

INFORMATICA LES 10

SORTEREN EN INTERNET

JAN LEMEIRE

INFORMATICA 2^E SEMESTER

FEBRUARI – MEI 2021

Wat is 134.184.129.2?

Vandaag

- 1. Sorteren (2e deel)**
- 2. Internet: geschiedenis**
- 3. Internet: technologie**

Sorteren
vervolg

1. Selection Sort

Idee: zoek kleinste, dan tweede kleinste, enzovoorts

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	03	24	86	45	30	27	63	96	50	10
1	03	51	24	86	45	30	27	63	96	50	10
2	03	10	24	86	45	30	27	63	96	50	51
3	03	10	24	86	45	30	27	63	96	50	51
4	03	10	24	27	45	30	86	63	96	50	51
5	03	10	24	27	30	45	86	63	96	50	51
6	03	10	24	27	30	45	86	63	96	50	51
7	03	10	24	27	30	45	50	63	96	86	51
8	03	10	24	27	30	45	50	51	96	86	63
9	03	10	24	27	30	45	50	51	63	86	96
10	03	10	24	27	30	45	50	51	63	86	96
11	03	10	24	27	30	45	50	51	63	86	96

n stappen

2. Bubble Sort

Idee: 'bubbel' kleinste-tot-dan-toe naar boven

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	<u>03</u>	24	86	45	30	27	63	96	50	10
1	<u>03</u>	51	<u>10</u>	24	86	45	30	27	63	96	50
2	03	<u>10</u>	51	24	<u>27</u>	86	45	30	<u>50</u>	63	96
3	03	10	<u>24</u>	51	27	<u>30</u>	86	45	50	63	96
4	03	10	24	<u>27</u>	51	30	<u>45</u>	86	50	63	96
5	03	10	24	27	<u>30</u>	51	45	<u>50</u>	86	63	96
6	03	10	24	27	30	<u>45</u>	51	50	63	86	96
7	03	10	24	27	30	45	<u>50</u>	51	63	86	96
8	03	10	24	27	30	45	50	51	63	86	96

Performantie Bubble Sort

◆ Aantal vergelijkingen/verwisselingen?

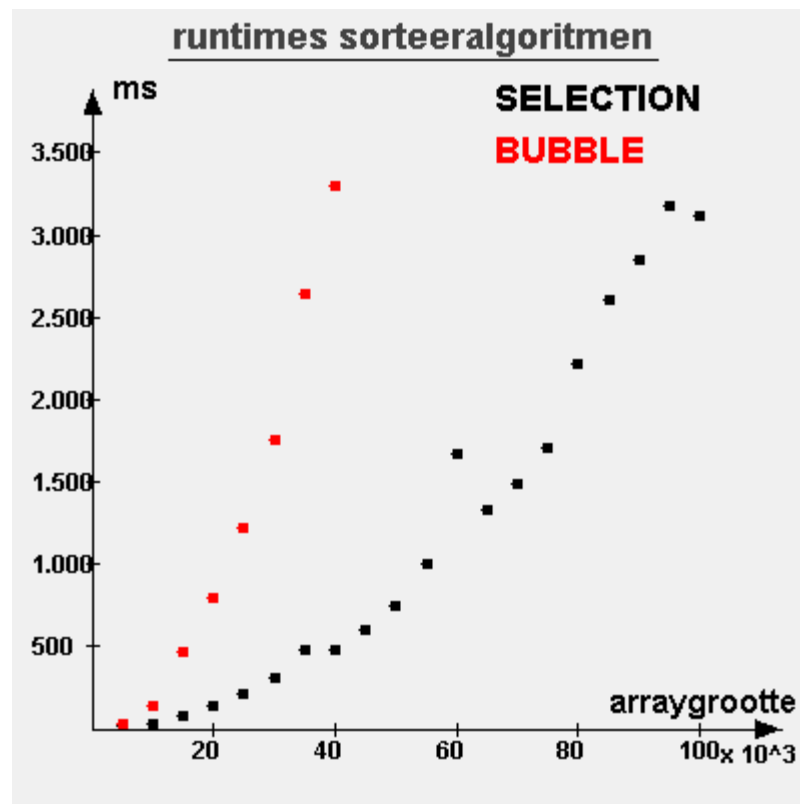
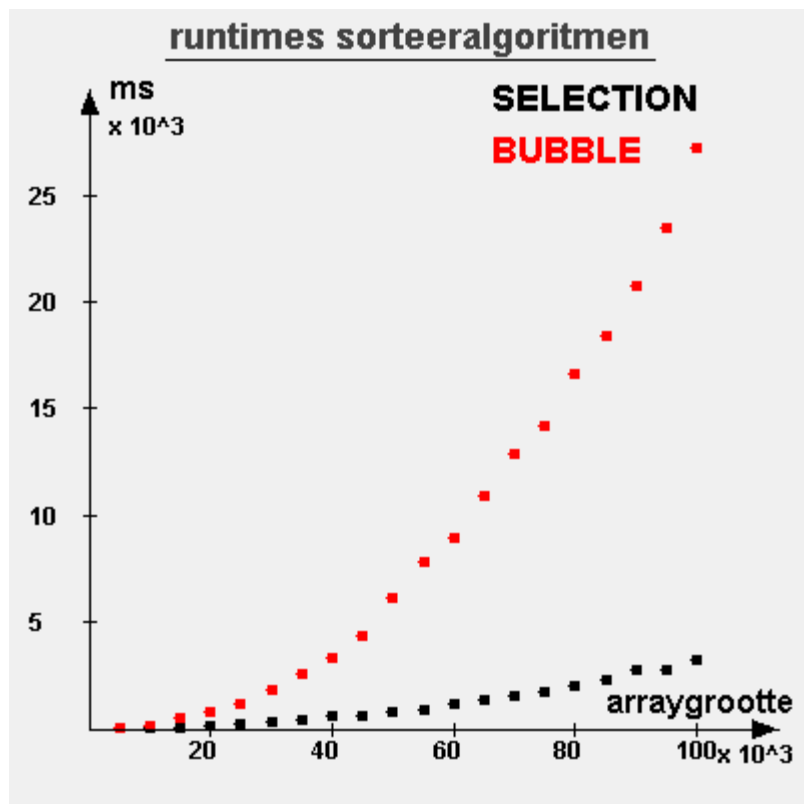
◆ Worst case: evenveel als selection sort

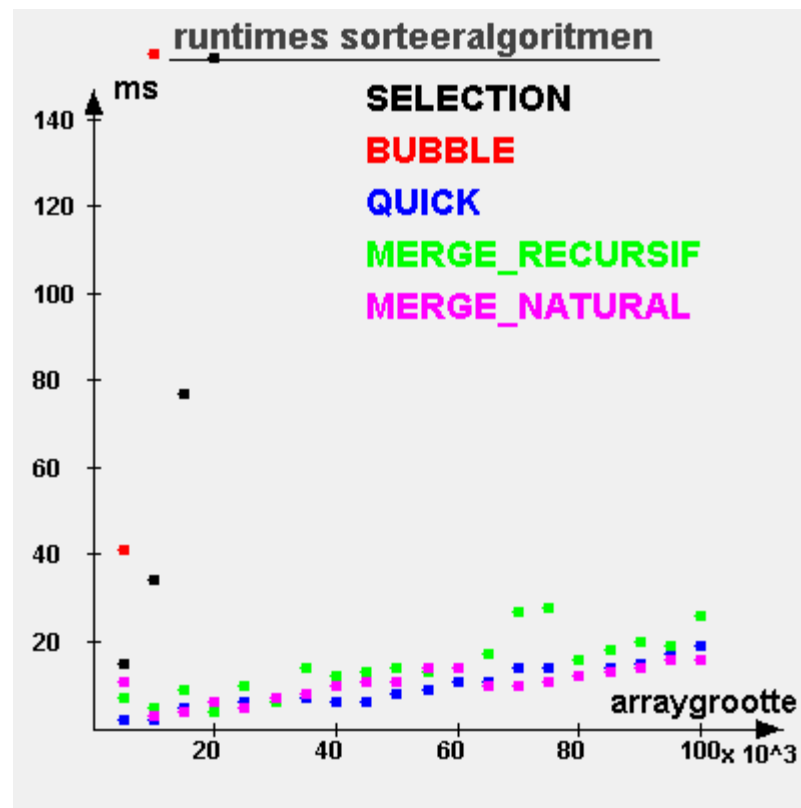
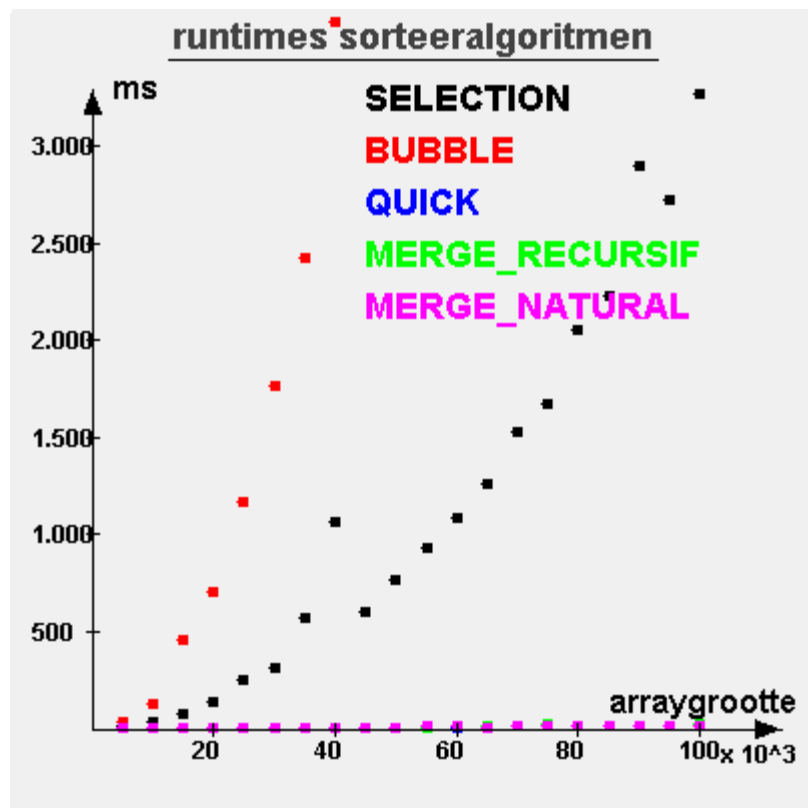
✦ Bvb in omgekeerde volgorde $O(n^2)$

◆ Best case? $O(n)$

✦ Array is al gesorteerd

✦ Of slechts enkelen niet op hun plaats





3. Quicksort

Idee: splits in deel met kleine elementen en deel met grote elementen

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
1	51	03	24	86	45	30	27	63	96	50	<u>10</u>
	3	<u>10</u>	24	86	45	30	27	63	96	50	51
2	3	10	24	86	45	30	27	63	96	50	<u>51</u>
	3	10	24	50	45	30	27	<u>51</u>	96	86	63
3	3	10	24	50	45	30	<u>27</u>	51	96	86	63
	3	10	24	<u>27</u>	45	30	50	51	96	86	63
4	3	10	24	27	45	30	<u>50</u>	51	96	86	63
	3	10	24	27	45	30	<u>50</u>	51	96	86	63
5	3	10	24	27	45	<u>30</u>	50	51	96	86	63
	3	10	24	27	<u>30</u>	45	50	51	96	86	63
6	3	10	24	27	30	45	50	51	96	86	<u>63</u>
	3	10	24	27	30	45	50	51	<u>63</u>	86	96
7	3	10	24	27	30	45	50	51	63	86	<u>96</u>
	3	10	24	27	30	45	50	51	63	86	<u>96</u>

Code I

Zie programma **Sorting**
in package **algoritmen**

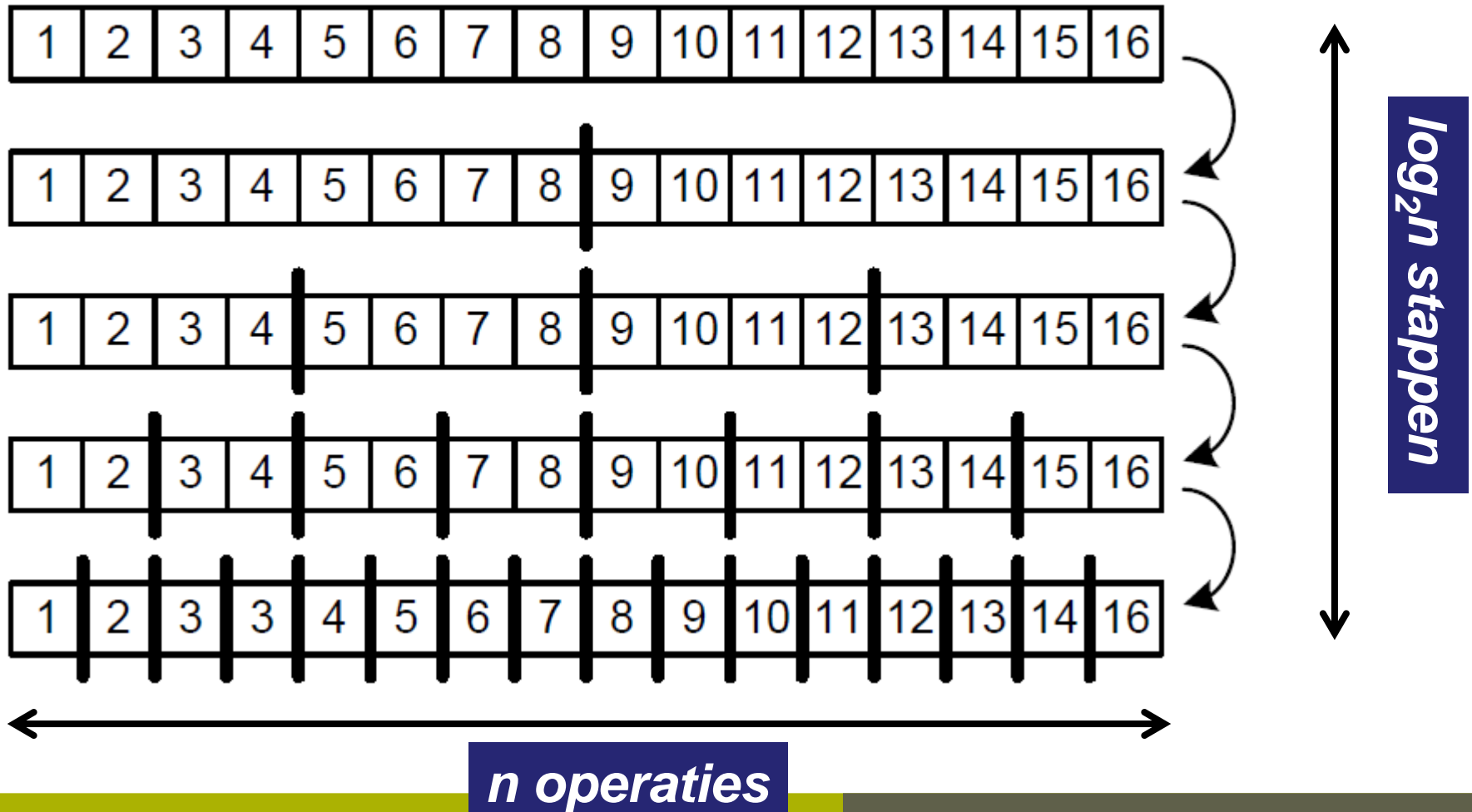
```
public static void quickSort(int[] array){
    aantalVergelijkingen=0;
    aantalKopies = 0;
    quicksort(array, 0, array.length - 1);
}
private static void quicksort(int[] array, int left, int right)
{
    if (right <= left) return;
    int i = partition(array, left, right);
    if (PRINT_TUSSEN_RESULTATEN)
System.out.println(" > ["+left+" | "+i+" | "+right+"]
"+Arrays.toString(array));
    quicksort(array, left, i-1);
    quicksort(array, i+1, right);
}
```

```

private static int partition(int[] a, int left, int right) {
    // a[right] is ons pivot-element
    int i = left;
    int j = right - 1;
    while (true) {
        while (a[i] < a[right]){ // vind links een element > pivot
            i++;
            aantalVergelijkingen++;
        }
        while (a[right] < a[j]){ // vind rechts een element < pivot
            aantalVergelijkingen++;
            if (j == left) // ga niet buiten array
                break;
            j--;
        }
        if (i >= j) // tests of indexen mekaar hebben gekruisd
            break;
        swap(a, i, j); // verwissel beide elementen
        i++;
        j--;
    }
    swap(a, i, right); // verwissel met pivot
    return i;
}

```

Aantal compare-swaps



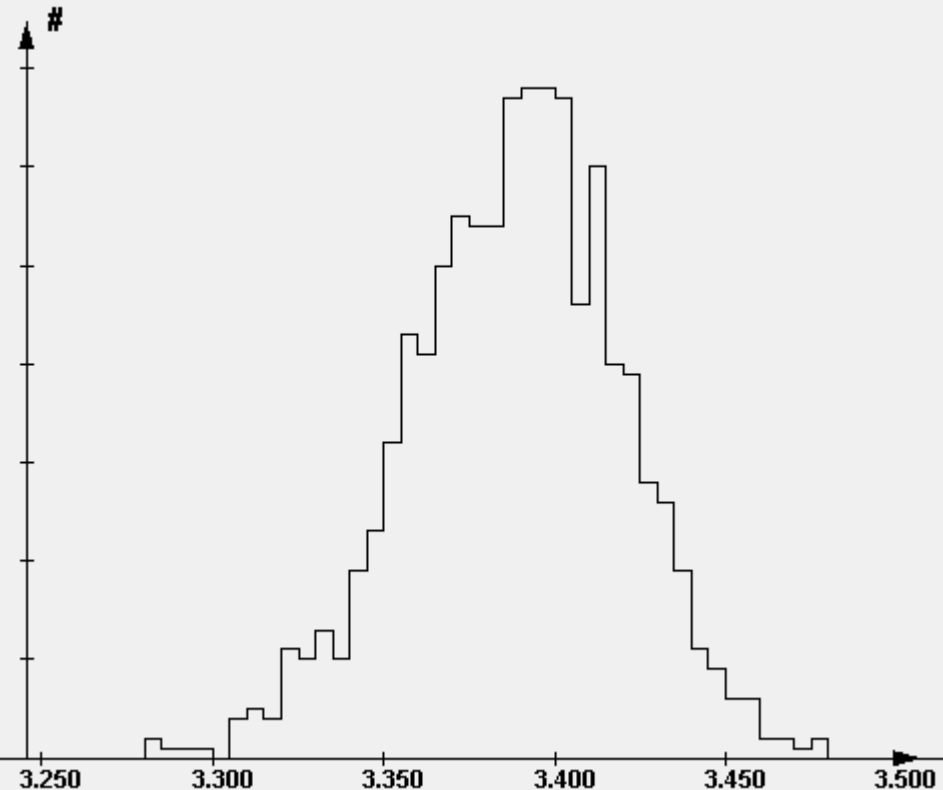
Let op: het recursieve algoritme doet dit eerst links

Performantie Quicksort

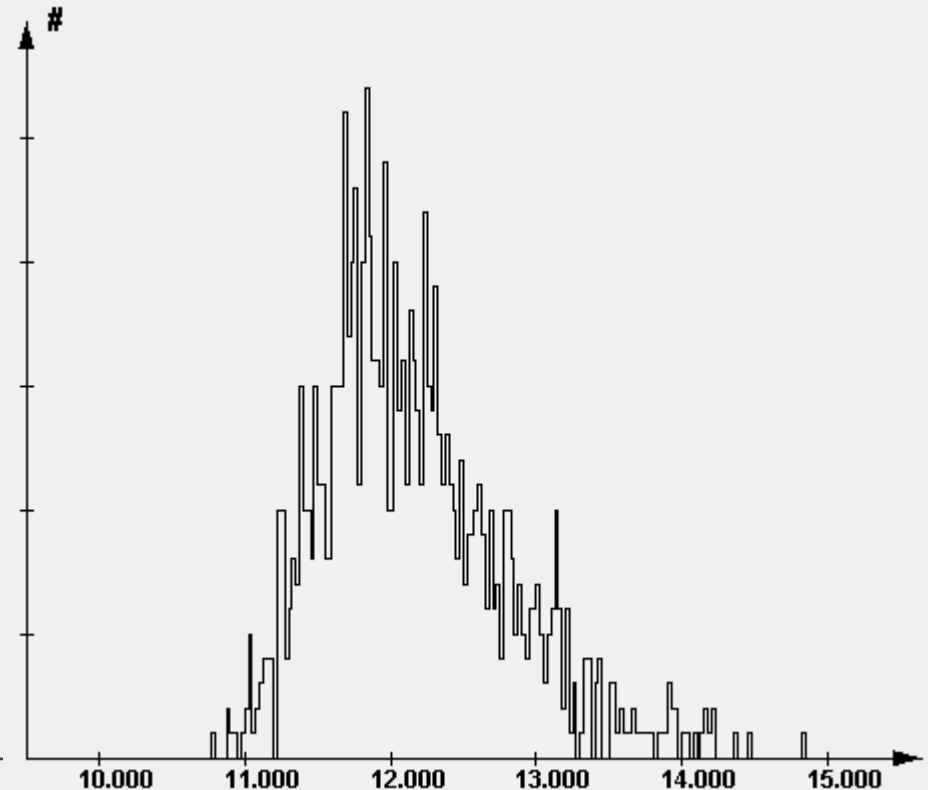
- ◆ Per niveau van opsplitsen: globaal ongeveer n vergelijkingen
- ◆ Aantal opsplitsingen: gemiddeld $\log_2 n$
 - ✦ *Afhankelijk kwaliteit van pivot element!*
- ◆ Performantie = $O(n \cdot \log_2 n)$
 - ✦ *Gemiddeld $1,39 \cdot n \cdot \log_2 n$*
 - ✦ *Is bewezen dat 't niet sneller kan voor een willekeurige array!*
- ◆ Vb: $n=1000$ elementen
 - ✦ $n^2 = 1.000.000 <> n \cdot \log_2 n \approx 10.000$

Altijd $n \cdot \log_2 n$?

Aantal uitwisselingen voor 1000 elementen.



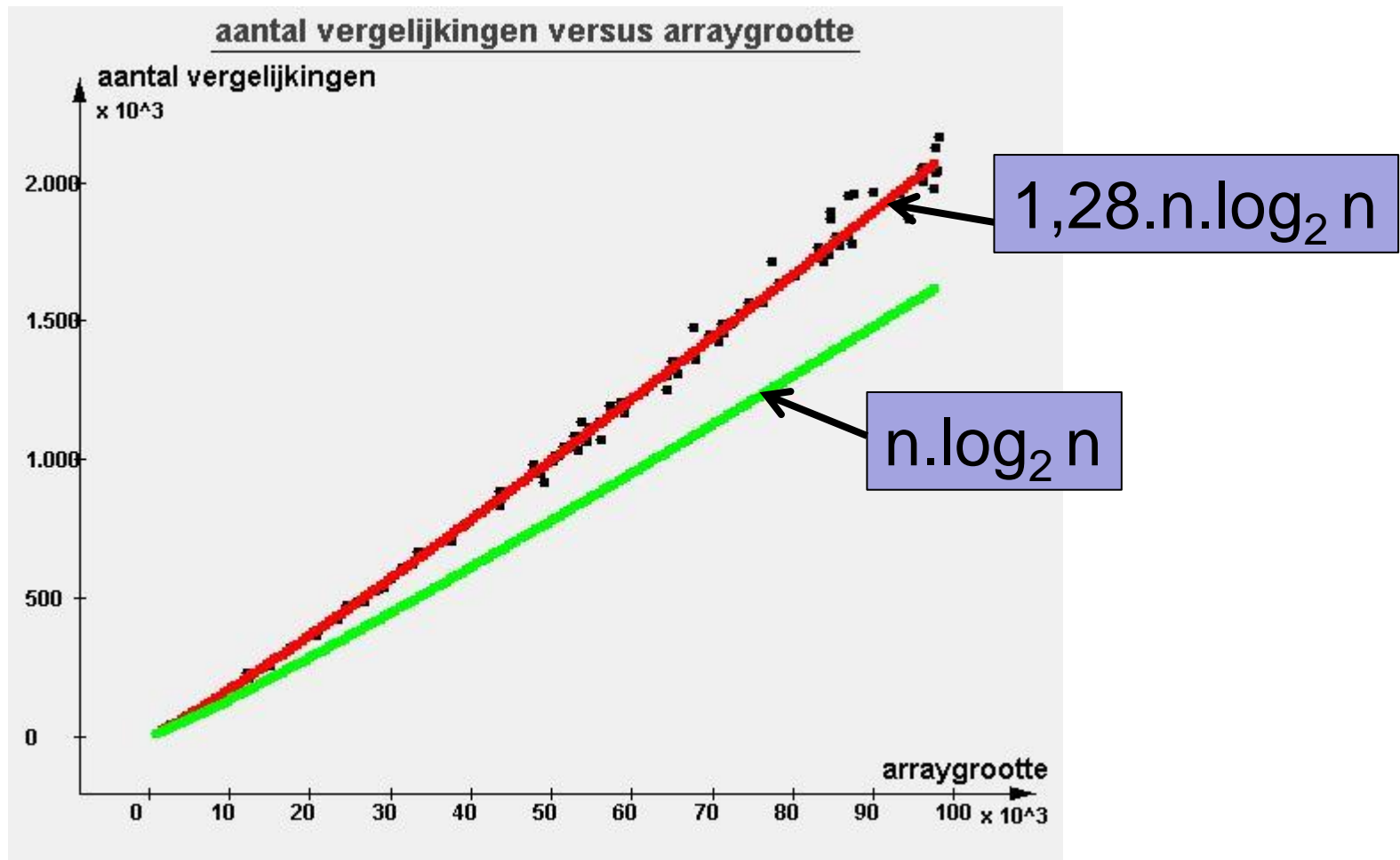
Aantal vergelijkingen voor 1000 elementen



$$n \cdot \log_2 n = 9965$$

$$1,39 \cdot n \cdot \log_2 n = 13851$$

Performantie i.f.v. n



Worst-case arrays?

- ◆ Wanneer slechte pivot?
- ◆ Als pivot grootste of kleinste element is
 - ✦ Gesorteerd in volgorde
 - ✦ Gesorteerd in omgekeerde volgorde

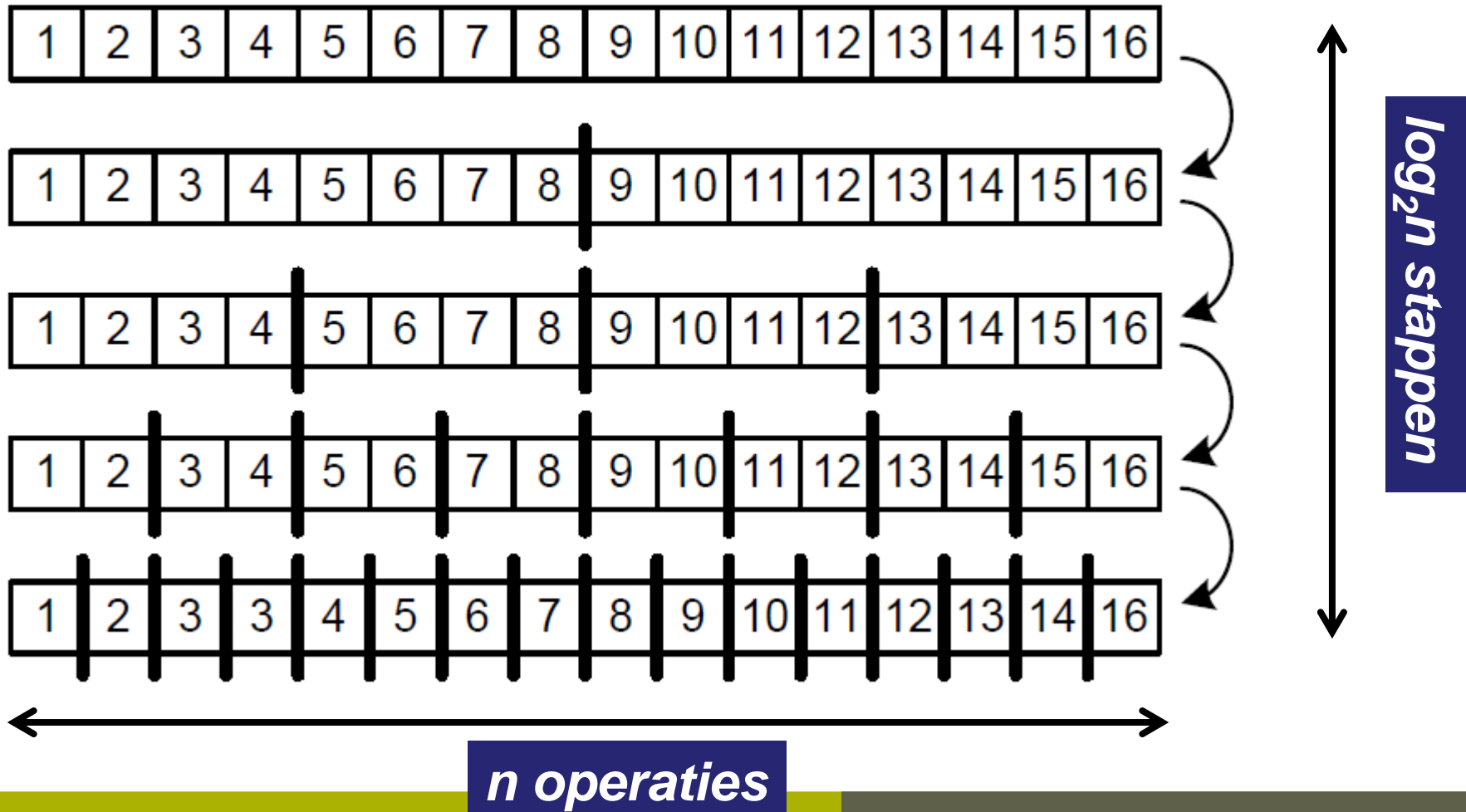
Selection & Bubble Sort: n^2

Idee: zoek kleinste, dan tweede kleinste, enzovoorts

Step	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]	A[10]
0	51	03	24	86	45	30	27	63	96	50	10
1	03	51	24	86	45	30	27	63	96	50	10
2	03	10	24	86	45	30	27	63	96	50	51
3	03	10	24	86	45	30	27	63	96	50	51
4	03	10	24	27	45	30	86	63	96	50	51
5	03	10	24	27	30	45	86	63	96	50	51
6	03	10	24	27	30	45	86	63	96	50	51
7	03	10	24	27	30	45	50	63	96	86	51
8	03	10	24	27	30	45	50	51	96	86	63
9	03	10	24	27	30	45	50	51	63	86	96
10	03	10	24	27	30	45	50	51	63	86	96
11	03	10	24	27	30	45	50	51	63	86	96

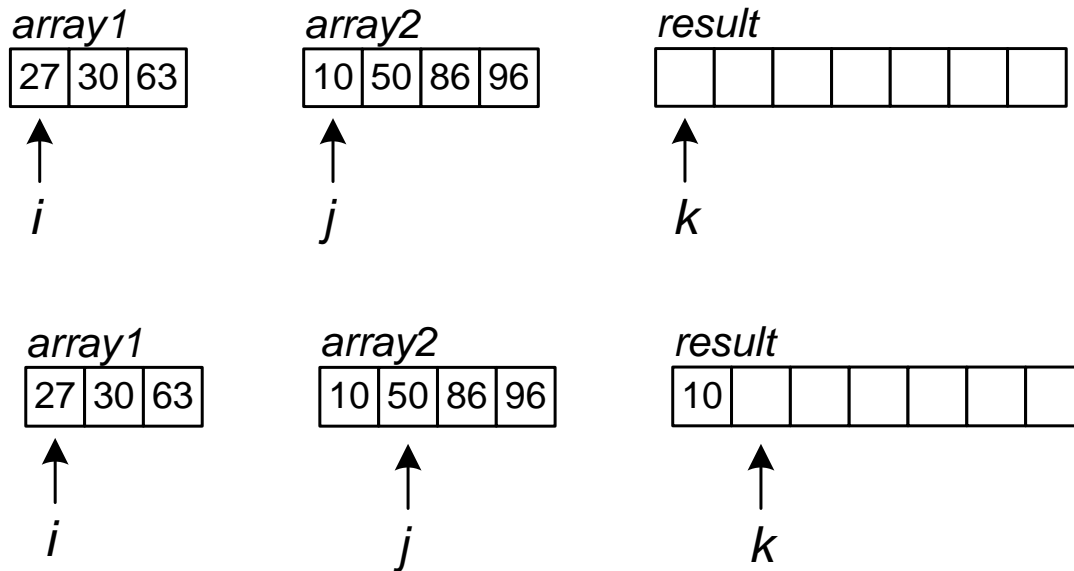
n stappen

Quicksort: $n \cdot \log_2 n$



4. Mergesort

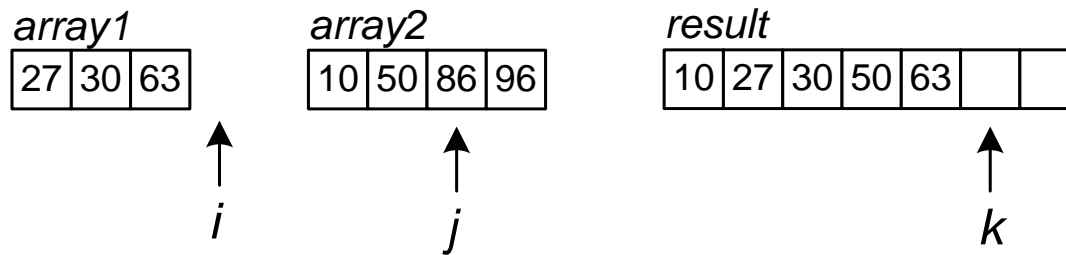
Idee: Voeg twee gesorteerde delen bij elkaar (mergen)



```
private static int[] merge(int[] array1, int[] array2){
    int[] result = new int[array1.length + array2.length];
    int i=0, j=0;// i is om door array1 te lopen, j voor array2
    int k=0; // k is voor de result array
    while(i < array1.length && j < array2.length){
        if (array1[i] < array2[j]){
            result[k] = array1[i];
            i++;
        } else {
            result[k] = array2[j];
            j++;
        }
        k++;
    }
    while(i < array1.length){
        result[k] = array1[i];
        i++;
        k++;
    }
    while(j < array2.length){
        result[k] = array2[j];
        j++;
        k++;
    }
    return result;
}
```

2e deel van *merge*-methode

Na de 1^e while:



Nu enkel *array2* nog kopiëren...

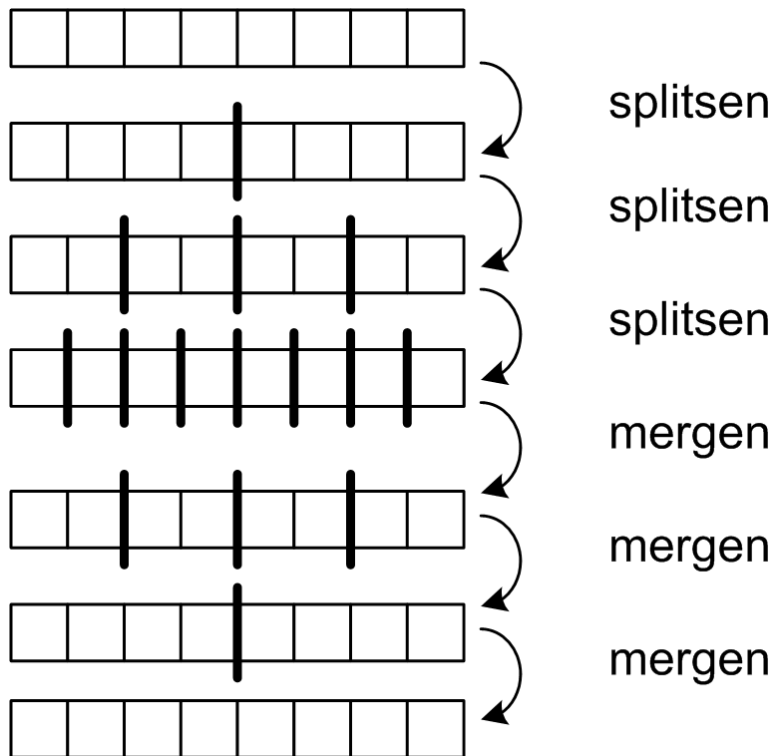
mergesort recursief

```
public static int[] mergeSort(int[] array){
    // deel array op in twee
    int l1 = array.length/2;
    int[] arr1 = Arrays.copyOf(array, l1);
    int[] arr2 = Arrays.copyOfRange(array, l1, array.length);
    // sorteer beide delen
    if (arr1.length > 1)
        arr1 = mergeSort(arr1);
    if (arr2.length > 1)
        arr2 = mergeSort(arr2);

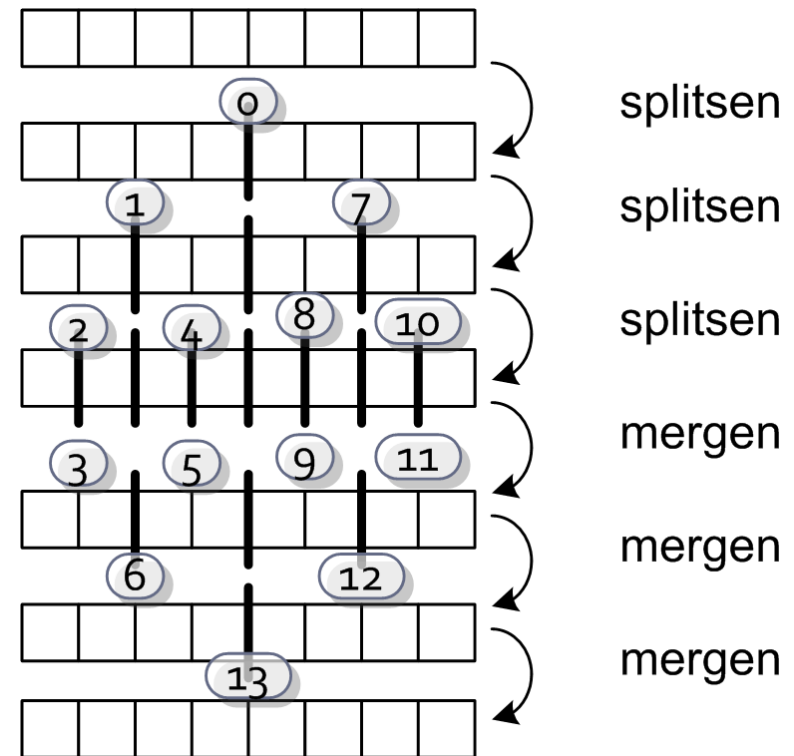
    // nu beide onderdelen gesorteerd zijn, kunnen we ze 'mergen'
    return merge(arr1, arr2);
}
```

Mergesort recursief

Dit is wat er gebeurt:



Opgelet!! Dit is de volgorde van de stappen:



Performantie mergesort

- ◆ Stap 1: mergen arrays van grootte 1
 - ◆ Stap 2: mergen arrays van grootte 2
 - ◆ Stap 3: mergen arrays van grootte 4
 - ◆ ...
 - ◆ Stap $\log_2 n$: mergen arrays van grootte $n/2$
- ◆ Elke stap: n operaties (vergelijkingen en swaps)
- ➔ $n \cdot \log_2 n$ operaties

Dit is gegarandeerd en altijd!!! Geen worst-case.


Is zelfde performantie als quicksort!

Maar: kopies van arrays nodig, niet “in-place” zoals bij quicksort


Natuurlijke mergesort

Stap 1: opsplitsen

Vervolgens 2 delen 'mergen'



0	qw erty u iop as dfghjklz x cv bn m
1	u iop as dfghjklz x cv bn m eqrtwy
2	as dfghjklz x cv bn m eqrtwy iopu
3	x cv bn m eqrtwy iopu adefghjklz
4	bn m eqrtwy iopu adefghjklz cvx
5	eqrtwy iopu adefghjklz cvx bmn
6	adefghjklz cvx bmn eiopqrtuwy
7	bmn eiopqrtuwy acdefghjklsvxz
8	acdefghjklsvxz beimnopqrtuwy
9	abcdefghijklmnopqrstuvwxyz



We gaan alle stukjes bijhouden in een queue en 2 per 2 mergen.

Dit zou je typisch recursief doen, hier zie je hoe dit anders kan, met een queue.

```

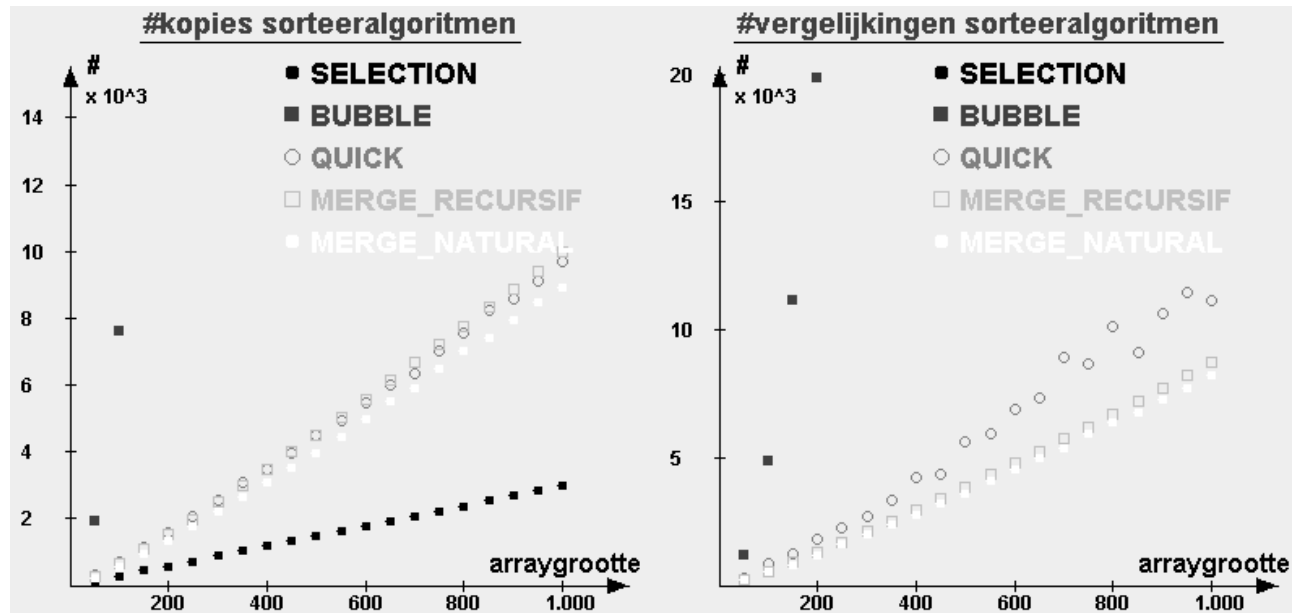
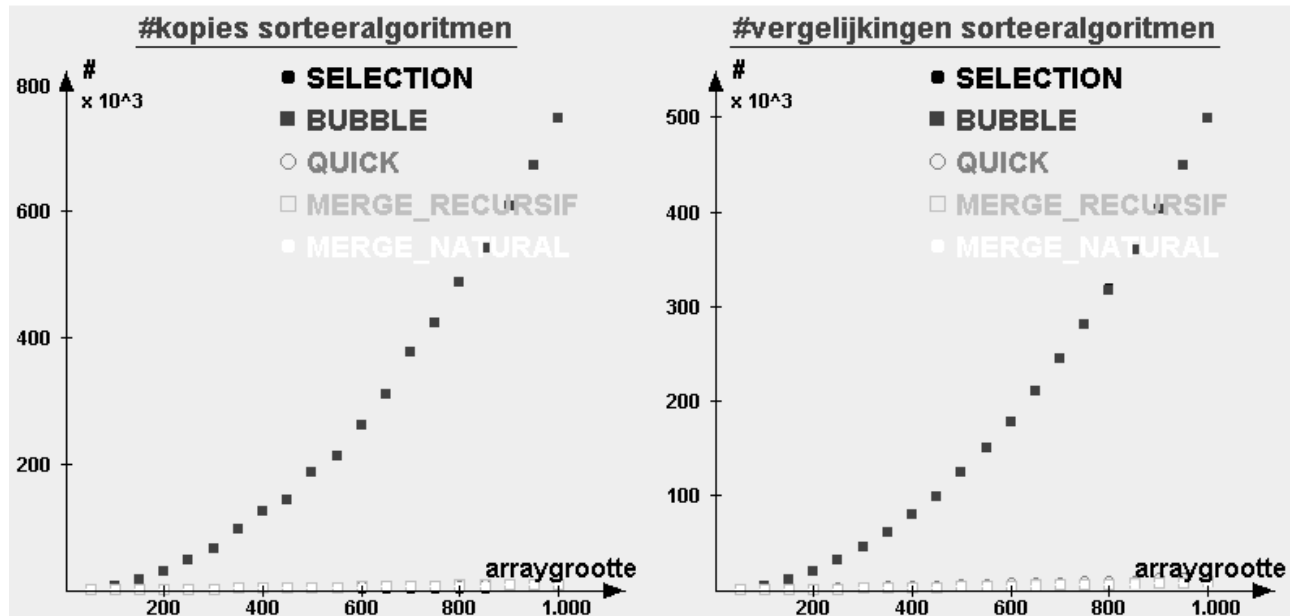
public static int[] naturalMergeSort(int[] array){
    FIFOQueue<int[]> queue = new FIFOQueue<int[]>(array.length);
    // 1. we kappen de originele array eerst in gesorteerde stukjes
    int start=0;
    for(int i=1;i<array.length;i++){
        if (array[i] < array[i-1]){
            queue.add(Arrays.copyOfRange(array, start, i));
            start = i;
        }
    } // het laatste stukje
    queue.add(Arrays.copyOfRange(array, start, array.length));
    // 2. en sorteren we alle stukjes
    aantalVergelijkingen=0;
    aantalKopies = 0;
    while(queue.size() > 1){
        int[] arr1 = queue.get();
        int[] arr2 = queue.get();
        int[] arr3 = merge(arr1, arr2);
        queue.add(arr3);
    }
    return queue.get();
}

```

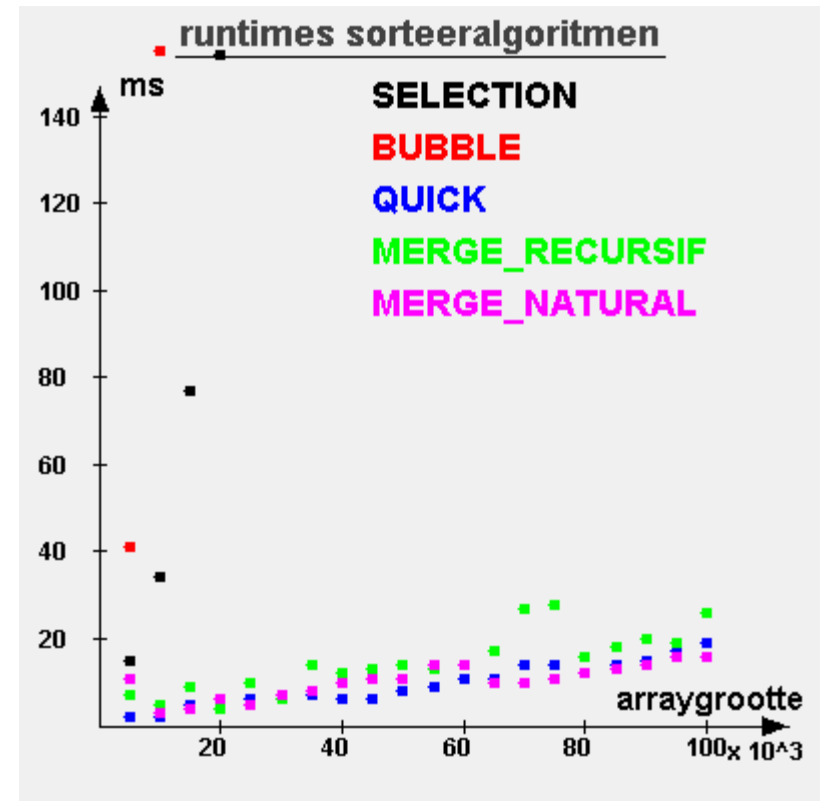
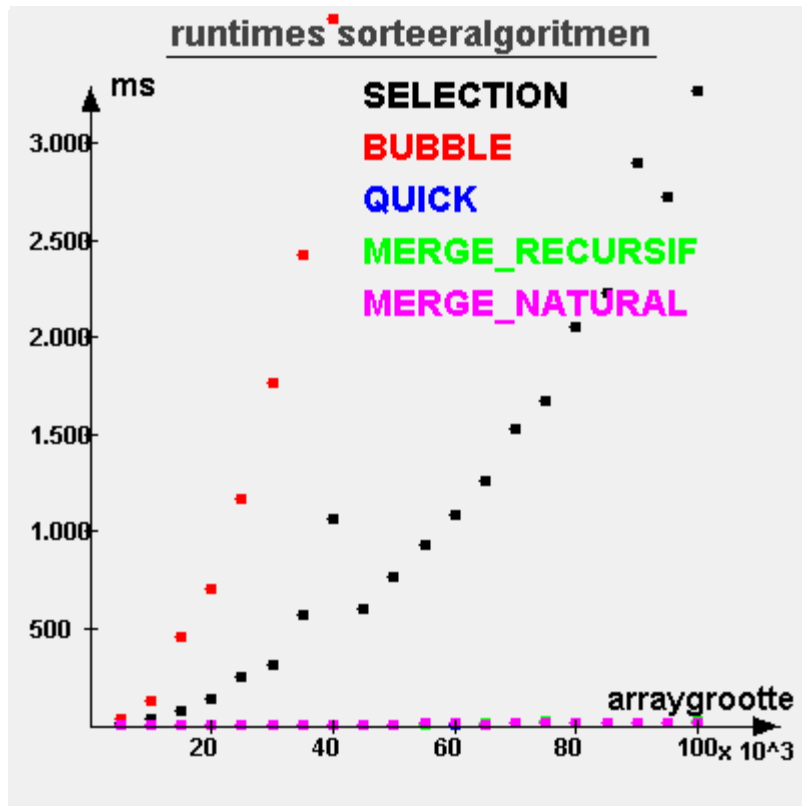
Performantie

- ◆ Selection & Bubble Sort: **n^2**
- ◆ Mergesort & Quicksort: **$n \cdot \log_2 n$**

Aantal basisoperaties



Uitvoeringstijden in milliseconden (ms)



Quicksort en mergesort ongeveer even lang

Generic sorting

Arrays class:

Method Summary

`static void sort(int[] a)`

Sorts the specified array of ints into ascending numerical order.

`static void sort(int[] a, int fromIndex, int toIndex)`

Sorts the specified range of the specified array of ints into ascending numerical order.

`static void sort(Object[] a)`

Sorts the specified array of objects into ascending order, according to the *natural ordering* of its elements.

`static void sort(Object[] a, int fromIndex, int toIndex)`

Sorts the specified range of the specified array of objects into ascending order, according to the *natural ordering* of its elements.

`static void sort(T[] a, Comparator<? super T> c)`

Sorts the specified array of objects according to the order induced by the specified comparator.

`static void sort(T[] a, int fromIndex, int toIndex, Comparator<? super T> c)`

Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.

Voor lijsten

Collections class

```
static void sort(List list)
```

Sorts the specified list into ascending order, according to the *natural ordering* of its elements.

```
static void sort(List list, Comparator c)
```

Sorts the specified list according to the order induced by the specified comparator.

1) “Natuurlijke” ordening

Interface Comparable

Method Summary

`int compareTo(Object o)`

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Voorbeeld

```
public class Student implements Comparable<Student>{
    String voornaam, naam;
    int rolnummer, score;
    Vak[] vakken;
    int[] punten;

    Student(String voornaam, String naam, int rolnummer){
        this.voornaam=voornaam;
        this.naam=naam;
        this.rolnummer=rolnummer;
        vakken = new Vak[4];
    }
    @Override
    public int compareTo(Student student2) {
        int ordeNaam = this.naam.compareTo(student2.naam);
        if (ordeNaam == 0) // beide dezelfde naam
            return this.voornaam.compareTo(student2.voornaam);
        else
            return ordeNaam;
    }
}
```

2) Specifieke ordening

java.util

Interface Comparator<T>

Method Summary

int compare(T o1, T o2)

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

boolean

equals(Object obj)

Indicates whether some other object is "equal to" this comparator.

Voorbeeld

```
class ComparatorOnScore implements Comparator<Student>{  
    @Override  
    public int compare(Student student1, Student student2) {  
        return student1.score - student2.score;  
    }  
}
```

Sorteren gebeurt nu als volgt, met eigen comparator-object:

```
Arrays.sort(studenten, new ComparatorOnScore());
```



Hoofdstuk 8: Internet - geschiedenis

Geschiedenis: ARPANET

- ◆ TCP/IP ontstond uit Arpanet, ontwikkeld door Amerikaans leger
- ◆ Eisen aan digitaal communicatiesysteem:
 - ✦ **Flexibel**
 - ✦ **Gedecentraliseerd**, geen 'baas'
 - ✦ **Onafhankelijkheid**: delen kunnen op zich werken
 - ✦ **Zelf-regulerend**
 - ➔ **robustness and survivability**
 - “including the capability to withstand losses of large portions of the underlying networks (due to a nuclear attack)”
 - pakketjes kunnen verloren gaan: fouten of 'overlopen' van de queues bij bottlenecks

Het internet werkt compleet anders dan een traditioneel telefonienetwerk. Bij telefonie zijn er centrales die een exclusieve connectie reserveren tussen de 3 belpunten. Internet is gedecentraliseerd, gegevens worden in pakketjes opgedeeld die elk hun eigen route mogen volgen over gedeelde lijnen.

Microsoft vòòr 1995

- ◆ Microsoft: door Windows, grootste speler in IT
- ◆ Maar is niet geïnteresseerd in internet (ik beroepsmatig ook niet)
- ◆ Op CERN wordt Netscape Navigator ontwikkeld, de *eerste browser*
 - ✦ Om informatie (file) van een remote computer te visualiseren
- ◆ Netscape wordt de grootste speler
 - ✦ Is nu Firefox/Seamonkey

De volgende revolutie werd...

INTERNET!

Microsoft reageert

◆ Lanceert Internet Explorer in 1995

- ✦ Gratis bij Windows waardoor iedereen het begint te gebruiken

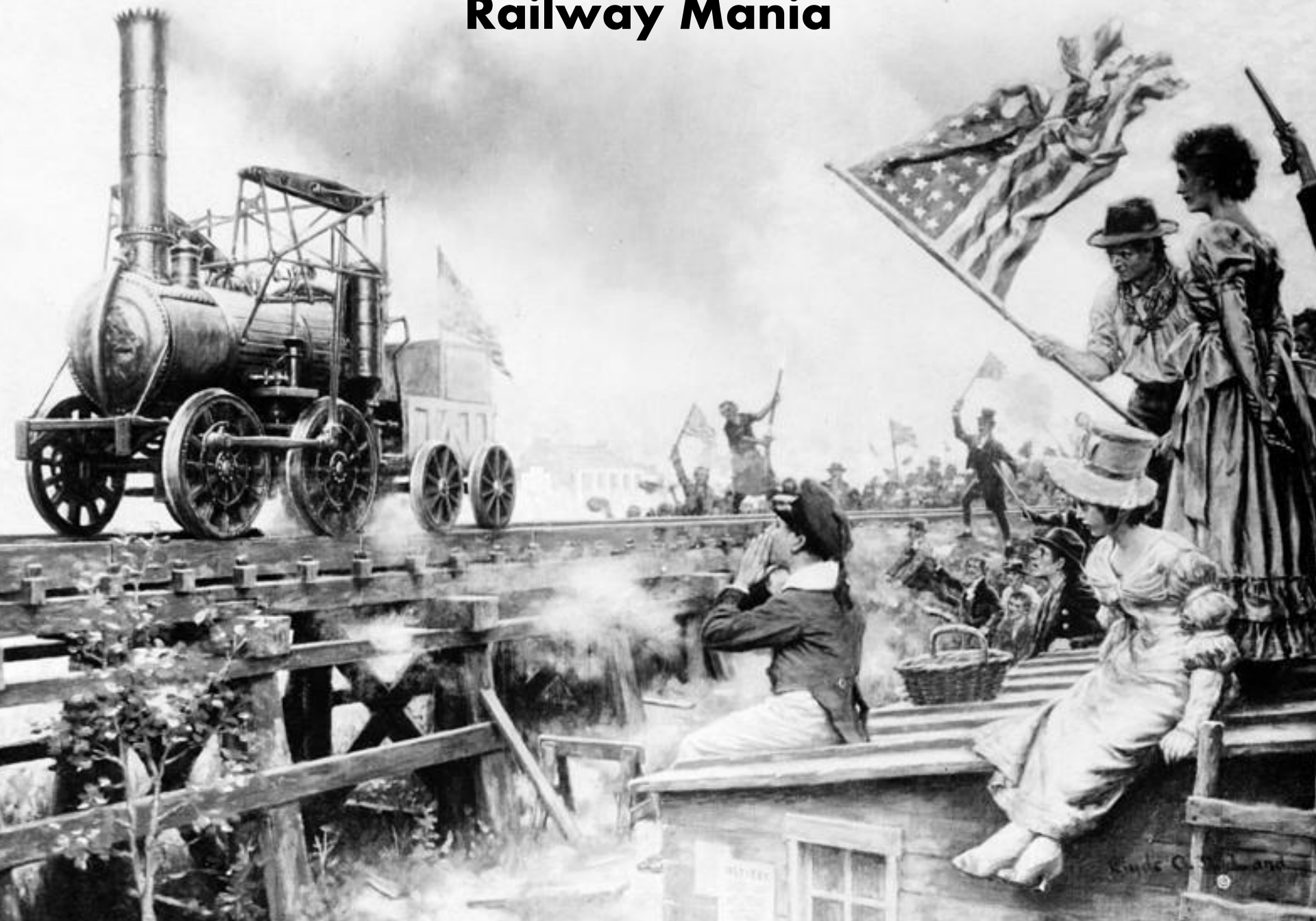
◆ Koopt hotmail op

- ✦ En onlangs nog Skype en Nokia (maar dominante positie lijkt nu definitief verloren)

◆ Veroveret het internet

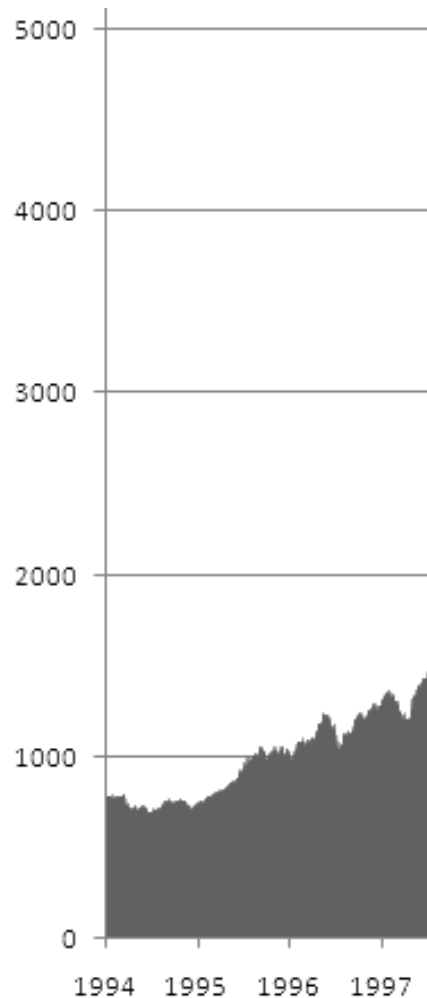
- ✦ Door macht van Windows

Railway Mania



De internetrevolutie

Technologie-index van USA: Nasdaq



NASDAQ Composite

INDEXNASDAQ: .IXIC

8.809,12 +98,41 (1,13%) ↑

5 mei 17:15 EDT · Disclaimer



2000 tot nu: + 77%

De internetbubbel

Of internetzeepbel of dotcom-crisis

- ◆ Nieuwe technologie opent nieuwe mogelijkheden, creëert (te) hoge verwachtingen

➔ **Investeren geblazen!**

- ◆ Professionals/leken steken hun geld in aandelenfondsen en aandelenclubs om de boot niet te missen
- ◆ Bubbel \leq Hebzucht!

IT-sector: the winner takes it all

- ◆ Windows, office, TCP/IP-protocol, pdf, Google, facebook, twitter, ...
- ↔ in andere sectoren kunnen er meerdere spelers zijn (bvb meerdere automerken, wasmachines, ...)
- ◆ Kan leiden tot monopolieposities en oneerlijke concurrentie (verstoorde marktwerking).
 - ◆ Microsoft kreeg verscheidene veroordelingen van de Europese Commissie
- ◆ *Alternatief*: standaards waar iedereen zich aan houdt

De IT-sector heeft zo zijn specifieke economische wetten. Eentje is dat het meestal 1 speler is die de markt veroverd doordat de consument er alle baat bij heeft dat iedereen hetzelfde product gebruikt, wegens gemak van interoperabiliteit en uniformiteit. Dit wordt ook bereikt door standaards.

De economie zou totaal veranderen => 'dotcom'-economie

- ◆ Internet: ongekennde commerciële mogelijkheden
 - ✦ *Anders communiceren*
 - ✦ *Anders kopen*
 - Ook kerstbomen kopen op het internet...
- ◆ Belangrijkste: =**aandacht** (hits/leden/...)
 - ✦ *"Get large or get lost"*
 - ✦ agressieve marktpenetratie door middel van het uitbouwen van netwerken.
- ◆ Extreme verliezen in het begin werden gezien als slechts investerings. Winst/omzet maken zou later komen.

Velen geloofden dat we totaal anders zouden gaan consumeren. Je moest dus wel investeren in het internet, of je zou er binnen de kortste keren uit liggen.

30.14 +0.10(0.35%) 3:26PM EST - Nasdaq Real Time Price

Enter name(s) or symbol(s)

GET CHART

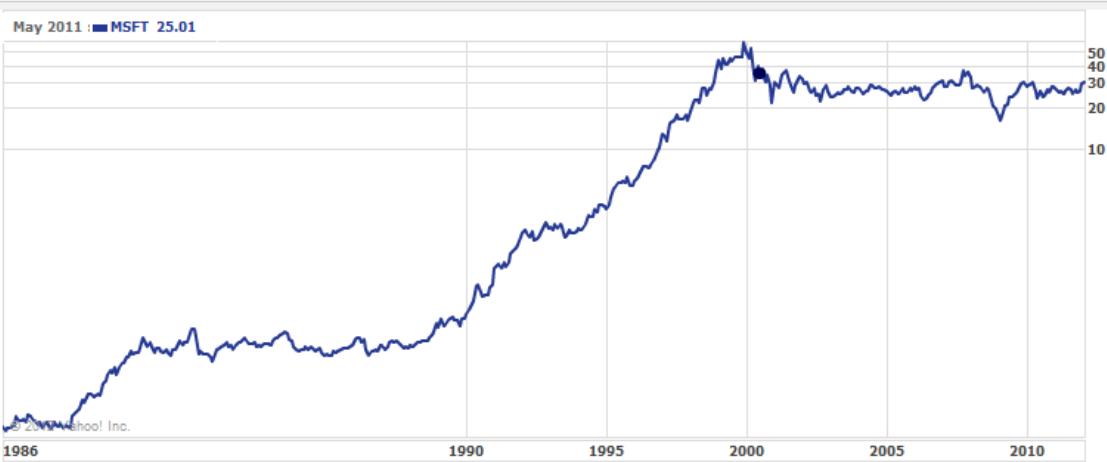
COMPARE

EVENTS

TECHNICAL INDICATORS

CHART SETTINGS

RESET



15.80 +0.12(0.75%) 3:18PM EST - Nasdaq Real Time Price

Enter name(s) or symbol(s)

GET CHART

COMPARE

EVENTS

TECHNICAL INDICATORS

CHART SETTINGS

RESET



Na het springen van de bubble (2000)

- ◆ Veel geld verloren
 - ✦ Investeerders en ook de 'gewone man', enkel de slimmeriken zullen er (net) op tijd uit gestapt zijn
- ◆ De droom spatte echter uit elkaar...
- ➔ Niemand wilt/durft meer te investeren in IT-technologie
- ◆ Tot... **Google** komt (2003-2004) en toont aan dat je wèl geld kunt verdienen op het Internet
 - ✦ Zie Nasdaq-index: begint weer te klimmen

Moraal van het verhaal: technologie heeft tijd nodig om te 'rijpen'.



Hoofdstuk 8: Internet - technologie

Technologie 1: netwerk

◆ Lokaal netwerk:

electrische kabel

◆ Glasvezel verbindt

lokale netwerken

➡ informatie via licht

◆ The world's cable map:

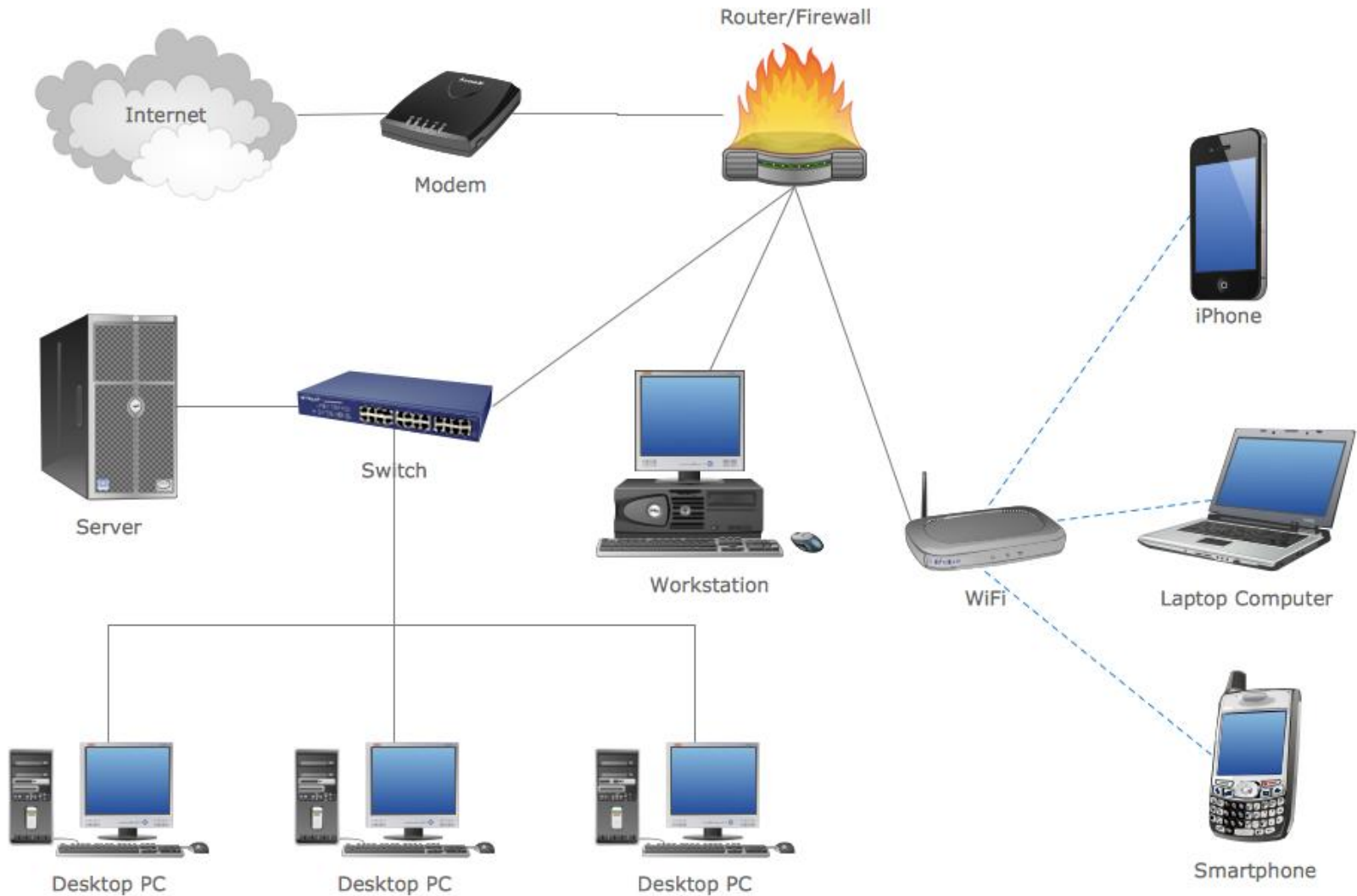
✦ <https://beta.infrapedia.com/app>
(<http://www.cablemap.info/>)



Technologie 2: componenten

- ◆ **Netwerkkkaart:** toegang van je PC/laptop tot het internet, via een netwerkkabel of wireless (wifi)
- ◆ **Switch:** netwerkknooppunt, gebruik je om meerdere connecties te verbinden
- ◆ **Router:** ook een netwerkknooppunt, maar bepaalt ook de route van het pakketje. Het is de toegang van een lokaal network tot het internet en het vormt de knooppunten op het internet
- ◆ **Modem:** maakt informatiesignalen geschikt om over een medium te worden getransporteerd, bvb wireless, telefoon- (Belgacom) of kabelnetwerk (Telenet).

Lokaal network (LAN)

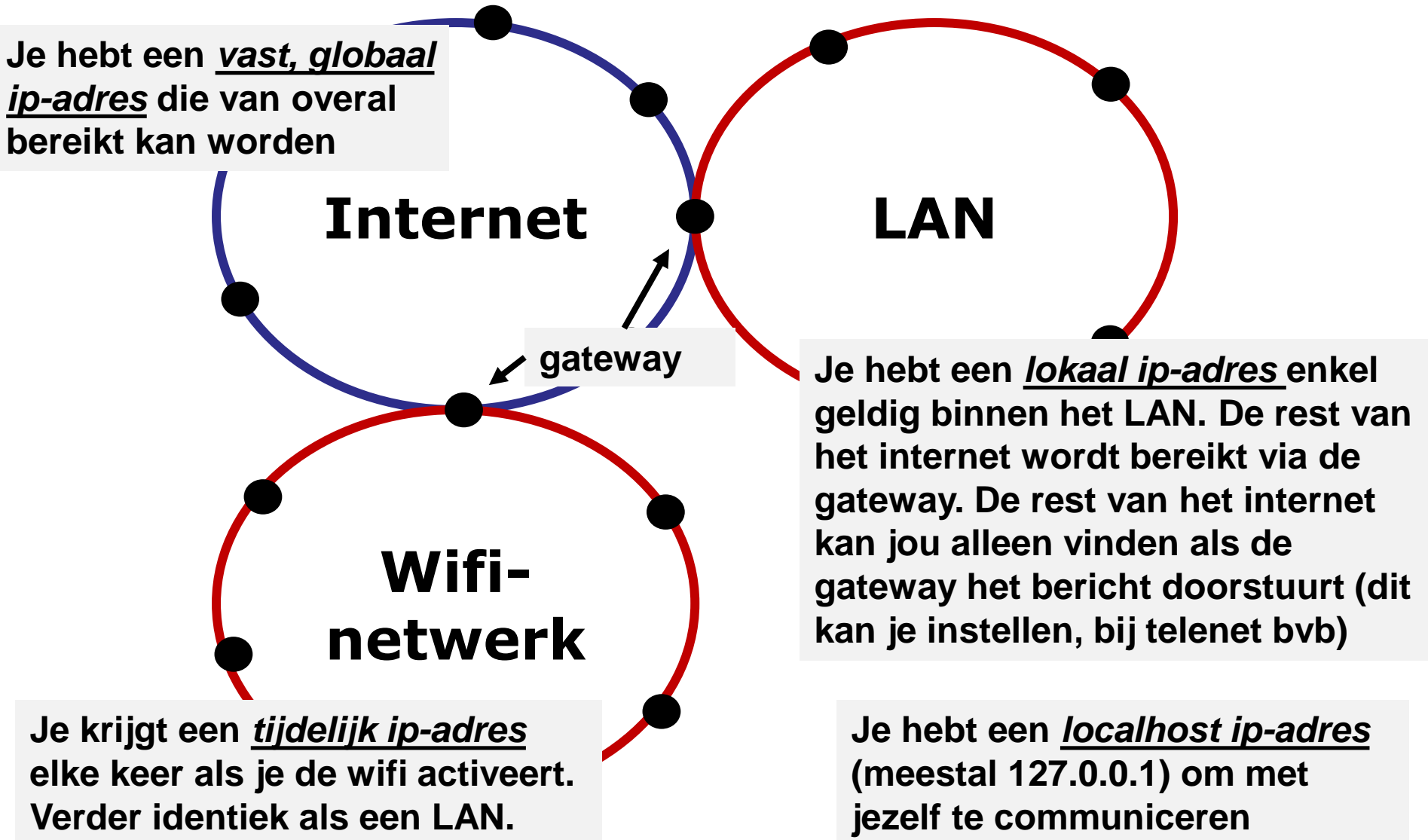


Technologie 3: protocol

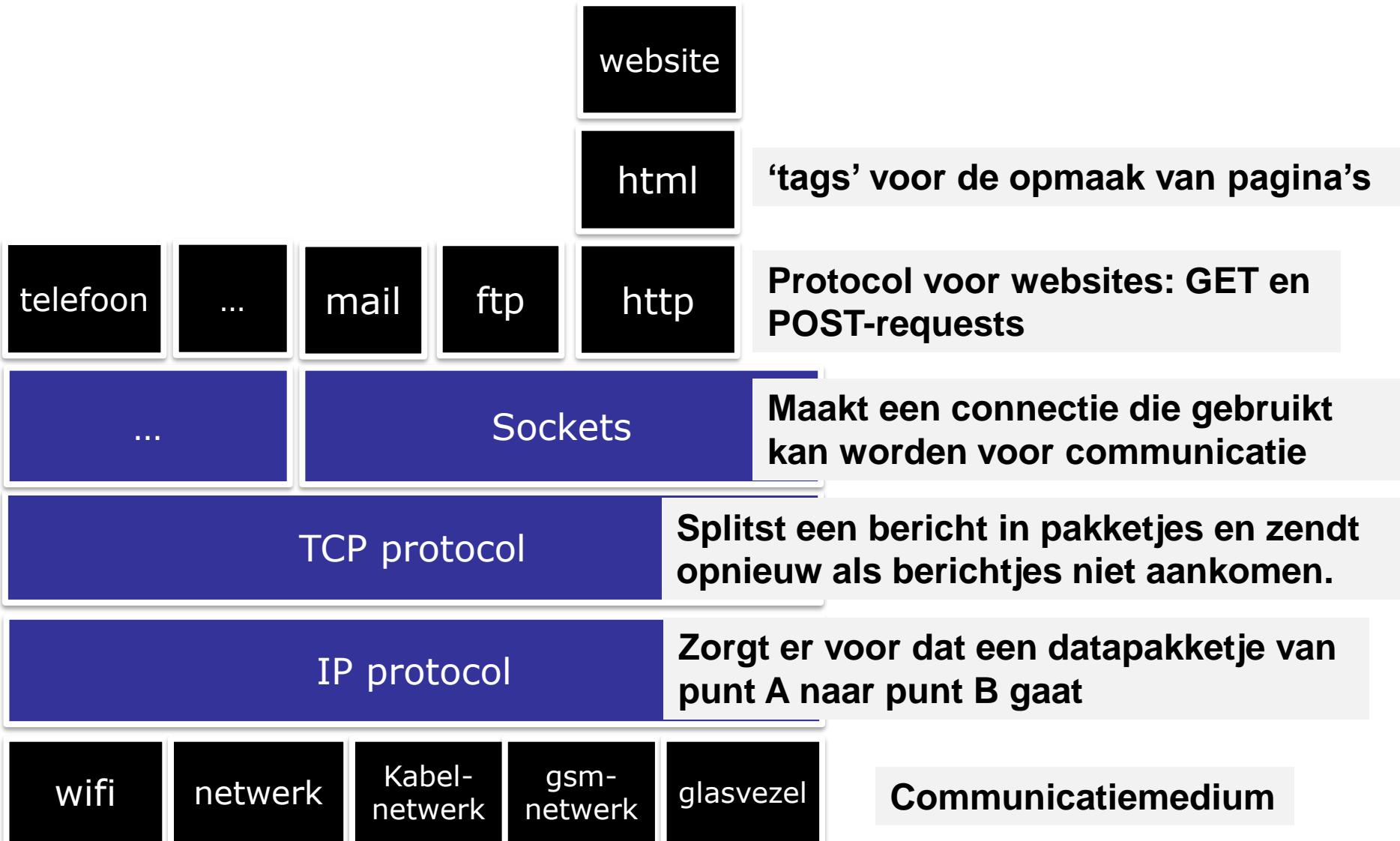
- ◆ Protocol = afgesproken formeel communicatiewijze
- ◆ IP= Internet Protocol
 - ✦ Adres van elke computer: IP-adres
 - Bvb 192.168.0.233 in het nieuwe IPv6 is het een langer getal
 - Windows-commando *ipconfig* (start command prompt met *Windows-teken* + *cmd*) of onder *Performance* -> *Wifi* in de Task Manager
 - Domeinnaam is een *alias* voor het nummer ('naam')
omzetting kan je doen via <http://www.hcidata.info/host2ip.cgi>
- ◆ Bericht wordt opgedeeld in IP-pakketjes die hun weg naar de bestemming zoeken over het net
 - ↔ Oud telefoonnetwerk: je had de hele lijn voor je gereserveerd
 - ✦ Het zoeken van de weg: TCP-protocol (Transmission Control Protocol)

➡ **TCP/IP-protocol**

Local Area Network (LAN) & Wifi



Technologie: softwarecomponenten



Internetconnecties via sockets

Client - server

- ◆ **Client** tracht een connectie te maken met een server via zijn ip-adres en 'poort'
- ◆ **Server** luistert op een 'poort' naar binnenkomende connecties
 - ✦ Een poort is een natuurlijk getal
 - ✦ Elke applicatie gebruikt een eigen poort
 - Websites (http): 80
 - File transfer (ftp): 20
 - http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
 - ✦ Zo kan de communicatie van de verschillende applicaties uit elkaar gehouden worden.

Zie programma **MyClient**
en **MyServer** in package
internet

Browser: HTTP-protocol

Te bekijken met browser-add-on **HTTP Header Live**

- ◆ Website opvragen gebeurt via een GET- of POST – request
 - ✦ Je stuurt www-adres
 - ✦ POST is voor het meezenden van gegevens
- ◆ Antwoord: OK/found met inhoud die getoond wordt of foutmelding (vb page not found)
- ◆ De basis is een html-pagina die als file op server staat
 - ✦ Figuren staan apart als file op server en worden 1-voor-1 opgestuurd
 - ✦ 'Webserver.java' is voorbeeldcode van een eenvoudige webserver die html-files verstuurt naar de browser
- ◆ Eenvoudig websites maken: zie link op parallel

Een html-webpagina

Op te vragen via rechtermuisknop
-> View Page Source

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=
<title>Webpaginavoorbeeld</title>
</head>
<body>
<div align="center"><big><big><b>De titel</b></big></big><b>
</div>
<a href="http://parallel.vub.ac.be/education/java/theorie.html">een
link</a><br>
een lijst:<br>
<ul>
<li>item 1</li>
<li>item 2</li>
</ul>
een foto:<br>
<br>
<br>
<hr size="2" width="100%"><br>
</body>
</html>
```

← TAG

← BEGIN VAN WAT GETOOND WORDT

← FIGUREN STAAN APART

← EINDE VAN TAG

De titel

[een link](#)

een lijst:

- item 1
- item 2

een foto:



Test het uit

- ◆ Zoek het ip-adres op van enkele websites (domeinnaam)
 - ✦ <http://www.hcidata.info/host2ip.cgi>
- ◆ Zoek je eigen ip-adres op
 - ✦ Windows-commando *ipconfig* (Windows-teken en dan 'cmd' typen) of onder *Performance -> Wifi* in de *Task Manager* (via Windows-teken + x)
 - ✦ Gebruik het javaprogramma 'MyIpAddress'
- ◆ Test de connectie met een ander ip-adres
 - ✦ Commando (Windows-teken + cmd): **ping <ip-adres>**
- ◆ Maak een connectie via sockets
 - ✦ Start een server met Javaprogramma 'MyServer' (poort 6667) of MyPingPongServer (poort 6657 – verander NETWORK_TYPE)
 - Gebruik NETWORK_TYPE = LOCALHOST om op je eigen computer te testen
 - ✦ Maak een clientconnectie met de server met 'MyClient':
 - Specifieer ip-adres op lijn 20: `String IP_ADDRESS = "192.168.1.100";`
 - Als server op eigen computer staat: `IP_ADDRESS = "127.0.0.1"`
 - ✦ Maak een connectie met PingPongServer met 'MyClient': zet poort op 6657 en geef ip-adres

MyClient en MyServer staan in package internet