

Vortrag am 11.06.2008

Fach: Informatik



# Sortieralgorithmen: Heapsort

Ein Vortrag von Florian Sachs und Josephin Rodenstein  
Werner-von-Siemens-Gymnasium Magdeburg

# Gliederung



- 1 Einleitung
- 2 Die Datenstruktur „Heap“
- 3 Überführung in Max-Heap
- 4 Das Sortieren
- 5 Varianten von Heapsort
- 6 Quellen

# 1 Einleitung

- 1964 entwickelt von Robert W. Floyd und J. W. J. Williams
- Verbesserung von Selectionsort
- einer der besten vergleichsbasierten Algorithmen
- Komplexität:  $O(n \log(n))$
- 2 Phasen:
  1. Erstellung des Heap
  2. Verarbeitung des Heap
- In-place → spart Speicherplatz
- Nicht stabil

# 1 Einleitung

## Stabiles Sortierverfahren nach Zahlen:

1 Anton		1 Anton
4 Karl		1 Paul
3 Otto		3 Otto
5 Bernd	=>	3 Herbert
3 Herbert		4 Karl
8 Alfred		5 Bernd
1 Paul		8 Alfred

Bewahrt ursprüngliche  
Reihenfolge der  
Datensätze deren  
Sortierschlüssel gleich ist

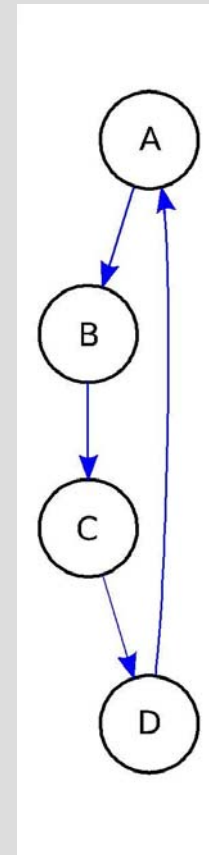
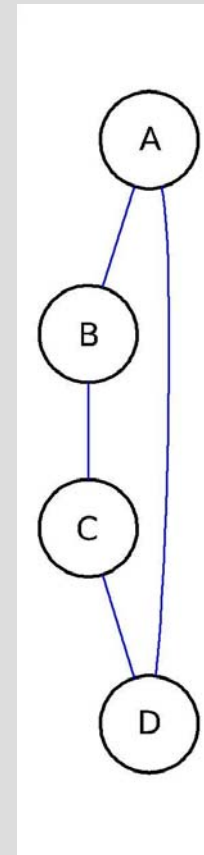
## Instabiles Sortierverfahren nach Zahlen:

1 Anton		1 Paul		1 Anton
4 Karl		1 Anton		1 Paul
3 Otto		3 Otto		3 Otto
5 Bernd	=>	3 Herbert	oder	3 Otto
3 Herbert		4 Karl		4 Karl
8 Alfred		5 Bernd		5 Bernd
1 Paul		8 Alfred		8 Alfred

# 2 Die Datenstruktur „Heap“



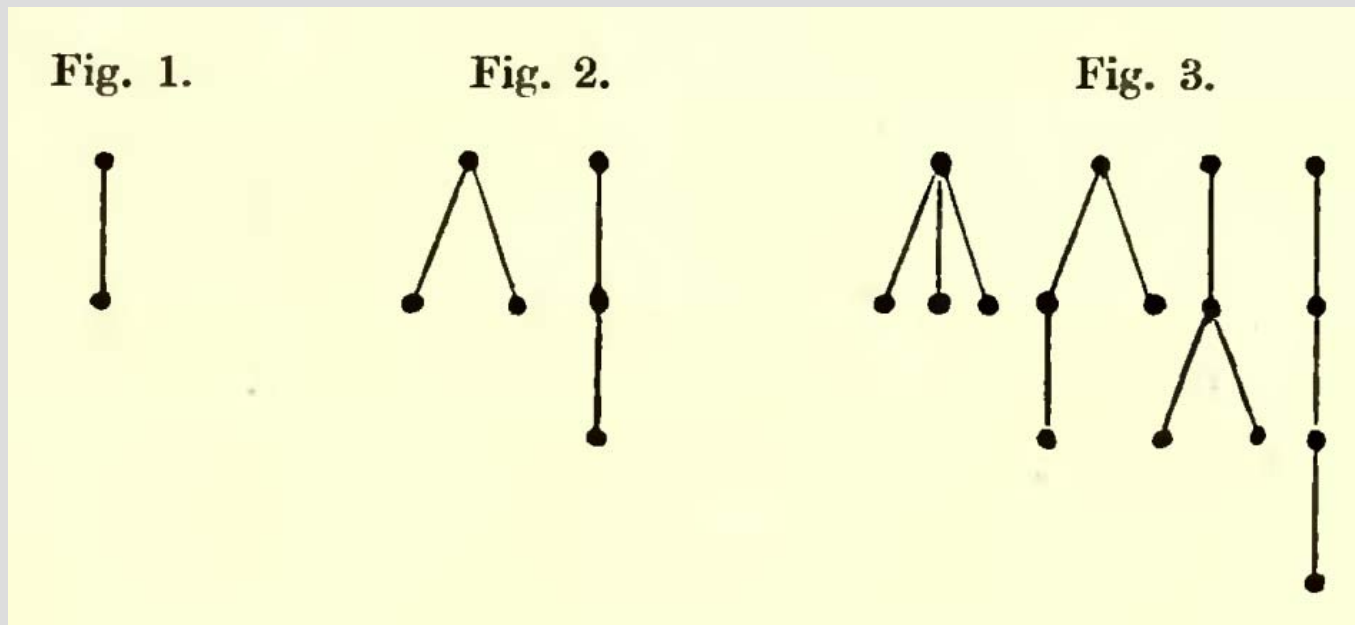
- Graph: Menge von Punkten, zwischen denen Linien verlaufen
- Punkte: Knoten
- Linien: Kanten



# 2 Die Datenstruktur „Heap“



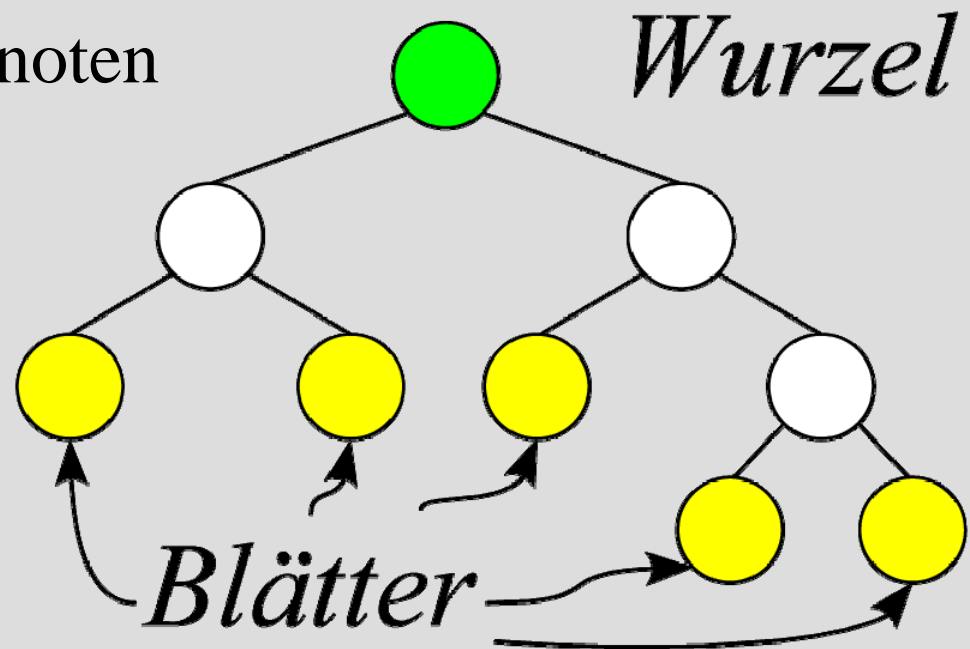
- Baum: spezieller Graph  $\rightarrow$  Hierarchie
- Ungerichtete oder gewurzelte Bäume



# 2 Die Datenstruktur „Heap“



- Binärbaum: gewurzelter Baum  
(Baum mit gerichteten Kanten)
- Hat nur zwei Kindknoten



# 2 Die Datenstruktur „Heap“



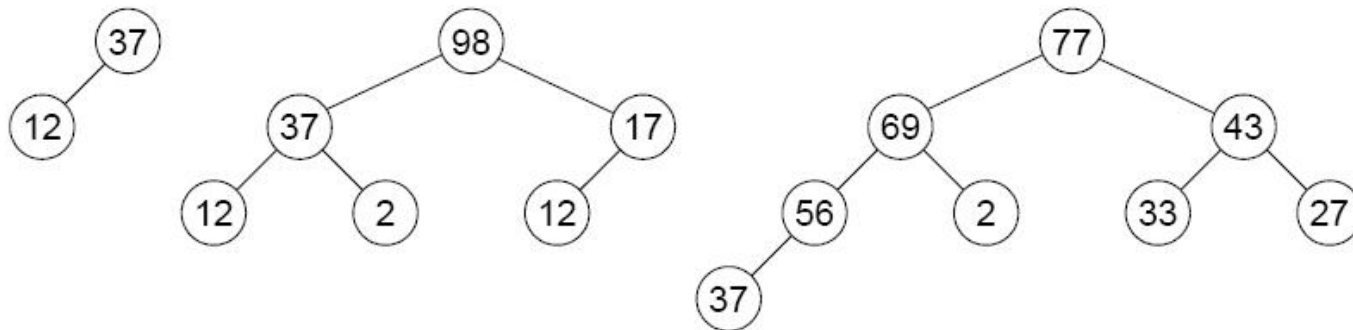
- Wörtlich: „Haufen“
- Meist auf Bäumen basierende Datenstruktur
- Verschiedene Varianten
- Max-Heap für uns wichtig
  - Heapbedingung:  
Elternknoten größer als Kinderknoten



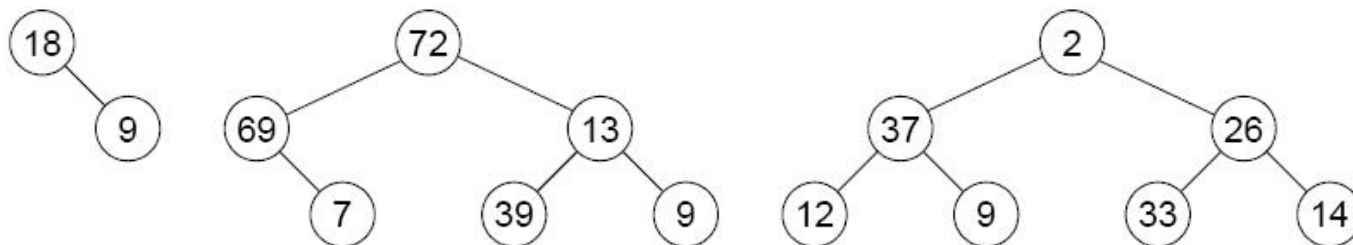
# 2 Die Datenstruktur „Heap“



Heap-Strukturen



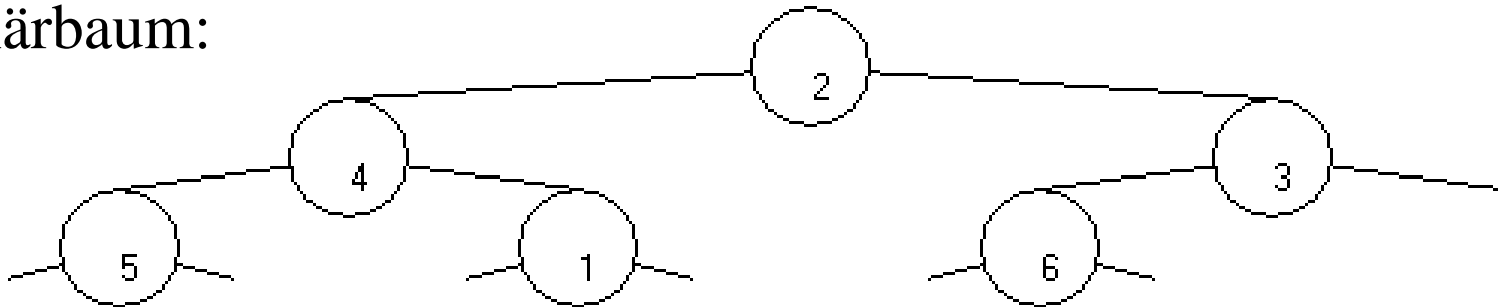
keine Heap-Strukturen



# 3 Überführung in Max-Heap



Binärbaum:



Array:

Index	1	2	3	4	5	6
Element	2	4	3	5	1	6

# 3 Überführung in Max-Heap

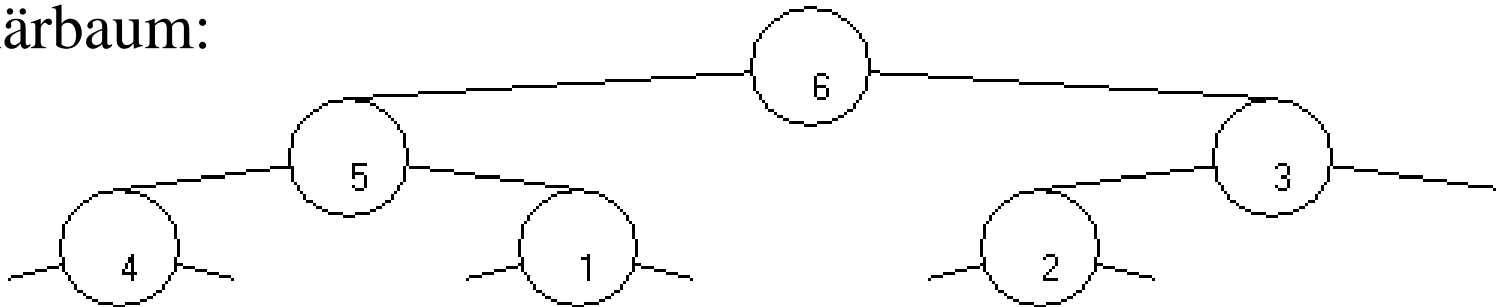


- Zweite Hälfte des Array ist Blatt
  - keine Kinder
  - beginn bei Mitte
- Für Arbeiten mit Array:
  - Elternknoten:  $i$
  - Linkes Kind:  $2i$
  - Rechtes Kind:  $2i + 1$

# 3 Überführung in Max-Heap



Binärbaum:



Array:

Index	1	2	3	4	5	6
Element	6	5	3	4	1	2

# 3 Überführung in Max-Heap

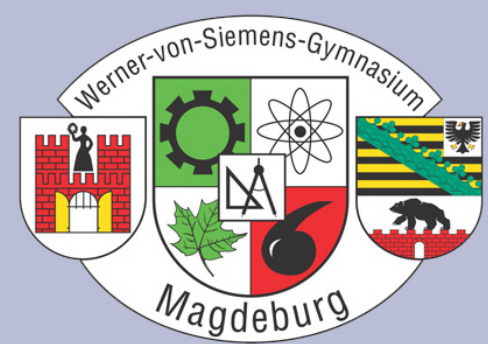


```
Var Input: array of Char;  
  
Size:=Length(Edit1.Text);  
for i:=1 to Size do Input[i]:=Edit1.Text[i];  
for i:=(Size div 2) downto 1 do SiftDown(Size, i);  
  
procedure TForm1.SiftDown(Size: Cardinal; j: Cardinal);  
var Parent, Left, Right, Buffer: Char;  
    Largest: Cardinal;  
begin  
    while (j<=(Size div 2)) do  
        begin  
            Parent:=Input[j];  
            Left:=Input[2*j];  
            if (2*j+1)<=Size then Right:=Input[2*j+1]  
            else Right:=Chr(0);
```

# 3 Überführung in Max-Heap



```
if(Left>Parent) then Largest:=2*j
  else Largest:=j;
if(Right>Parent) and (Right>Left) then Largest:=2*j+1;
if Largest<>j then
  begin
    Buffer:=Input[Largest];
    Input[Largest]:=Input[j];
    Input[j]:=Buffer;
    j:=Largest;
  end
  else Exit;
end;
end;
```

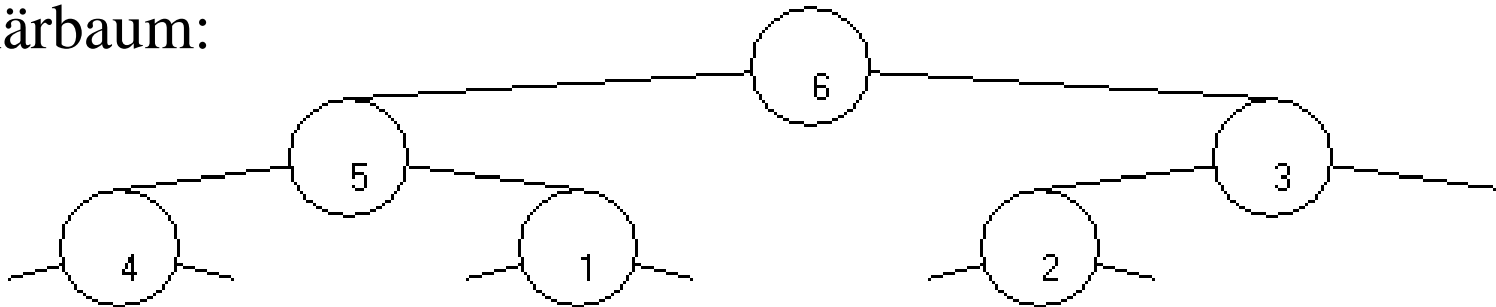


# 4 Das Sortieren

- Max-Heap  $\rightarrow$  Wurzel ist größtes Element
- Vertauschung erstes und letztes Element
- Letztes Element nicht mehr betrachten
- Max-Heap bilden

# 4 Das Sortieren

Binärbaum:



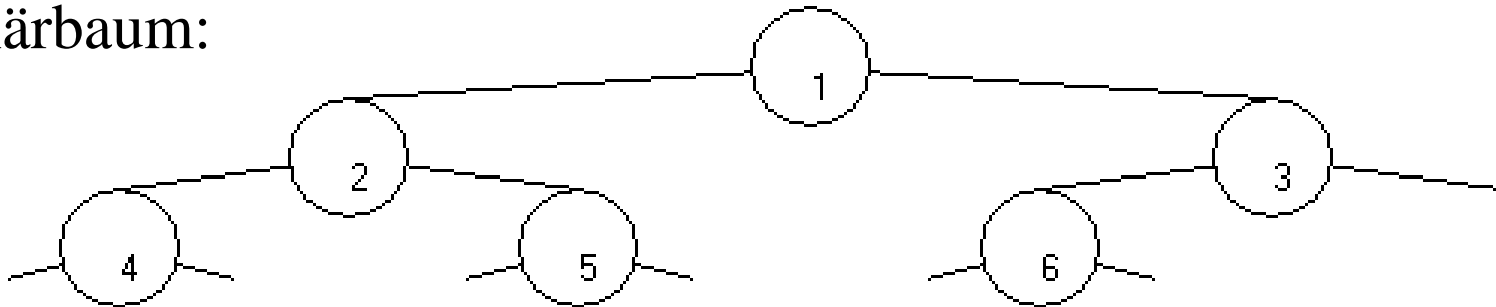
Array:

Index	1	2	3	4	5	6
Element	6	5	3	4	1	2



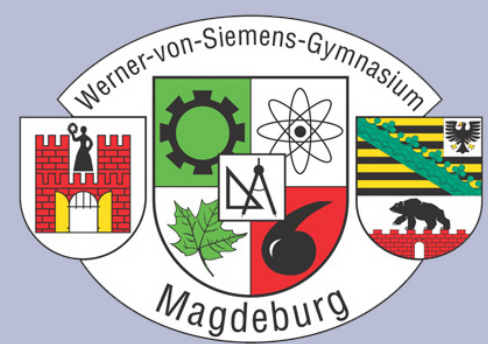
# 4 Das Sortieren

Binärbaum:



Array:

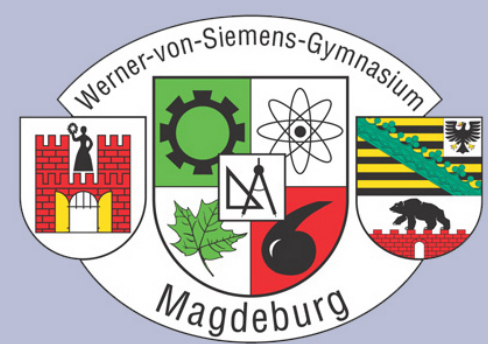
Index	1	2	3	4	5	6
Element	1	2	3	4	5	6



# 4 Das Sortieren

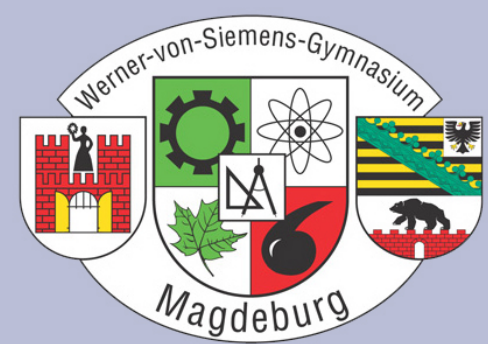
```
procedure TForm1.SetMax(Size: Cardinal);  
var Buffer: Char;  
begin  
    Buffer:=Input[Size];  
    Input[Size]:=Input[1];  
    Input[1]:=Buffer;  
    SiftDown(Size-1,1);  
end;
```

# 5 Varianten von Heapsort

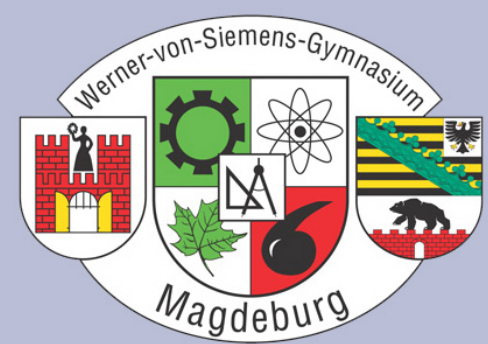


- Bottom-Up-Heapsort
  - Spart Vergleiche,
  - sofort zum Ende versickert → wiederaufsteigen
- Smoothsort
  - Umkehren des Heaps → Vorteil bei vorsortierten Folgen
- Ternäre Heaps
  - 3 statt 2 Kindsnoten → Reduzierung der Vergleiche
  - Bei 1 Mio. Elementen 20-30% Zeitersparnis
- n-äre Heaps
  - n Kindsnoten
  - Mehr Vergleiche, dafür weniger Vertauschungen

# 6 Quellen



- <http://www.wikipedia.org>
- <http://www.wanginator.de>
- <http://www.hipphampel.de>
- <http://www.activevb.de>
- <http://wwwdvs.informatik.uni-kl.de>
- <http://www.inf.fh-flensburg.de>
- <http://lernen.bildung.hessen.de>
- Vorträge unter \\zeus\pub



Wir bedanken uns für Eure  
Aufmerksamkeit und stehen Euch  
nun für Fragen zur Verfügung.

Vortrag unter  
[www.florian-sachs.de](http://www.florian-sachs.de)