

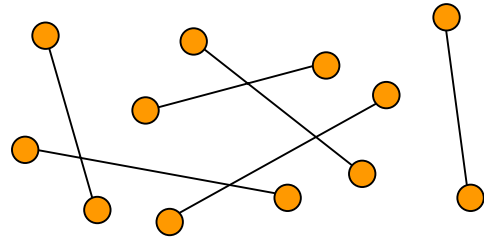
Wie schnell kann man Kreuzungen zählen?

VO Algorithm Engineering
Professor Dr. Petra Mutzel
Lehrstuhl für Algorithm Engineering, LS11

25. VO

31.01.2006

Allgemeines Problem

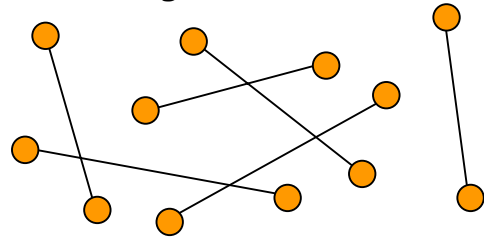


Gegeben: Menge S von Geradensegmenten
Gesucht: Anzahl der Schnittpunkte

Literatur für diese VO

- W. Barth, M. Jünger und P. Mutzel: Simple and Efficient 2-Layer Cross Counting, Journal of Graph Algorithms and Applications (JGAA), 2003

Allgemeiner Fall

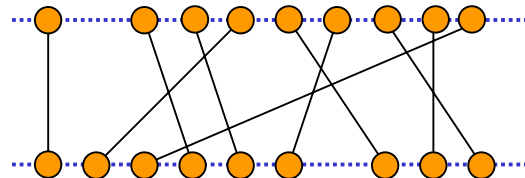


Ausgeben der Kreuzungen:
Chazelle & Edelsbrunner [1992] $O(|E| \log |E| + |C|)$
Nur Zählen:
Chazelle [1985] $O(|E|^{1.695})$

Überblick

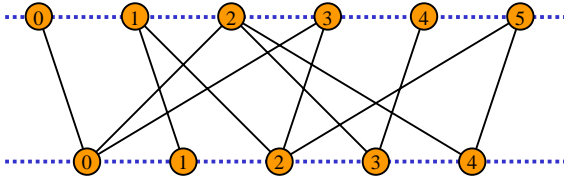
- Einführung: Problem
- Algorithmen
 - verschiedene klassische Algorithmen
 - BJM-Algorithmus
- Experimente

Spezielles Problem



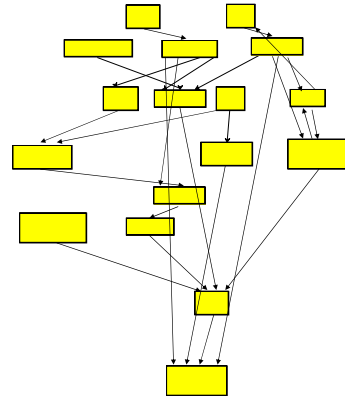
Endpunkte der Segmente
liegen auf parallelen Geraden

...eigentlich: bipartiter Graph

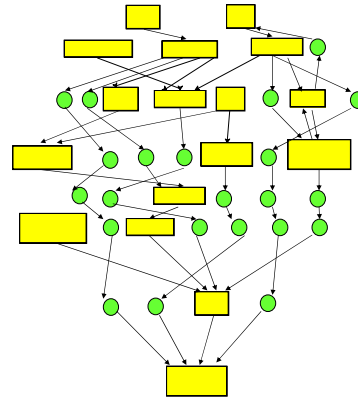


Anwendung:
Automatisiertes Zeichnen von Graphen

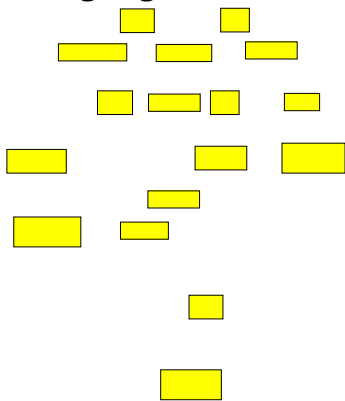
Festlegung der Schichten



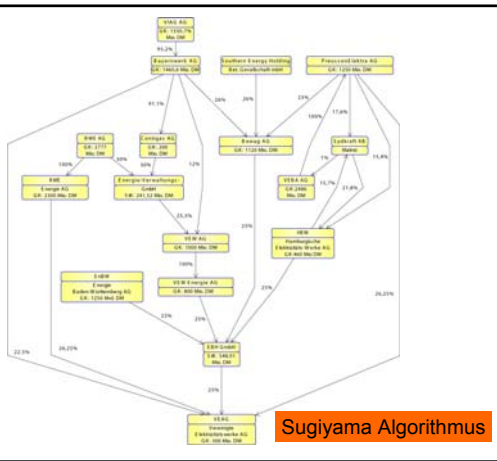
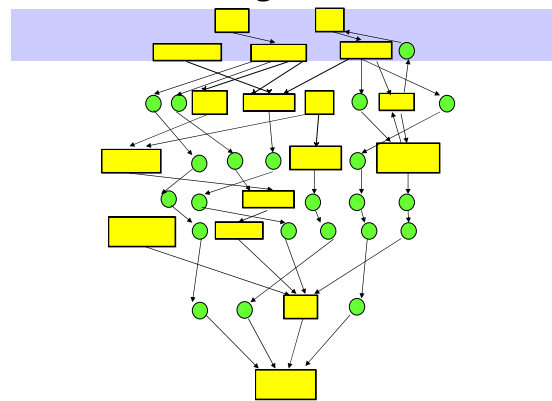
Einfügung künstlicher Knoten

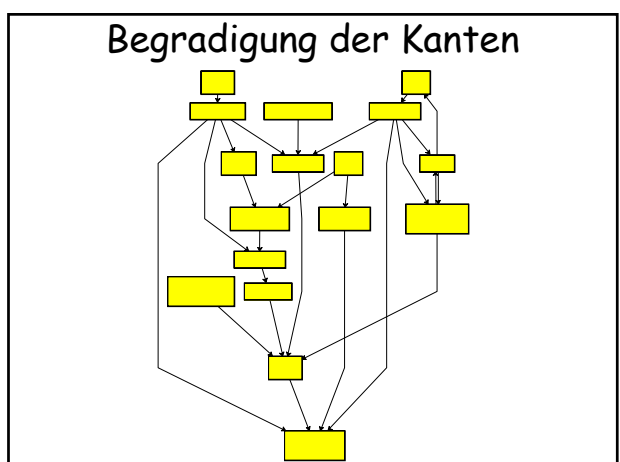
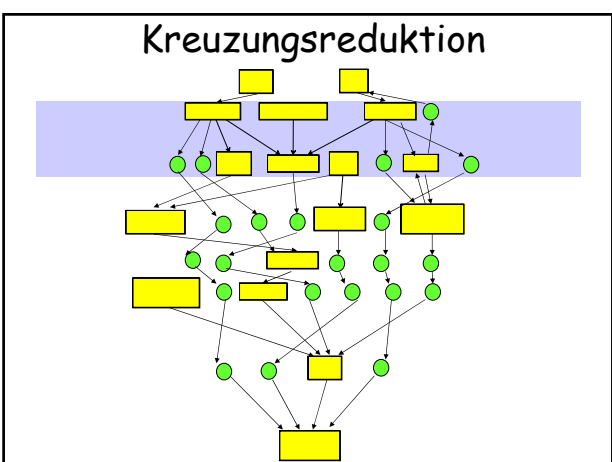
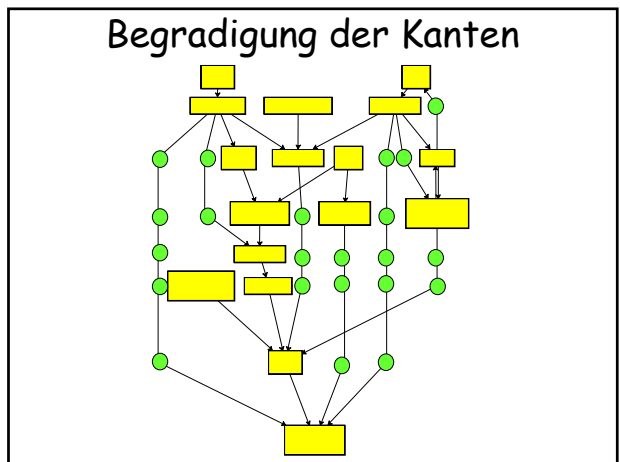
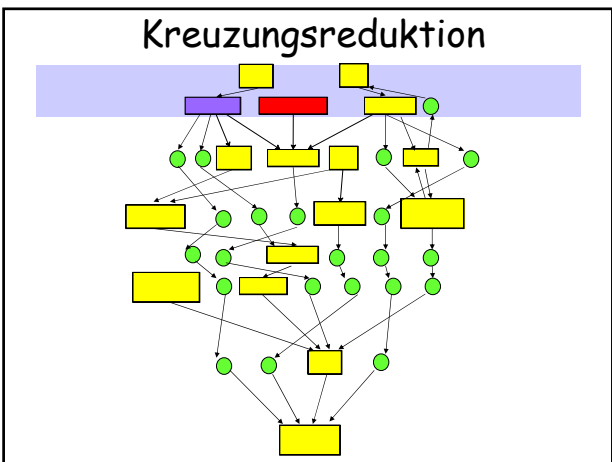
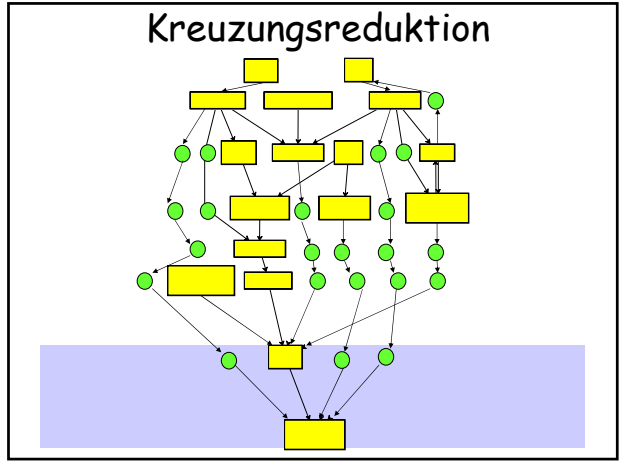
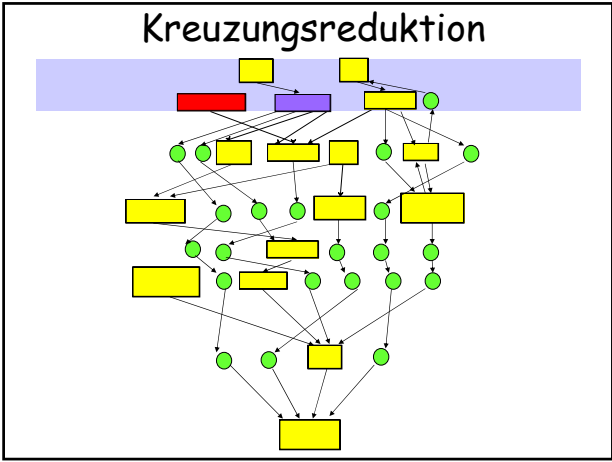


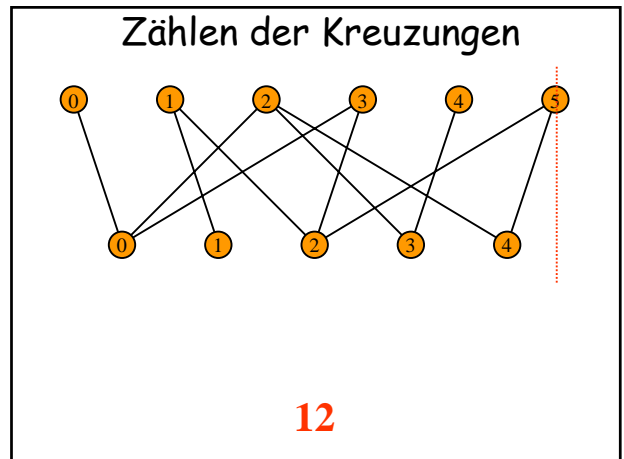
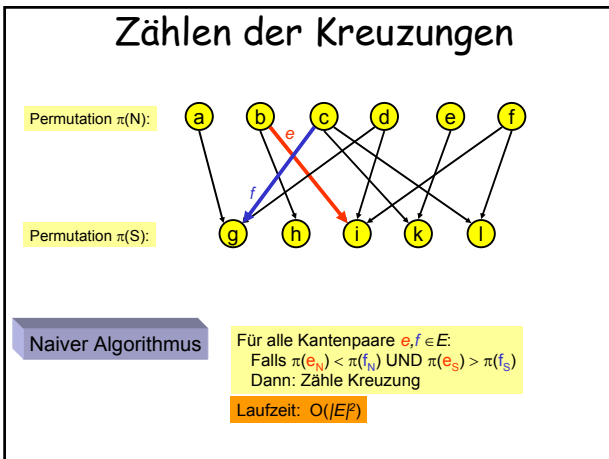
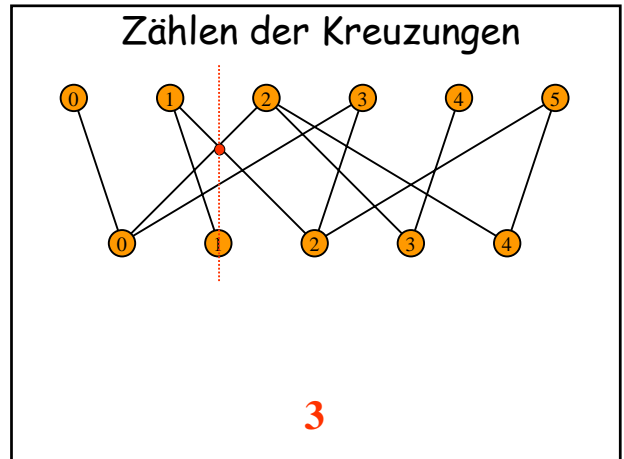
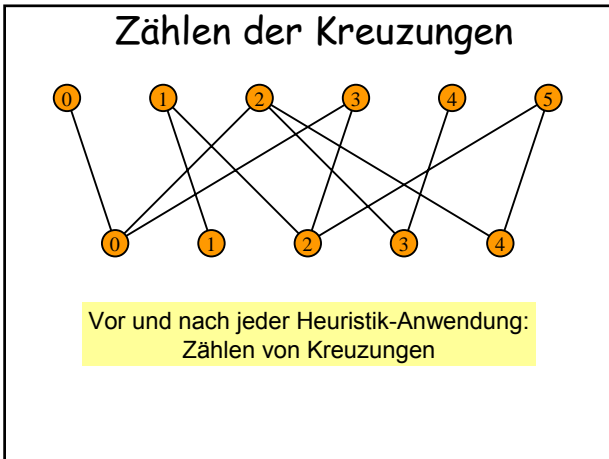
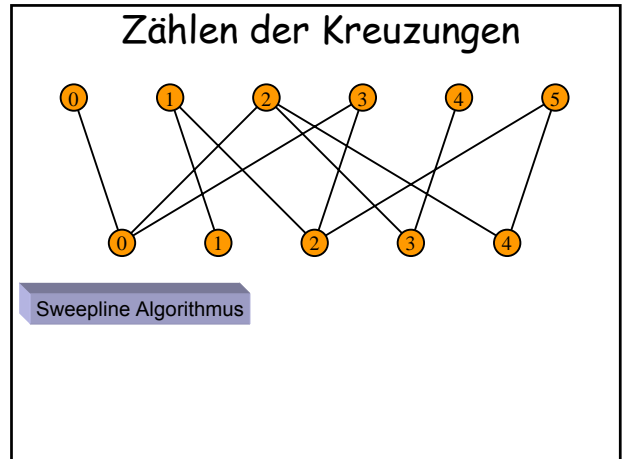
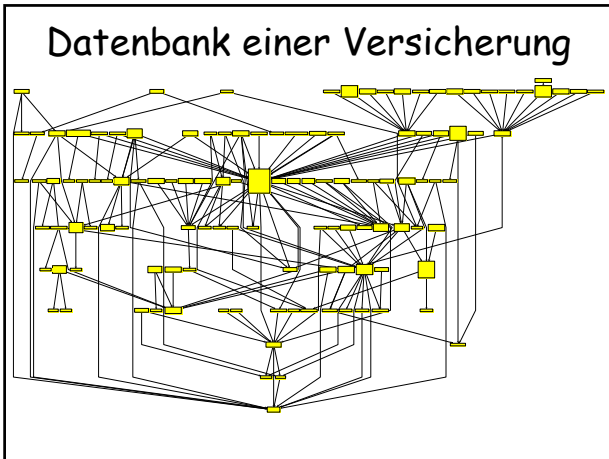
Festlegung der Schichten



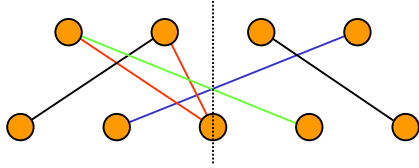
Kreuzungsreduktion







Sweepline Algorithmus

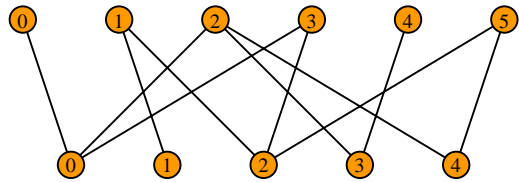


Bewege Sweepline von links nach rechts:

Teilung in

- bereits beendete Kanten (links),
- noch nicht erreichte Kanten (rechts), und
- aktive Kanten (1 Endknoten erreicht)

Zählen der Kreuzungen



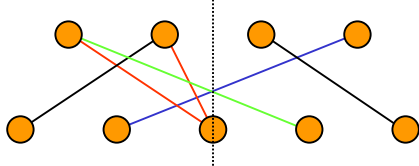
Ein zweiter Sweepline Algorithmus:

$O(|E| \log |V|)$ Waddle & Malhotra [1999]

Probleme:

- viele komplizierte Fallunterscheidungen
- aufwändiger Algorithmus (viele Seiten)
- aufwändige Implementierung

Sweepline von Sander 1995



Zwei geordnete Listen UL und LL:

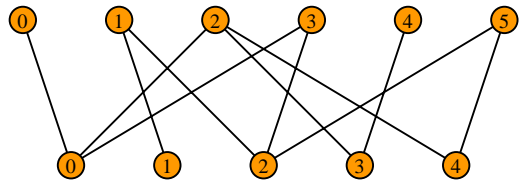
Endknoten der aktiven Kanten wird in der

- UL-Liste in der **oberen** Schicht deaktiviert
- LL-Liste in der **unteren** Schicht deaktiviert

Laufzeit: $O(|E| + |C|)$

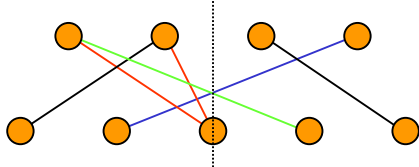
(Graph sei zusammenhängend)

Zählen der Kreuzungen



Ein einfacher
 $O(|E| \log |V|)$ Algorithmus

Geht es schneller als $O(|E| + |C|)$?



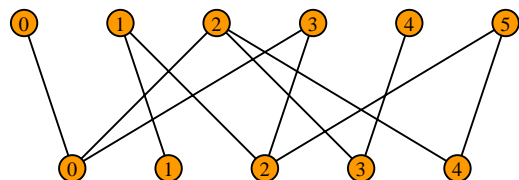
Anzahl der Kreuzungen:

- Worst Case: $|E|^2$
- Average Case: $\frac{1}{4} |E|^2$

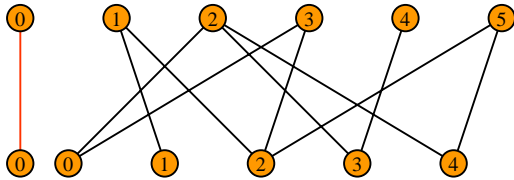
Auflisten von Kreuzungen: **NEIN!**

Zählen von Kreuzungen: **JA!**

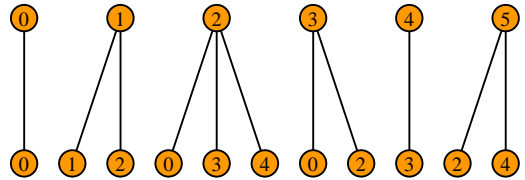
Lexikographisches Sortieren



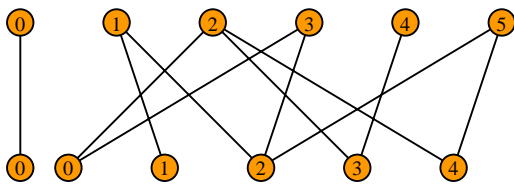
Lexikographisches Sortieren



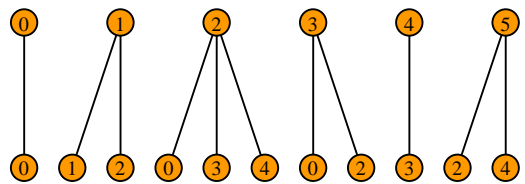
Lexikographisches Sortieren



Lexikographisches Sortieren

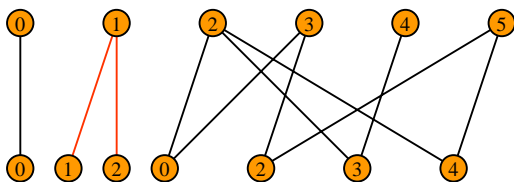


Beobachtung



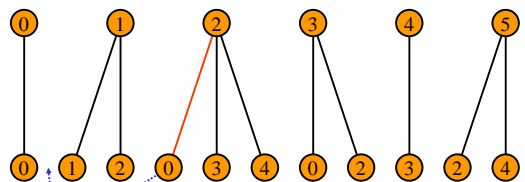
Die Anzahl der Kreuzungen ist gleich der Anzahl der Inversionen der unteren Folge.

Lexikographisches Sortieren

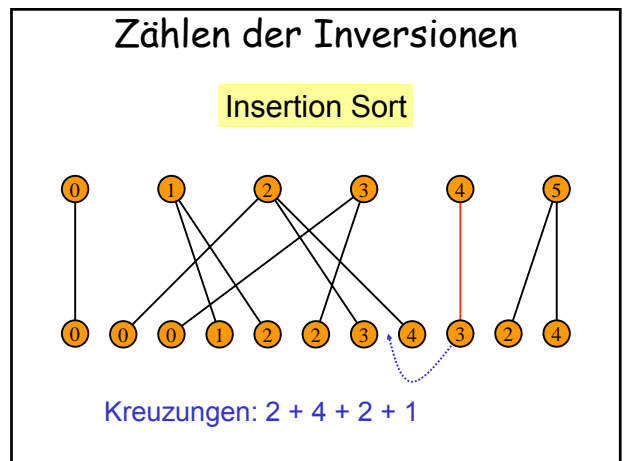
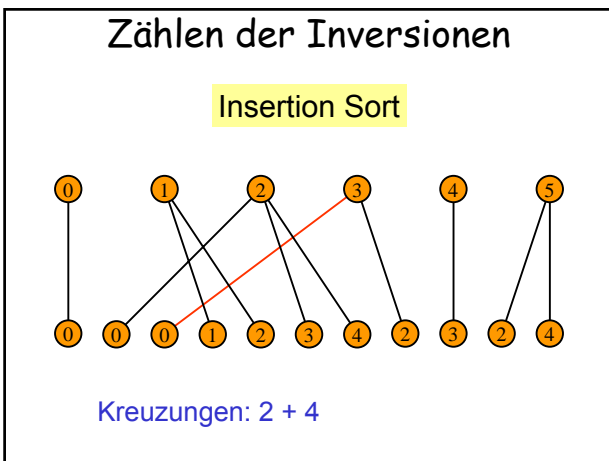
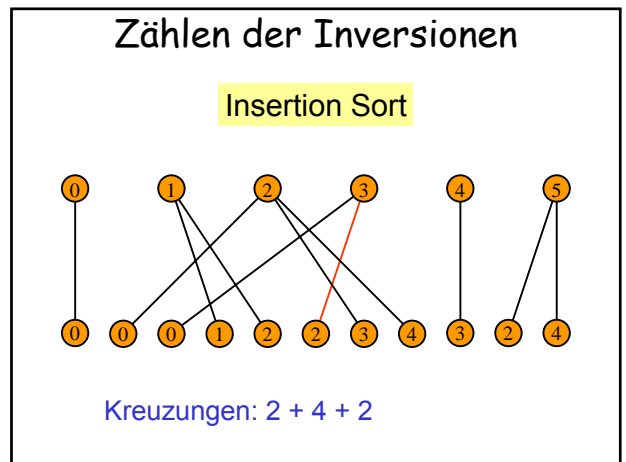
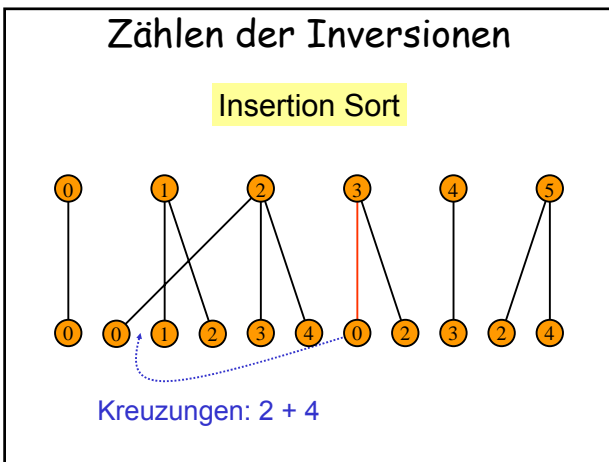
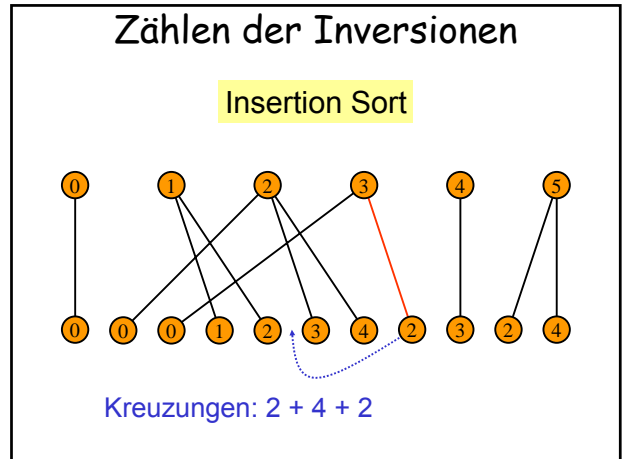
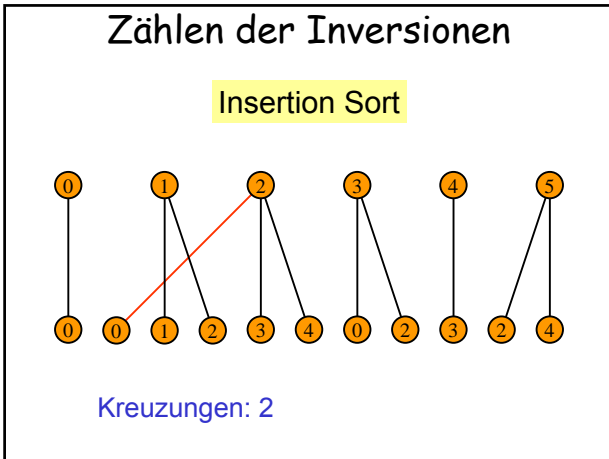


Zählen der Inversionen

Insertion Sort

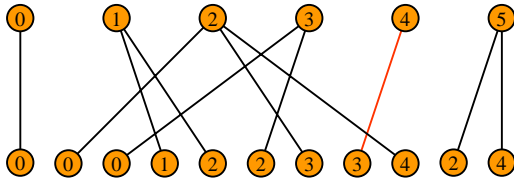


Kreuzungen: 2



Zählen der Inversionen

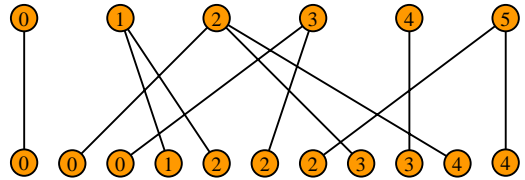
Insertion Sort



Kreuzungen: $2 + 4 + 2 + 1$

Zählen der Inversionen

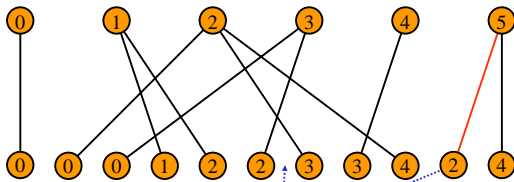
Insertion Sort



Kreuzungen: $2 + 4 + 2 + 1 + 3 = 12$

Zählen der Inversionen

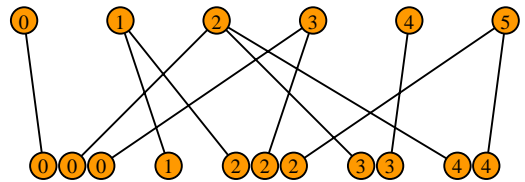
Insertion Sort



Kreuzungen: $2 + 4 + 2 + 1 + 3$

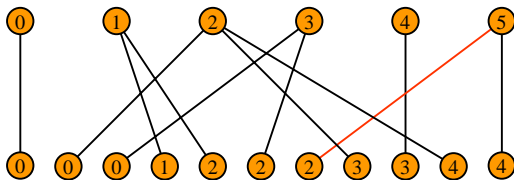
Zählen der Inversionen

Insertion Sort



Zählen der Inversionen

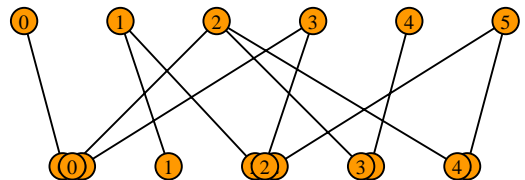
Insertion Sort



Kreuzungen: $2 + 4 + 2 + 1 + 3$

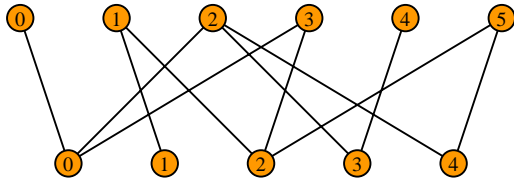
Zählen der Inversionen

Insertion Sort



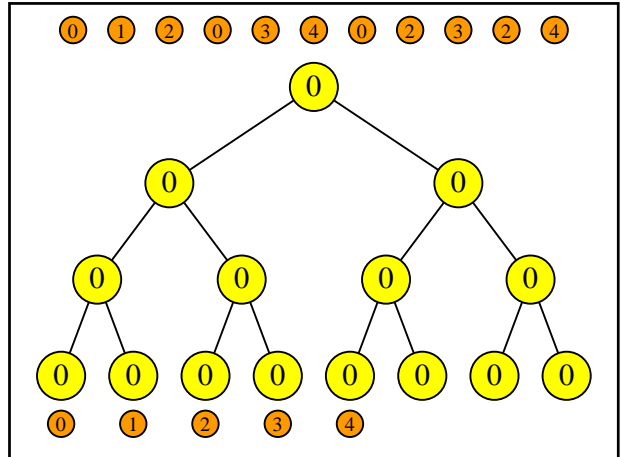
Zählen der Inversionen

Insertion Sort
 Laufzeit: $O(|C|+|E|)$

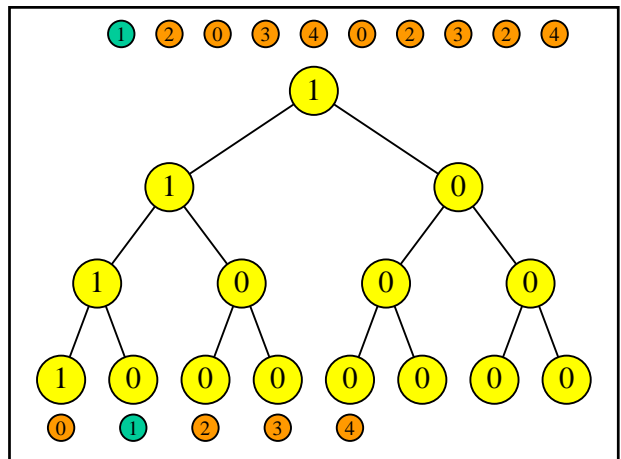
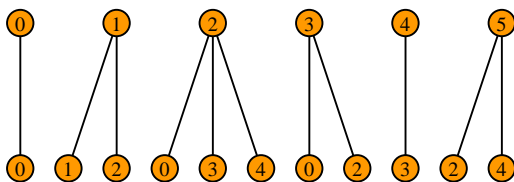
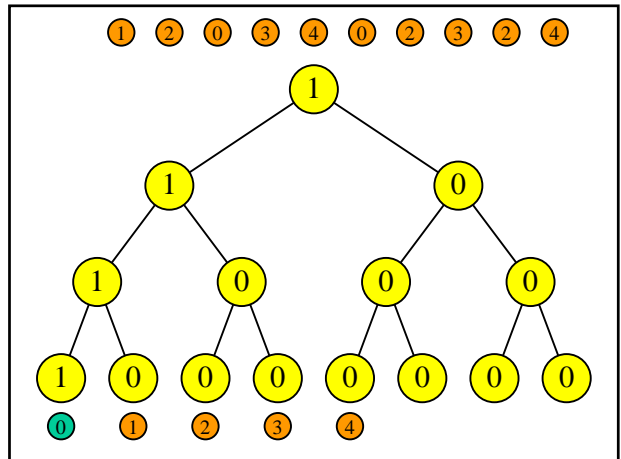
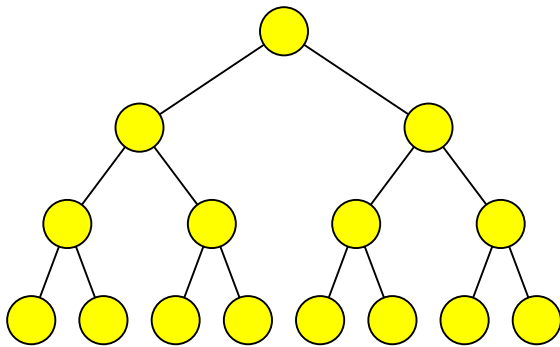


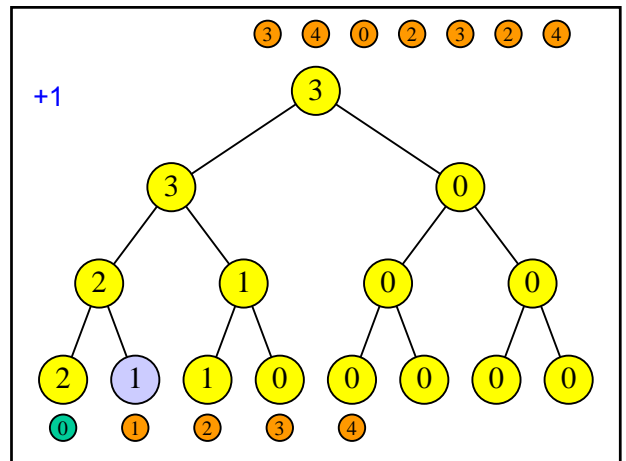
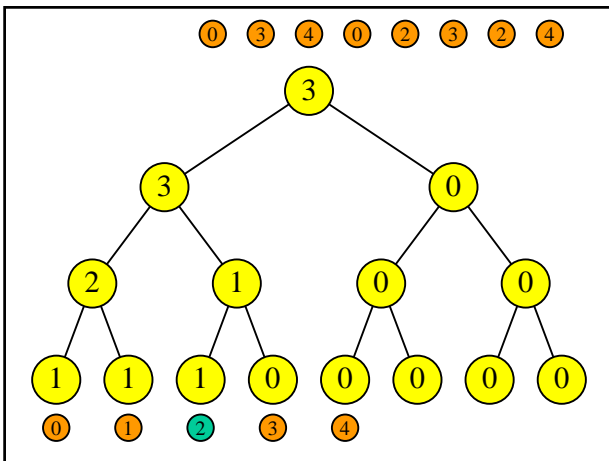
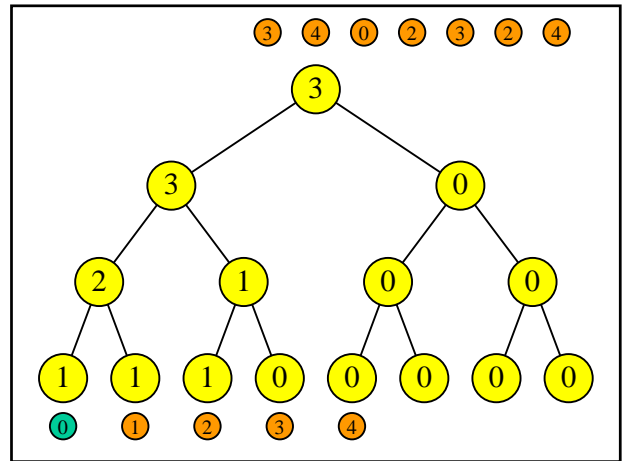
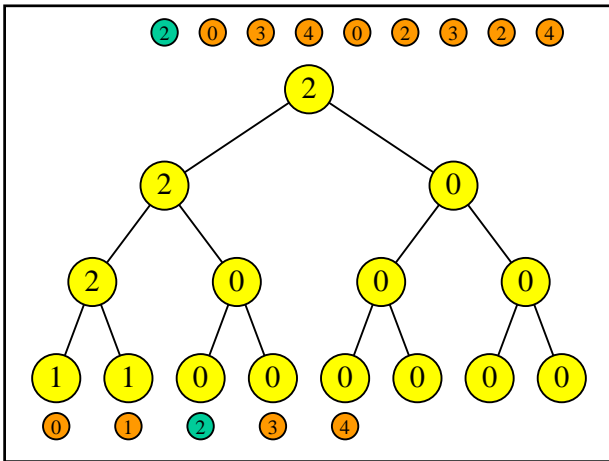
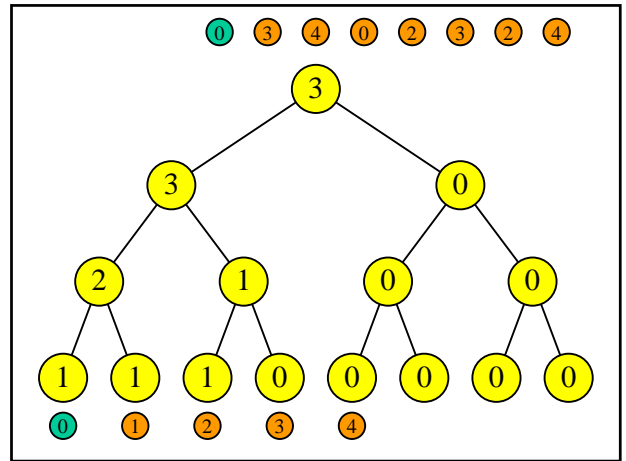
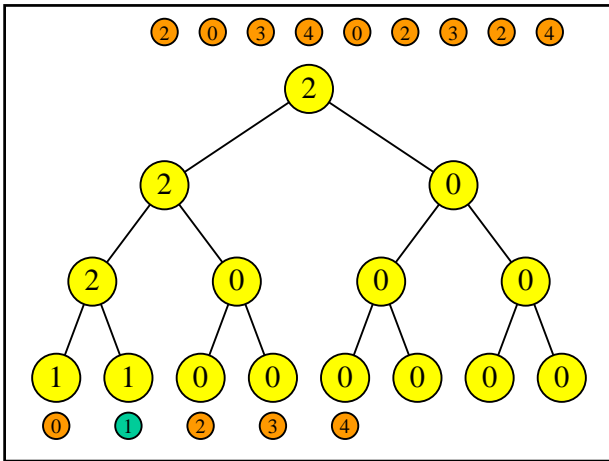
Alternative 1: Merge Sort: $O(|E| \log |E|)$

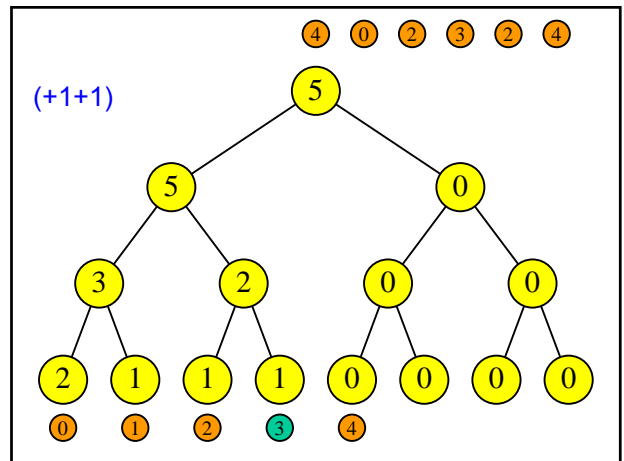
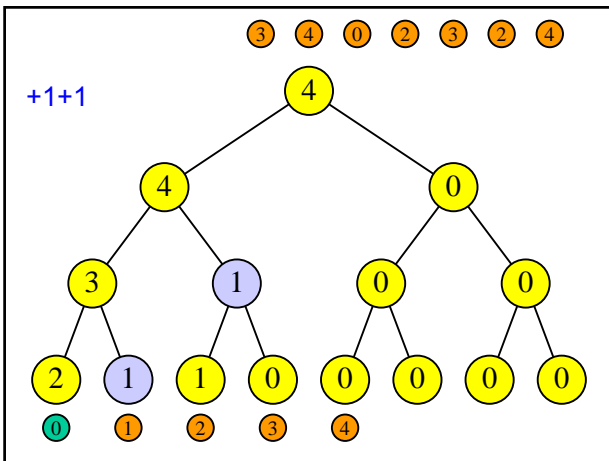
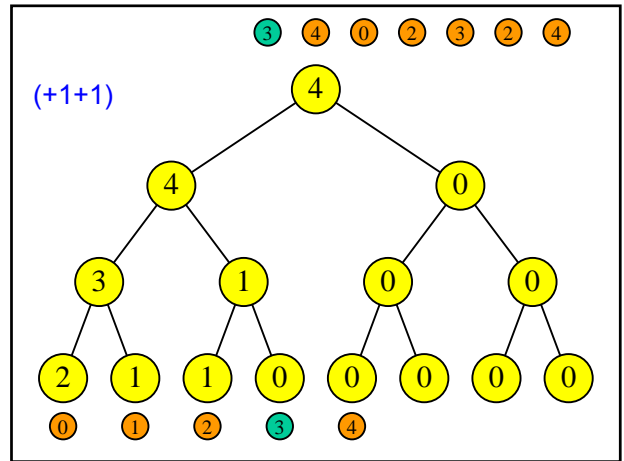
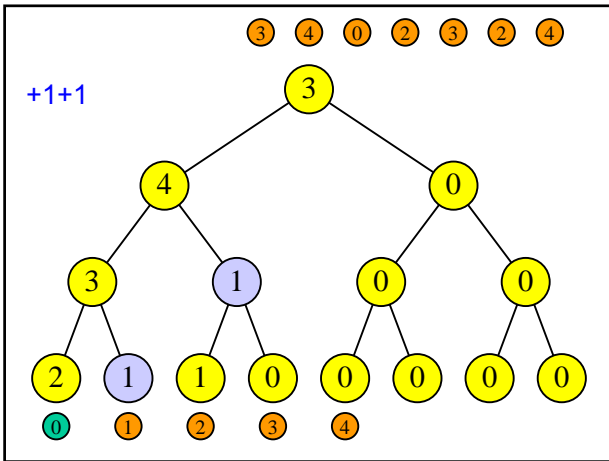
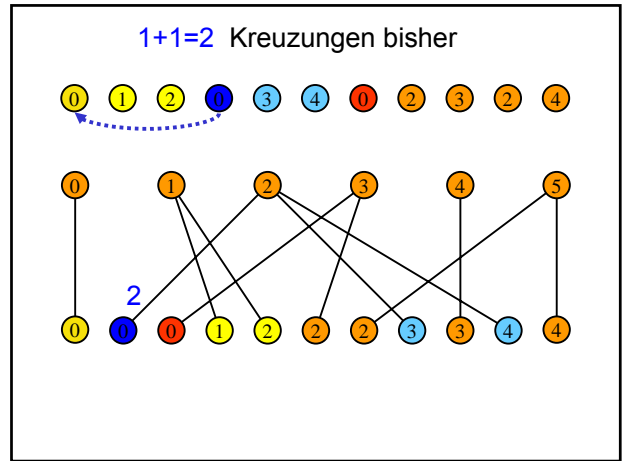
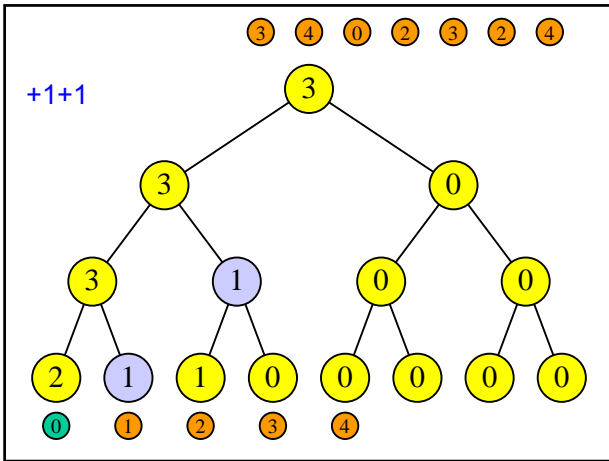
Alternative 2: Verbesserung auf $O(|E| \log |V_{\text{small}}|)$

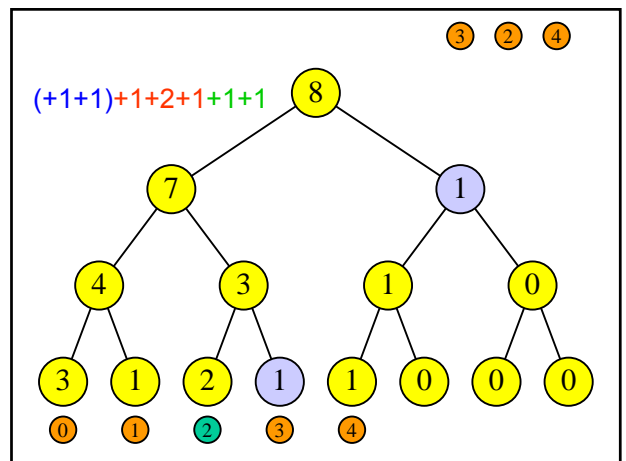
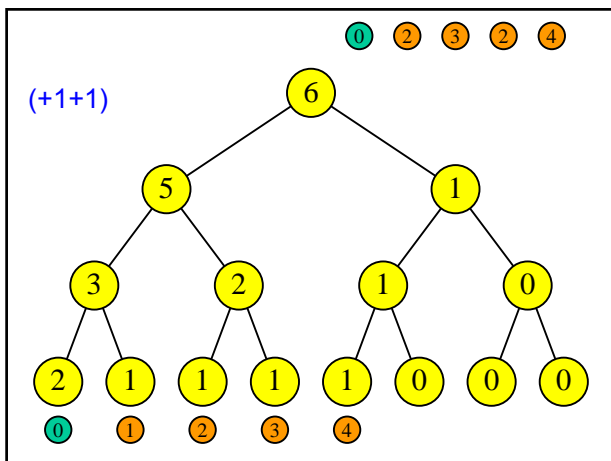
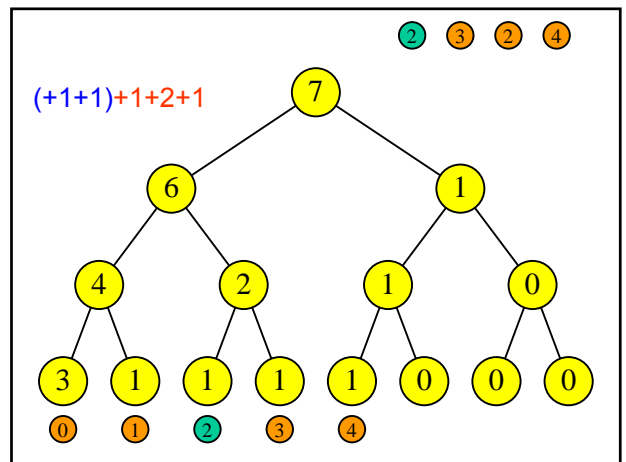
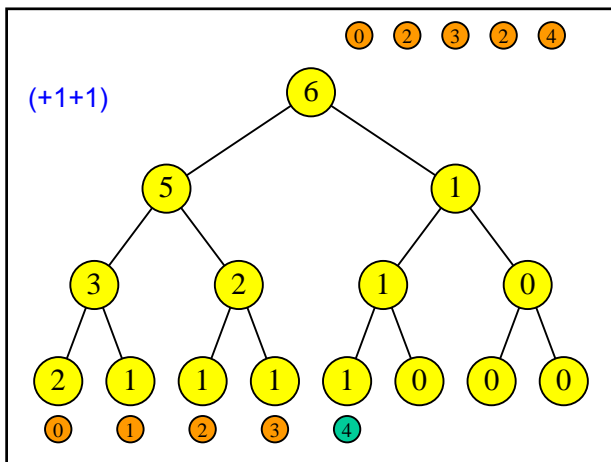
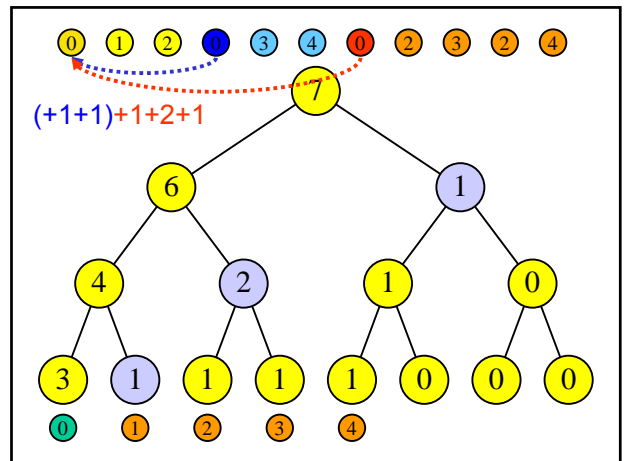
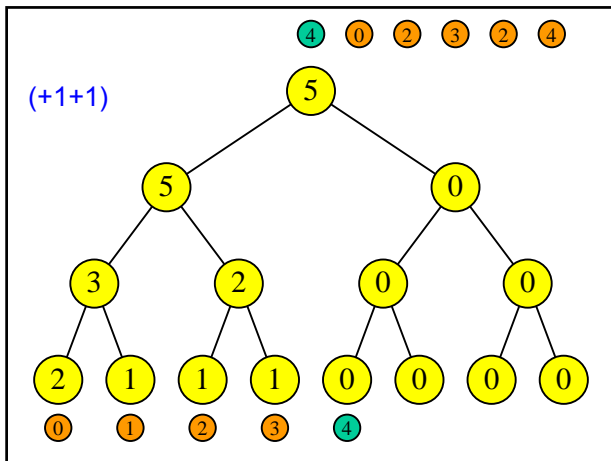


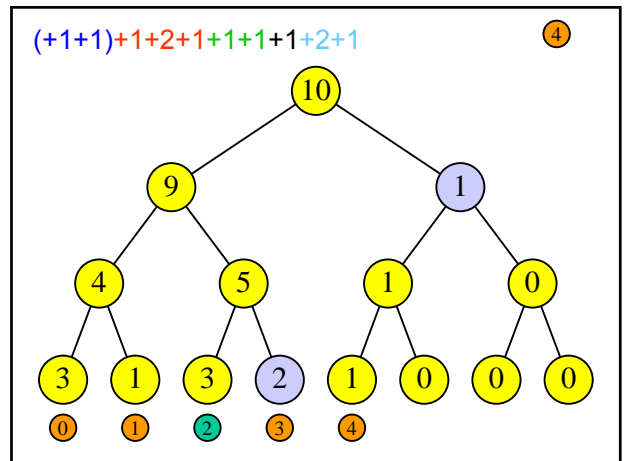
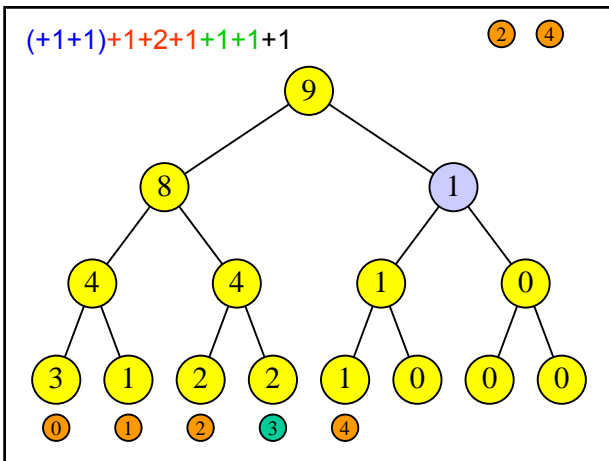
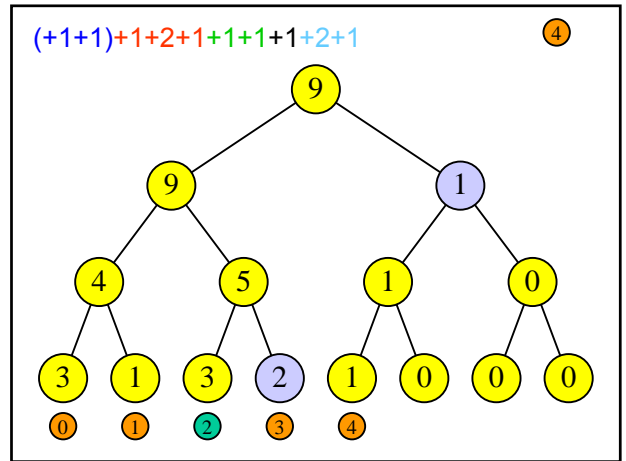
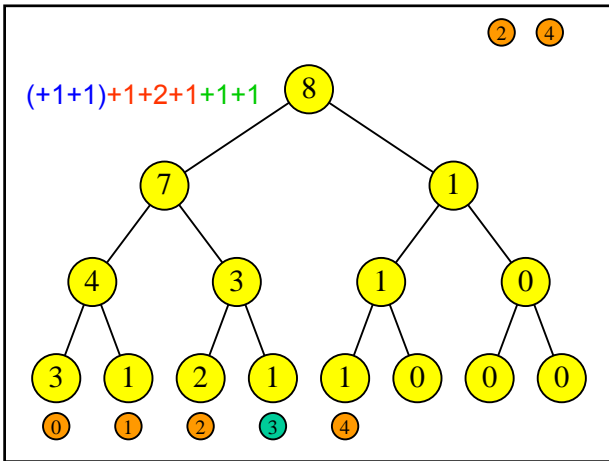
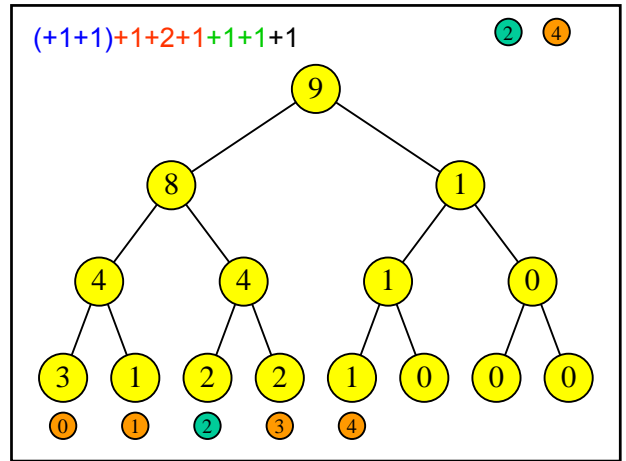
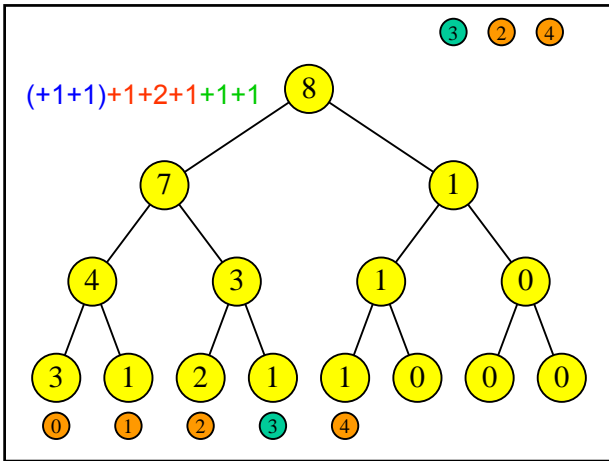
Accumulator Tree

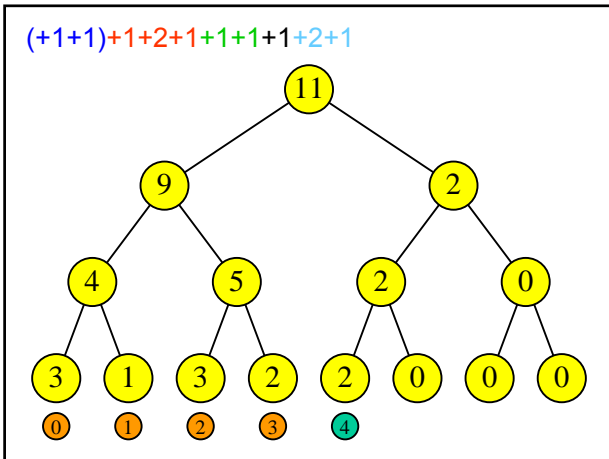
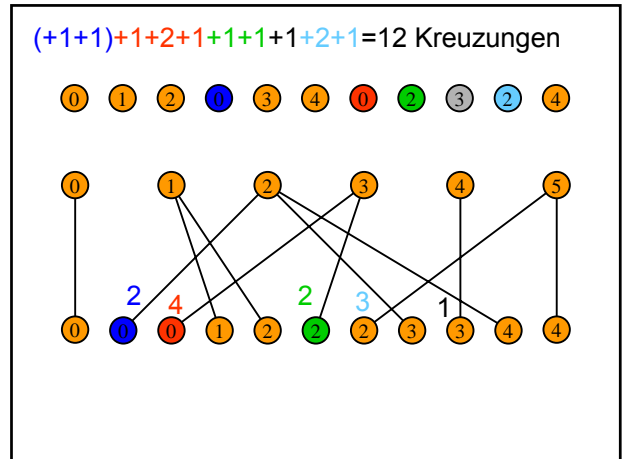
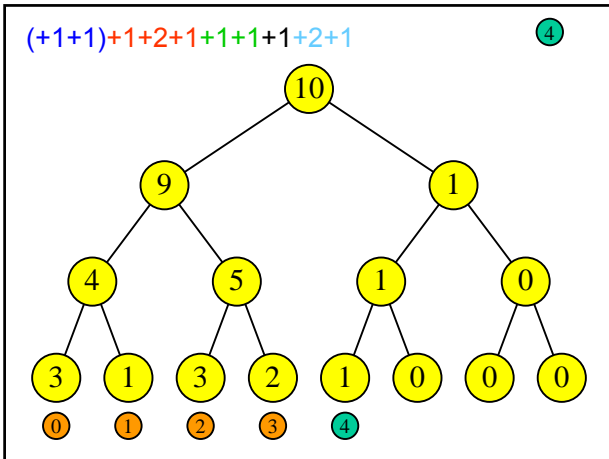












C-Program-Fragment

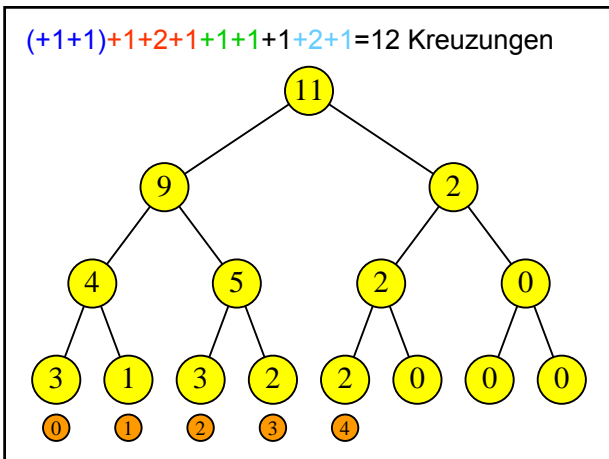
```

/* build the accumulator tree */
firstindex = 1;
while (firstindex < q) firstindex *= 2;
treesize = 2*firstindex - 1; /* number of tree nodes */
firstindex -= 1; /* index of leftmost leaf */
tree = (int *) malloc(treesize*sizeof(int));
for (t=0; t<treesize; t++) tree[t] = 0;

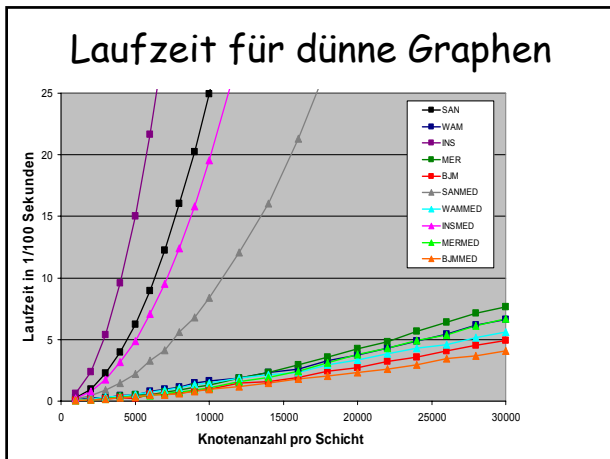
/* count the crossings */

crosscount = 0; /* number of crossings */
for (k=0; k<r; k++) { /* insert edge k */
  index = southsequence[k] + firstindex;
  tree[index]++;
  while (index > 0) {
    if (index % 2) crosscount += tree[index + 1];
    index = (index - 1) / 2;
    tree[index]++;
  }
}
printf("Number of crossings: %d\n", crosscount);
free(tree);

```



- ### Praktische Experimente
- SAN Sander [1995] $O(|E| + |C|)$
 - WAM Waddle & Malhotra [1999] $O(|E| \log |V|)$
 - INS Insertion Sort $O(|E| + |C|)$
 - MER Merge Sort $O(|E| \log \text{run}(p))$
 - BJM Neuer Algorithmus $O(|E| \log |V_{\text{small}}|)$



Computational Experiments

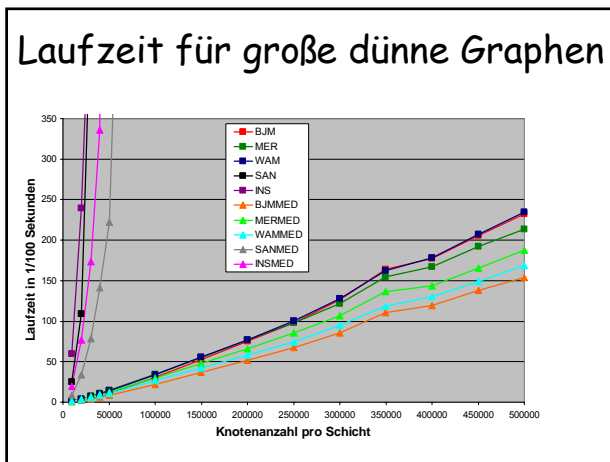
AT&T Graphen

1. Phase: Schichteneinteilung von AGD

- longest-path-layering: 30,061 Instanzen
- Coffman-Graham-layering: 57,300 Instanzen

Für jede Instanz:
10 zufällige Umsortierungen
gefolgt durch einen Median-Sortierschritt

Anzahl an Testläufen:
601,220 "longest path"
1,146,000 "Coffman-Graham"



Computational Experiments

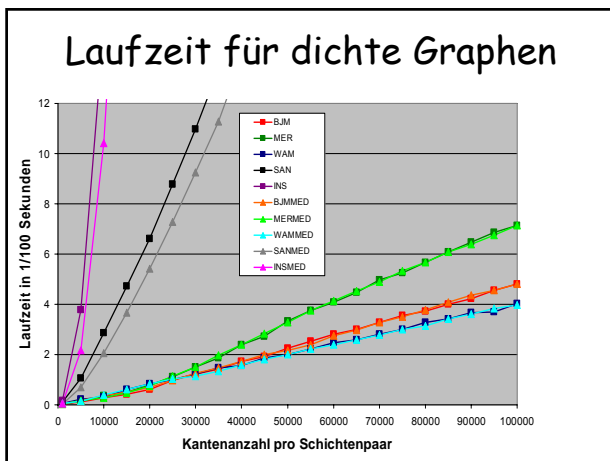
AT&T Graphen

Longest path layering

- 1 bis 6,566 obere Knoten, 63 durchschnittlich
- 1 bis 5,755 untere Knoten, 57 durchschnittlich
- 1 bis 6,566 Kanten, 64 durchschnittlich

Zufällige Umsortierungen:
0 bis 10,155,835 Kreuzungen, 24,472 durchschnittlich

Median-sortierte Schichten:
0 bis 780,017 Kreuzungen, 182 durchschnittlich



Computational Experiments

AT&T Graphen

Coffman-Graham layering

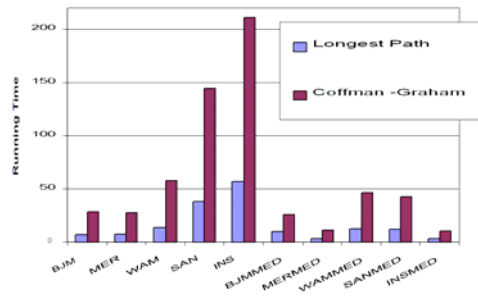
- 1 bis 3,278 obere Knoten, 142 durchschnittlich
- 1 bis 3,278 untere Knoten, 137 durchschnittlich
- 1 bis 3,276 Kanten, 141 durchschnittlich

Zufällige Umsortierungen:
0 bis 2,760,466 Kreuzungen, 47,559 durchschnittlich

Median-sortierte Schichten:
0 bis 2,872 Kreuzungen, 4 durchschnittlich

Graphen aus der Praxis

AT&T Graphen



Folgerung

WAM, MER, BJM
Reduzieren die Laufzeit
des Sugiyama-Algorithmus
signifikant!

BJM ist sehr einfach
zu implementieren!

Offenes Problem

Gegeben: eine Permutation von $0..n-1$:

Ist wirklich Zeit $\Theta(n \log n)$ nötig, um
die Anzahl der Inversionen zu zählen?

Vielen Dank