

Kürzeste Wege

Distributed Systems
and Parallel Processing

Julian Vollmer (525904)
Philip Stewart (526571)

Agenda

- Motivation
- Shortest Path
- Realisierung
 - Multiprocessor
 - Singleprocessor
- Probleme
- Ergebnisse



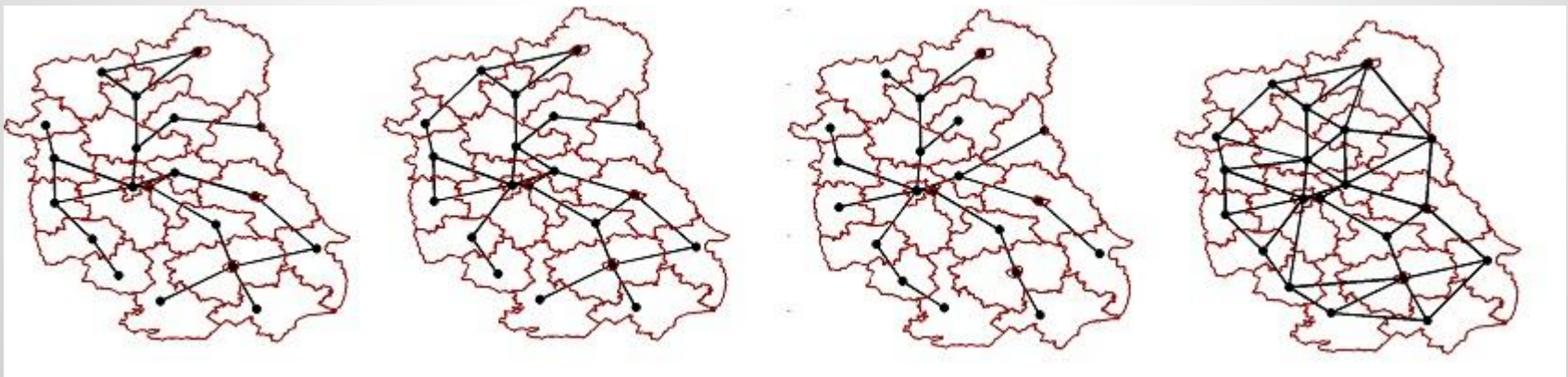
Motivation

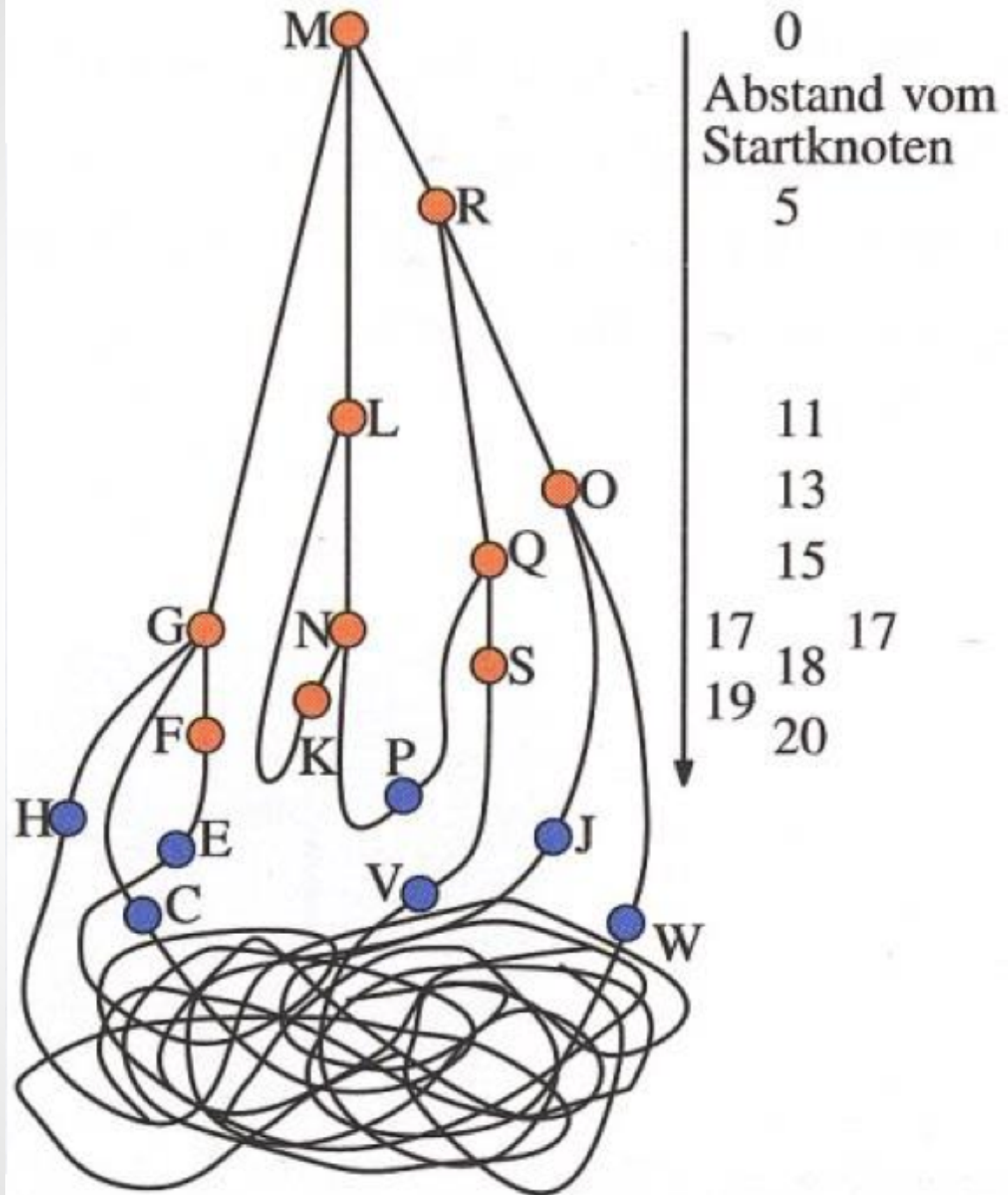
- Die Graphentheorie ist eine vielseitig angewandte Technik
- Dabei wird der Shortest Path Algorithmus oft verwendet
- In der Praxis sind Graphen mit Millionen Knoten üblich
- Parallelisierung (mit GPU) bietet hohe Rechenleistung bei geringem Preis



Shortest Path

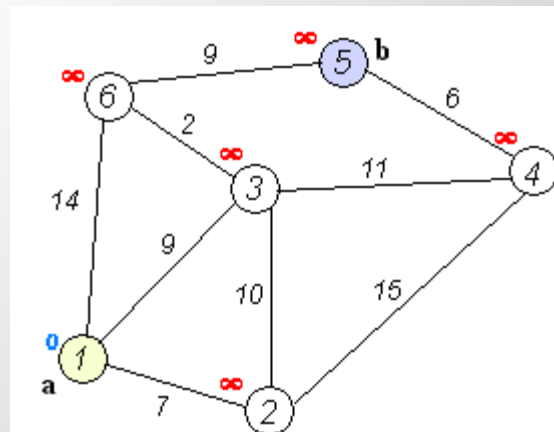
- Ausgangssituation:
 - Graph mit n Knoten
 - Knoten sind mit gewichteten Kanten verbunden
- Suche des kürzesten Weges
 - von einem Punkt zu allen anderen
 - von allen zu allen Punkten
- Distanz = Gewicht der Kanten

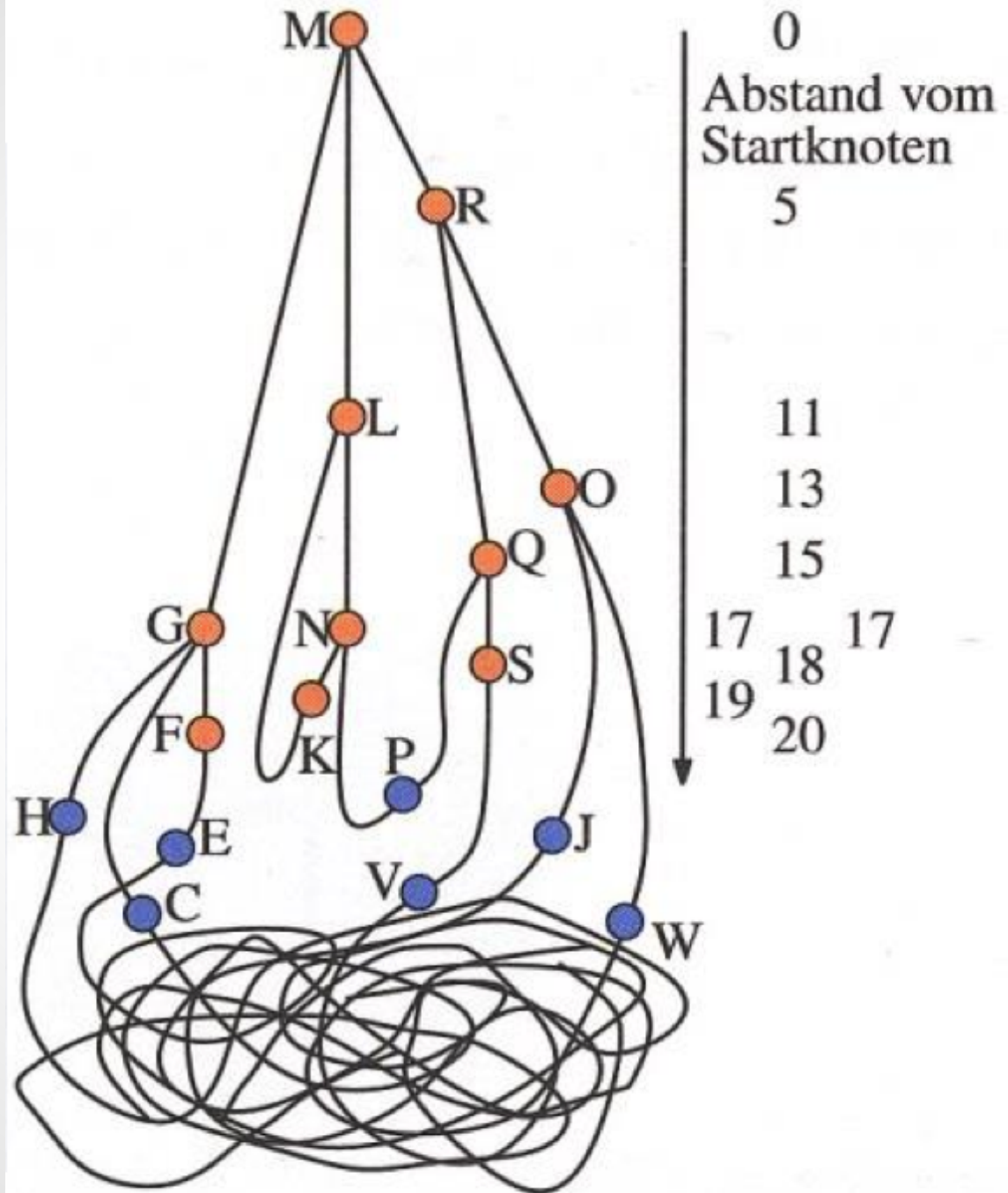




Dijkstra-Algorithmus

- Jeder Knoten erhält zusätzlich eine Distanz, einen Vorgänger und eine Markierung
- Solange es unmarkierte Knoten gibt:
 - Markiere den aktuellen Knoten
 - Berechne Distanz benachbarter unmarkierter Knoten
 - Wenn Wert kleiner als dessen gespeicherte Distanz:
Ersetze Distanz und setze aktuellen Knoten als Vorgänger





Parallelisierung Ansätze

- OpenMP
 - Analyse der Schleifen
 - einsetzen von passenden `#pragma`
 - definieren der bestmöglichen Anzahl an Threads
- OpenCL
 - Speicher allokieren
 - Context holen
 - ...



OpenMP

- Parallelisierung der Schleifen:
 - Testen aller Folgeknoten des aktuellen
 - Berechnung der kuerzesten Distanz
- `omp for schedule(guided)`
- Eergebniss: ca. 4 mal schneller
(bei 4 Threads)



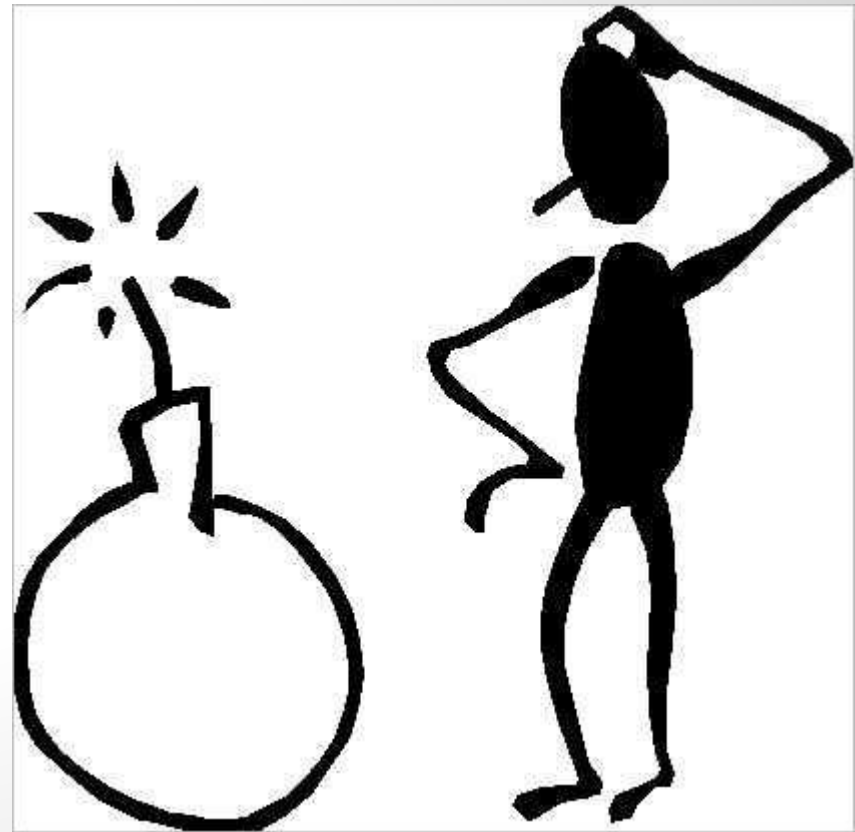
OpenCL

- Nicht lauffähig
 - Mac !
 - Linux !
- Nur auf NVIDIA ?



Probleme

- OpenCL
- Zeitmessung
 - `omp_get_wtime()`
 - `clock()`
- Parallelisierung
 - Deadlock ?



Ergebnis - Lines of Code

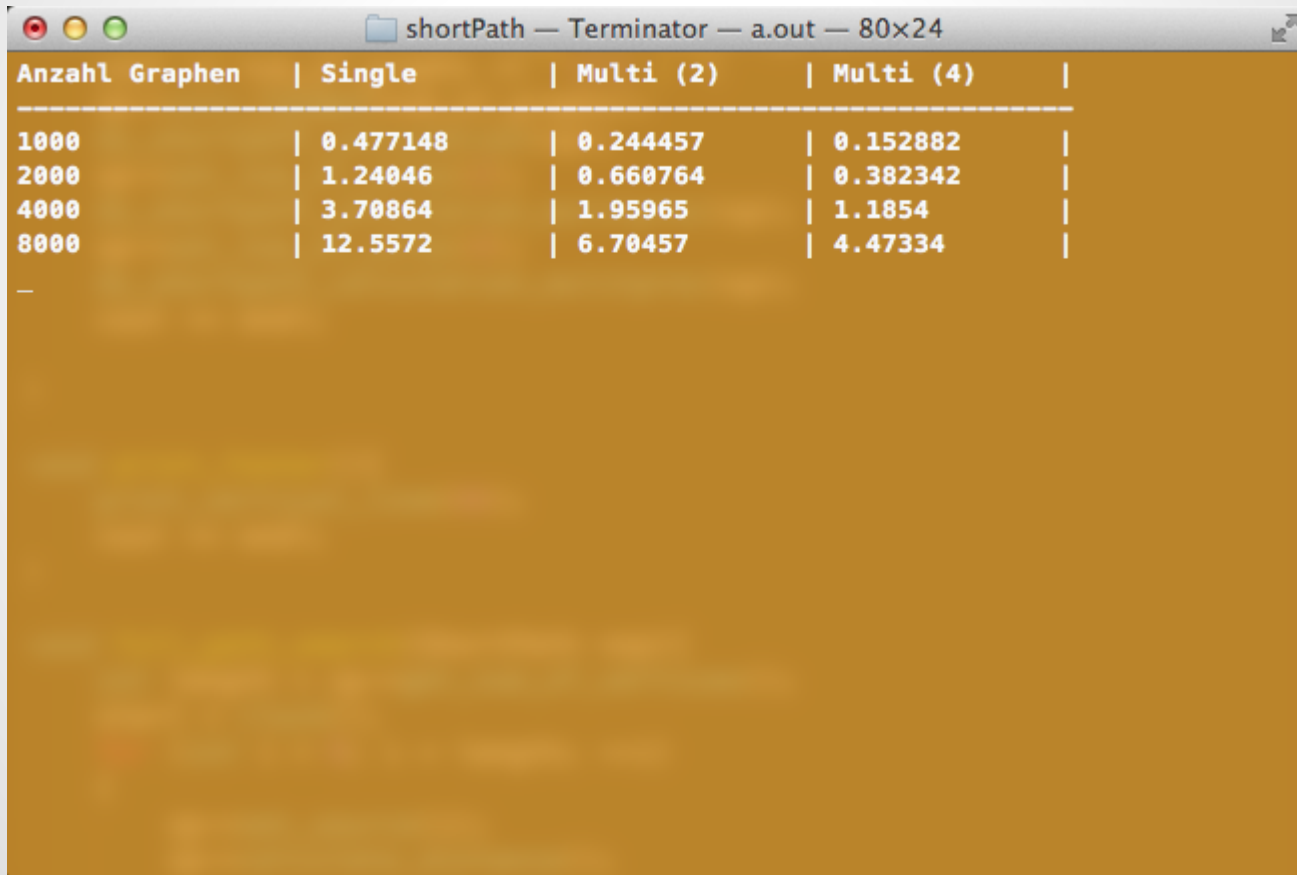
Language	files	blank	comment	code
C++	10	193	563	885
C/C++ Header	9	70	18	232
make	2	12	19	28
SUM:	21	275	600	1145

Ergebnis - UI

```
shortPath — Terminator — a.out — 80x24
#####
# Kuerzeste Wegeberechnung mit Multiprocessor #
#####

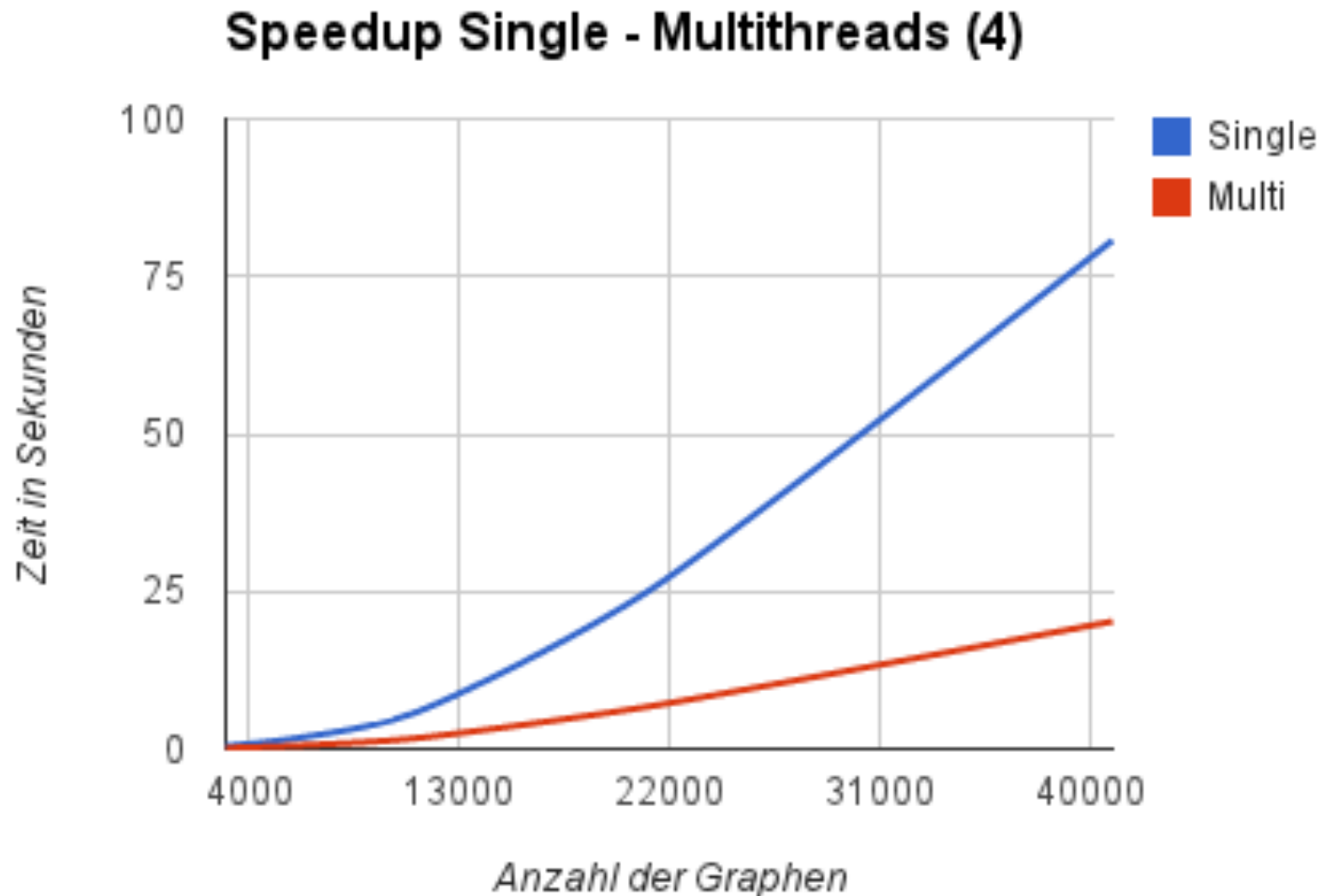
(1)Datei einlesen
(2)Kuerzte Wege (ohne Threads)
(3)Kuerzte Wege (mit Threads)
(4)Setze Anzahl der zu berechnenden Graphen
(5)Anzahl der Threads fuer Multicore
(6)Schreibe Adjazenzmatrix auf Terminal
(7)Schreibe Wegeberechnung auf Terminal
(8)Speichere Adjazenzmatrix
(9)Show Main Test!
Druecken Sie 0 zum Beeneden
Wie moechten Sie fortfahren [ 0 - 9 ] :
```

Ergebnis - UI



Anzahl Graphen	Single	Multi (2)	Multi (4)	
1000	0.477148	0.244457	0.152882	
2000	1.24046	0.660764	0.382342	
4000	3.70864	1.95965	1.1854	
8000	12.5572	6.70457	4.47334	

Ergebnis - Geschwindigkeit





Vielen Dank fuer die
Aufmerksamkeit!