

Kap. 2: Hierarchisches Graphenzeichnen

2.1. Kreuzungszählen

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

2. VO

5. April 2007

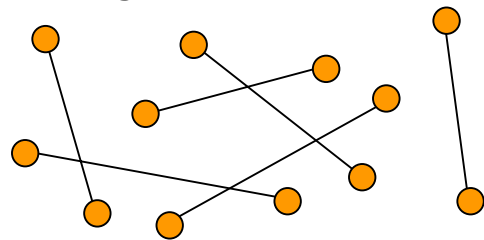
Literatur für diese VO

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

Überblick

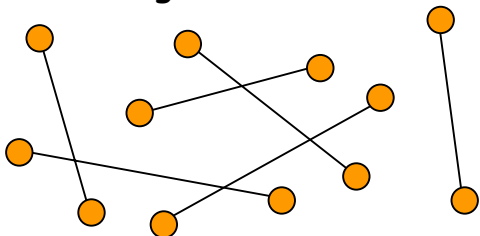
- Einführung
 - Hierarchisches Graphzeichnen
 - Kreuzungszählen
- Algorithmen
 - verschiedene klassische Algorithmen
 - BJM-Algorithmus
- Experimente

Allgemeines Problem



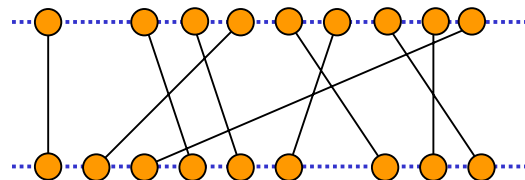
Gegeben: Menge S von Geradensegmenten
Gesucht: Anzahl der Schnittpunkte

Allgemeiner Fall



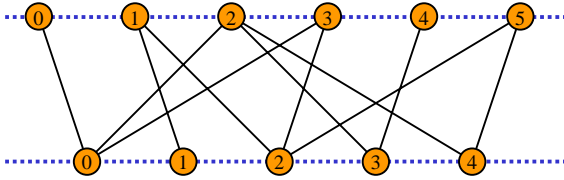
Ausgeben der Kreuzungen:
Chazelle & Edelsbrunner [1992] $O(|E| \log |E| + |C|)$
Nur Zählen:
Chazelle [1985] $O(|E|^{1.695})$

Spezielles Problem



Endpunkte der Segmente
liegen auf parallelen Geraden

...eigentlich: bipartiter Graph



Anwendung:
Automatisiertes Zeichnen von Graphen

■ **Mysterien und Leberwurstbrote oder: Die wirrsten Grafiken der Welt (8)**
Die Verflechtung der Stromwirtschaft (nach Michael Stelte) taz

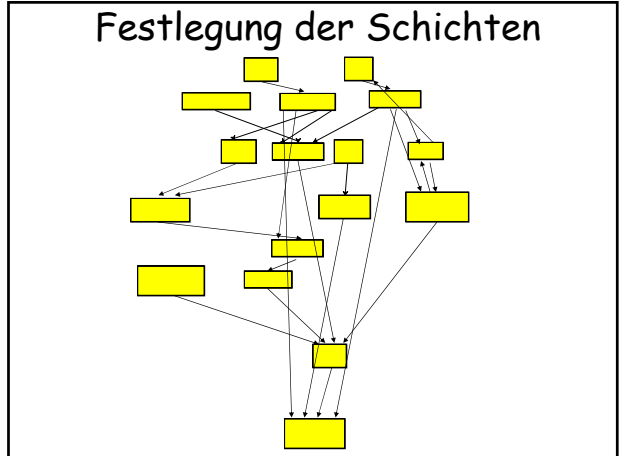
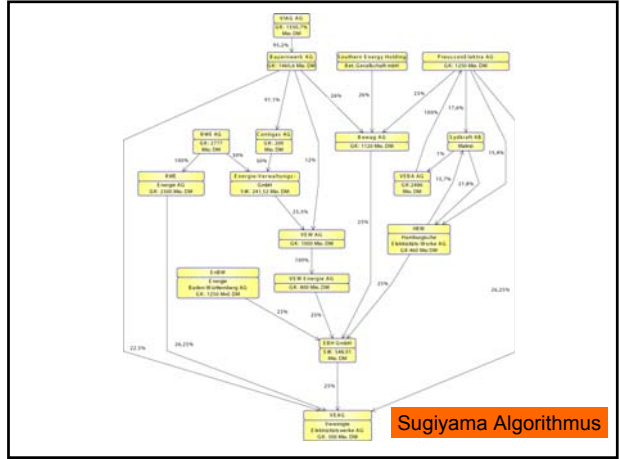
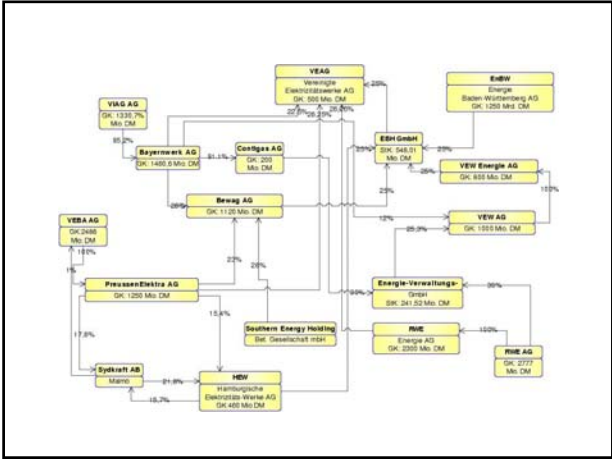
Wir sind verflochten die Stromwirtschaft in Deutschland ist, sagt eine illustrierte Grafik aus Michael Steltes Berliner Energiezeitschrift: Die Hamburgerische Electricitäts-Werke AG ist beispielsweise mit 15,7 Prozent an der Sydnah AG Mainz beteiligt, diese mit einem Prozent an der Veba AG, diese mit 100 Prozent an der PreussenElektra AG, diese mit 17,8 Prozent an der Sydnah AB Mainz und diese wiederum mit 21,8 Prozent an der Hamburgerischen Electricitäts-Werke AG. Ob es wohl Steuerpflöge gibt, die es wegen, sich solche komplizierten Besitzverhältnisse eintrüpfeln und mit Tachemeter, Aggregat und Leberwurstbrot bewaffnet in die Mysterien der Stromwirtschaft-Buchhaltung einzudringen? Oder überlässt das Finanzamt diese Arbeit lieber Sisyphos? Und sich ständerten verflochten Klumpen auszuwaschen?

Das wäre vorstellbar, denn die Verflechtung der Stromwirtschaft ist selbst unersieglbar, auch wenn die Grafik auf den ersten Blick sehr übersichtlich wirkt. Einmal angenommen, die Vloger wüssten alles Markt-Geschehen. Dann gehören 25 Prozent davon der E.ON GmbH (ca. 2.500 Mrd. DM), das sind 25 Prozent der VEW Energie AG (ca. 0,2474275 Mrd. DM) und deren 21,1 Prozent ebenfalls der Bayerwerk AG (ca. 0,049215625 Mrd. DM).

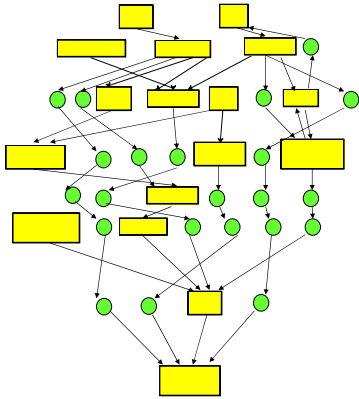
Damit hat die Bayerwerk AG über 25,3 Prozent von den 0,25 Mrd. DM des VEW Energie AG. Aber 25,3 Prozent von 0,25 Mrd. DM ergibt 0,13125 Mrd. DM. Aber die Bayerwerk ist ja zudem auch noch mit weiteren 22,5 Prozent direkt an der Vloger beteiligt, so dass von den 0,25 Mrd. DM noch einmal 2,25 Mrd. DM abfallen und sich so mit einer Gesamtsumme von 2,38125 Mrd. DM ergibt.

2) Wieviel schuldet die Bayerwerk AG zusätzlich der Vloger AG? 3) Ruf bei der Bayerwerk AG in München an (E.ON-Geschäftsbereich) 12241) und erkundige dich danach, ob deine Rechnung richtig ist!

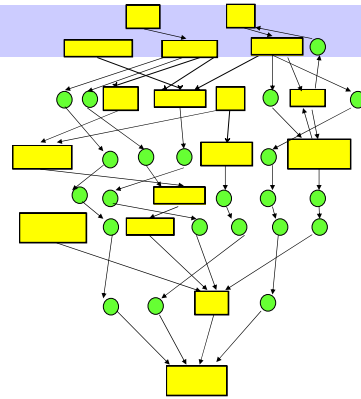
Gerhard Henschel



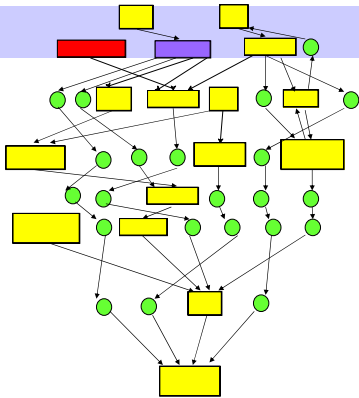
Einfügung künstlicher Knoten



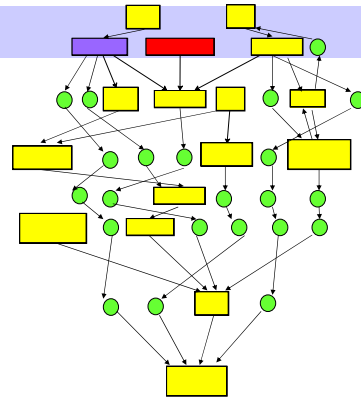
Kreuzungsreduktion



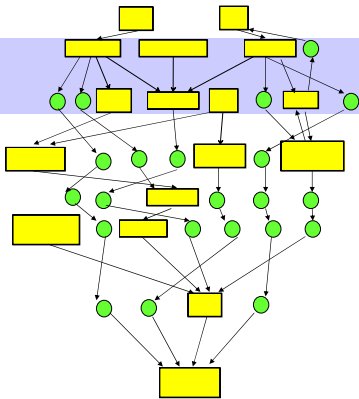
Kreuzungsreduktion



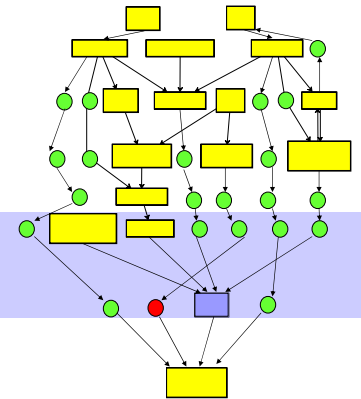
Kreuzungsreduktion



Kreuzungsreduktion

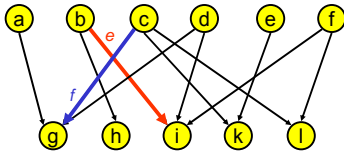


Kreuzungsreduktion



Zählen der Kreuzungen

Permutation $\pi(N)$:



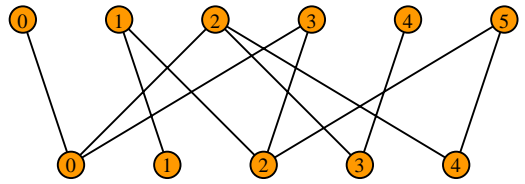
Permutation $\pi(S)$:

Naiver Algorithmus

Für alle Kantenpaare $e, f \in E$:
 Falls $\pi(e_N) < \pi(f_N)$ UND $\pi(e_S) > \pi(f_S)$
 Dann: Zähle Kreuzung

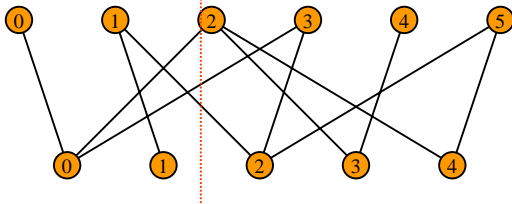
Laufzeit: $O(|E|^2)$

Zählen der Kreuzungen



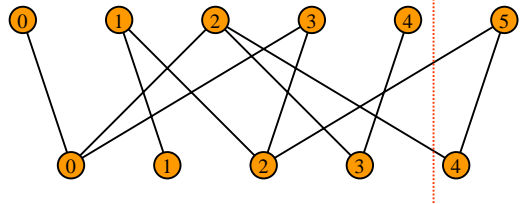
Sweepline Algorithmus

Zählen der Kreuzungen



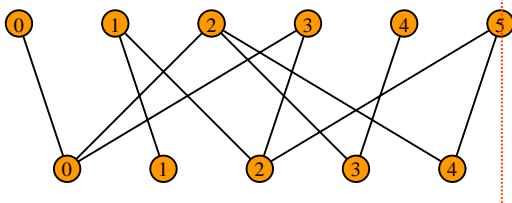
4

Zählen der Kreuzungen



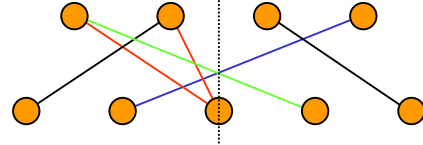
12

Zählen der Kreuzungen



12

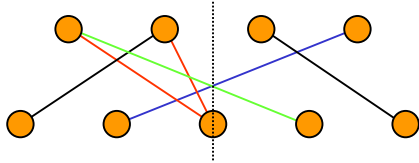
Sweepline Algorithmus



Bewege Sweepline von links nach rechts:
 Teilung in

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

Sweepeline von Sander 1995



Zwei geordnete Listen UL und LL:

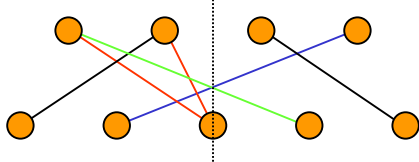
- Endknoten der aktiven Kanten wird in der
- UL-Liste in der **oberen** Schicht deaktiviert
 - LL-Liste in der **unteren** Schicht deaktiviert

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

Sweepeline von Sander 1995

0. Vorverarbeitung: Splitte Knoten, so dass Knotengrad = 1 für alle Knoten.
- Durchlaufe die Knoten nacheinander (von links nach rechts).
- Fall A: Sei v der aktuelle Knoten in unteren Schicht von Kante (w,v)
3. Falls v Endknoten ist, dann
4. CrossCount += |LL|
5. $u = \text{first}(UL)$
6. While u links von w liegt {
7. CrossCount += 1; $u = \text{next}(UL)$
8. }
9. /* jetzt gilt: $u=w$ /* entferne w aus Liste UL
10. Sonst: aktiviere neue Kante: Eintrag in Liste LL
11. Fall B: vertausche Rollen von UL und LL

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



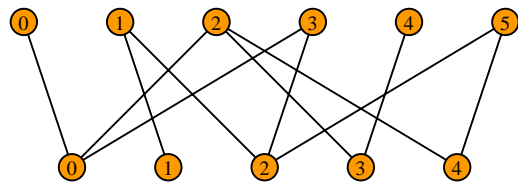
Anzahl der Kreuzungen:

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

Aufzählen von Kreuzungen: **NEIN!**

Zählen von Kreuzungen: **JA!**

Zählen der Kreuzungen



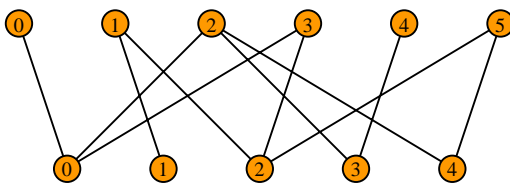
Ein zweiter Sweepeline Algorithmus:

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

Probleme:

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

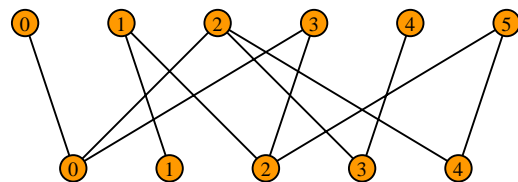
Zählen der Kreuzungen



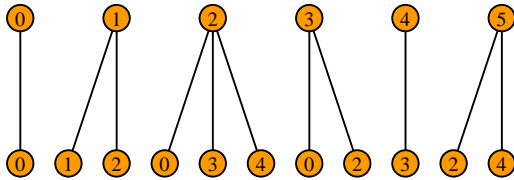
Ein einfacher

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

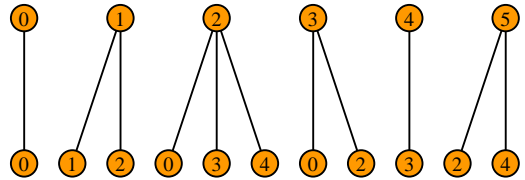
Lexikographisches Sortieren



Lexikographisches Sortieren



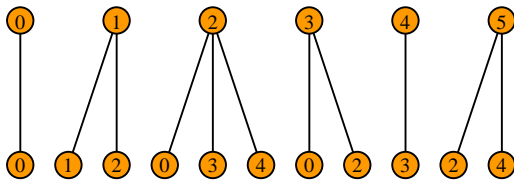
Beobachtung



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

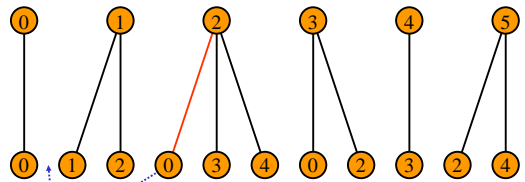
Zählen der Inversionen

Insertion Sort



Zählen der Inversionen

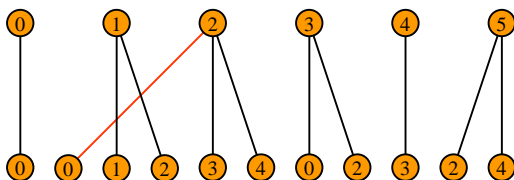
Insertion Sort



Kreuzungen: 2

Zählen der Inversionen

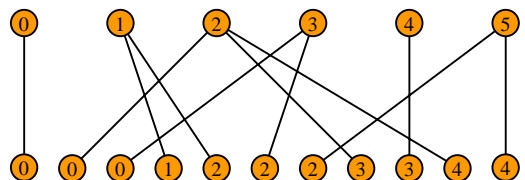
Insertion Sort



Kreuzungen: 2

Zählen der Inversionen

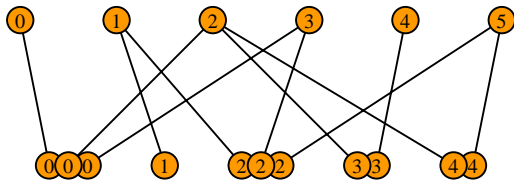
Insertion Sort



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

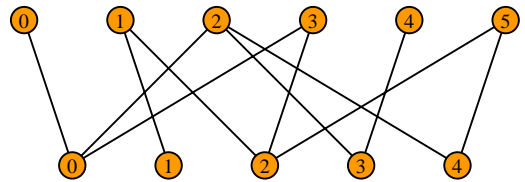
Zählen der Inversionen

Insertion Sort



Zählen der Inversionen

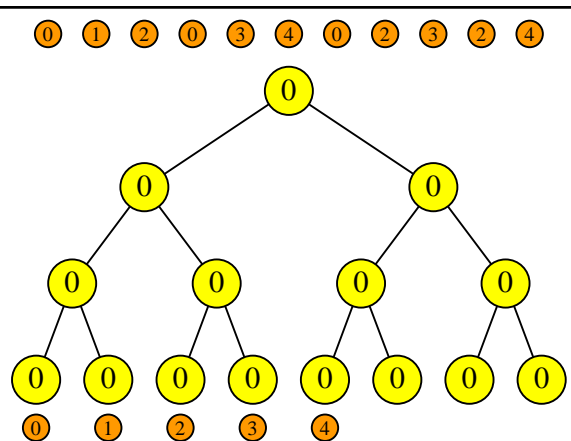
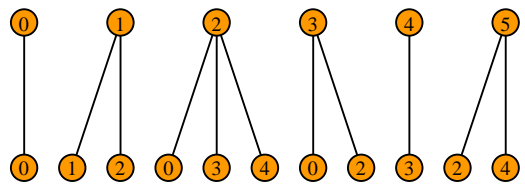
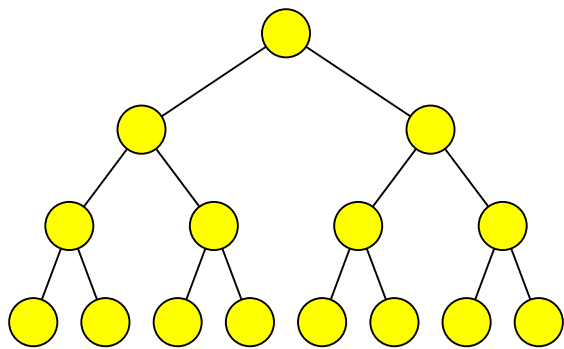
Insertion Sort
Laufzeit: $O(|C|)$

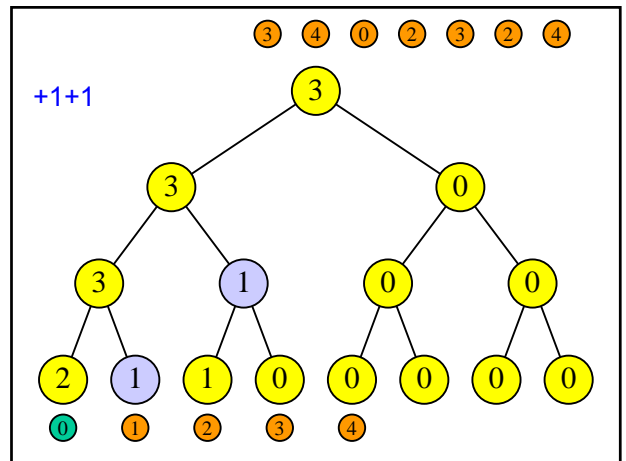
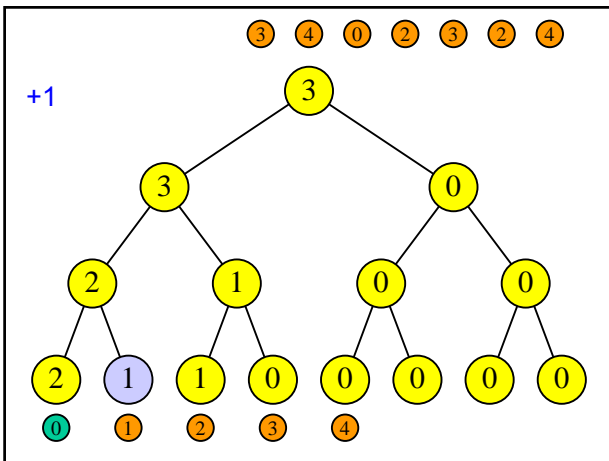
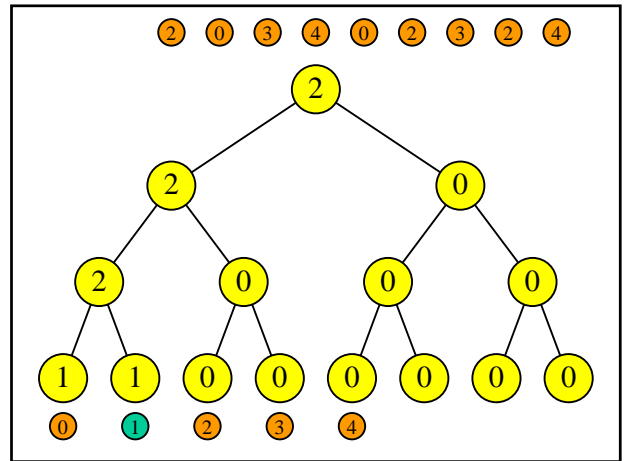
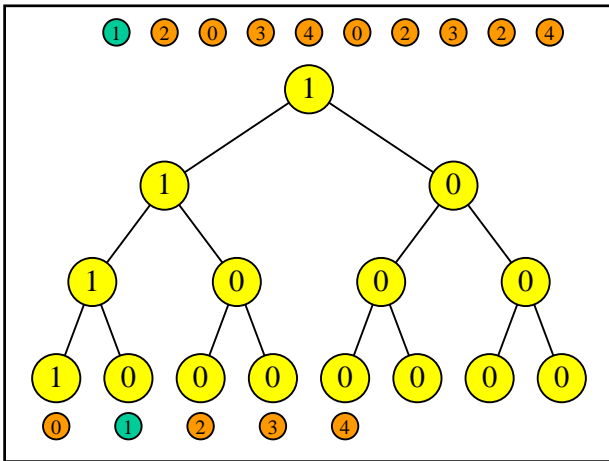
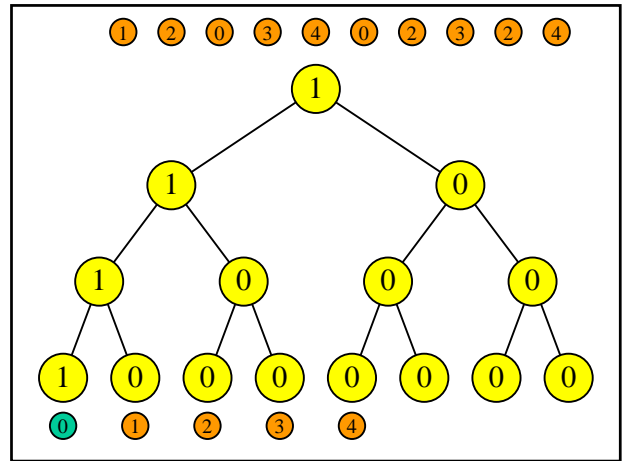
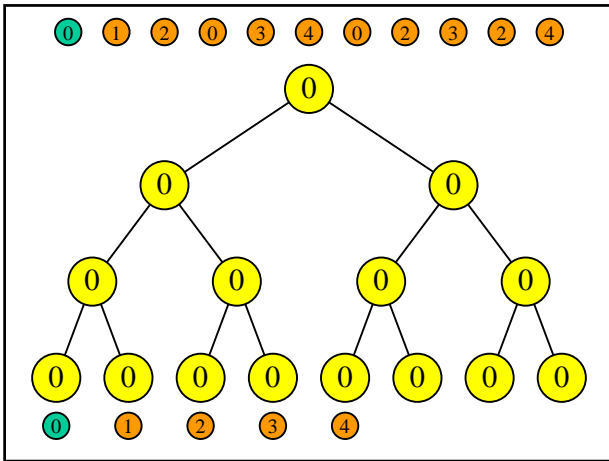


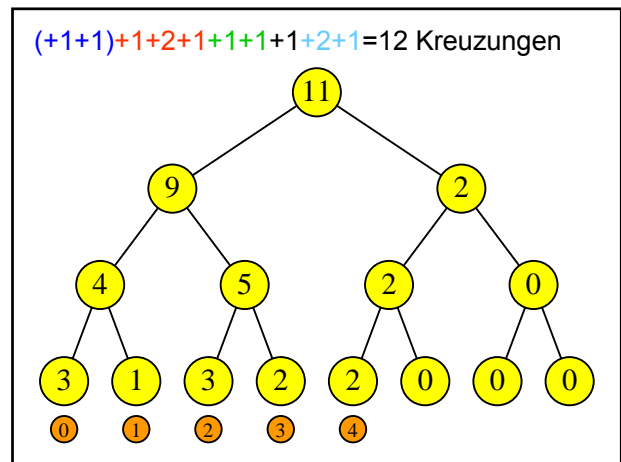
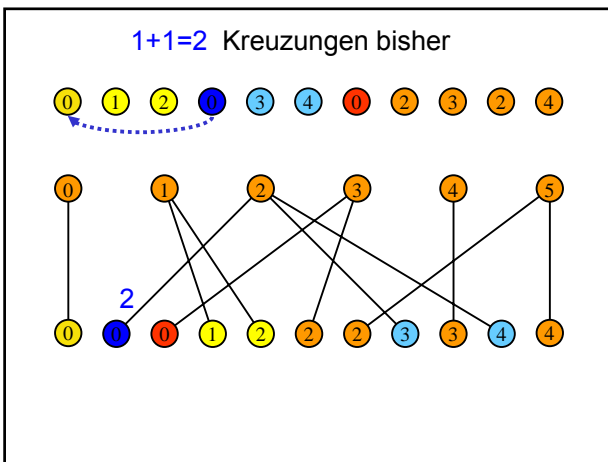
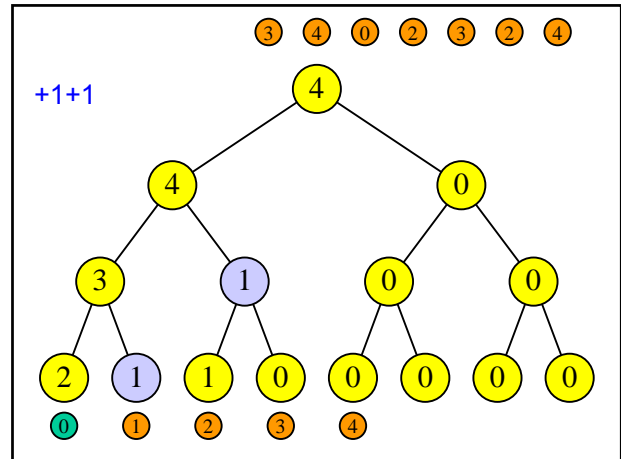
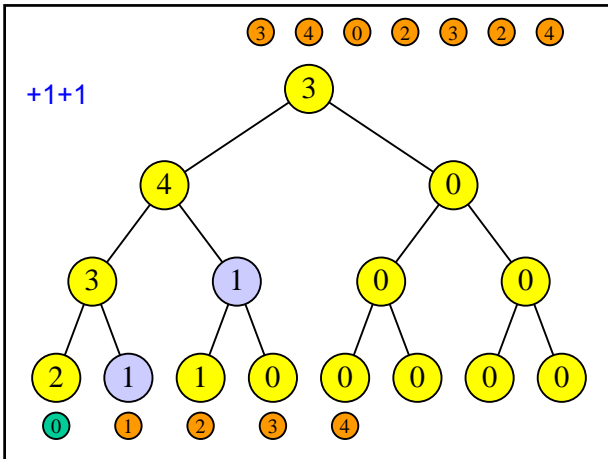
Alternative 1: Merge Sort: $O(|E| \log |E|)$

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

Accumulator Tree







Algorithmus CrossCount

1. Sortiere die untere Schicht lexikographisch → southsequence[0..r-1]
2. Initialisierung des Accumulator Tree
3. CrossCount = 0; /* Kreuzungszahl */
4. For (k=0; k<r; k++) { /* füge Kante k ein */
5. index = southsequence[k] + startindex
6. tree[index]++; /* Eintrag auf Blattebene */
7. While (index>0) { /* laufe Baum hoch */
8. If (index zu linkem Kind gehörig)
9. crosscount += tree[index + 1];
9. Wandere eine Schicht nach oben: neuer index
10. tree[index]++; /* Eintrag auf aktueller Schicht */
11. }
12. }

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] $O(|E| + |C|)$

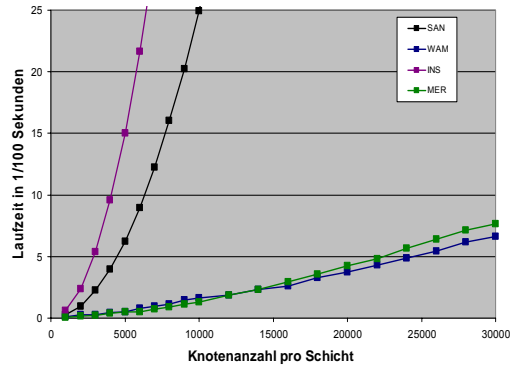
WAM Waddle & Malhotra [1999] $O(|E| \log |V|)$

INS Insertion Sort $O(|E| + |C|)$

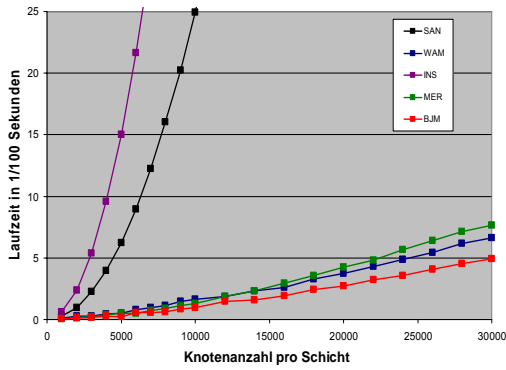
MER Merge Sort $O(|E| \log \text{run}(p))$

BJM Neuer Algorithmus $O(|E| \log |V_{\text{small}}|)$

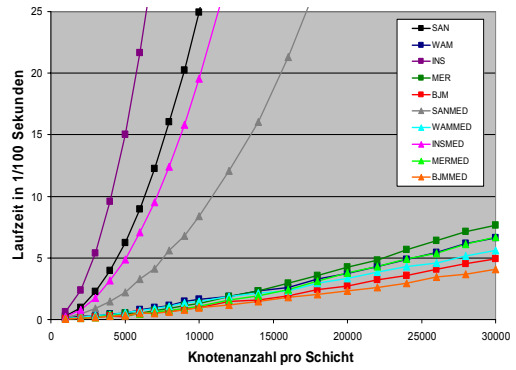
Laufzeit für dünne Graphen



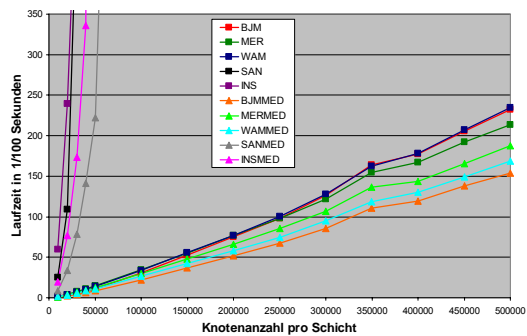
Laufzeit für dünne Graphen



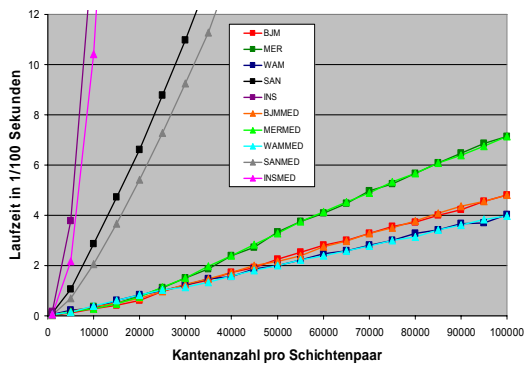
Laufzeit für dünne Graphen



Laufzeit für große dünne Graphen



Laufzeit für dichte Graphen



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

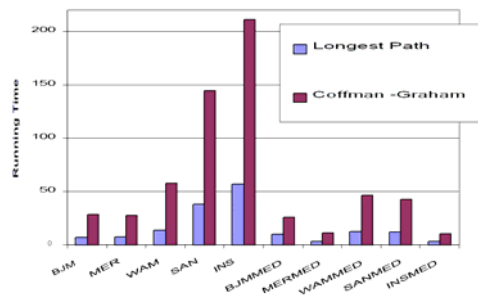
1 bis 3,278 obere Knoten, 142 durchschnittlich
1 bis 3,278 untere Knoten, 137 durchschnittlich
1 bis 3,276 Kanten, 141 durchschnittlich

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