METODE DE SORTARE INTERNA

Metode directe de sortare: interschimbare, selectie, insertie. Metode specifice de sortare: radix, numarare, bucket

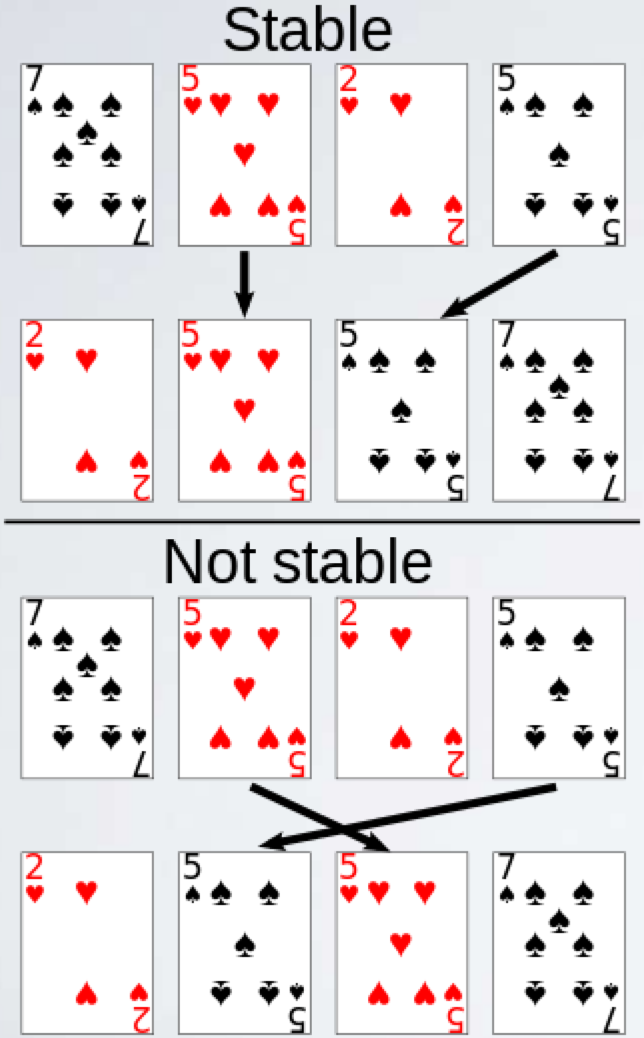
PROBLEMA SORTARII

- Exista o relatie de ordine liniara intre cheile obiectelor pe care dorim sa le sortam
- Avem o secventa de obiecte (inregistrari, elemente) $\boldsymbol{r}_1$, $\boldsymbol{r}_2,\ldots, \boldsymbol{r}_n$ cu cheile $\boldsymbol{k}_1, \boldsymbol{k}_2,\ldots, \boldsymbol{k}_n$, respectiv, trebuie sa re-aranjam obiectele in ordinea $\boldsymbol{r}_{i1}, \boldsymbol{r}_{i2},\ldots, \boldsymbol{r}_{in},$ astfel incat $\boldsymbol{k}_{i1}\leq \boldsymbol{k}_{i2}\leq\ldots \leq\boldsymbol{k}_{in}$
  - i.e. sa generam o permutare crescatoare

ALGORITMI DE SORTARE - DIMENSIUNI DE ANALIZA

- Complexitatea computationala (defav, mediu, fav)
- Memoria aditionala utilizata
- Stabilitate
  - Vezi exemplul urmator

ALGORITMI DE SORTARE - DIMENSIUNI DE ANALIZA



https://commons.wikimedia.org/w/index.php?curid=25670889

- Stabilitate
  - pastreaza ordinea relativa a elementelor egale
  - Daca doua obiecte cu chei egale apar in aceeasi ordine in sirul ordonat ca si in sirul initial – neordonat atunci algoritmul de sortare este stabil.

ALGORITMI DE SORTARE - DIMENSIUNI DE ANALIZA

- Complexitatea computationala (defav, mediu, fav)
- Memoria aditionala utilizata
- Stabilitate
  - pastreaza ordinea relativa a elementelor egale
  - Daca doua obiecte cu chei egale apar in aceeasi ordine in sirul ordonat ca si in sirul initial – neordonat atunci algoritmul de sortare este stabil.
- Daca e bazata pe comparatii sau nu
- Strategia generala
  - inserare, interschimbare, selectie, interclasare
- Adaptivitate
  - cat de mult variaza timpul de rulare in functie de cum arata intrarea
  - Cum este influentat timpul de rulare daca sirul este gata ordonat inainte sa se ruleze algoritmul de sortare. Algoritmii care considera acest lucru sunt algoritmi adaptabili..

ALGORITMI DE SORTARE - DIMENSIUNI DE ANALIZA

- Complexitatea computationala = timpul de rulare
- Cum evaluam timpul de rulare
  - Numar de pasi pt a sorta *n* elemente
  - Numar de comparatii
  - Numar de “mutari” (atribuiri)

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

		i	$i+1$	
A		a_i	a_{i+1}	

- Facem o trecere prin sir, pornind de la primul element, si, pentru fiecare element:
    - Comparam cu elementul urmator (tot timpul comparam elemente adiacente-vecine)

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

		i	$i+1$	
A		a_{i+1}	a_i	

- Facem o trecere prin sir, pornind de la primul element, si, pentru fiecare element:
  - Comparam cu elementul urmator (tot timpul comparam elemente adiacente)
  - Daca ordinea lor nu e cea corecta, le interschimbam

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

		i	i+1	i+2	
A		a_{i+1}	a_i	a_{i+2}	

- Facem o trecere prin sir, pornind de la primul element, si, pentru fiecare element:
  - Comparam cu elementul urmator (tot timpul comparam elemente adiacente)
  - Daca ordinea lor nu e cea corecta, le interschimbam
  - Avansam la urmatorul element

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

- Facem o trecere prin sir, pornind de la primul element, si, pentru fiecare element:
  - Comparam cu elementul urmator (tot timpul comparam elemente adiacente)
  - Daca ordinea lor nu e cea corecta, le interschimbam
  - Avansam la urmatorul element

1. Ce se intampla dupa prima parcurgere a sirului?

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

- Facem o trecere prin sir, pornind de la primul element, si, pentru fiecare element:
  - Comparam cu elementul urmator (tot timpul comparam elemente adiacente)
  - Daca ordinea lor nu e cea corecta, le interschimbam
  - Avansam la urmatorul element

1. Ce se intampla dupa prima parcurgere a sirului?

A

	n
	Max(a_i)

2. De cate treceri prin sir este nevoie pentru a fi siguri ca sirul este sortat?

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

n=8
swapped = false

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

A

1	2	3	4	5	6	7	8
3	9	12	5	7	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

	1	2	3	4	5	6	7	8
A	3	9	5	12	7	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

	1	2	3	4	5	6	7	8
A	3	9	5	12	7	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	5	7	12	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

	1	2	3	4	5	6	7	8
A	3	9	5	7	12	2	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	5	7	2	12	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	5	7	2	12	9	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

1	2	3	4	5	6	7	8
3	9	5	7	2	9	12	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

	1	2	3	4	5	6	7	8
A	3	9	5	7	2	9	12	5

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	5	7	2	9	5	12

n=8
swapped = true

```
BUBBLE-SORT (A[1…n])
 n = length(A)
   repeat
     swapped = false
     for i = 1 to n-1 inclusive do
       /* if this pair is out of order */
       if A[i] > A[i+1] then
         /* swap them and remember something changed */
         swap(A[i], A[i+1])
         swapped = true
     n=n-1
   until not swapped
```

Dupa prima trecere:

A

1	2	3	4	5	6	7	8
3	9	5	7	2	9	5	12

n=7
swapped = true

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

Dupa a doua trecere completa:

	1	2	3	4	5	6	7	8
A	3	5	7	2	9	5	9	12

n=6
swapped = true

Dupa a treia trecere completa:

	1	2	3	4	5	6	7	8
A	3	5	2	7	5	9	9	12

n=5
swapped = true

etc.

SORTAREA PRIN INTERSCHIMBARE (BUBBLESORT)

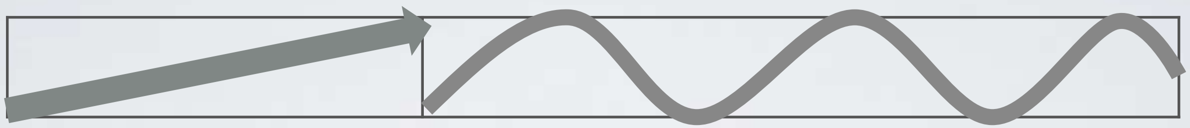
UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- *Rabbits and turtles (maxime vs minime)*
  - Versiuni imbunatatite (see Wikipedia)
    - Cocktail sort
    - Comb sort
- Complexitate?
  - favorabil, defavorabil
- Stabilitate?
- Adaptivitate?
- Memorie aditionala?

SELECTIE SI INSERARE

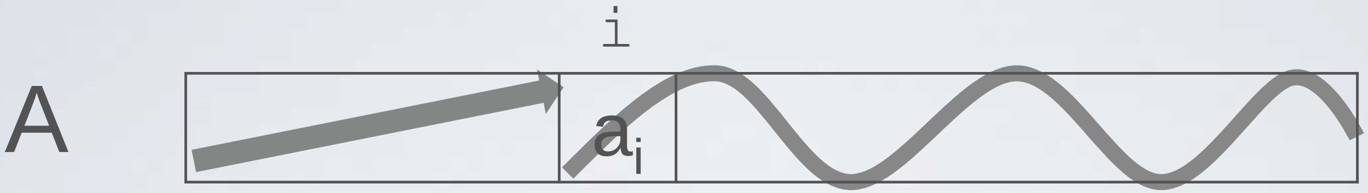
$sortat_i$ $nesortat_i$

A

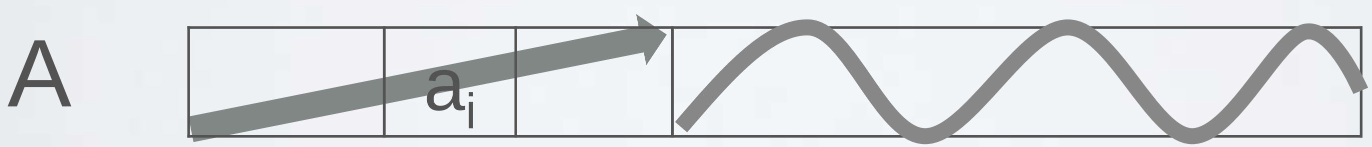


- *La fiecare iteratie, sirul = parte <u>sortata</u> si parte <u>nesortata</u>*
- *Se alege un element din partea nesortata, si se adauga partii sortate (astfel partea sortata creste cu `1` la fiecare iteratie*
  - *INSERARE: se alege un element OARECARE (primul) din partea nesortata si se **cauta** pozitia lui in partea sortata*
  - *SELECTIE: se **cauta** un element anume (minimul) din partea nesortata si se adauga pe o pozitie OARECARE (urmatoarea) din partea sortata*

SORTAREA PRIN INSERARE



- *Pentru fiecare din elementele de la 2 la n*
  - *Cauta pozitia in partea sortata (`1…i-1`)*
  - *Insereaza elementul pe pozitia aceea*



SORTAREA PRIN INSERARE

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	3	9	12	5	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	5	9	12	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	5	9	12	7	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	5	7	9	12	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	3	5	7	9	12	2	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	2	3	5	7	9	12	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	2	3	5	7	9	12	9	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	2	3	5	7	9	9	12	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

	1	2	3	4	5	6	7	8
A	2	3	5	7	9	9	12	5

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

SORTAREA PRIN INSERARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

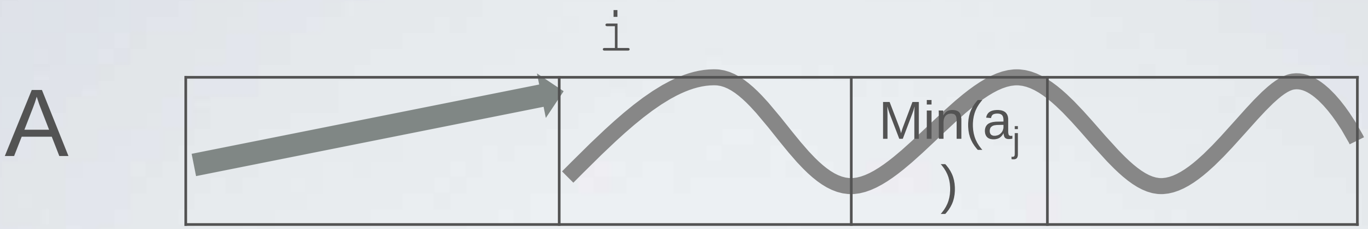
	1	2	3	4	5	6	7	8
A	2	3	5	5	7	9	9	12

```
INSERT-SORT(A[1…n])
for i = 2 to length(A)
    x = A[i]
    j = i - 1
    while j >= 1 and A[j] > x
        A[j+1] = A[j]
        j = j - 1
    A[j+1] = x
```

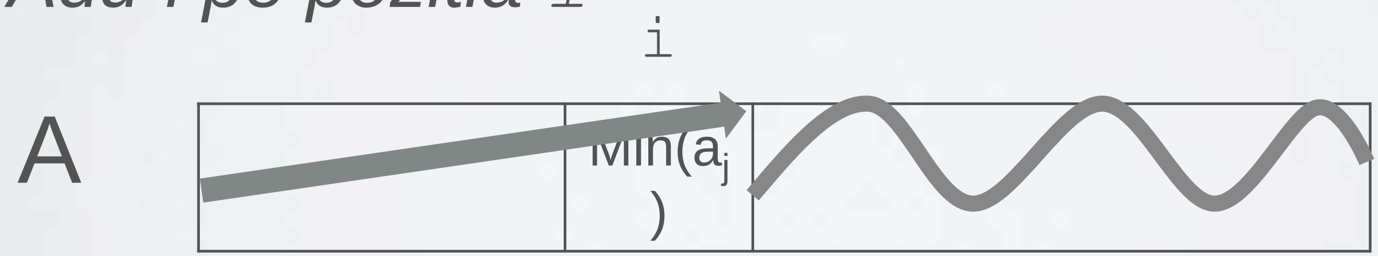
SORTAREA PRIN INSERARE

- *Cautarea binara a pozitiei de inserare in partea sortata*
- Complexitate?
  - favorabil, defavorabil
- Stabilitate?
- Adaptivitate?
- Memorie aditionala?

SORTAREA PRIN SELECTIE



- *Pentru fiecare element* `i` *de la primul pana la penultimul*
  - *Cauta minimul intre elementele* `i…n`
  - *Adu-l pe pozitia* `i`



SORTAREA PRIN SELECTIE

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	12	5	7	9	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	2	3	12	5	7	9	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	2	3	12	5	7	9	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
          then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	2	3	5	12	7	9	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
          then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	5	12	7	9	9	5

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	5	5	7	9	9	12

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	5	5	7	9	9	12

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	5	5	7	9	9	12

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
           then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

	1	2	3	4	5	6	7	8
A	2	3	5	5	7	9	9	12

```
SELECTION-SORT(A[1…n])
  for i = 1 to n-1
    /* find the min element in the unsorted A[i .. n] */
    jMin = i;
    /* test against elements after i to find the min */
    for j = i+1 to n
        if A[j] < A[jMin]
          then jMin = j
    if jMin != i
        then swap(a[i], a[jMin])
```

SORTAREA PRIN SELECTIE

- Complexitate?
  - favorabil, defavorabil
- Stabilitate?
- Memorie?
- Adaptivitate?
  - Dintre cei 3 algoritmi de pana acum, care este cel mai adaptabil?
  - Dar cel mai putin adaptabil?

SORTARI BAZATE PE COMPARATII

- Pana acum, am vazut sortari bazate pe comparatii – decizii luate in urma *compararii* a 2 elemente
  - Mergesort, Quicksort
  - Selection, Insertion, Bubble
- Avantaj – algoritm general; aplicabil pe orice fel de chei, atata timp cat se pot compara (suprascriem comparatorul)
- Dezavantaj – algoritmul utilizeaza doar un bit de informatie la fiecare apel de *comparare*
  - Avem n! ordonari posibile => avem nevoie de cel putin log(n!) = O(nlogn) apeluri de *comparare*

SORTARI SPECIFICE

- Ce facem daca avem de sortat n stringuri a cate m caractere fiecare?
  - Algoritm bazat pe comparatii: `O(nmlogn)`
- Dar daca cunoastem de la inceput ca avem chei intregi in intervalul `1…k`?
  - Nu putem sorta oare mai eficient in aceasta situatie?

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

- Cheile sunt intregi in intervalul `1…k`, deci pot fi folosite pt. indexarea unui vector (nu se bazeaza pe comparatii)
- Se numara aparitiile fiecarei chei din A, si se stocheaza aceasta informatie in vectorul C

	1	2	3	4	5	6
***A* =**	3	4	1	4	1	1
***C* =**	3	0	1	2		

- Apoi se determina numarul de elemente care sunt <= cu fiecare dintre chei, prin calcularea *sumelor prefix*

C = 3 3 4 6

SORTAREA PRIN NUMARARE

- Rezultatul se genereaza intr-un vector nou, B, parcurgand A in sens invers si determinand pozitia elementului prin informatia stocata in C; se scade cu `1` valoarea intrarii elementului in C

	1	2	3	4	5	6
A =	3	4	1	4	1	1
B =			1			
C =	2	3	4	6		

SORTAREA PRIN NUMARARE

```
COUNTING-SORT(A,B,k)
let C[1…k] be a new array
for i=1 to k
    C[i] = 0
for j=1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements = i
for i=2 to k
    C[i] = C[i] + C[i-1]
// C[i]i now contains the number of elements <= i
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

- Stabilitate?
- Memorie?

SORTAREA PRIN NUMARARE

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0

```
let C[1…k] be a new array
for i=1 to k
    C[i] = 0
```

SORTAREA PRIN NUMARARE

A

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	0	0	0	0	0	0	0	0	0	0

```
4 for j=1 to A.length
5     C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

	j	1	2	3	4	5	6	7	8
A		9	3	12	5	7	2	9	5

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	0	0	0	0	0	0	1	0	0	0

```
4 for j=1 to A.length
5     C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	0	0	0	0	0	1	0	0	0

```
4 for j=1 to A.length
5      C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	0	0	0	0	0	1	0	0	1

```
4 for j=1 to A.length
5     C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	0	1	0	0	0	1	0	0	1

```
for j=1 to A.length
    C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	0	1	0	1	0	1	0	0	1

```
for j=1 to A.length
     C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

1	2	3	4	5	6	7	8	9	10	11	12
0	1	1	0	1	0	1	0	1	0	0	1

```
4 for j=1 to A.length
5       C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	1	0	1	0	1	0	2	0	0	1

```
4 for j=1 to A.length
5       C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	1	0	2	0	1	0	2	0	0	1

```
for j=1 to A.length
    C[A[j]] = C[A[j]] + 1
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	1	0	2	0	1	0	2	0	0	1

```
7 for i=2 to k
8     C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	1	0	2	0	1	0	2	0	0	1

```
7 for i=2 to k
8       C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	0	2	0	1	0	2	0	0	1

```
7 for i=2 to k
8     C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

	j	1	2	3	4	5	6	7	8
A		9	3	12	5	7	2	9	5

	i	1	2	3	4	5	6	7	8	9	10	11	12
C		0	1	2	2	2	0	1	0	2	0	0	1

```
7 for i=2 to k
8      C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	0	1	0	2	0	0	1

```
7 for i=2 to k
8       C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	1	0	2	0	0	1

```
7 for i=2 to k
8      C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	0	2	0	0	1

```
for i=2 to k
    C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	2	0	0	1

```
7 for i=2 to k
8      C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	7	0	0	1

```
for i=2 to k
    C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	7	7	0	1

```
7 for i=2 to k
8      C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	7	7	7	1

```
for i=2 to k
      C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

j	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	7	7	7	8

```
for i=2 to k
    C[i] = C[i] + C[i-1]
```

SORTAREA PRIN NUMARARE

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
	0	1	2	2	4	4	5	5	7	7	7	8

B

j	1	2	3	4	5	6	7	8
	9	3	12	5	7	2	9	5

```
10 for j=A.length downto 1
11     B[C[A[j]]] = A[j]
12     C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	4	4	5	5	7	7	7	8

j	1	2	3	4	5	6	7	8
B				5				

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	3	4	5	5	7	7	7	8

j	1	2	3	4	5	6	7	8
B				5				

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	3	4	5	5	7	7	7	8

j	1	2	3	4	5	6	7	8
B				5			9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	3	4	5	5	6	7	7	8

j	1	2	3	4	5	6	7	8
B				5			9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	1	2	2	3	4	5	5	6	7	7	8

j	1	2	3	4	5	6	7	8
B	2			5			9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	2	2	3	4	5	5	6	7	7	8

j	1	2	3	4	5	6	7	8
B	2			5			9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	2	2	3	4	5	5	6	7	7	8

j	1	2	3	4	5	6	7	8
B	2			5	7		9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	2	2	3	4	4	5	6	7	7	8

B

j	1	2	3	4	5	6	7	8
	2			5	7		9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	2	2	3	4	4	5	6	7	7	8

B

j	1	2	3	4	5	6	7	8
B	2		5	5	7		9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	2	2	2	4	4	5	6	7	7	8

j	1	2	3	4	5	6	7	8
B	2		5	5	7		9	

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	2	2	2	4	4	5	6	7	7	8

B

j	1	2	3	4	5	6	7	8
	2		5	5	7		9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

1	2	3	4	5	6	7	8
9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	2	2	2	4	4	5	6	7	7	7

B

j	1	2	3	4	5	6	7	8
	2		5	5	7		9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	2	2	2	4	4	5	6	7	7	7

j	1	2	3	4	5	6	7	8
B	2	3	5	5	7		9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	2	2	4	4	5	6	7	7	7

j	1	2	3	4	5	6	7	8
B	2	3	5	5	7		9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	2	2	4	4	5	6	7	7	7

j	1	2	3	4	5	6	7	8
B	2	3	5	5	7	9	9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

A

	1	2	3	4	5	6	7	8
A	9	3	12	5	7	2	9	5

C

i	1	2	3	4	5	6	7	8	9	10	11	12
C	0	0	1	2	2	4	4	5	5	7	7	7

B

j	1	2	3	4	5	6	7	8
B	2	3	5	5	7	9	9	12

```
for j=A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

SORTAREA PRIN NUMARARE

- Complexitate: O(n+k)
- Memorie: O(n+k)
- Stabilitate ?
- Se utilizeaza de regula ca sub-rutina in radix sort

BUCKET SORT

- Elementele din sir sunt impartite intr-un numar de partitii (en. *buckets*)
- Sortam *bucketurile* cu un alt algoritm - e.g. insertion sort
- Sortare distribuita
- $\Theta(n)$ in cazul mediu, presupunand distributie uniforma a cheilor, si am ales intervalele suficient de mici
  - E.g. avem de sortat numere reale intre 0.0 si 1.0, uniform distribuite
- Poate/poate sa nu fie bazata pe comparatii (e.g. pt chei intregi, daca avem 10 partitii, se impart cheile la 10, si luam partea intreaga pt a selecta partitia)

BUCKET SORT

```
BUCKET-SORT(A[1…n])
Let B[0…n-1] be a new array                Ω(1)
for i =1 to n                              O(n) * Ω(1)
   insert A[i] into bucket B[translate(A[i],n)]
for i = 0 to n-1                           O(n) * O(ni^2)
   sort(bucket B[i]) //typically w. insertion
 Concatenate buckets B[0],B[1],…, B[n-1]   O(n)
```

n_i este dimensiunea *bucketului* $\boldsymbol{B}[\boldsymbol{i}]$

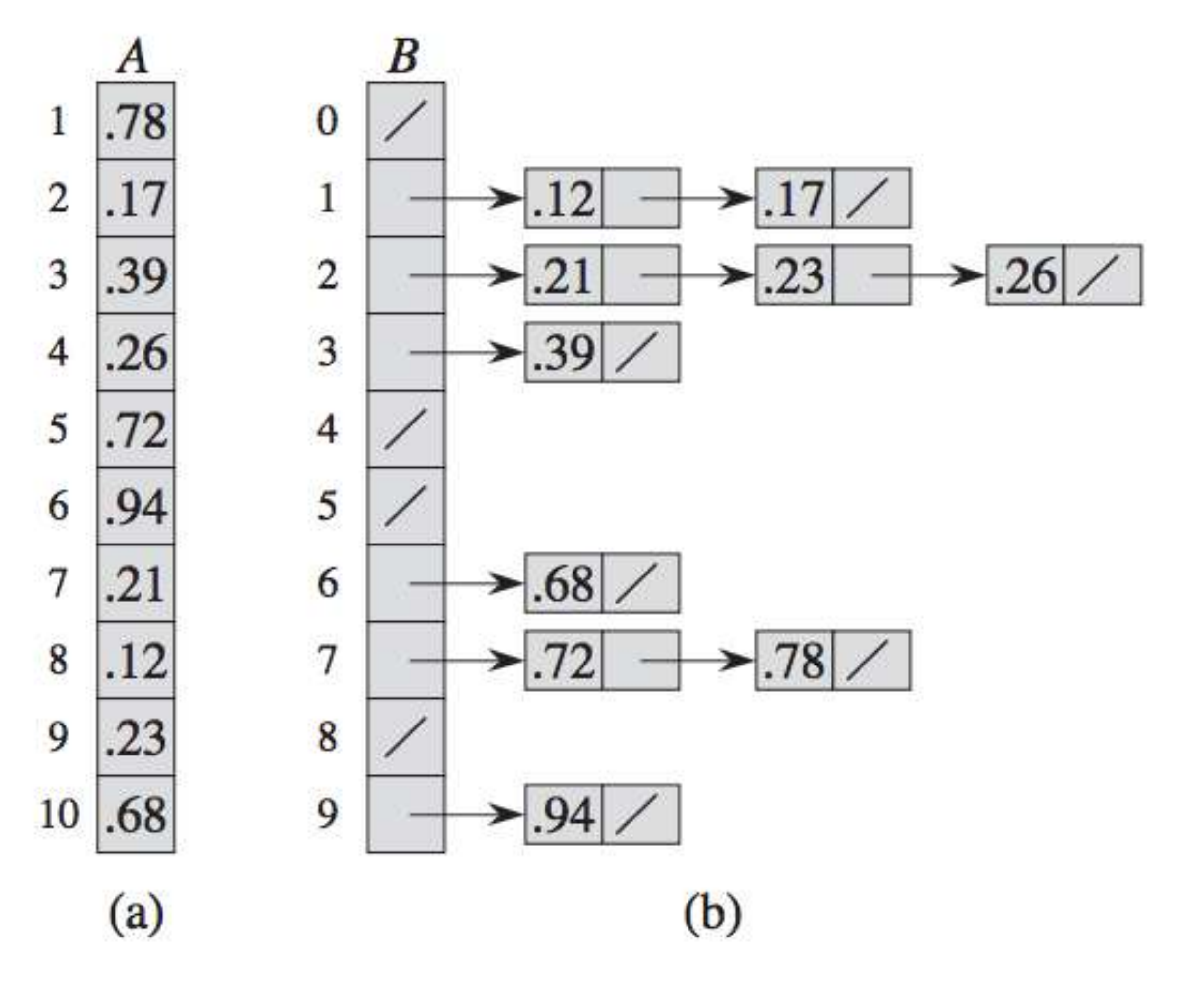
$$T(n) = \Theta(n) + \sum_{i=0}^{n-1} O({n_i}^2)$$

Cazul mediu: $\mathrm{E}[T(n)] = E[\Theta(n) + \sum_{i=0}^{n-1} O({n_i}^2)] =$

$$= \Theta(n) + \sum_{i=0}^{n-1} O(E[{n_i}^2]) =$$

$$\ldots = \Theta(n) + nO\left(2 - \frac{1}{n}\right) = \Theta(n)$$

BUCKET SORT – EXEMPLU



RADIX SORT

- Pp. ca avem de sortat o secventa de chei care pot fi comparate pe bucati: e.g. intregi de 3 cifre

```
RADIX-SORT(A[1…n], d)
   for i=1 to d
      use a stable sort to sort array A on digit i
```

De ce sortare stabila?

Exemplu:

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

RADIX SORT

- Eficienta
  - daca avem cifrele de la 1 la k, putem folosi sortarea prin numarare pt. a sorta cifra *i:* $\Theta(n+k)$
  - Facem d treceri prin sir: $\Theta(d(n+k))$
- Utilizare:
  - Sortare stringuri
  - Sortare date (an-luna-zi)
- Varianta LSB, exista si varianta MSB

CUM ALEGEM ALGORITMUL POTRIVIT

- Dimensiunea problemei
  - toate cursurile vs cursurile unui singur student
- Elementele de sortat ocupa memorie multa?
  - de evitat mutarile in cazul acesta
  - utilizare de structuri auxiliare - pointeri - mutam pointerii, nu elementele
- Avem nevoie de garantii asupra timpului de sortare (e.g. sisteme de control, retele)
  - nu putem utiliza QuickSort datorita comportamentului sau in cazul defavorabil

CUM ALEGEM ALGORITMUL POTRIVIT

- Elementele pot avea aceleasi chei? Avem nevoie de algoritm stabil?
  - *$O(n^2)$* tind sa fie stabili, *$O(nlgn)$* nu prea
  - Algoritmii instabili pot fi transformati in algoritmi stabili - cheie cu pozitia
    - costa spatiu si timp
  - Avem spatiu limitat?
    - MergeSort - O(n) - memorie
    - QuickSort - O(n) memorie in cazul defav.; O(logn) mediu

CUM ALEGEM ALGORITMUL POTRIVIT

- S-ar putea ca secventa sa nu incapa in memorie? (memorie virtuala)
  - Daca da se prefera algoritmi cu comportament local
    - HeapSort, QuickSort, met. directe
- S-ar putea ca secventa sa nu incapa nici in memoria virtuala?
  - sortari externe
- Ce stim despre intrare?
  - Daca sunt intr-un domeniu mic -> CountingSort
  - Daca avem chei compuse din parti ce se compara individual -> RadixSort
  - Daca avem chei numere reale distribuite uniform intr-un anumit interval -> BucketSort

COMPARATIE ALGORITMI DE SORTARE

Algoritm	Timp (Mediu/Defav)		Mem.	Stabil	Observatii
BubbleSort	O(n^2)	O(n^2)	O(1)	DA	‘’Tiny code size” (Wikipedia)
InsertionSort	O(n^2)	O(n^2)	O(1)	DA	Cautarea binara imbunatateste timpul de cautare a pozitiei de inserare
SelectionSort	O(n^2)	O(n^2)	O(1)	NU	Cel mai putin adaptiv
MergeSort	O(nlogn)	O(nlogn)	O(n)	DA	Not in-place; daca se aplica pe liste, nu necesita memorie aditionala; se poate aplica eficient pe liste; extrem de paralelizabil
QuickSort	O(nlogn)	O(n^2)	O(logn)	NU	Selectia aleatoare a pivotului, pt a evita cazul defvorabil. Se poate face sa fie stabila.
RadixSort	O(d(n+k))	O(d(n+k))	O(n+k)	DA	Daca se utilizeaza counting sort pt. sortarea pe digit
BucketSort	O(n)	O(n^2)	O(n)	DA	
CountingSort	O(n+k)	O(n+k)	O(n+k)	DA	Not in-place

PROBLEME PROPUSE

1. Fiind date doua siruri X si Y de numere intregi, pozitive, determinati toate perechile (x,y) astfel incat $x^y > y^x$ unde x este un element din sirul X si y este un element din Y.
   Exemplu: X={2,1,6}, Y = {1, 5}
   Iesire: Exista 3 perechi (x,y) si anume (2, 1), (2, 5) si (6, 1)

2. Folositi RadixSort ca sa ordonati crescator o multime de date de forma {zz,ll, an}, cu an >= 2000.
   Exemplu:

Datele de intrare:	Datele de iesire sortate:
{20, 1, 2014}	{ 3, 12, 2000}
{25, 3, 2010}	{18, 11, 2001}
{ 3, 12, 2000}	{ 9, 7, 2005}
{18, 11, 2001}	{25, 3, 2010}
{19, 4, 2015}	{20, 1, 2014}
{ 9, 7, 2005}	{19, 4, 2015}

3. Se da un sir de cuvinte. Ordonati sirul crescator in functie de lungimea fiecarui cuvant.
   Exemplu: cuvinte = {“invat”, “la”, “SDA”}
   Iesire: la SDA invat

4. Se da o lista simplu inalntuita care are proprietatea ca elementele sale sunt ordonate crescator apoi descrescator, apoi crescator s.a.m.d. ca in exemplu. Scrieti un algoritm care sorteaza lista crescator.
   Exemplu: Lista: 10->40->53->30->67->12->89->NULL
   Lista ordonata este: 10->12->30->43->53->67->89->NULL

BIBLIOGRAFIE

- Th. Cormen et al.: Introduction to Algorithms, cap. 8, sect. 2.1
- Animatii – algoritmi de sortare:
  - https://www.toptal.com/developers/sorting-algorithms/