

Seminar 2.2 - ALGORITMAT E RENDITJES

SELECTION Sort

- Algoritëm in-place sorting
- Më i mirë se bubble sort
- Vektori ndahet në dy nëngrupe vazhdimisht, pjesa e rradhitur dhe pjesa e parradhitur

Implementimi në Java

```
public class Selection
{
    public static void sort(Comparable[] a)
    {
        int N = a.length;
        for (int i = 0; i < N; i++)
        {
            int min = i;
            for (int j = i+1; j < N; j++)
                if (less(a[j], a[min]))
                    min = j;
            exch(a, i, min);
        }
    }

    private static void exch(Comparable[] a, int i, int j)
    {
        Comparable swap = a[i];
        a[i] = a[j];
        a[j] = swap;
    }
}
```

Shembull të iteracioneve për të rradhitur një vektor në rend rritës

	3	5	8	4	1	9	-2
$i = 0$	<u>-2</u>	5	8	4	1	9	3
$i = 1$	-2	<u>1</u>	8	4	5	9	3
$i = 2$	-2	1	<u>3</u>	4	5	9	8
$i = 3$	-2	1	3	<u>4</u>	5	9	8
$i = 4$	-2	1	3	4	<u>5</u>	9	8
$i = 5$	-2	1	3	4	5	<u>8</u>	9

Analiza e algoritmit

Kompleksiteti në kohë:

Rasti më i mirë (vektori i renditur): $O(n^2)$

Rasti mesatar: $O(n^2)$

Rasti më i keq (vektori i renditur në rend të kundërt): $O(n^2)$

Kompleksiteti në hapësirë: $O(1)$

INSERTION SORT

- Një algoritëm in-place sorting;
- I lehtë për tu përdorur në rastet kur i marrim elementët një e nga një dhe duam ti rradhisim hap pas hapi;
- Nëse vektori është i renditur nuk kryhen veprime zhvendosje;
-

Implementimi në Java

```
public void sort(Object[] data) {  
    for (int i= 1; i < data.length; i++) {  
        Comparable tmp= (Comparable)data[i];  
        for (j = i; j > 0 && tmp.compareTo(data[j-1]) < 0; j--)  
            data[j] = data[j-1];  
        data[j] = tmp;  
    }  
}
```

Shembull të iteracioneve për të rradhitur një vektor në rend rritës

```
i = 1  [ 3 8 5 4 1 9 -2 ]  
i = 2  [ 3 8 5 4 1 9 -2 ]  
i = 3  [ 3 5 8 4 1 9 -2 ]  
i = 4  [ 3 4 5 8 1 9 -2 ]  
i = 5  [ 1 3 4 5 8 9 -2 ]  
i = 6  [ 1 3 4 5 8 9 -2 ]  
      [ -2 1 3 4 5 8 9 ]
```

Analiza e algoritmit

Kompleksiteti në kohë:

Rasti më i mirë (vektori i renditur): $O(n)$

Rasti mesatar: $O(n^2)$

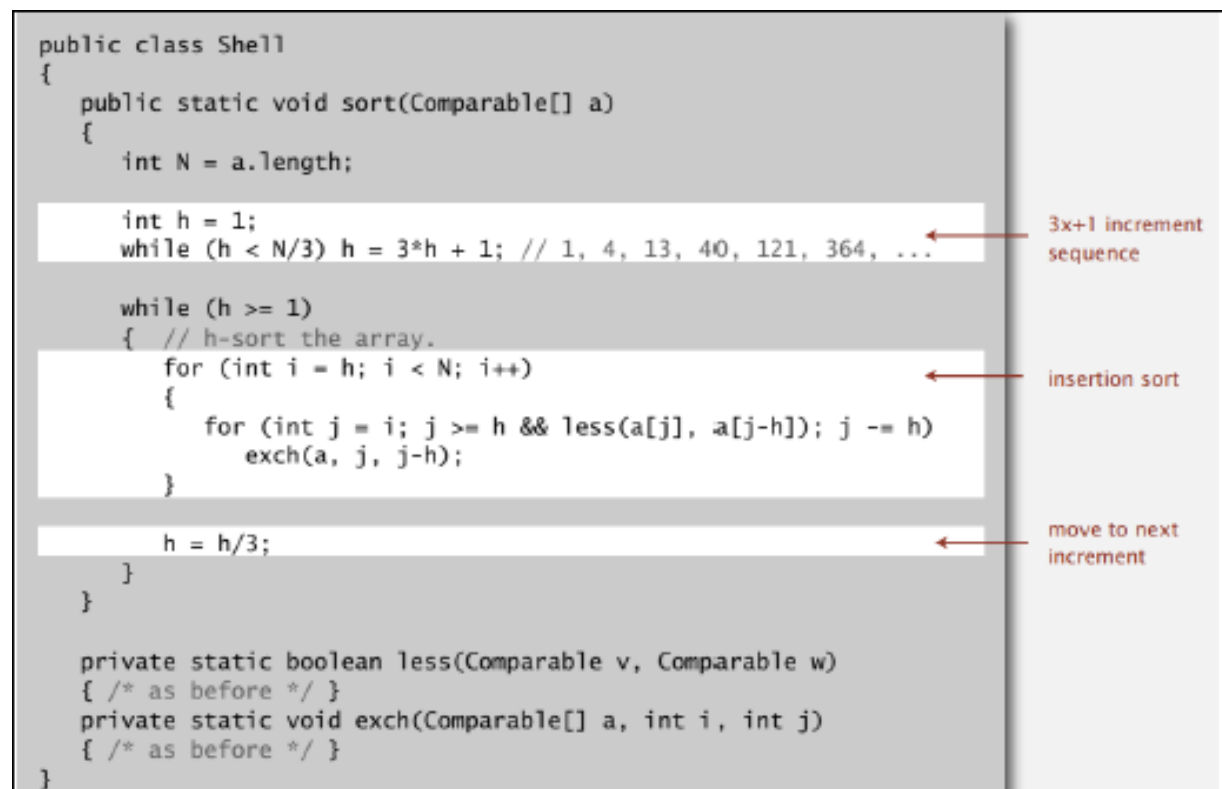
Rasti më i keq (vektori i renditur në rend të kundërt): $O(n^2)$

Kompleksiteti në hapësirë: $O(1)$

SHELL Sort

- Performancë më e mirë se insertion dhe selection sort
- Një përmirësim i insertion sort
- Algoritmi e ndan vektorin fillestar në nëngrupe më të vogla, ku secili nëngrup zakonisht rradhitet me insertion sort
- Nëngrupet (subsets) krijohen duke përdorur konceptin e intervalit. P.sh. nëse kemi intervalin ose hapësirën x , do të thotë që një nëngrup do të përfshijë të gjithë elementët që janë x pozicione larg.
- Eficient në rastin e listave me madhësi mesatare.

Implementimi në Java



```
public class Shell
{
    public static void sort(Comparable[] a)
    {
        int N = a.length;

        int h = 1;
        while (h < N/3) h = 3*h + 1; // 1, 4, 13, 40, 121, 364, ...

        while (h >= 1)
        { // h-sort the array.
            for (int i = h; i < N; i++)
            {
                for (int j = i; j >= h && less(a[j], a[j-h]); j -= h)
                    exch(a, j, j-h);
            }

            h = h/3;
        }

        private static boolean less(Comparable v, Comparable w)
        { /* as before */ }
        private static void exch(Comparable[] a, int i, int j)
        { /* as before */ }
    }
}
```

Annotations on the right side of the code block:

- 3x+1 increment sequence (points to the line `h = 3*h + 1;`)
- insertion sort (points to the inner loop of the `h-sort` block)
- move to next increment (points to the line `h = h/3;`)

Analiza e algoritmit

Kompleksiteti në kohë:

Varet nga mënyra se si përzgjidhet largësia (gap).

Rasti më i mirë për Shell Sort $(3x+1) = N \log N$

Rasti më i keq për Shell Sort $(3x+1) = N^{3/2}$

Kompleksiteti në hapësirë: $O(1)$

Pyetje

1. Pse ekzistojnë disa algoritme renditje?
2. Cili nga algoritmet (insertion, selection dhe shell sort) ka një kompleksitet të ndryshëm në rastin më të keq dhe në atë mesatar?
3. Cilat nga algoritmet (insertion, selection dhe shell sort) kanë një kompleksitet të ndryshëm në rastin më të mirë dhe atë mesatar?
4. Cili është numri maksimal i shkëmbimeve (exchange) për një element të caktuar x në algoritmin selection sort?
5. Cila metodë ekzekutohet më shpejt për një vektor me të gjithë vlerat identike, selection sort apo insertion sort?
6. Cila metodë ekzekutohet më shpejt për një vektor në rradhitje të kundërt, selection sort apo insertion sort?
7. Pse në h-sorting të algoritmit shellsort nuk përsoret selection sort?

Ushtrime

1. Tregoni të gjithë hapat se si rradhitet vektori i mëposhtëm:

E A S Y Q U E S T I O N

Duke përdorur:

- a. Selection Sort
 - b. Insertion Sort
2. Ndërttoni hap pas hapi të gjithë iteracionet që do të ndiqen për të rradhitur vektorin e mëposhtëm:
 $\text{int } a[] = \{44, 88, 55, 99, 66, 33, 22, 88, 77\}$
 - a. Selection sort
 - b. Insertion sort
 - c. Shell Sort
 3. Tregoni të gjithë hapat se si rradhitet vektori i mëposhtëm duke përdorur Shell Sort:

E A S Y S H E L L S O R T Q U E S T I O N

4. Përcaktoni një implementim 'Binary Insertion Sort' që përdor kërkimin binar për të gjetur pikën e shtimit në algoritmin insertion sort. Në momentin që gjendet pozicioni j për të vendosur vlerë, të gjithë vlerat e tjera zhvendosen me nga një pozicion djathtas. Numri i krahasimeve në këtë rast duhet të jetë $n \log n$. Ndërkohë numri i aksesimeve të vektorit do të jetë në nivel kuadratik për rastin më të keq.
5. **Dutch national flag.** Jepet një vektor me 0, 1 dhe 2. Shkruani një funksion i cili rradhit vektorin, duke vendosur të gjithë 0 në fillim dhe më pas të gjithë njëshat dhe dyshat. Zgjidhja duhet të jetë maksimumi e rendit n dhe pa hapësirë shtesë.

Shembull:

Input: {0, 1, 2, 0, 1, 2}

Output: {0, 0, 1, 1, 2, 2}

Input: {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}

Output: {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

6. Supozojmë se kemi një vektor me këto elemente: 6, 5, 4, 1, 2, 3. Tregoni sa shkëmbime (swaps) kryen Selection Sort për të renditur këtë vektor? Po Insertion Sort? Shpjegoni përgjigjen tuaj.
7. Për secilin nga algoritmet e mëposhtme të renditjes, tregoni sa është θ (si funksion i n -së) i numrit të krahasimeve që duhen për të renditur në rend rritës një matricë me n elemente të ndryshme edhe qe është e renditur në rend zbritës.
 - a. SelectionSort
 - b. Insertion sort
8. Ushtrime implementi i Shell Sort
<https://opensa-server.cs.vt.edu/OpenDSA/Books/Everything/html/Shellsort.html>