Základy algoritmizace

5. Vyhledávání a řazení 1

doc. Ing. Jiří Vokřínek, Ph.D.
Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Základy algoritmizace

- Dnes:
  - Vyhledávání v poli
  - Řazení pole
    - Insertion sort
    - Selection sort
    - Bubble sort

Vyhledávání

- Úloha

  Najděte prvek s danou hodnotou/vlastností v poli.

- Řešení
  - Vstup: pole *array* o velikosti *n*, hledaná vlastnost *val*
  - Výstup: prvek pole splňující hledané kritérium
  - Výstup (alternativní): *True* pokud pole obsahuje prvek

Je v tom rozdíl?

  - Algoritmus:

```
def searchLinear(array, val):
    for i in array:
        elem = array[i]
        if elem == val: return elem
```

Vyhledávání

- Podobné úlohy
  - Najděte největší číslo v poli celých čísel
  - Najděte nejmenší číslo
  - Najděte k-té největší číslo
  - Najděte k-té nejmenší číslo

- Sekvenční vyhledávání
  - Nejde to lépe?

Vyhledávání

- Hledání binárním půlením
  - Rychlejší prohledání
  - Vyhledávací pole musí být uspořádané

```
def binSearch(array, val):
    l, r = 0, len(array) - 1
    while r>=l:
        i = int((l + r)/2)
        if val == array[i]: return array[i]
        if val > array[i]:
            l = i + 1
        else:
            r = i - 1
```

  - Alternativně funkce vrací *True*/*False*, index *i, atd.*

Vyhledávání

- Zobecnění klíč-hodnota
  - Obecně hledáme prvky podle klíče který je asociován s daným prvkem
  - V dnešní přednášce uvažujeme jako klíč číselnou hodnotu prvku
  - V Pythonu můžeme použít tuple (klíč, hodnota)
  - Pro porovnávání lze definovat funkce *value*, *equals, greater,* atp. pro libovolné struktury hodnot (objektů)
  - Uvažujme, že hledané prvky lze porovnávat pomocí operátorů >, <, ==

Jak pole uspořádat?

Řazení

- Motivace – urychlení vyhledávání dotazů
- *A* … neprázdná posloupnost prvků $\langle a_1 \dots a_n \rangle$
- $|A|$ … délka posloupnosti
- $val(a_i)$ … hodnota prvku a_i pro porovnání

- Posloupnost je seřazená právě tehdy, když
  - $|A| < 2$,
  - $|A| \geq 2, val(a_1) \leq val(a_2)$ a posloupnost $\langle a_2 \dots a_n \rangle$ neobsahuje prvek a_1 a je seřazená

Řazení

- Dle způsobu uložení
  - Vnitřní řazení vs. vnější řazení
- Dle způsobu využití klíčů
  - Adresní řazení vs. asociativní řazení
- Kategorie řazení
  - Vkládáním
  - Výběrem
  - Výměnou
- Stabilní algoritmus řazení – pokud se relativní pořadí prvků se shodnou hodnotou nemění v průběhu řazení
- Metriky „kvality“ – počet porovnání *C(n)*, počet přesunů *M(n)*, paměťová náročnost *S(n)*

Řazení vkládáním
Insertion sort

Insert Sort

- Řazení přímým vkládáním

  for $i \in \langle 2, n \rangle$

  „vlož a_i na patřičné místo mezi $a_1, \ldots, a_i$“

```
def insertSort(array):
    for i in range(1, len(array)):
        currentvalue = array[i]
        position = i
        while position>0 and array[position-1]>currentvalue:
            array[position]=array[position - 1]
            position = position-1
        array[position]=currentvalue
```

Insert Sort

1		2		3		4		5		6
44		55		12		42		94		18

1

2

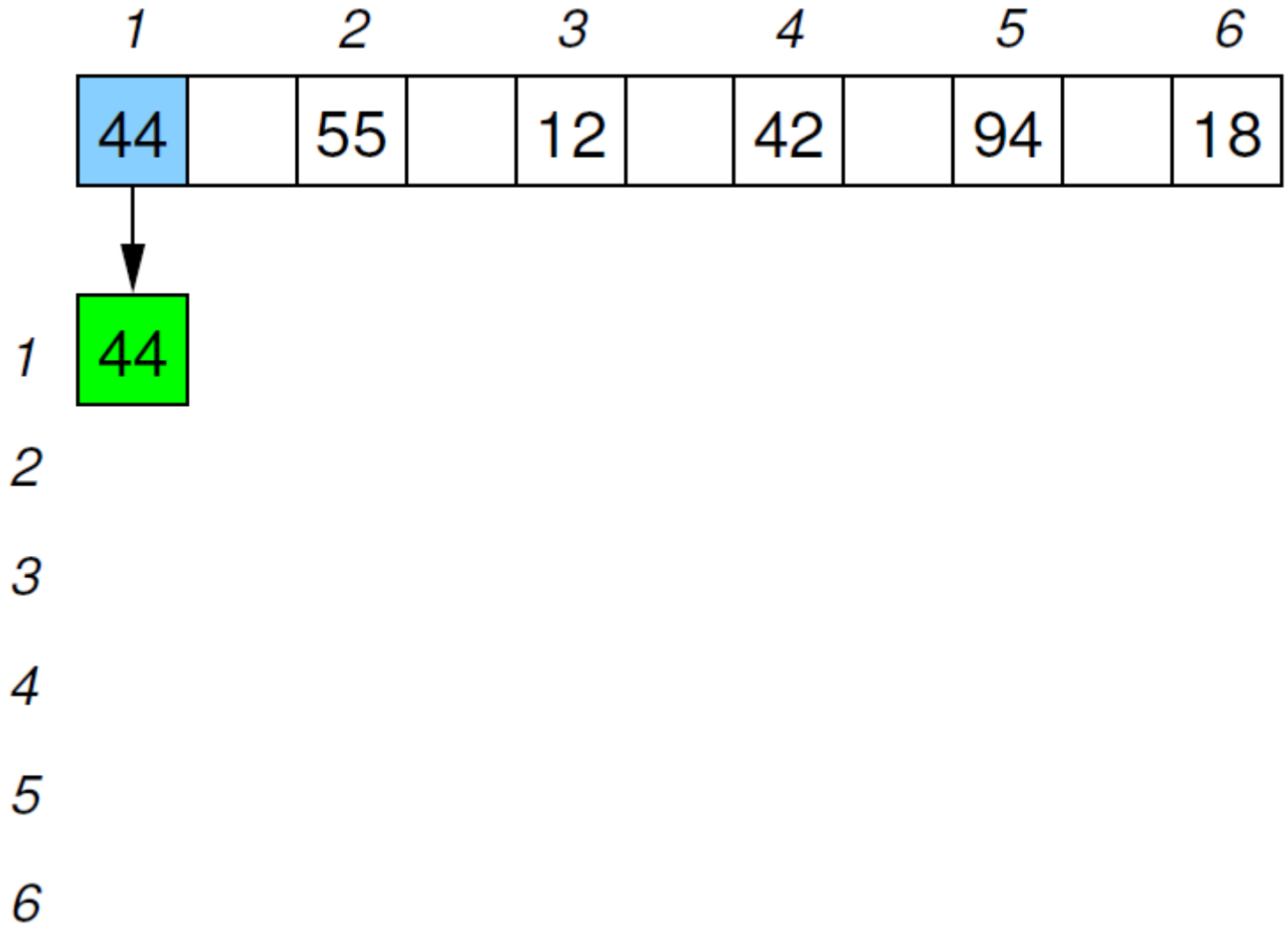
3

4

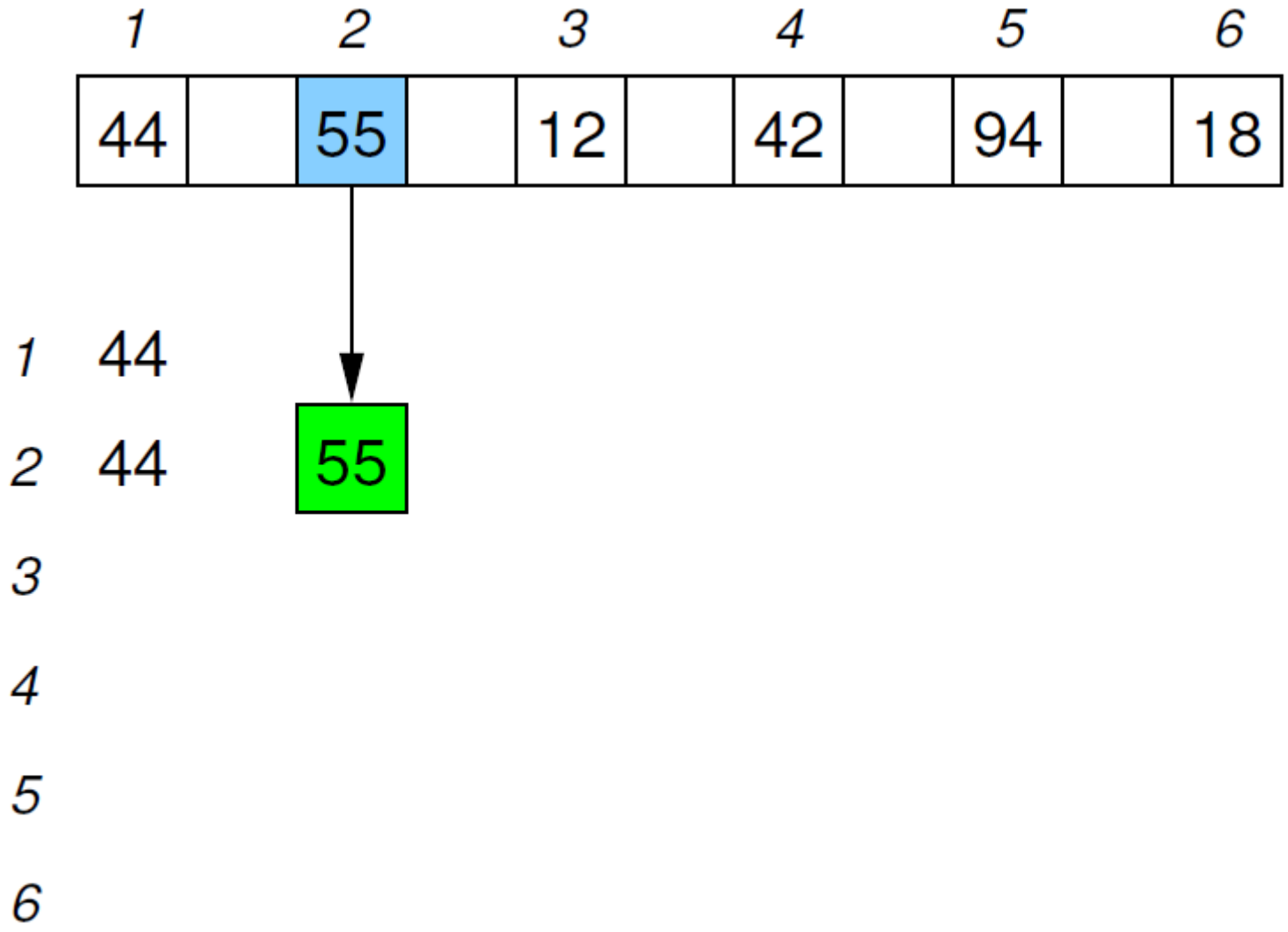
5

6

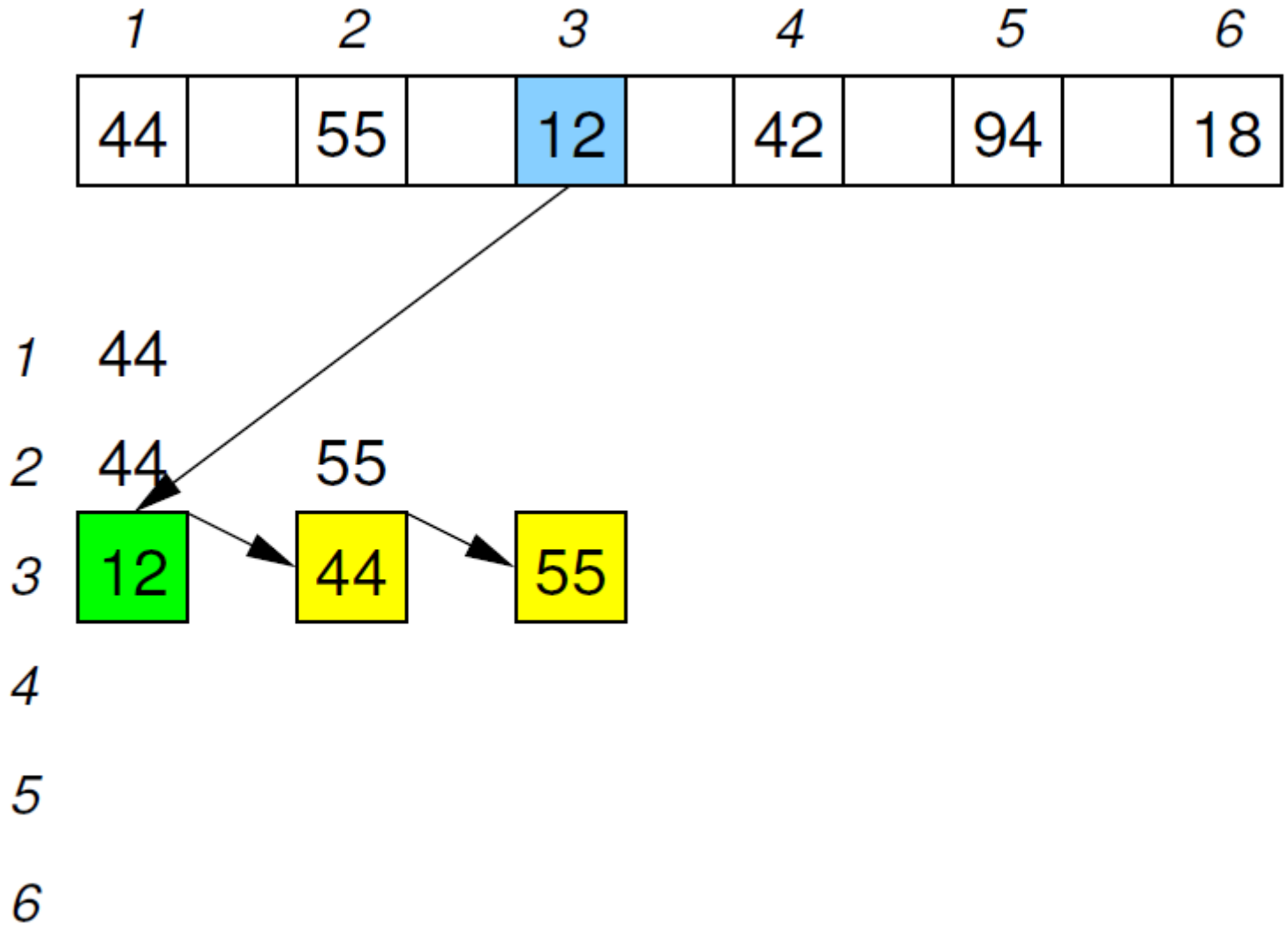
Insert Sort



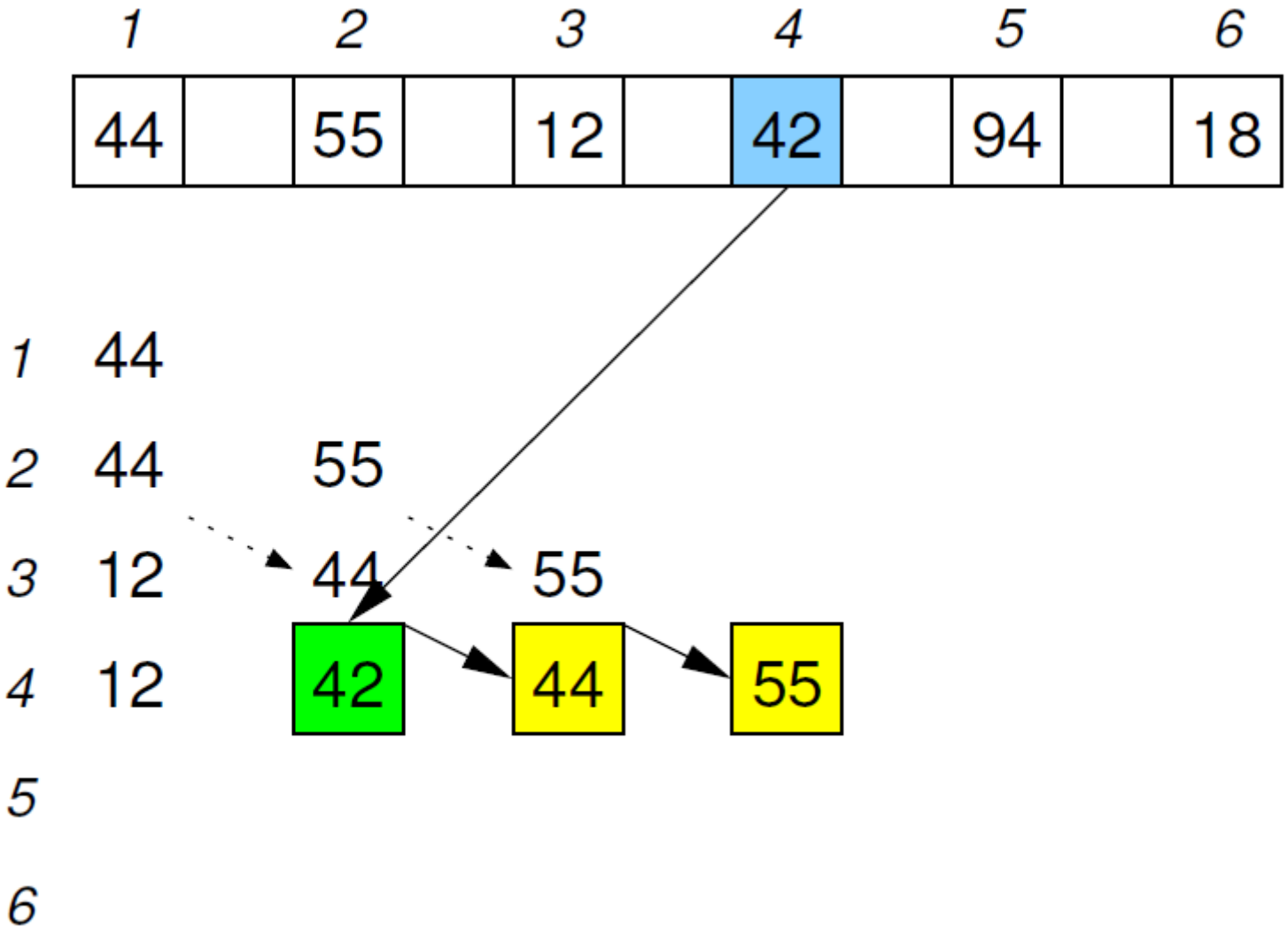
Insert Sort



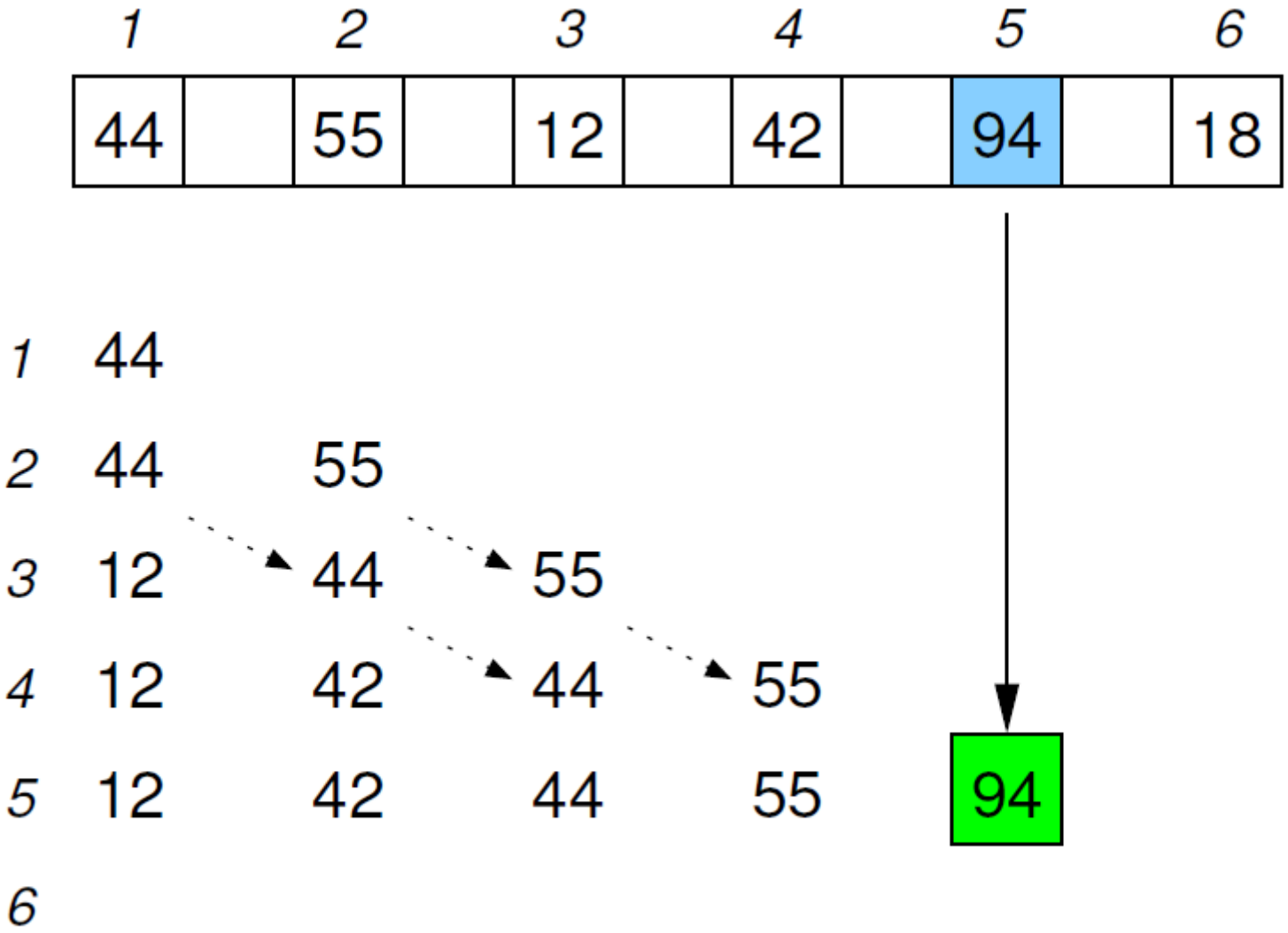
Insert Sort



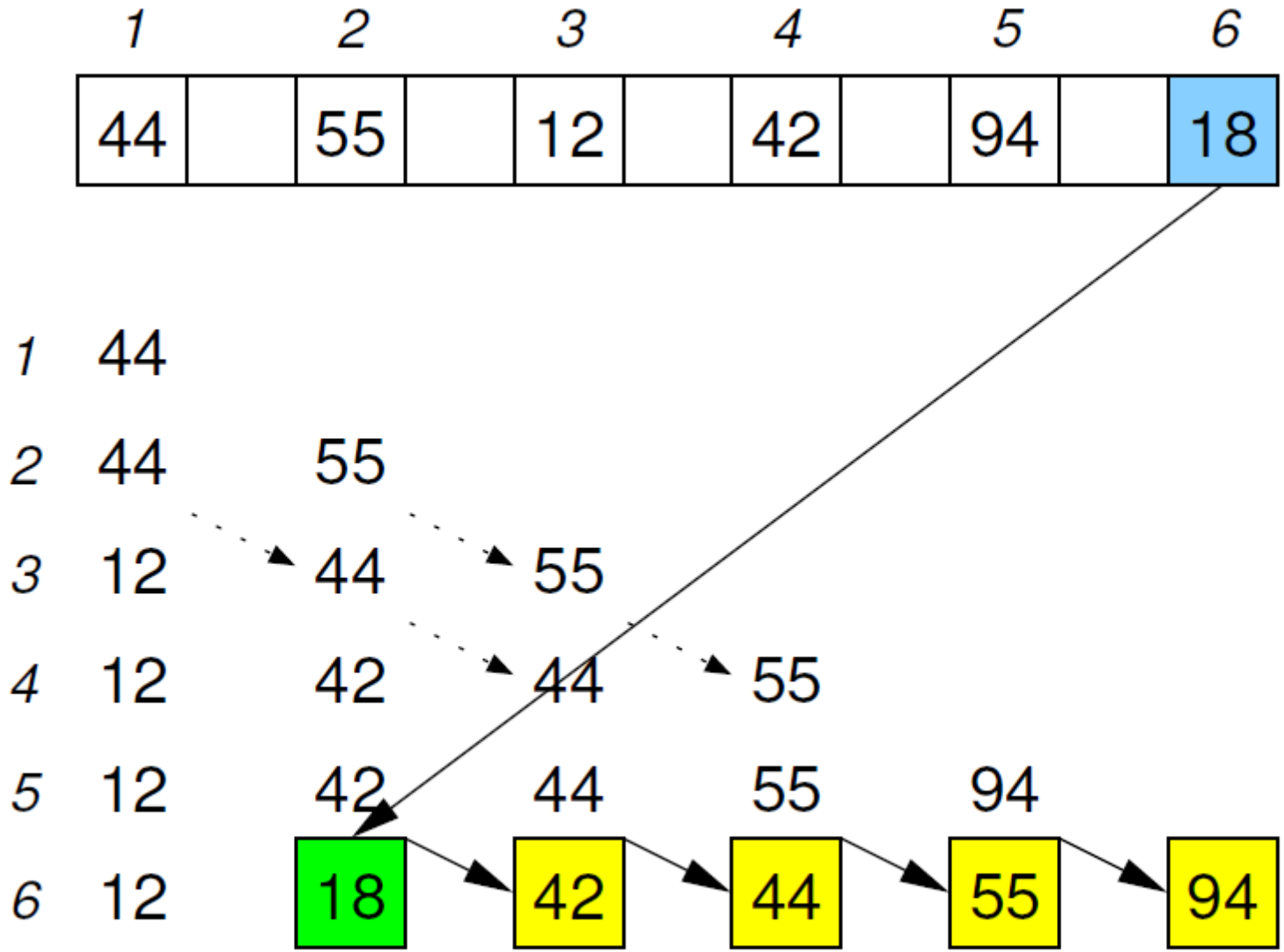
Insert Sort



Insert Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 44
2 44 55
3 12 44 55
4 12 42 44 55
5 12 42 44 55 94
6

Insert Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 44
2 44 55
3 12 44 55
4 12 42 44 55
5 12 42 44 55 94
6 12 18 42 44 55 94

Insert Sort

- Řazení přímým vkládáním

  for $i \in \langle 2, n \rangle$

  „vlož a_i na patřičné místo mezi $a_1, \ldots, a_i$“

```
def insertSort(array):
    for i in range(1, len(array)):
        currentvalue = array[i]
        position = i
        while position>0 and array[position-1]>currentvalue:
            array[position]=array[position - 1]
            position = position-1
        array[position]=currentvalue
```

Řazení výběrem
Selection sort

Select Sort

- Řazení přímým výběrem

  for $i \in \langle 1, n \rangle$

  „najdi index *k* nejmenšího prvku v $\langle a_i, \ldots, a_n \rangle$,

  $a_k = min\langle a_i, \ldots, a_n \rangle$,

  zaměň prvky a_i, a_k “

```
def selectSort(array):
    for i in range(len(array)):
        least = i
        for k in range(i + 1 , len(array)):
            if array[k] < array[least]: least = k
        array[least], array[i] =  array[i],array[least]
```

Select Sort

1	2	3	4	5	6
44	55	12	42	94	18

1

2

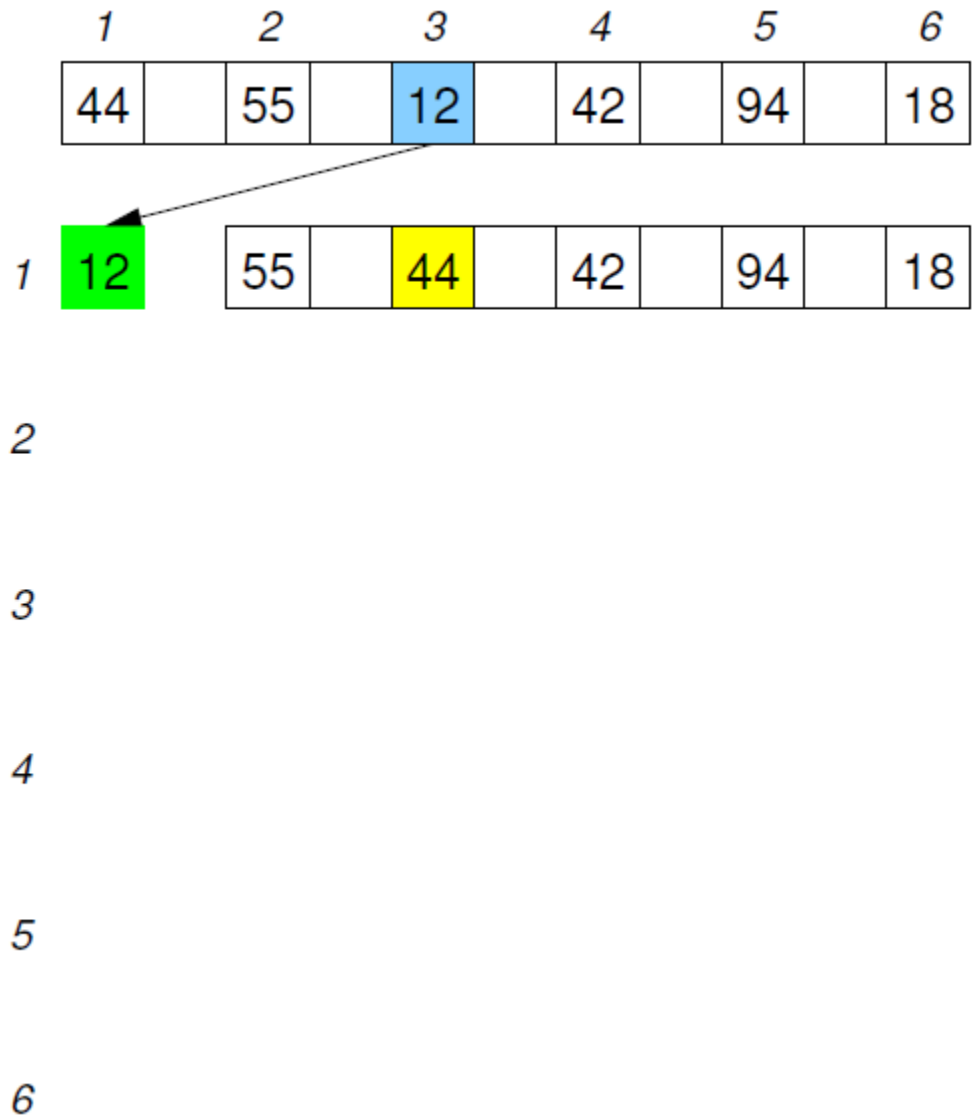
3

4

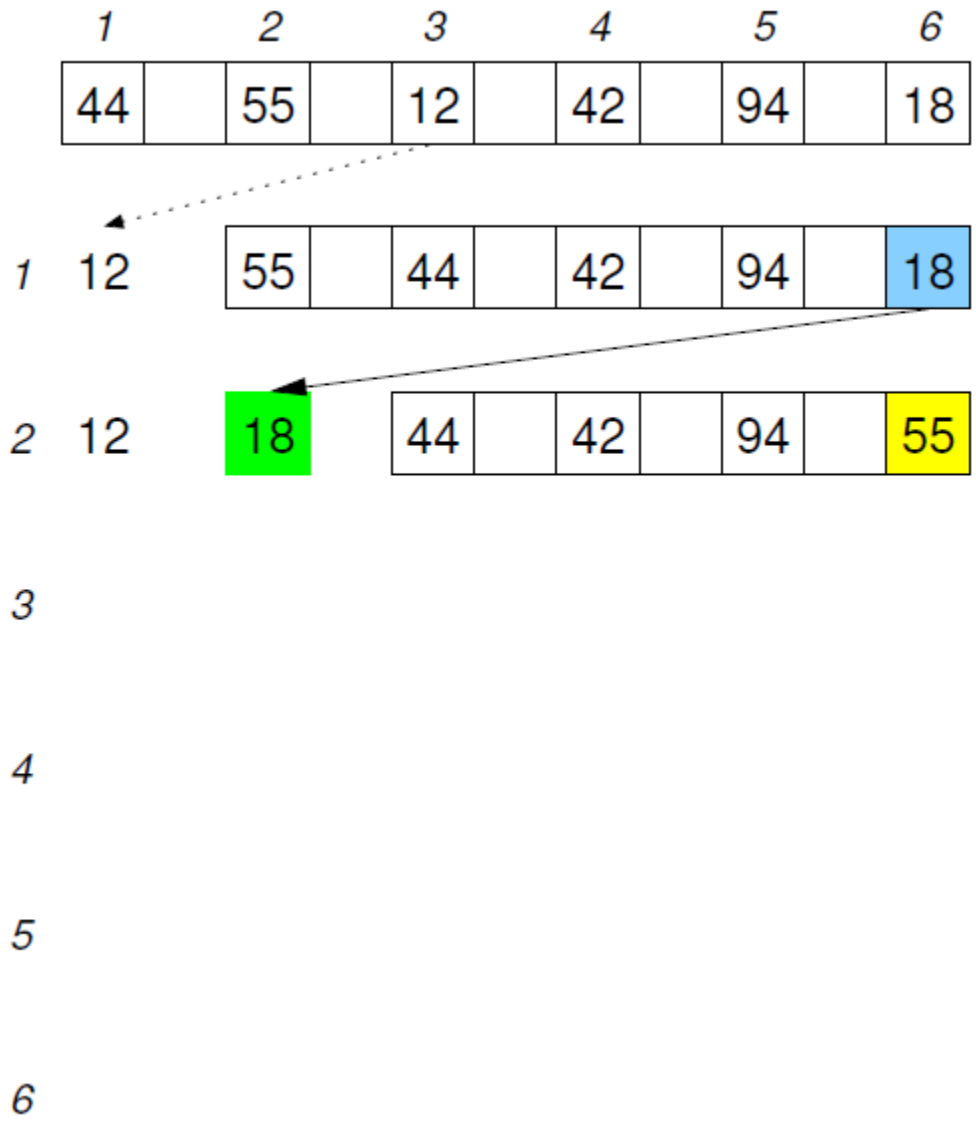
5

6

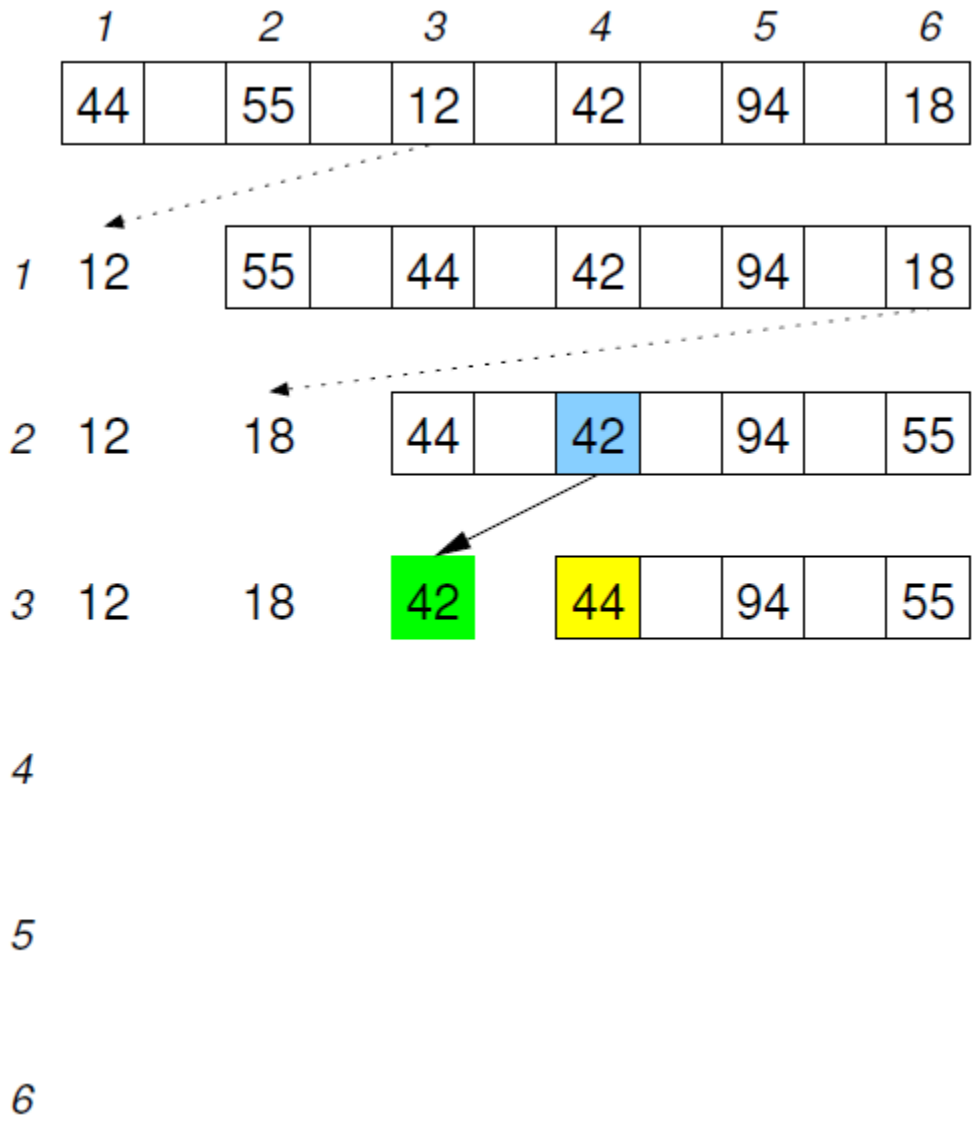
Select Sort



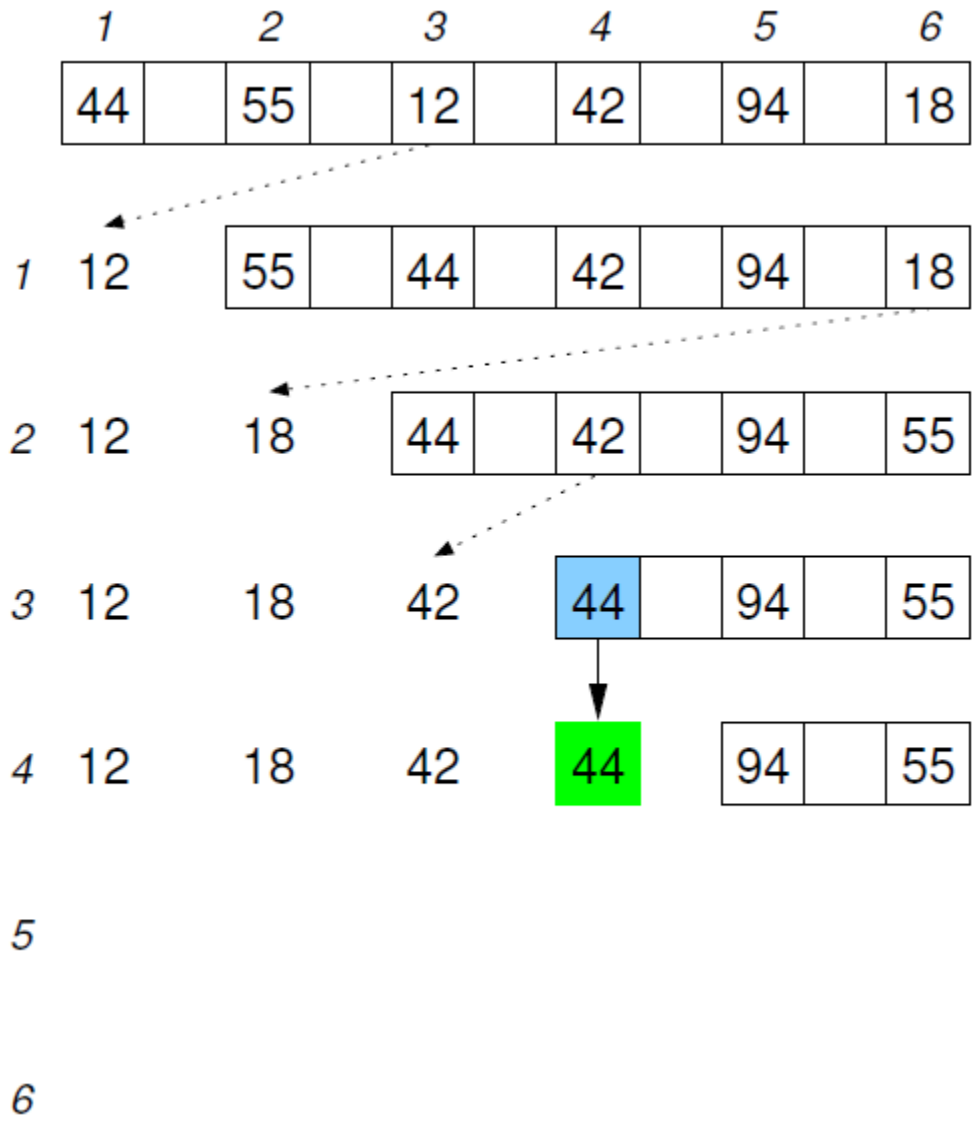
Select Sort



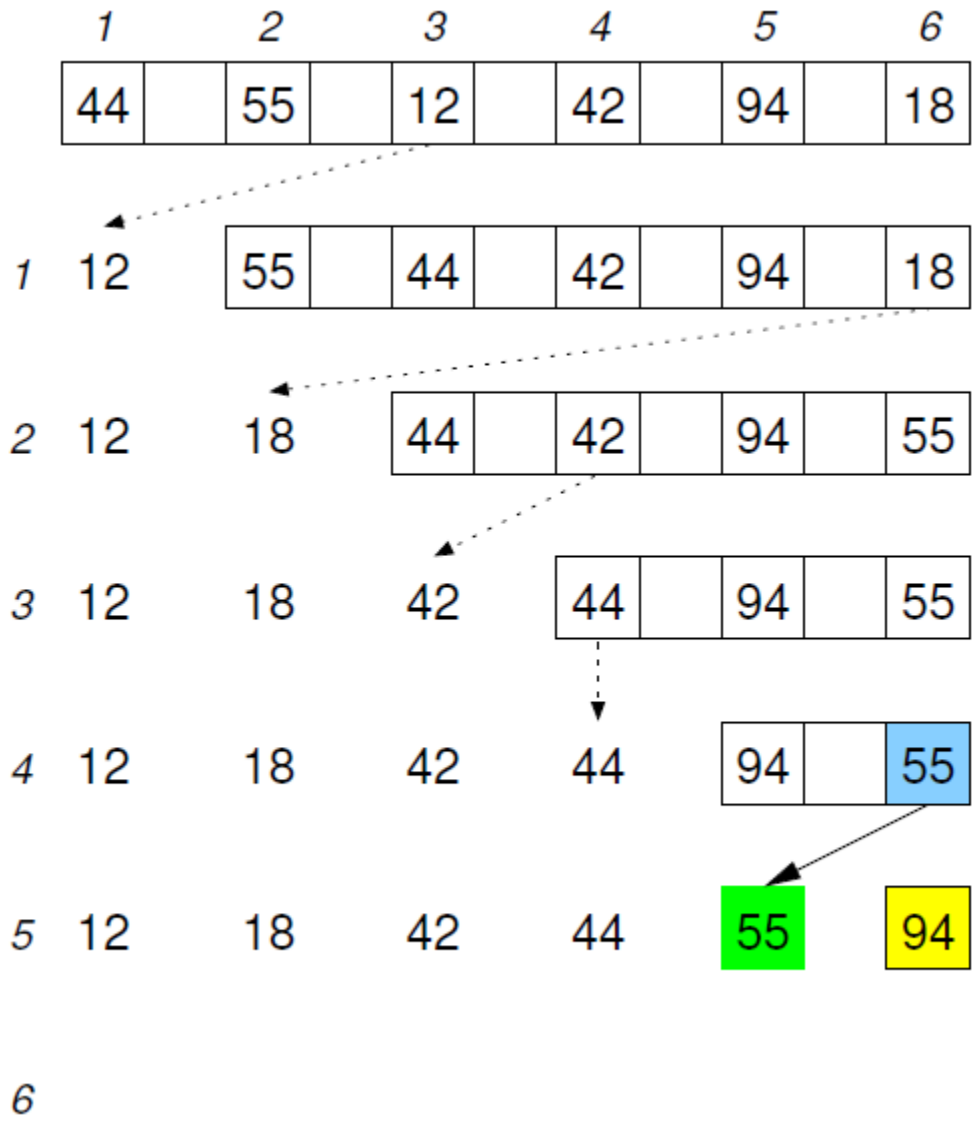
Select Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 12 55 44 42 94 18
2 12 18 44 42 94 55
3 12 18 42 44 94 55
4
5
6

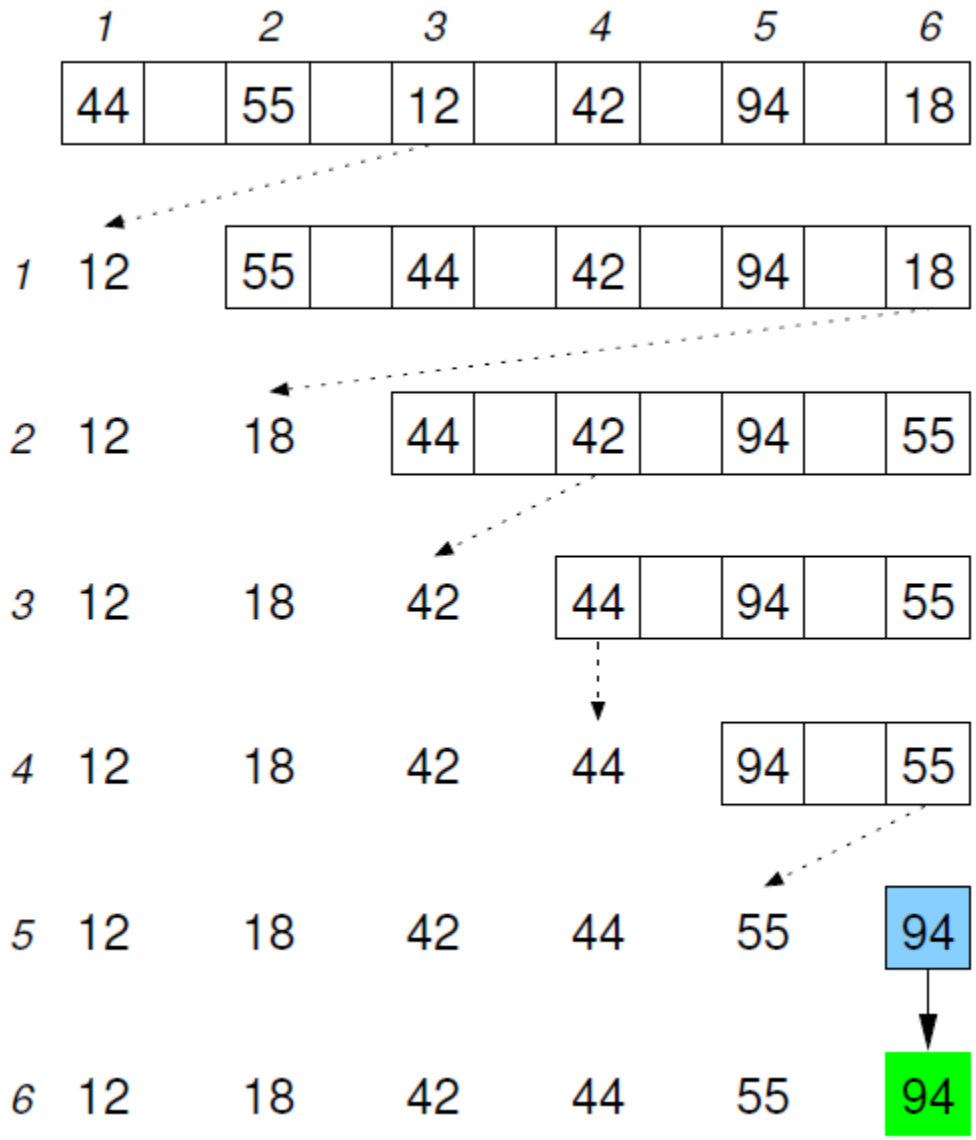
Select Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 12 55 44 42 94 18
2 12 18 44 42 94 55
3 12 18 42 44 94 55
4 12 18 42 44 94 55
5
6

Select Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 12 55 44 42 94 18
2 12 18 44 42 94 55
3 12 18 42 44 94 55
4 12 18 42 44 94 55
5 12 18 42 44 55 94
6

Select Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 12 55 44 42 94 18
2 12 18 44 42 94 55
3 12 18 42 44 94 55
4 12 18 42 44 94 55
5 12 18 42 44 55 94
6 12 18 42 44 55 94

Select Sort

- Řazení přímým výběrem

  for $i \in \langle 1, n \rangle$

  „najdi index *k* nejmenšího prvku v $\langle a_i, \ldots, a_n \rangle$,

  $a_k = min\langle a_i, \ldots, a_n \rangle$,

  zaměň prvky a_i, a_k “

```
def selectSort(array):
    for i in range(len(array)):
        least = i
        for k in range(i + 1 , len(array)):
            if array[k] < array[least]: least = k
        array[least], array[i] =  array[i],array[least]
```

Řazení výměnou
Bubble sort

Bubble Sort

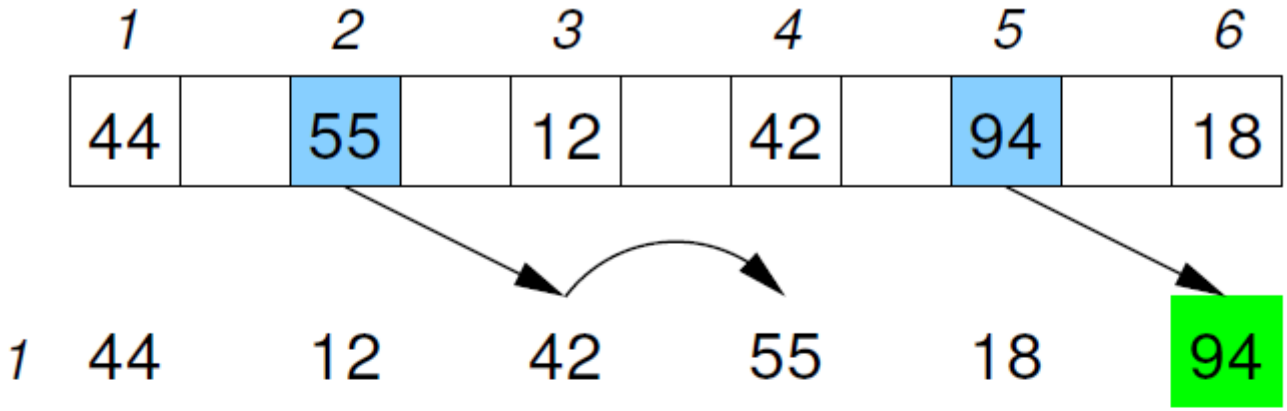
- „Maximální prvek probublává na i-tou pozici, kde i=n, n-1, … , 1“

```
def bubbleSort(array):
    for last in range(len(array)-1, 0, -1):
        for i in range(last):
            if array[i]>array[i+1]:
                array[i],array[i+1]=array[i+1],array[i]
```

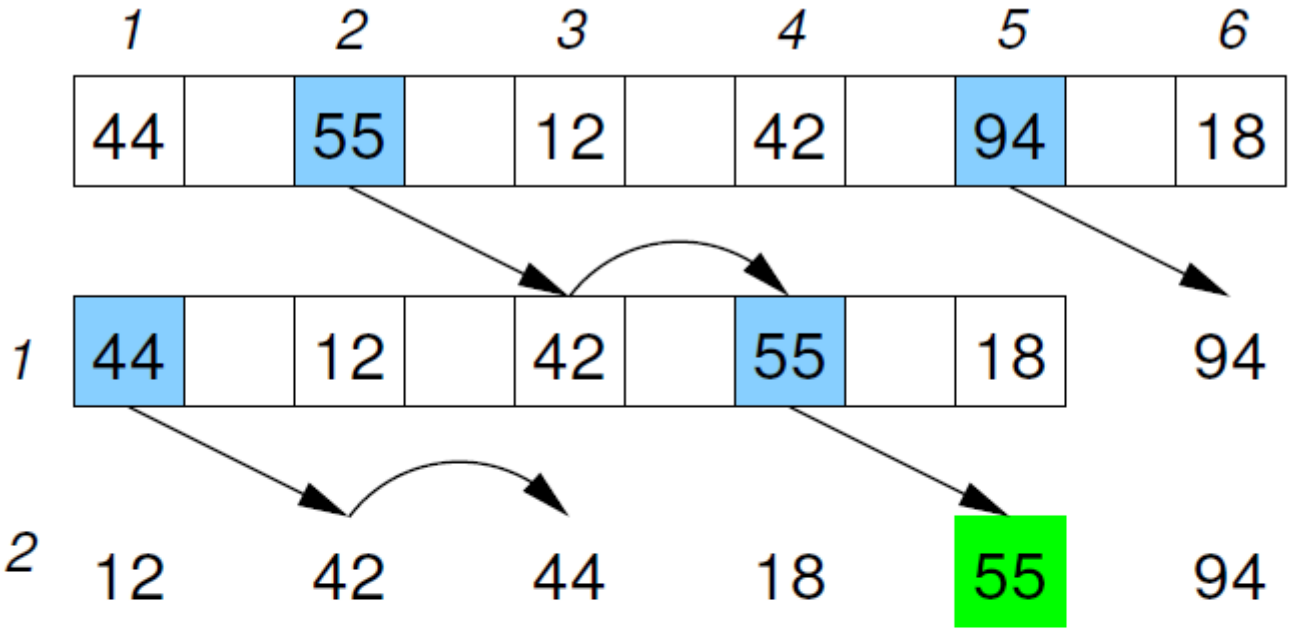
Bubble Sort

1		2		3		4		5		6
44		55		12		42		94		18

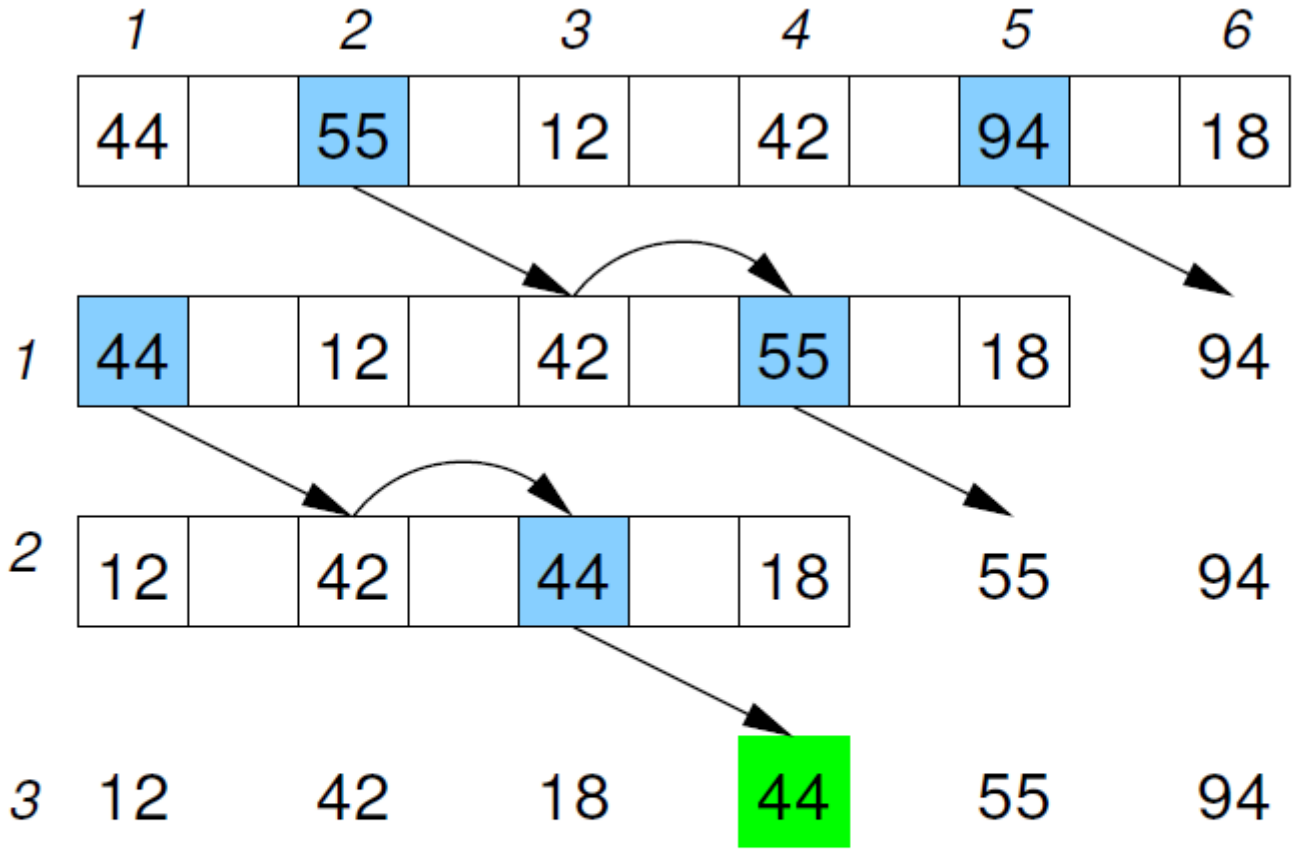
Bubble Sort



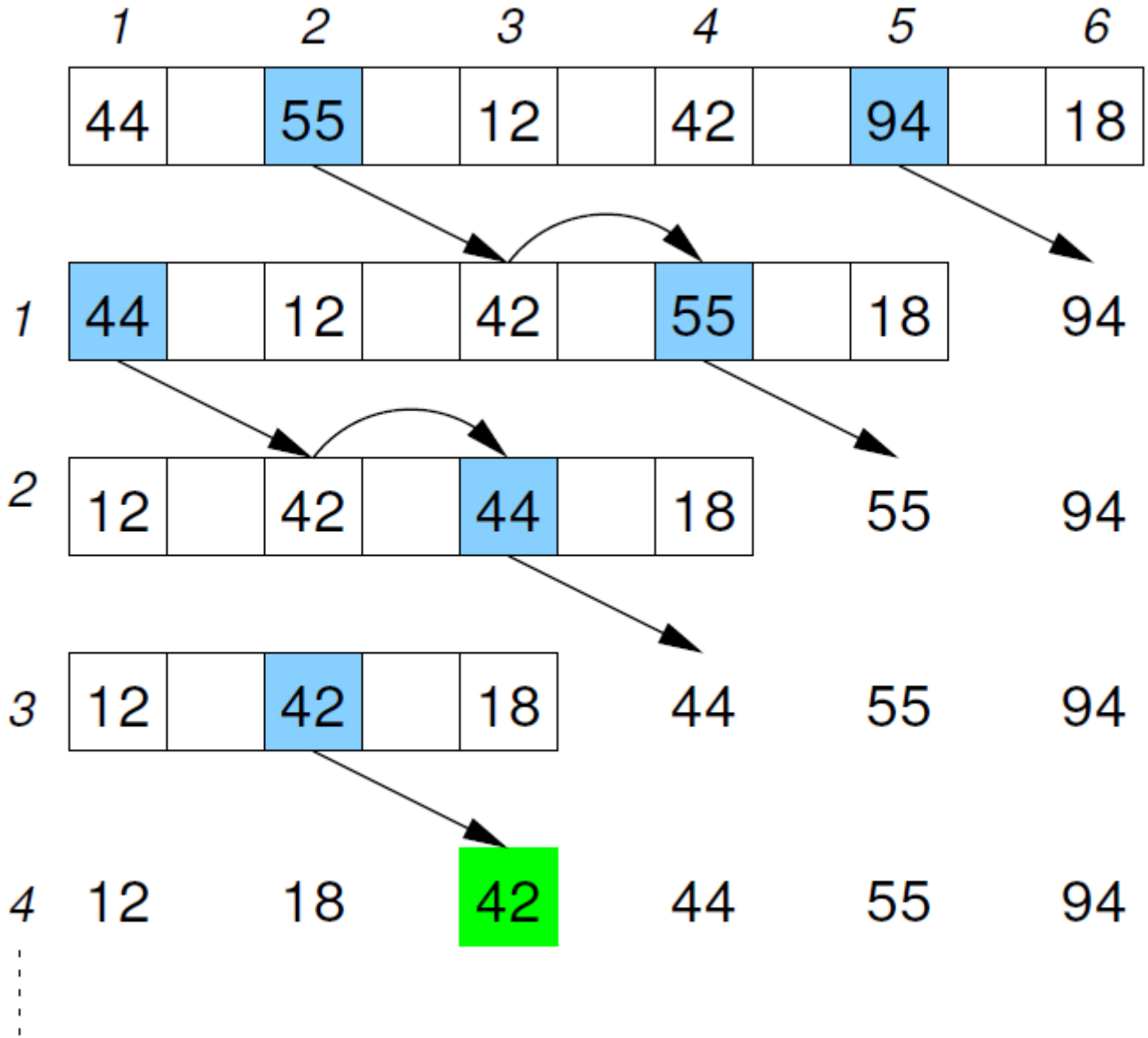
Bubble Sort



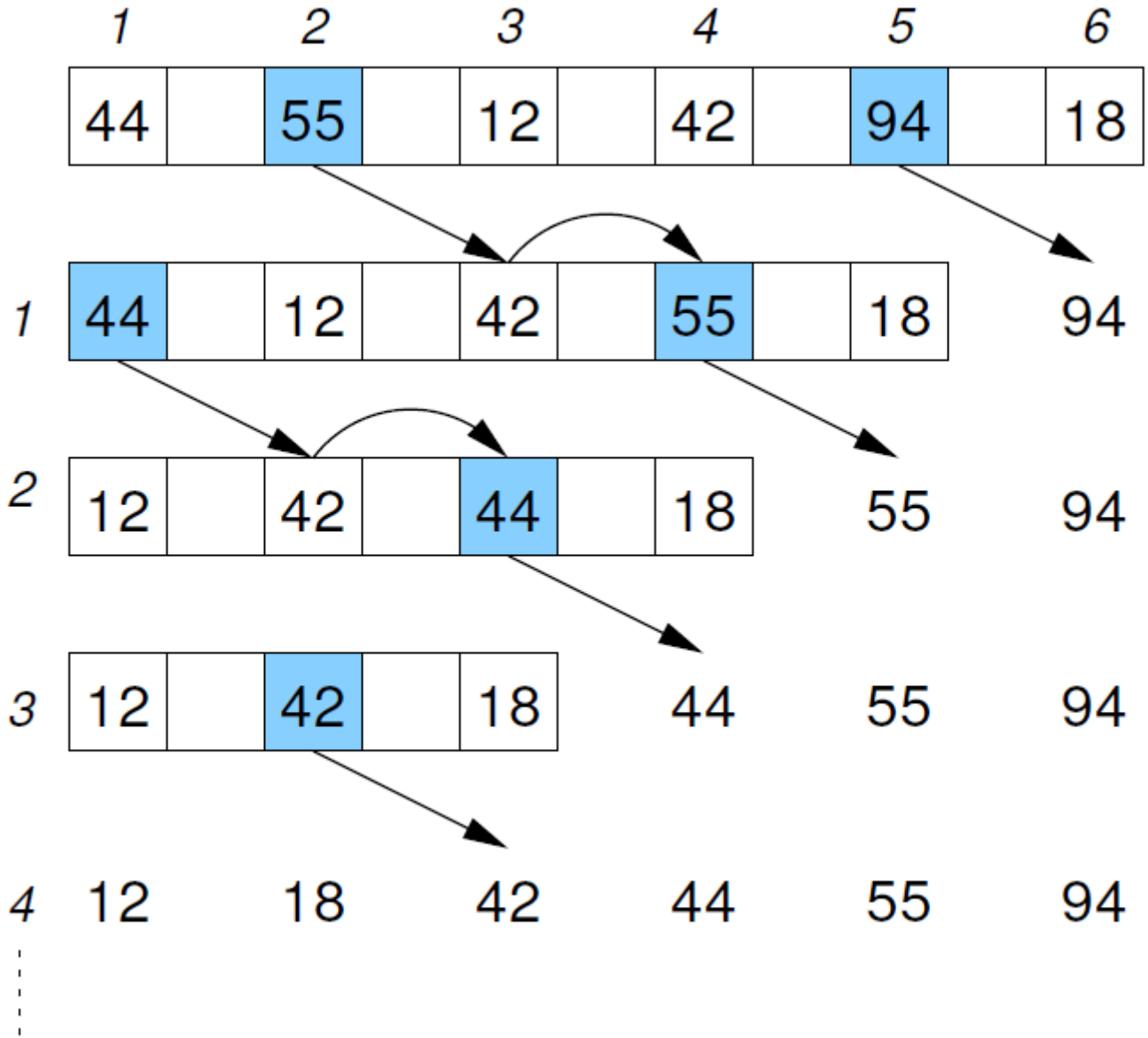
Bubble Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 44 12 42 55 18 94
2 12 42 44 18 55 94
3 12 42 18 44 55 94

Bubble Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 44 12 42 55 18 94
2 12 42 44 18 55 94
3 12 42 18 44 55 94
4 12 18 42 44 55 94

Bubble Sort


1 2 3 4 5 6
44 55 12 42 94 18
1 44 12 42 55 18 94
2 12 42 44 18 55 94
3 12 42 18 44 55 94
4 12 18 42 44 55 94

Bubble Sort

- „Maximální prvek probublává na i-tou pozici, kde i=n, n-1, … , 1“

```
def bubbleSort(array):
    for last in range(len(array)-1, 0, -1):
        for i in range(last):
            if array[i]>array[i+1]:
                array[i],array[i+1]=array[i+1],array[i]
```

Který je lepší?

Řazení

- Porovnání algoritmů
  - Časová náročnost (best, average, worst)

    Porovnávací algoritmy nemohou být lepší než $n \cdot log\, n$
  - Paměťová náročnost (memory)
  - Stabilita řazení
- Vše v „big O“ notaci

Více se tomuto tématu budeme věnovat v druhé půlce semestru

Algoritmus	Best	Average	Worst	Memory	Stable
Insertion sort	n	n^2	n^2	1	Yes
Selection sort	n^2	n^2	n^2	1	No
Bubble sort	n	n^2	n^2	1	Yes

Základy algoritmizace

- Dnes:
  - Vyhledávání v poli
    - Lineární vyhledávání
    - Binární vyhledávání
  - Řazení pole
    - Insertion sort
    - Selection sort
    - Bubble sort

Příště pokročilejší datové struktury