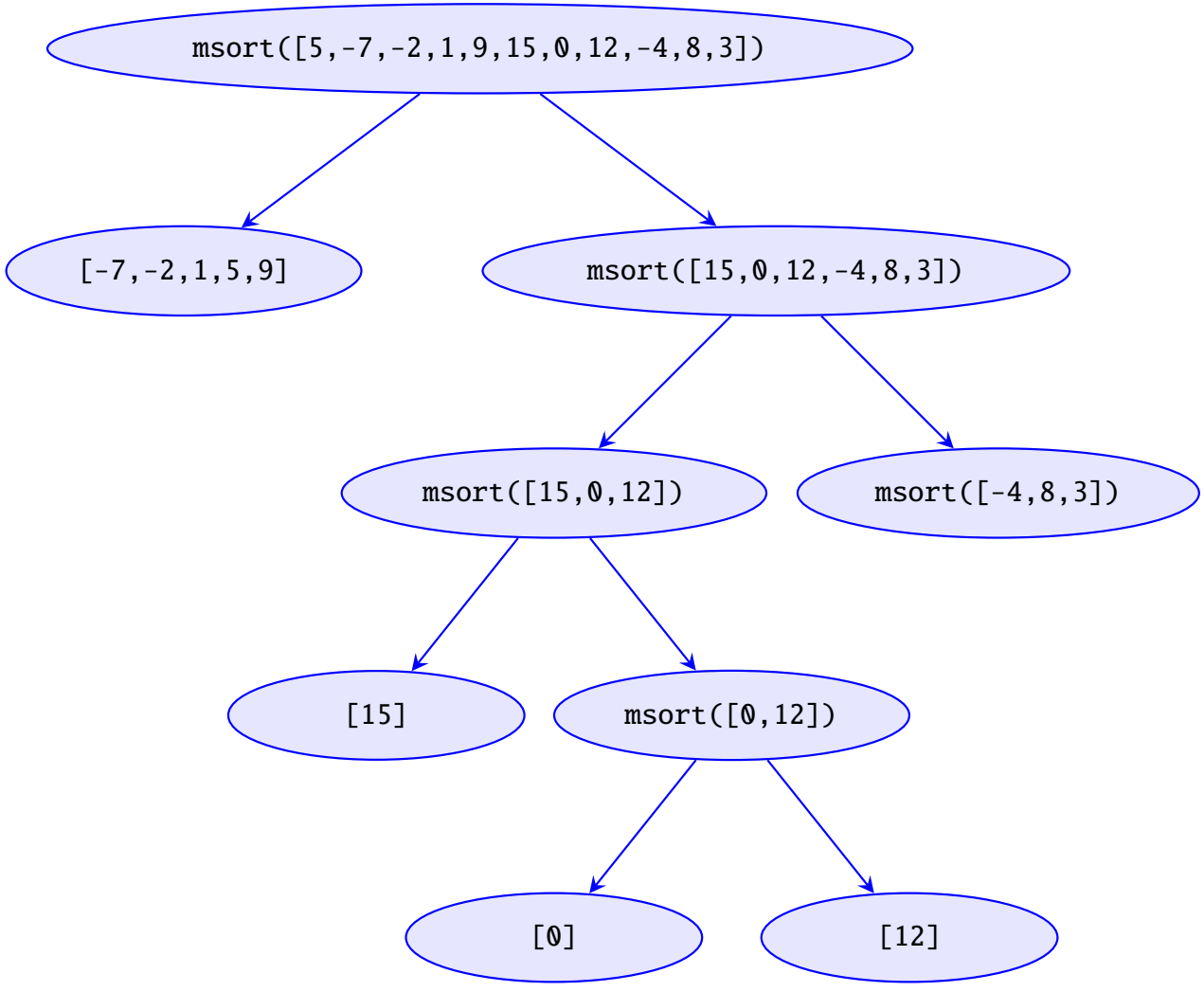
Ühildusmeetodil sorteerimine



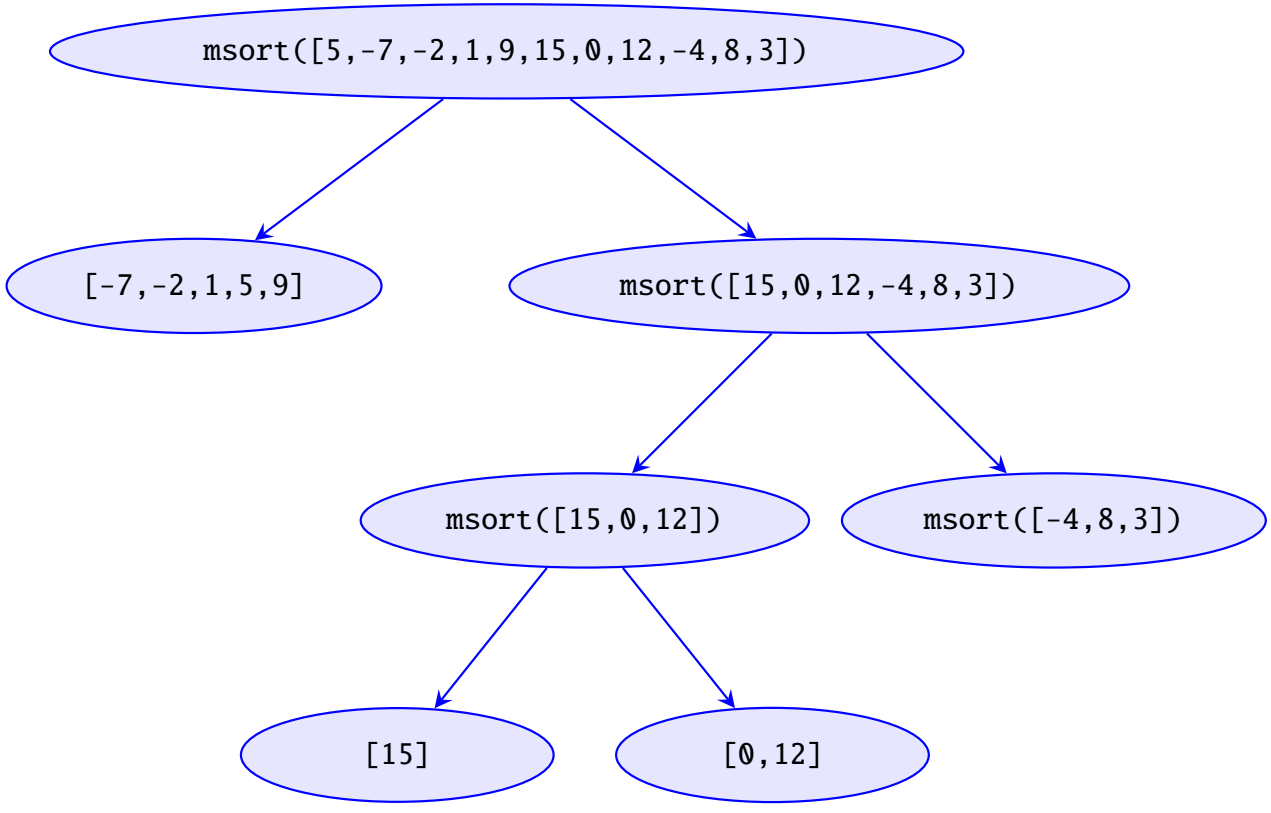
Ühildusmeetodil sorteerimine



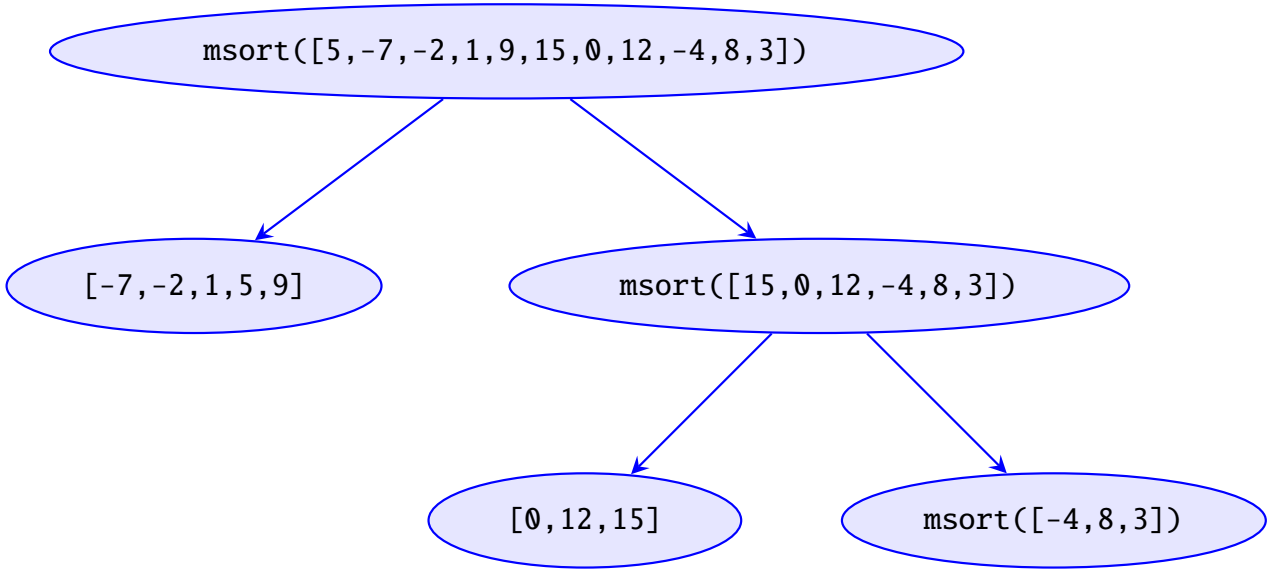
Ühildusmeetodil sorteerimine



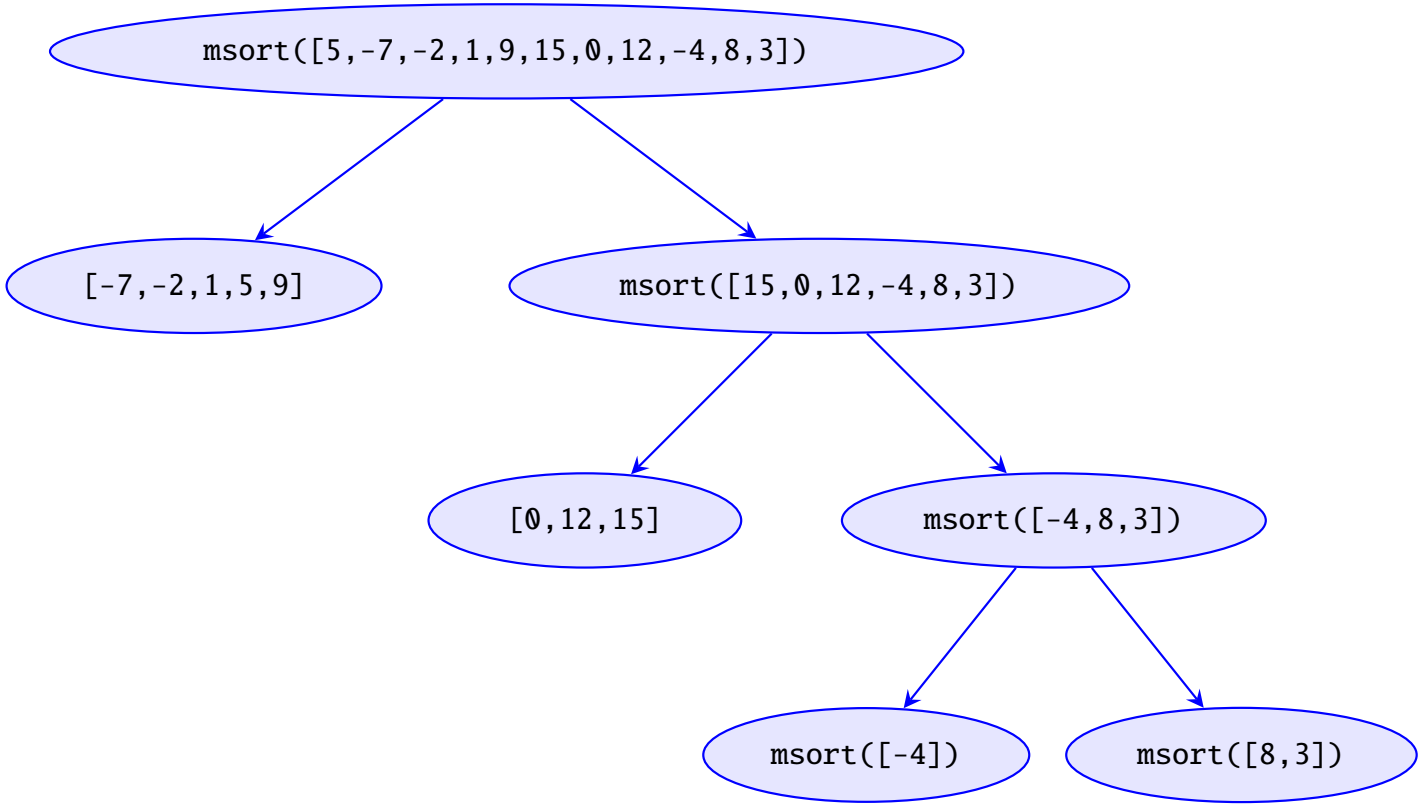
Ühildusmeetodil sorteerimine



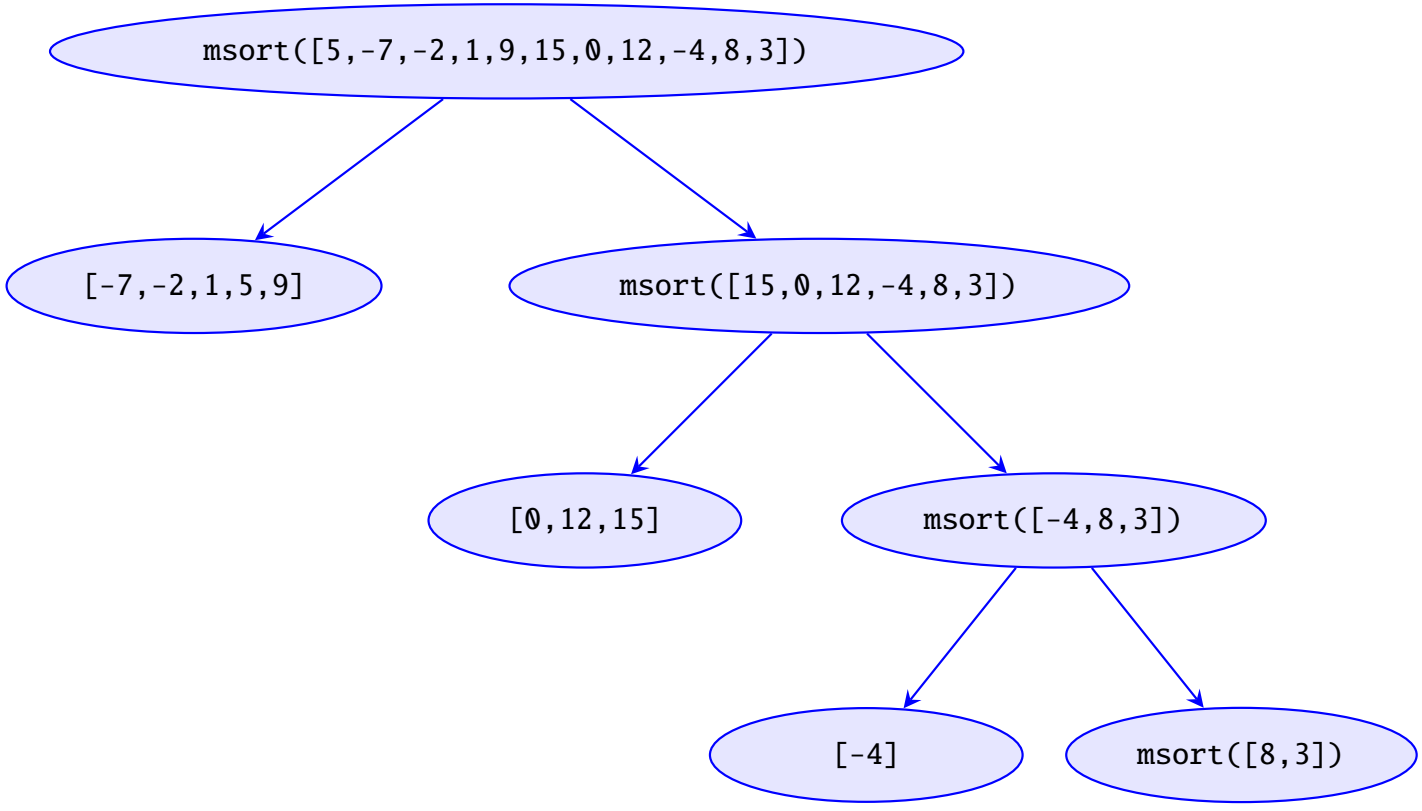
Ühildusmeetodil sorteerimine



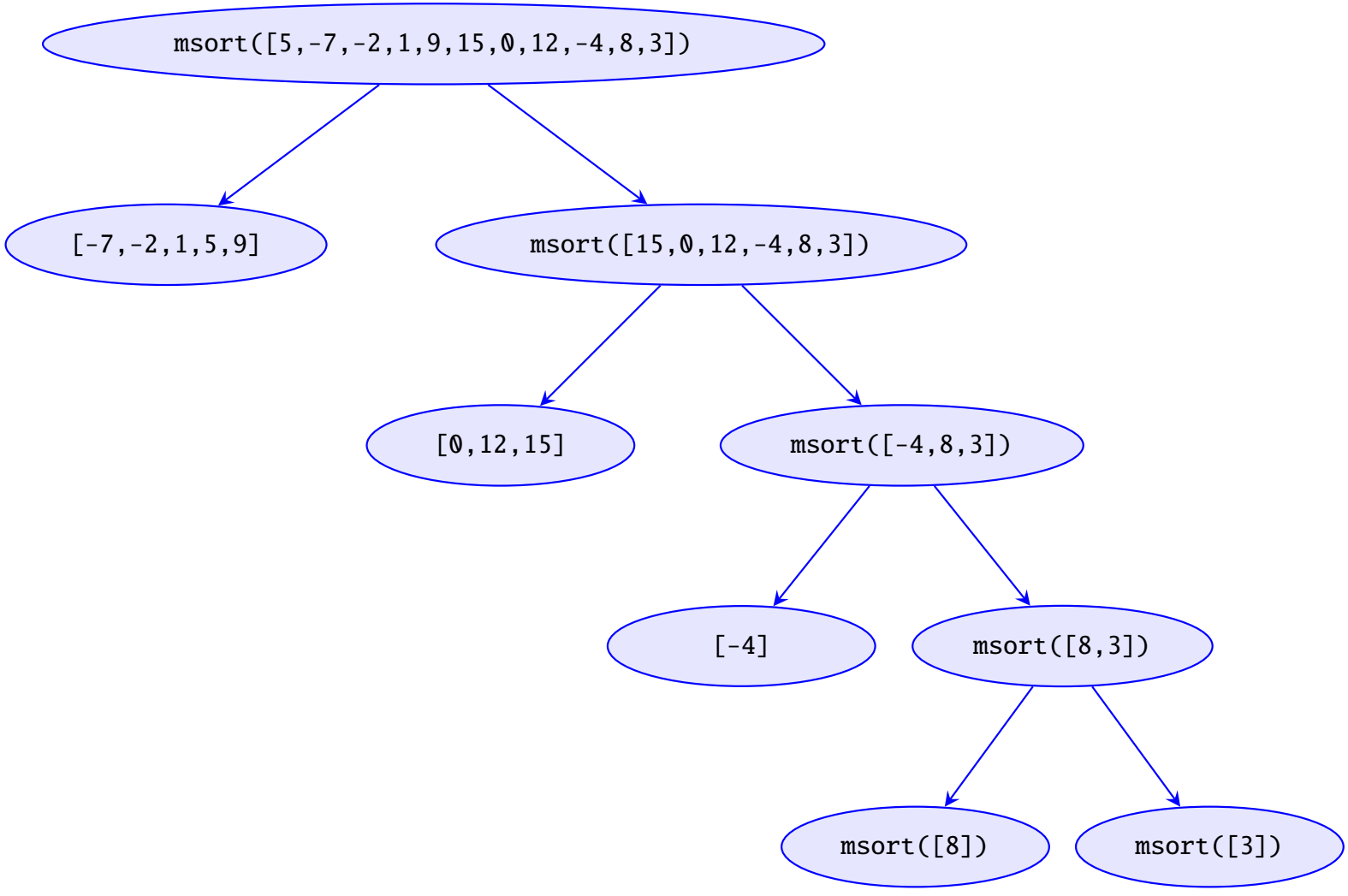
Ühildusmeetodil sorteerimine



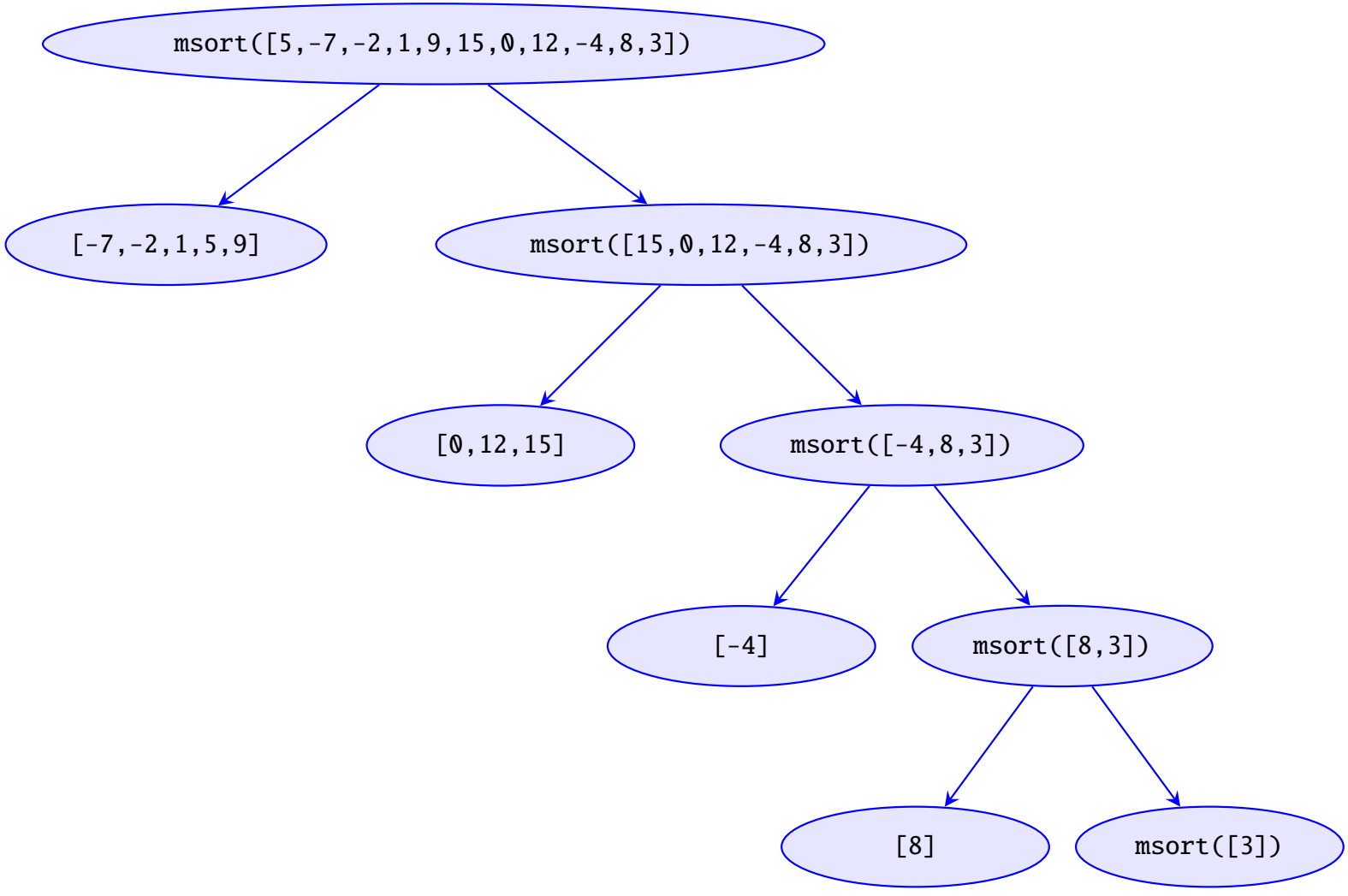
Ühildusmeetodil sorteerimine



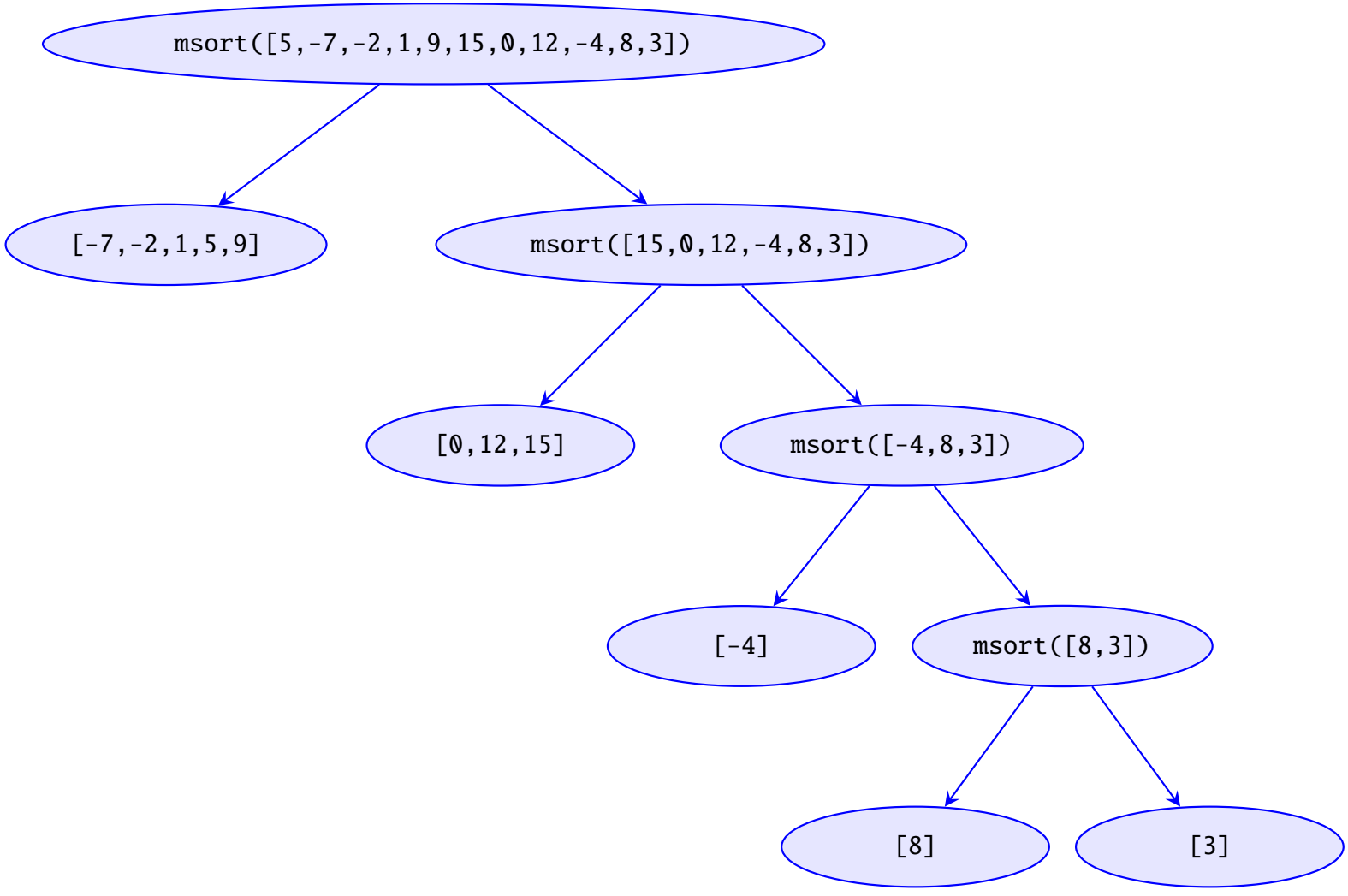
Ühildusmeetodil sorteerimine



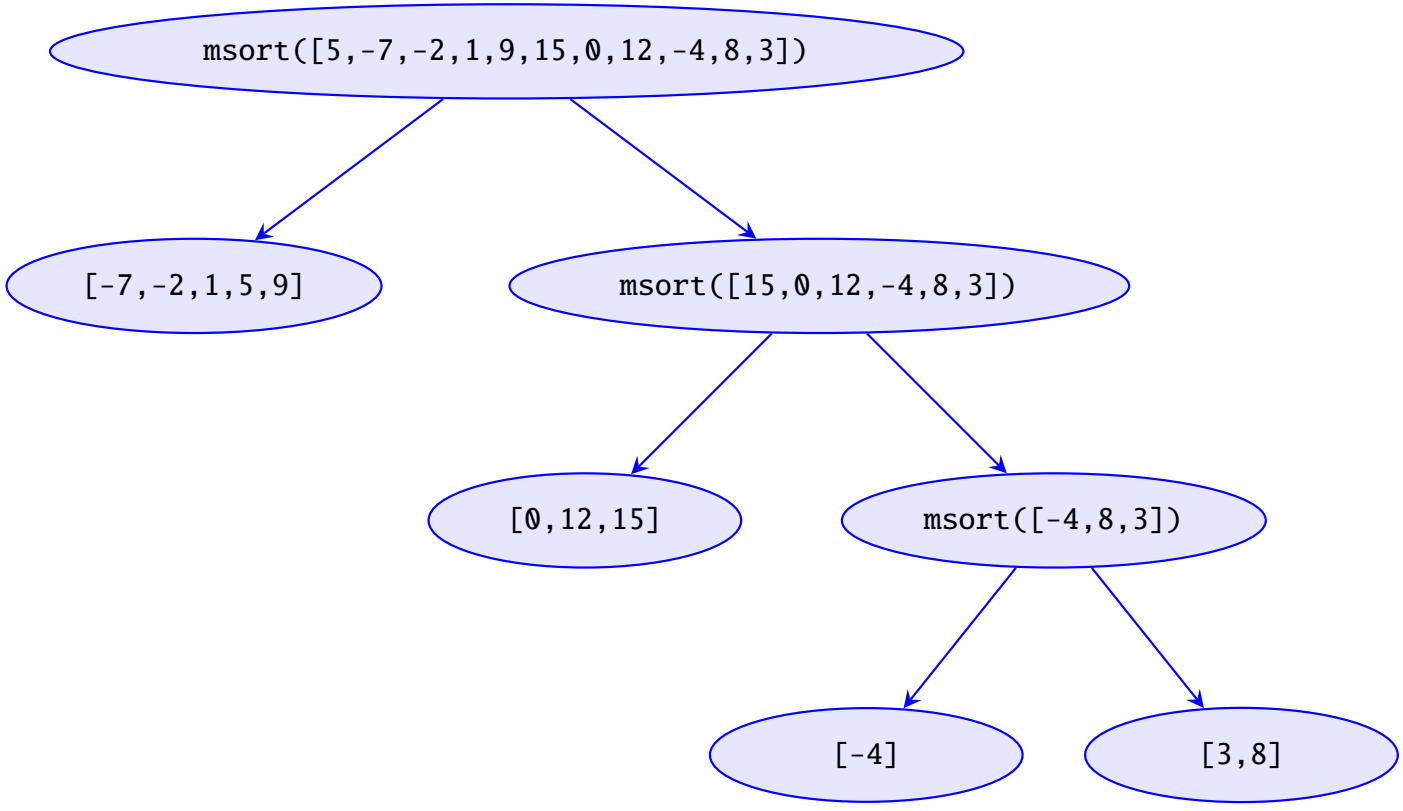
Ühildusmeetodil sorteerimine



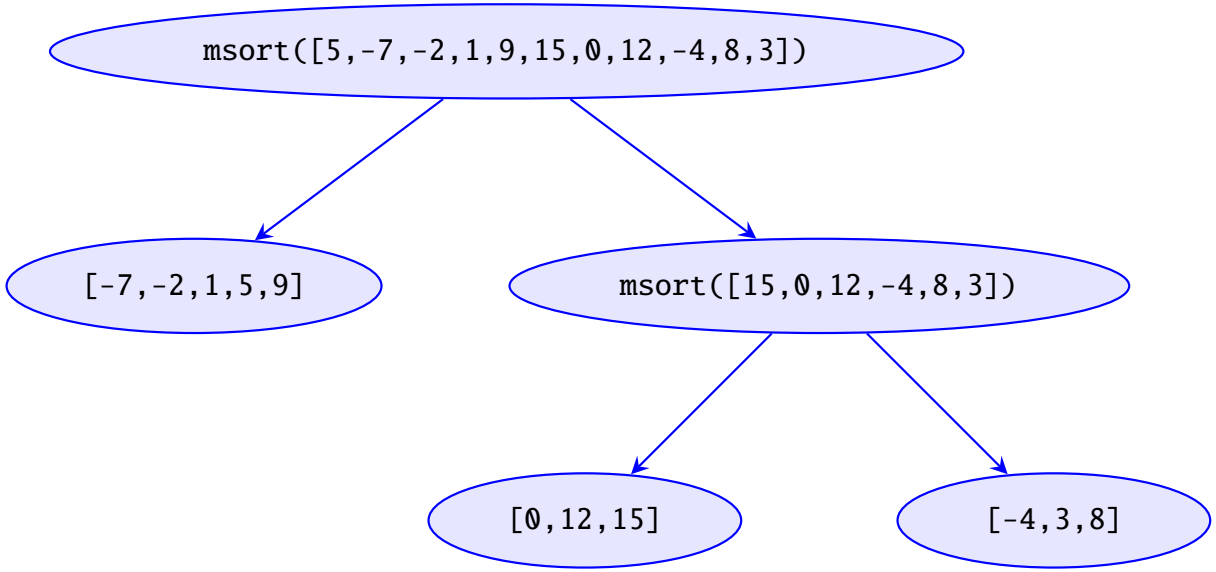
Ühildusmeetodil sorteerimine



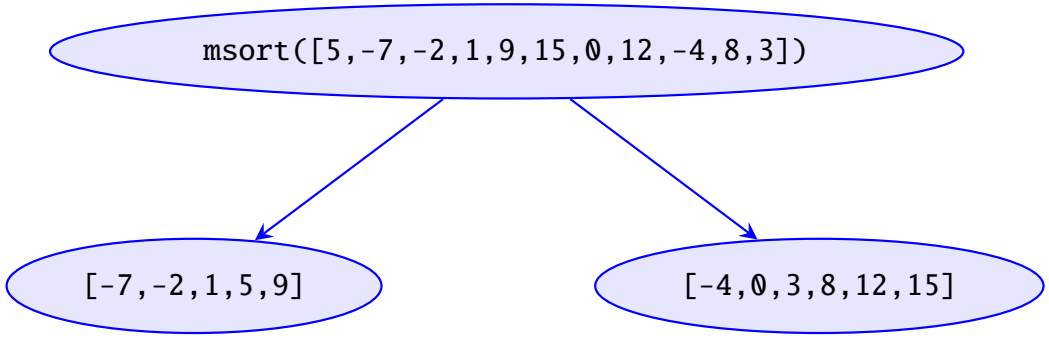
Ühildusmeetodil sorteerimine



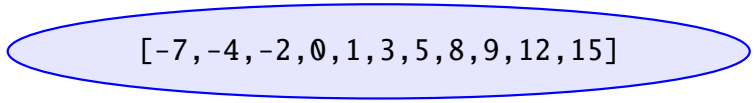
Ühildusmeetodil sorteerimine



Ühildusmeetodil sorteerimine



Ühildusmeetodil sorteerimine



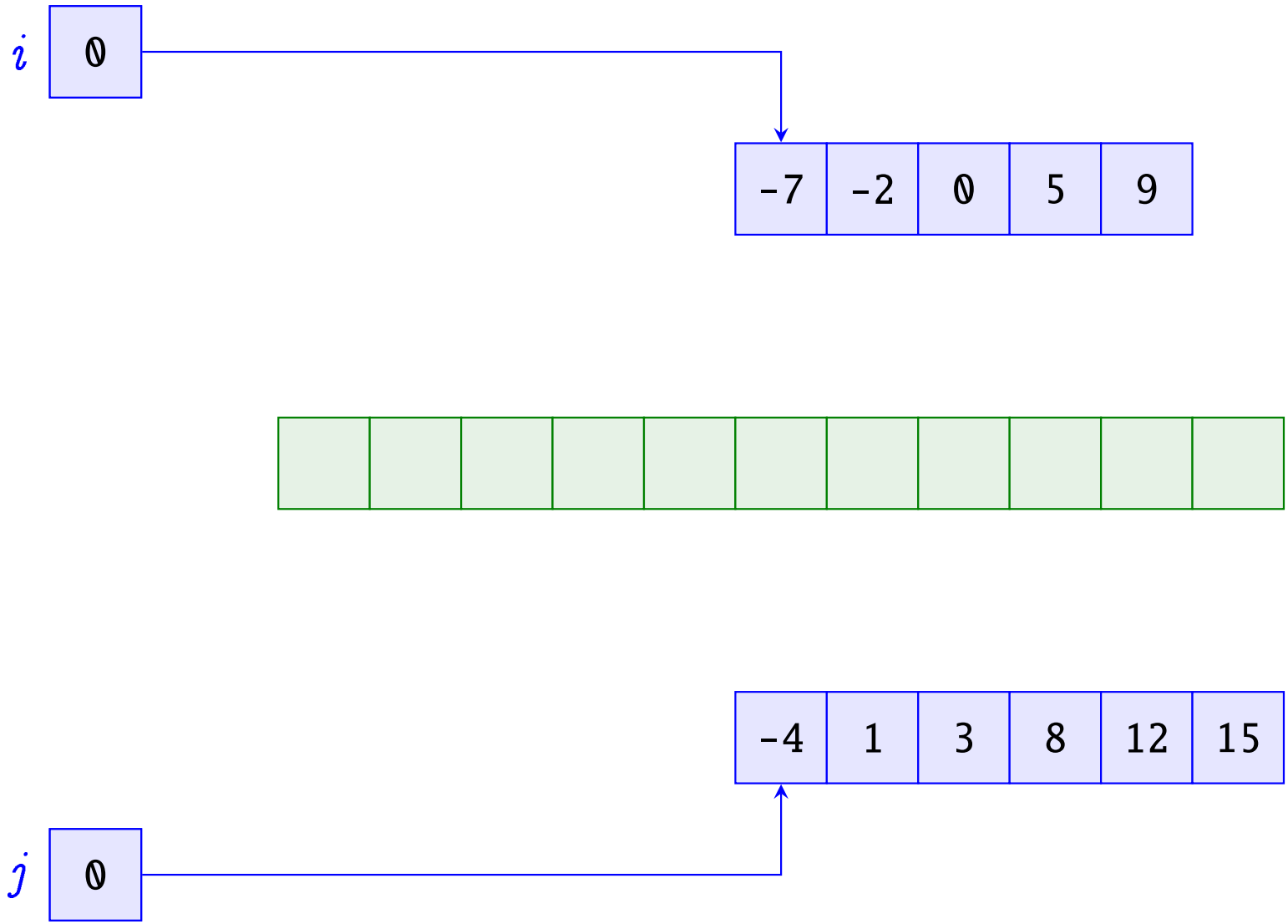
Ühildusmeetodil sorteerimine

Ühildusmeetodil sorteerimine

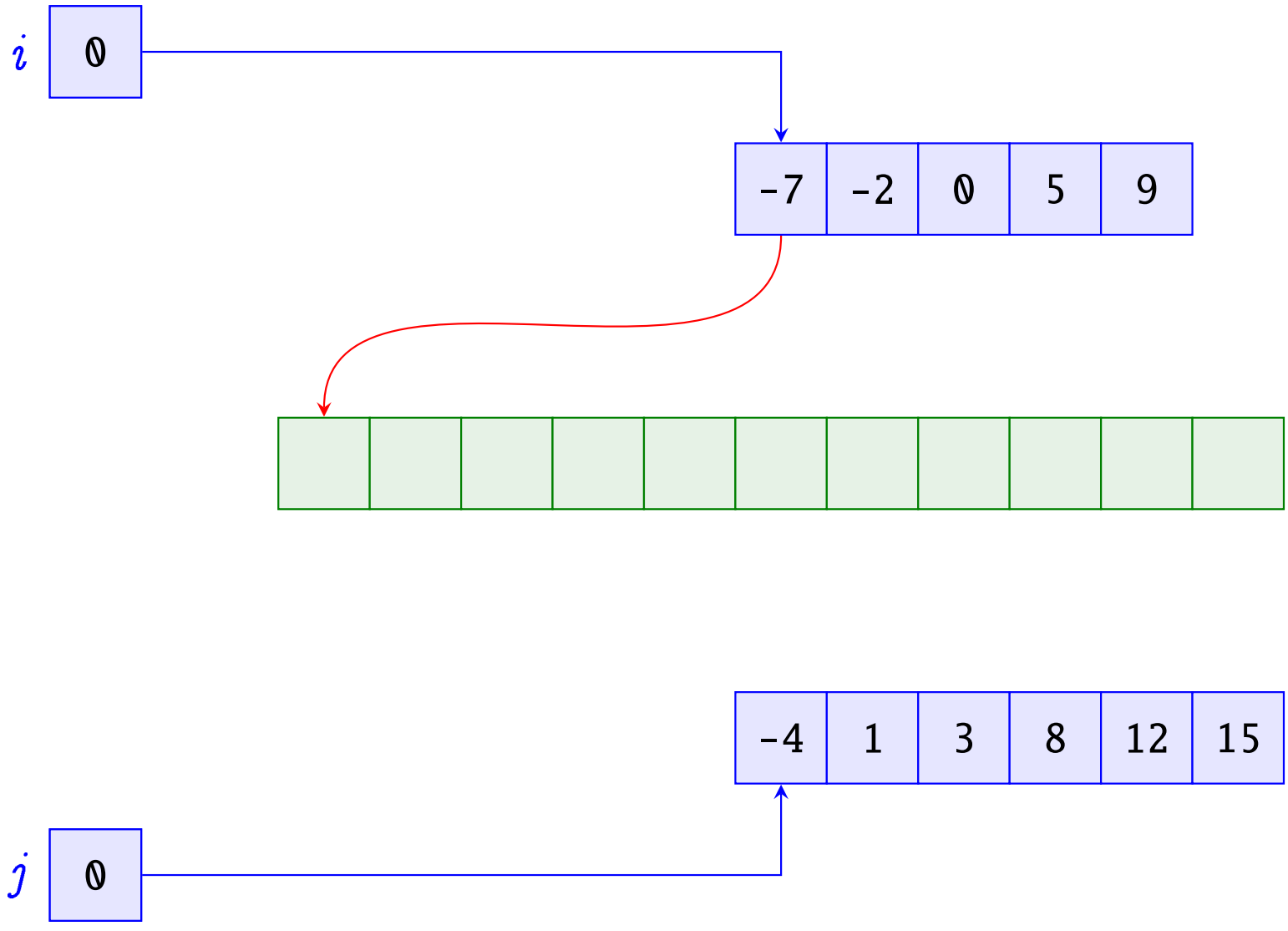
```
def merge(left, right):
    result = []; i = 0; j = 0
    while (i < len(left) and j < len(right)):
        if (left[i] <= right[j]):
            result.append(left[i]);   i += 1
        else:
            result.append(right[j]);  j += 1
    result += left[i:]
    result += right[j:]
    return result

def mergeSort(a):
    if len(a) < 2:    return a
    mid   = len(a) // 2
    left  = mergeSort(a[:mid])
    right = mergeSort(a[mid:])
    return merge(left,right)
```

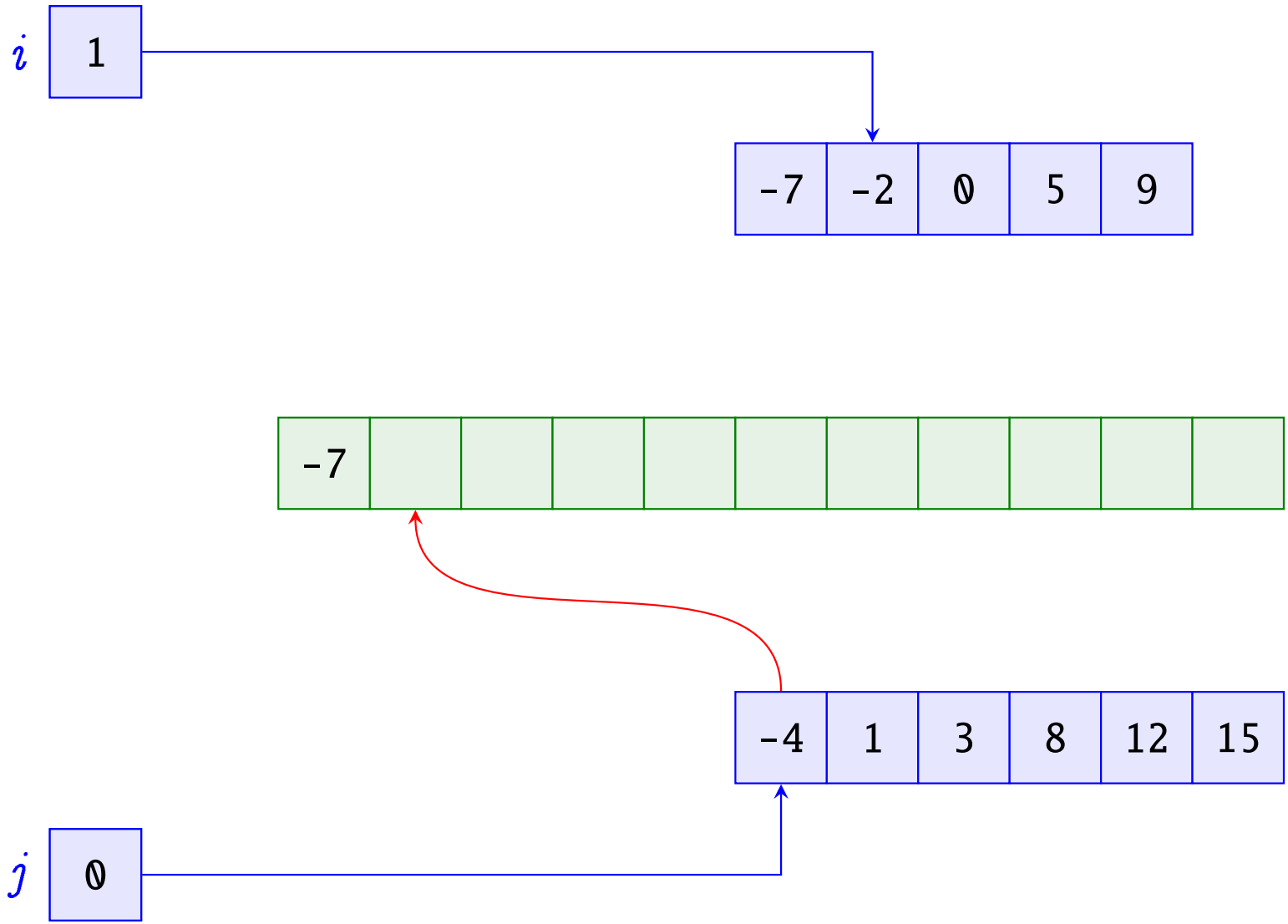
Ühildusmeetodil sorteerimine



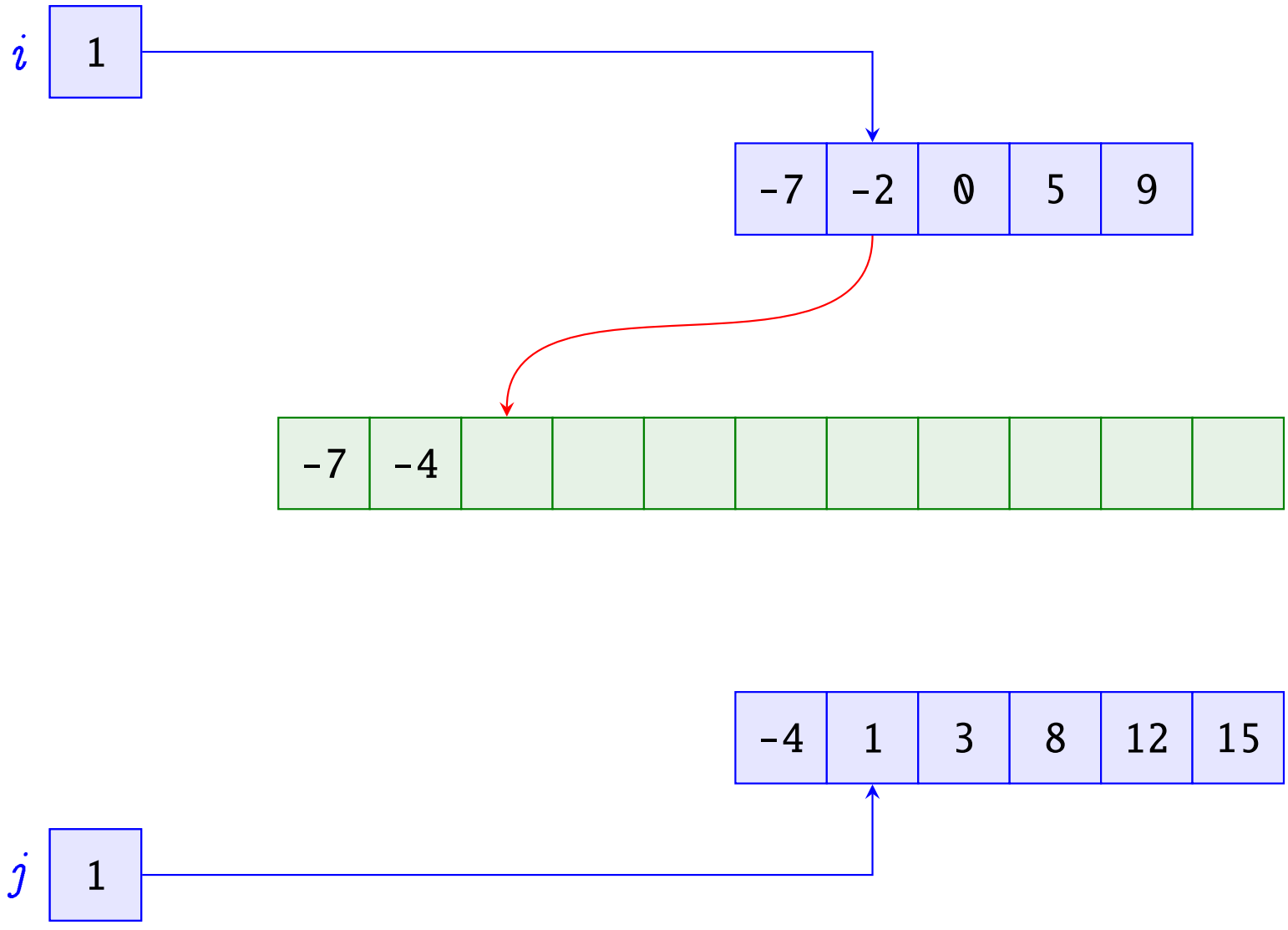
Ühildusmeetodil sorteerimine



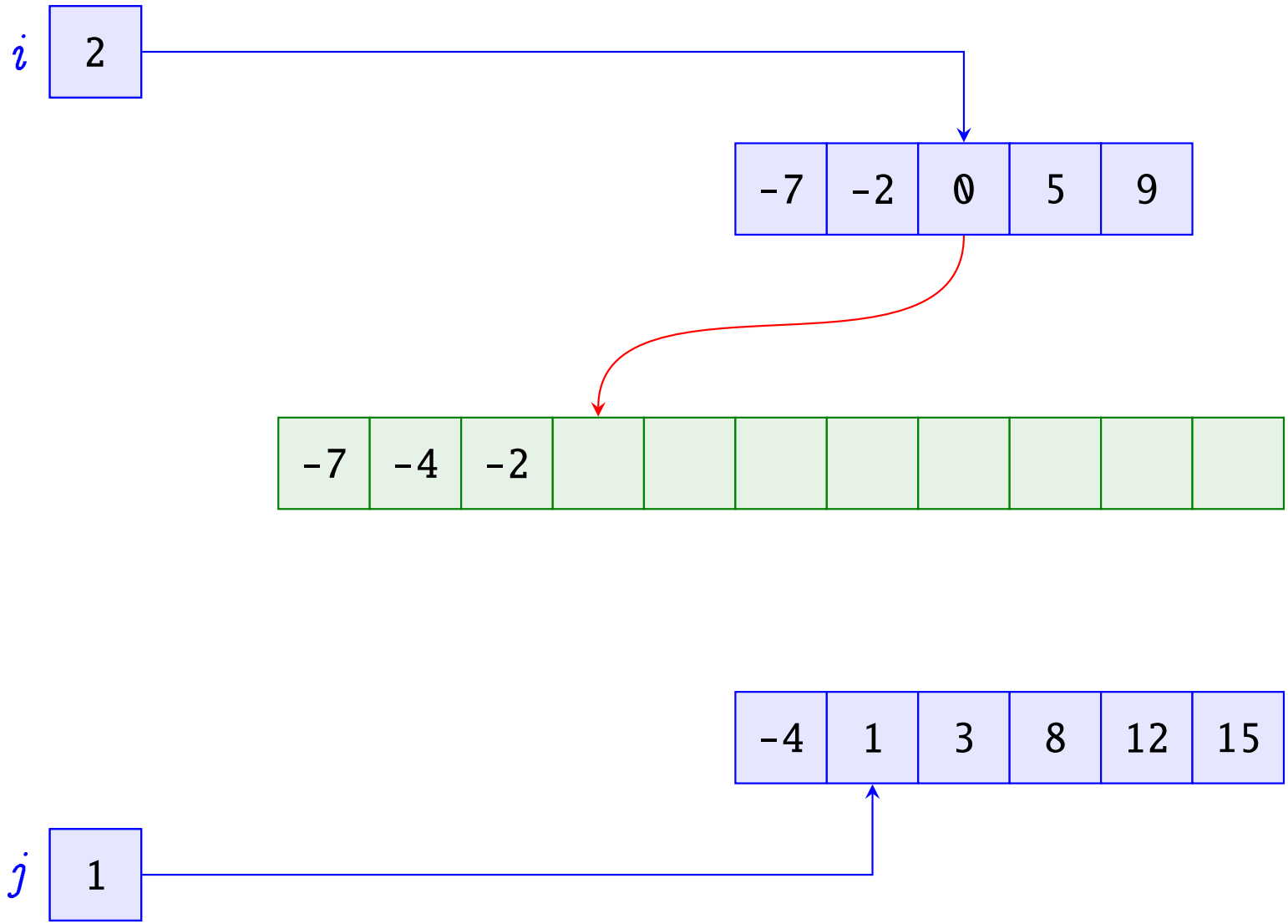
Ühildusmeetodil sorteerimine



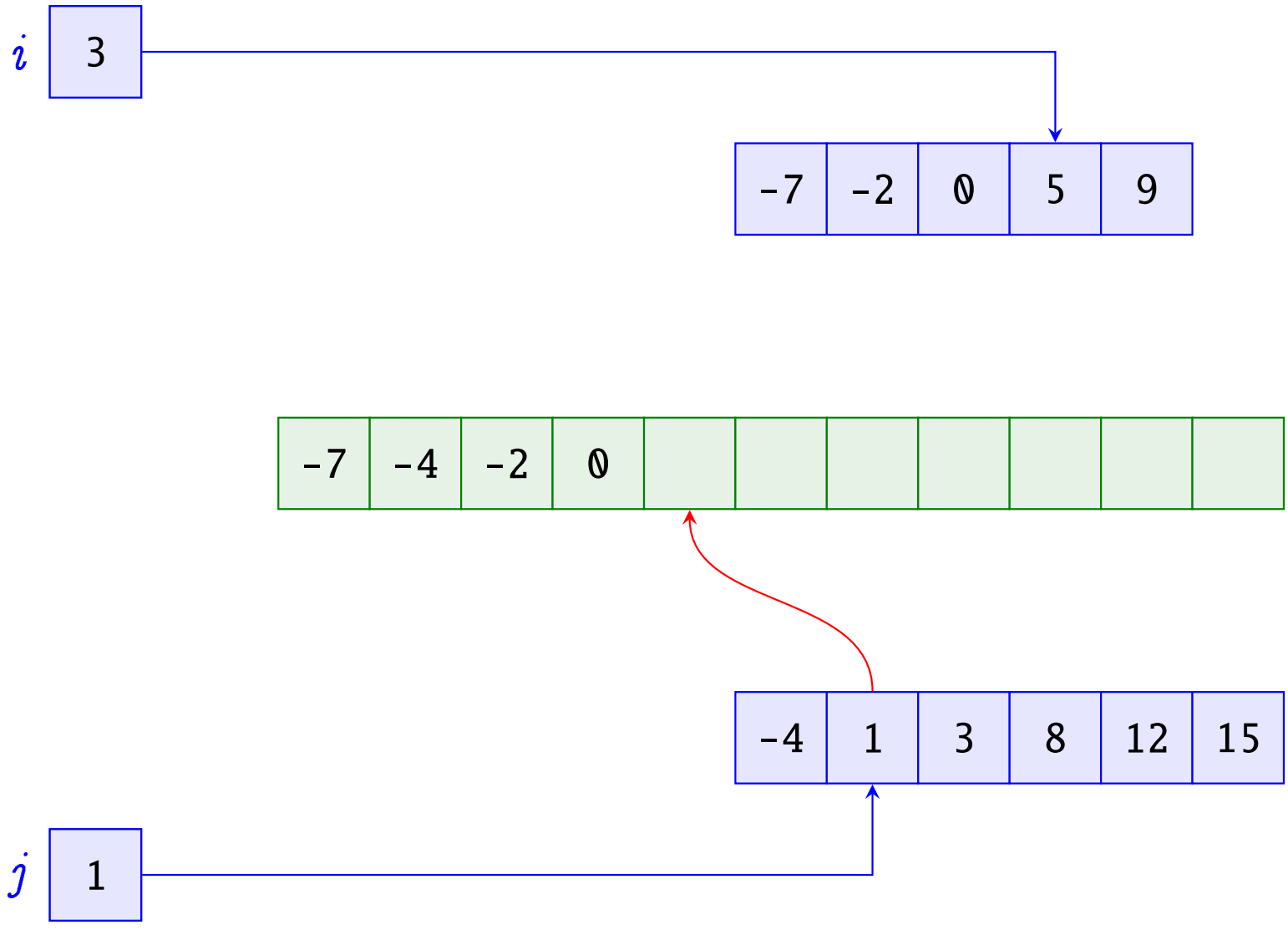
Ühildusmeetodil sorteerimine



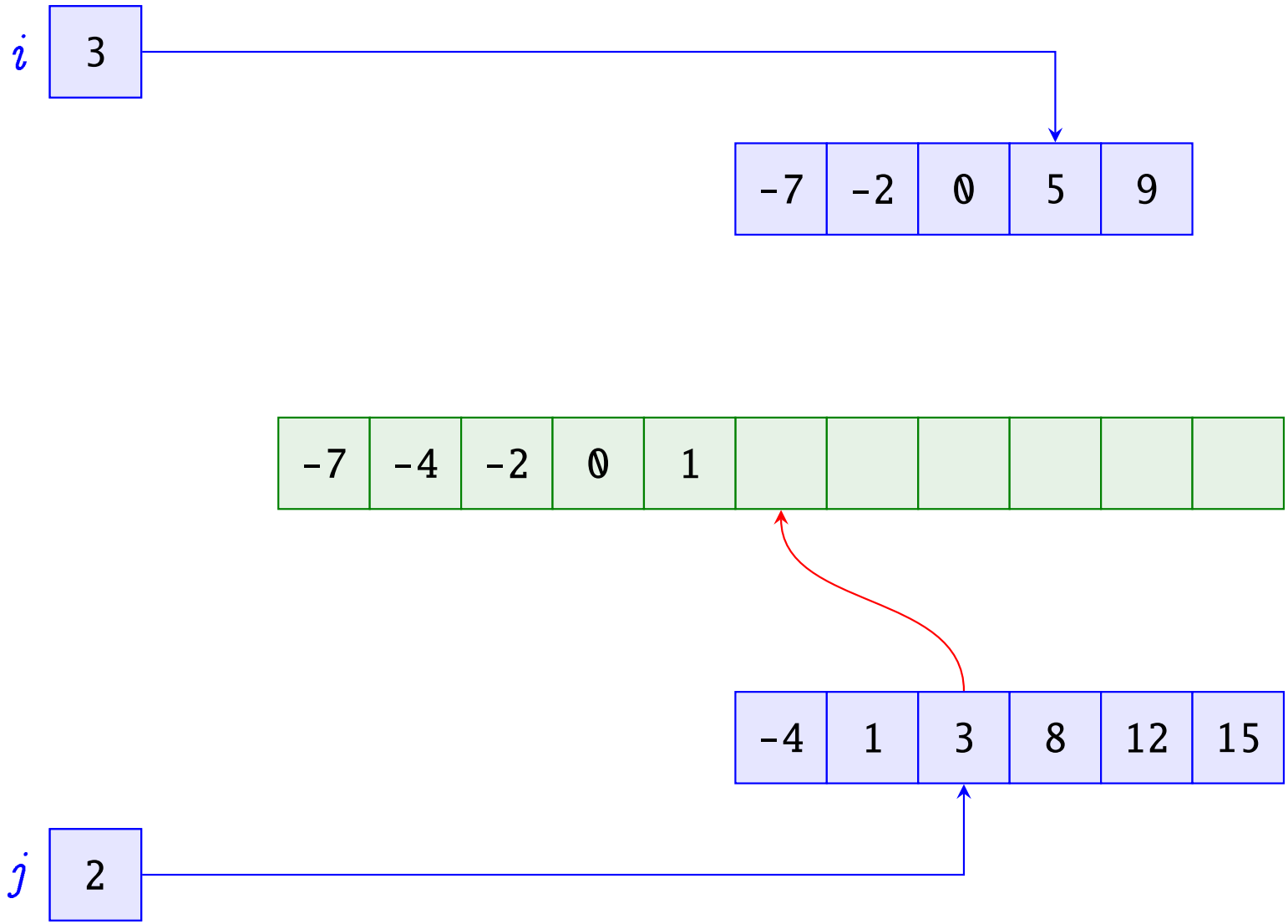
Ühildusmeetodil sorteerimine



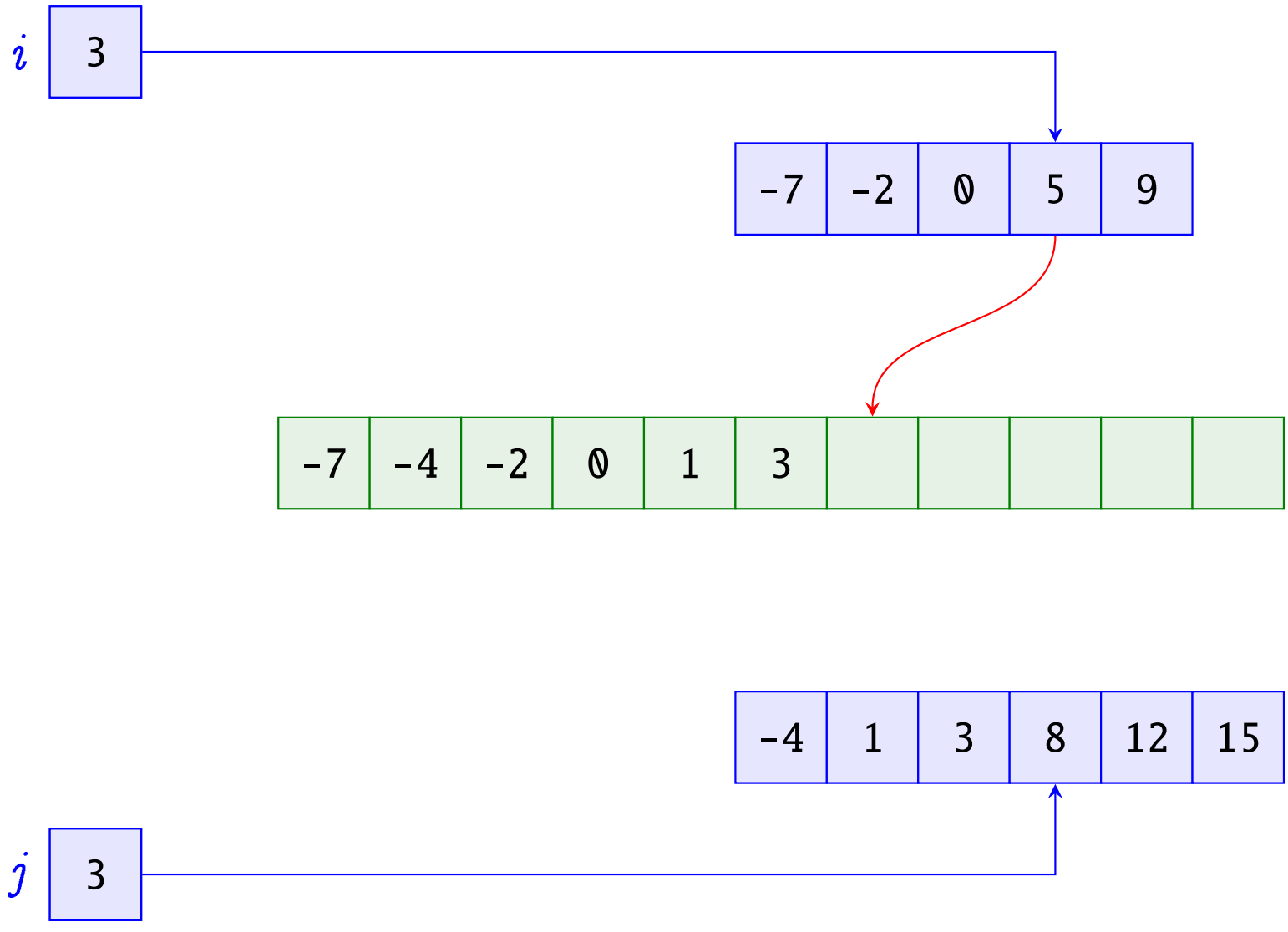
Ühildusmeetodil sorteerimine



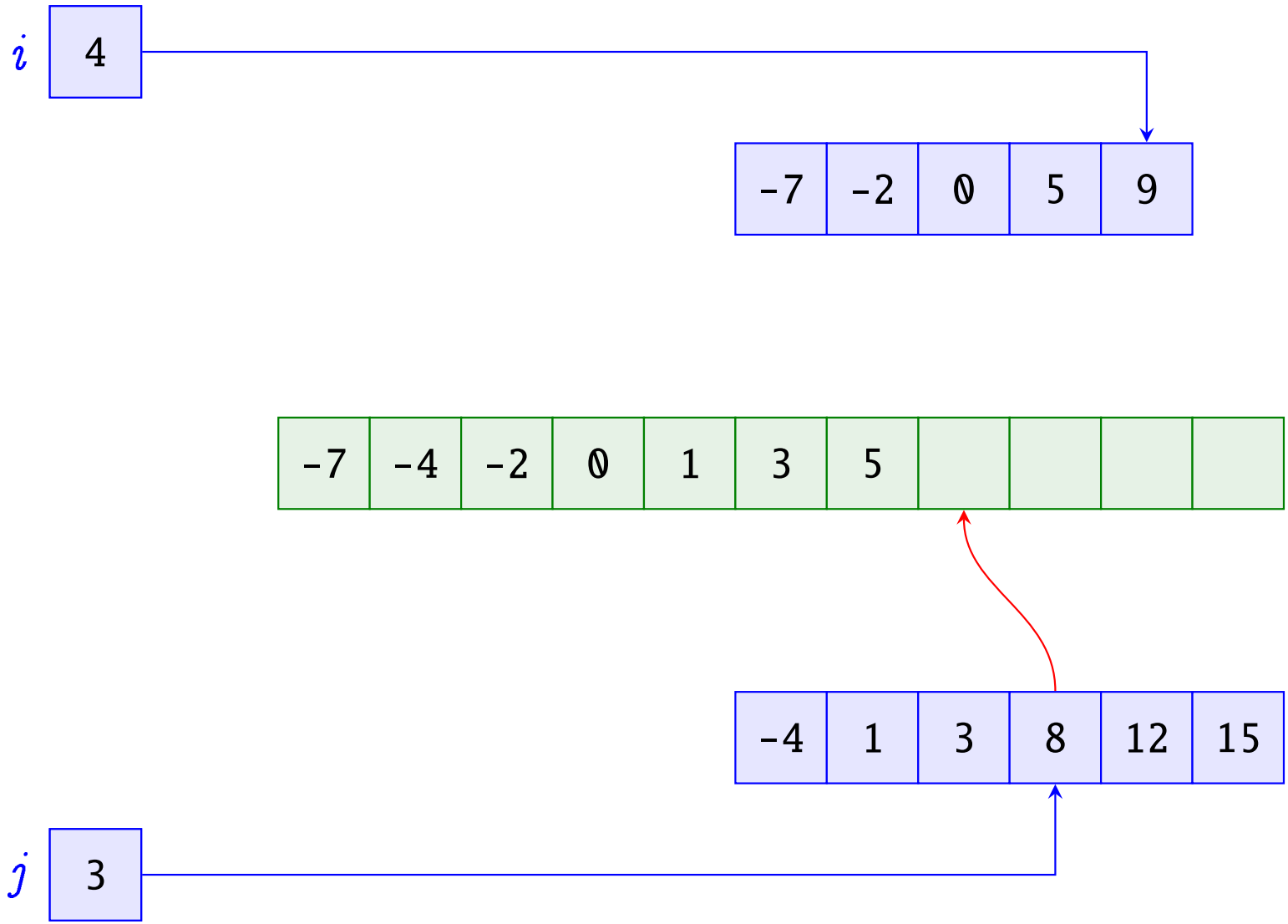
Ühildusmeetodil sorteerimine



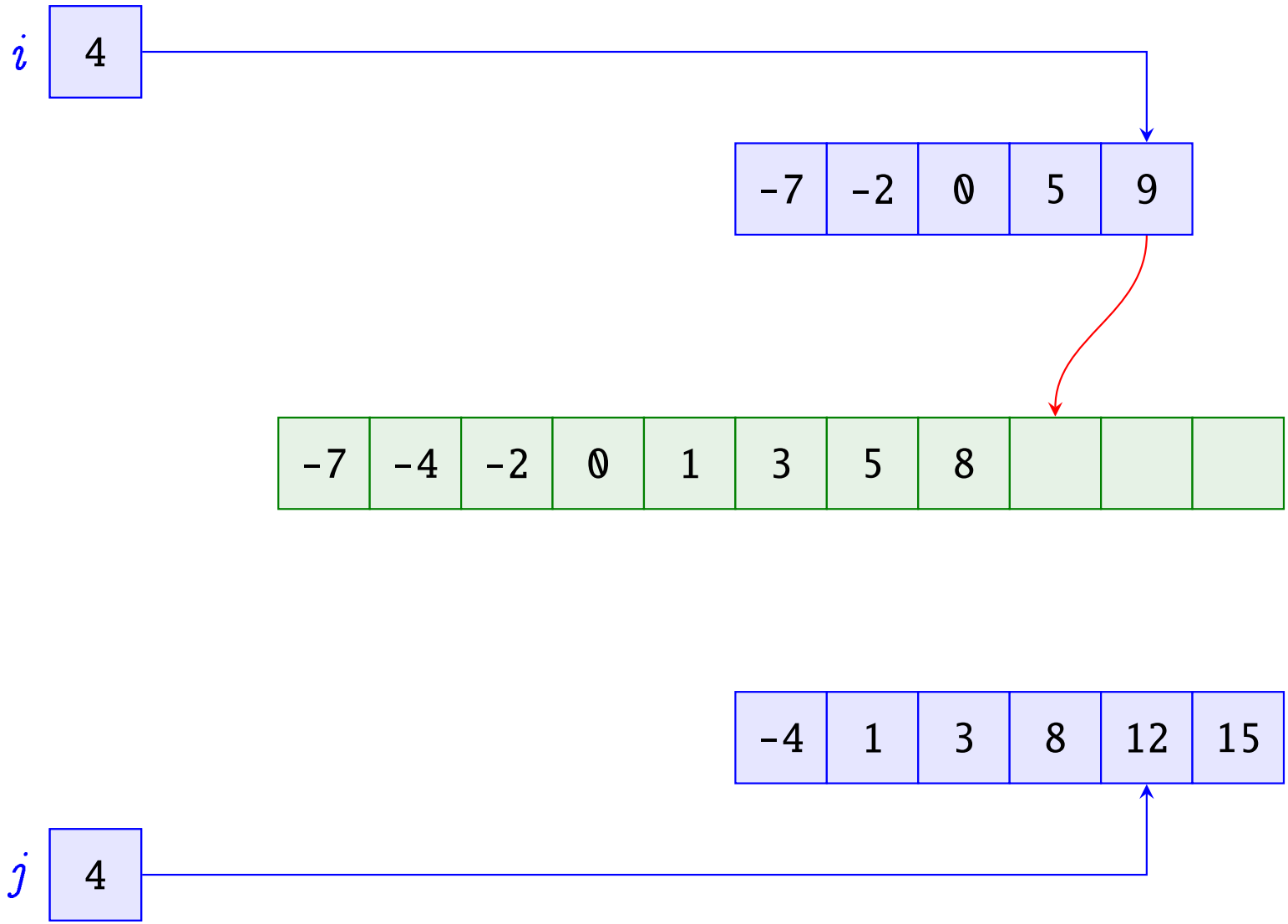
Ühildusmeetodil sorteerimine



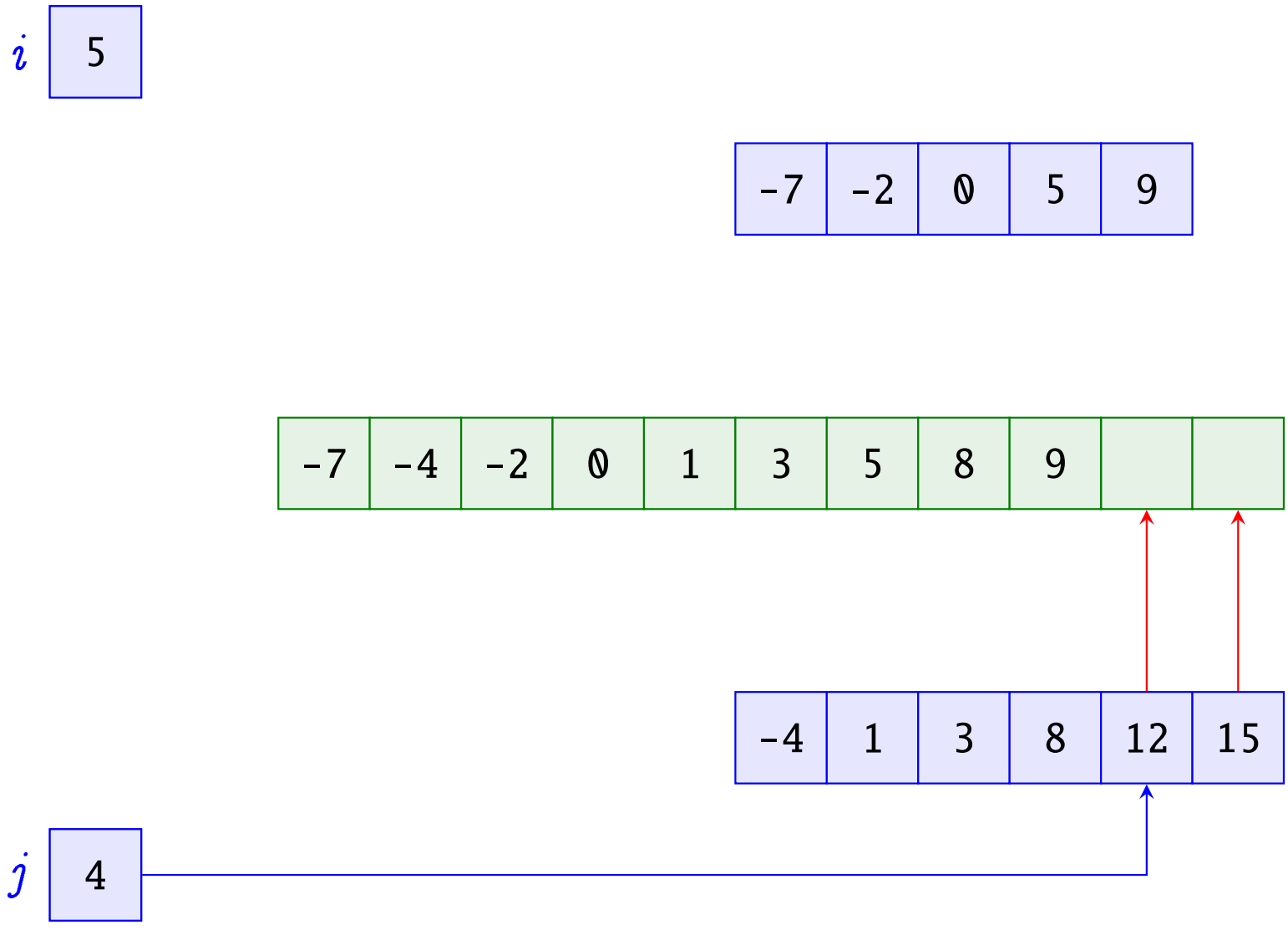
Ühildusmeetodil sorteerimine



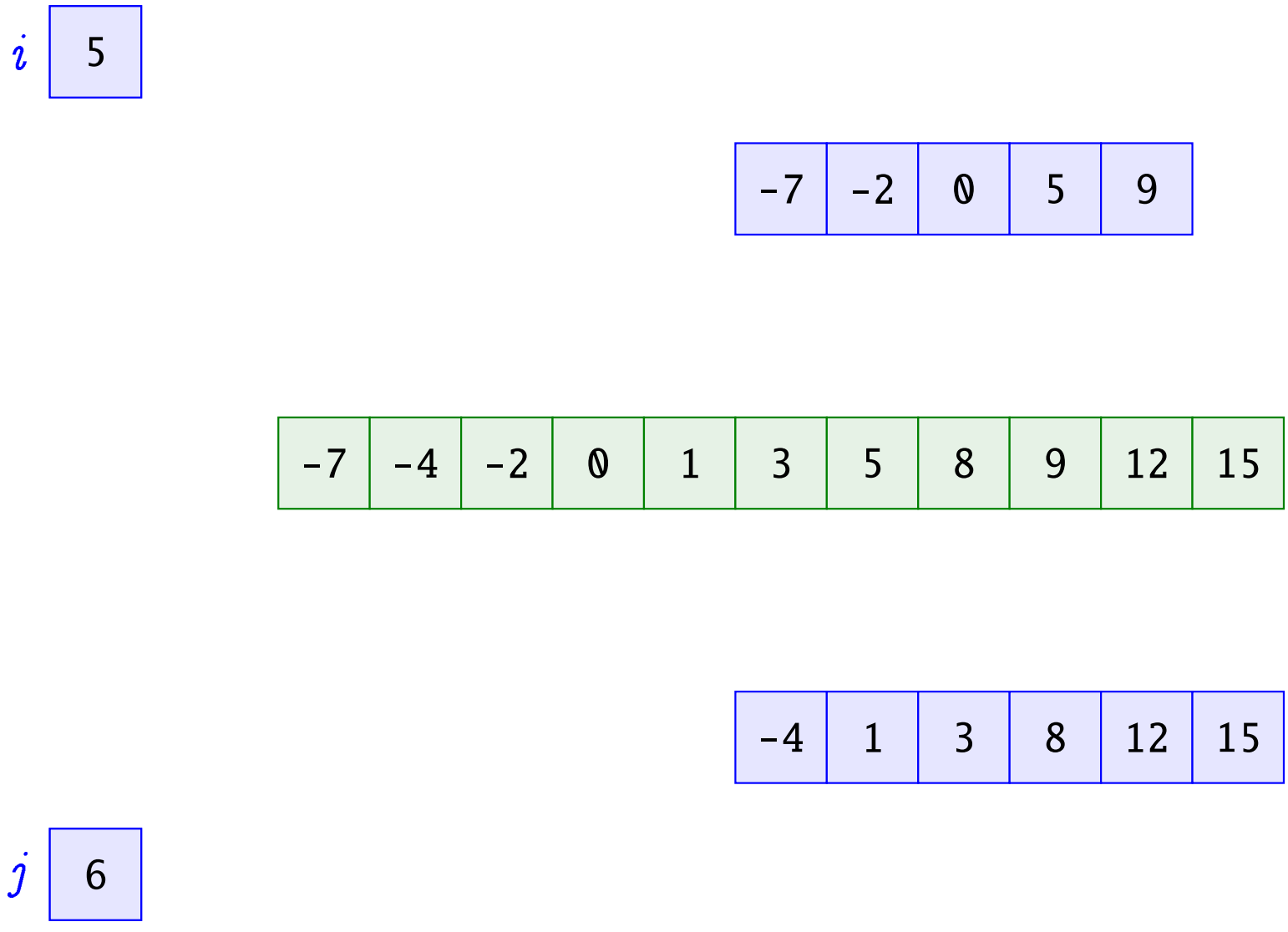
Ühildusmeetodil sorteerimine



Ühildusmeetodil sorteerimine



Ühildusmeetodil sorteerimine



Ühildusmeetodil sorteerimine

- Kahe järjestatud massiivi ühildamine on lineaarne.
- Igal rekusiooni väljakutsepuu tasemel kõik vastavad alammassiivid lõikumatud.
- Seega on ka igal tasemel tehtav töö lineaarne.
- Kuna massiivid tükeldatakse alati pooleks, siis on väljakutsepuu sügavus $\log n$.
- Seega on ühildusmeetodi keerukus $\mathcal{O}(n \cdot \log(n))$.

Suur tänu osalemast

ja

kohtumiseni!