1 Graphen

Um eine einheitliche Darstellung über alle Themen hinweg zu gewährleisten, wollen wir in diesem Seminar folgende Notation nutzen.

gerichtete Graphen Ein *gerichteter Graph* $G = (V, E)$ besteht aus einer Menge von *Knoten* $V = \{v_1, \ldots, v_n\}$ und einer Menge von gerichteten *Kanten* $E \subseteq V \times V$. Eine Kante $e = (u, v)$ von Knoten u zu Knoten v wird als Pfeil von u nach v gezeichnet.

Beachten Sie, dass bei dieser Definition keine parallelen Kanten in einem Graphen vorhanden sein können. Es kann aber Schleifen geben, also Kanten, die von einem Knoten u zu demselben Knoten u verlaufen.

ungerichtete Graphen Ein *ungerichteter Graph* $G = (V, E)$ besteht aus einer Menge von *Knoten* $V = \{v_1, \ldots, v_n\}$ und einer Menge von Kanten $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$, die als ungeordnete Paare repräsentiert werden. Eine Kante $e = \{u, v\}$ zwischen Knoten u und v wird als (gerade) Linie zwischen u und v gezeichnet.

Beachten Sie, dass bei dieser Definition keine parallelen Kanten und keine Schleifen in einem Graphen vorhanden sein können.

2 Themen

Da die ersten Vortragenden nur wenig Zeit für die Vorbereitung der Vorträge haben, sind die ersten sieben Themen bereits aus ALD bekannt. Bei diesen Themen geht es nicht darum, sich neues Wissen selbständig zu erarbeiten, sondern „nur“ darum, Stoff zu präsentieren und wissenschaftliches Schreiben zu üben.

Tiefensuche und deren Anwendung

1. Tiefensuche, Test auf Kreisfreiheit, topologische Sortierung
2. starke Zusammenhangskomponenten und Schnittpunkte

Kürzeste Wege Unterscheide single-source shortest path und all-pairs shortest path. Welche Algorithmen können auch negative Kantengewichte verarbeiten?

3. Dijkstra
4. Bellman/Ford, Floyd/Warshall
5. Bidirektionale Suche und $A^\star$-Algorithmus

Minimaler Spannbaum

6. nach Prim (inklusive Beweisidee der Korrektheit)
7. nach Kruskal (inklusive Union-Find-Datenstruktur)

Vorrangwarteschlangen Die Algorithmen von Prim (minimaler Spannbaum) und Dijkstra (kürzeste Wege) sowie der A*-Algorithmus benötigen eine Priority-Queue als Datentyp. Hier sollen verschiedene Implementierungen dieses Datentyps untersucht werden.

8. Linksbäume
9. Binomial-Queues
10. Fibonacci-Heaps
11. Amortisierte Laufzeitanalysen (speziell zu Fibonacci-Heaps)

Netzwerkfluss Welche Menge an Wasser kann die Kanalisation in Krefeld verkraften, ohne das es zu Überschwemmungen kommt? Über welche Wege können wir Fans am besten zum Stadion leiten? Solche Fragestellungen können mit den folgenden Algorithmen beantwortet werden.

12. Max-Flow Min-Cut Theorem, Idee von Ford/Folkerson
13. Idee von Edmond/Karp, Idee von Dinic
14. Push-Relabel-Algorithmus nach Goldberg und Tarjan

Maximum Matching Wie ordne ich die Arbeitskräfte einer Firma am besten denjenigen Tätigkeiten zu, für die die Arbeitskräfte am besten geeignet sind? Wie werden Studierende möglichst gerecht auf Wahlfächer verteilt?

15. auf bipartiten Graphen
16. auf allgemeinen Graphen

Kombinatorische Spiele als Suchprobleme in Graphen Bei Zwei-Personen-Spielen wie Schach, Dame, 4-Gewinnt, Othello oder auch bei Tic-Tac-Toe suchen wir in einer konkreten Spielsituation den bestmöglichen Zug. Manchmal kommt, wie bei Backgammon, auch noch der Zufall z.B. in Form von Würfeln ins Spiel.

17. Minimax-Strategie
18. Alpha/Beta-Pruning und Zufallselemente
19. Reinforcement Learning (Kombination mit neuronalen Netzen möglich)
20. Monte-Carlo Tree Search (Kombination mit parallelen Algorithmen möglich)

NP-vollständige Probleme Hier soll von den folgenden Problemen gezeigt werden, dass sie NP-vollständig sind. Dazu ist eine polynomielle Reduktion von einem bekannten Problem zu zeigen. Beim historisch ersten NP-vollständigen Problem SAT ist eine andere Beweisführung nötig.

21. SAT (satisfiability problem)
22. Knotenfärbung, Clique, Independet Set, Clique Cover
23. Hamiltonkreis (gerichtet/ungerichtet)
24. Vertex Cover, 3D-Matching

beschränkte Eingaben Schwere Graphenprobleme wie Clique, Cliquen-Überdeckung, Independent Set oder Färbung sind auf speziellen Graphklassen oft gar nicht so schwer wie im allgemeinen Fall. Oft finden wir effiziente Algorithmen für schwere Probleme, wenn wir die Menge der zulässigen Eingaben geeignet beschränken.

25. Co-Graphen
26. chordale Graphen
27. Vergleichbarkeitsgraphen
28. baumweite-beschränkte Graphen

Lösung schwerer Graphenprobleme Ist eine Beschränkung der Eingaben in der Praxis nicht möglich, so sind wir oft schon zufrieden, wenn möglichst gute Lösungen in akzeptabler Zeit gefunden werden.

29. Approximative Algorithmen
30. Branch-and-Cut
31. Lokale Suche und Simulated Annealing
32. Fixed-Parameter Tractable

Wenn Sie gerne an einem anderen Thema arbeiten möchten, sprechen Sie mich an. Diese Liste von Themen ist nicht abschließend sondern als Vorschlag zu verstehen.

3 Literatur

Graphentheorie

- Diestel: Graphentheorie. Springer Verlag.
- Krumke, Noltemeier: Graphentheoretische Konzepte und Algorithmen. Springer Vieweg
- Jungnickel: Graphen, Netzwerke und Algorithmen. Springer Verlag.
- Harary: Graph Theory. Addison-Wesley Publishing Company.
- Golumbic: Algorithmic Graph Theory and Perfect Graphs. Academic Press.
- Gurski, Rothe, Rothe, Wanke: Exakte Algorithmen für schwere Graphenprobleme. Springer Verlag.

Algorithmen und Datenstrukturen

- Ottmann, Widmayer: Algorithmen und Datenstrukturen. Springer Spektrum.
- Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms. MIT Press.
- Aho, Hopcroft, Ullman: Datastructures and Algorithms. Addison-Wesley.
- Jon Kleinberg, Éva Tardos: Algorithm Design. Pearson-Addison Wesley[1]
- Schöning: Algorithmen – kurz gefasst: Spektrum Akademischer Verlag.
- Horowitz, Sahni: Algorithmen. Springer.

Künstliche Intelligenz

- Russel, Norvig: Artificial Intelligence – A Modern Approach. Prentice Hall.
- Luger: Künstliche Intelligenz. Pearson Studium.
- Sutton, Barto: Reinforcement Learning: An Introduction (second edition). MIT Press.

Theoretische Informatik[2]

- Garey, Johnson: Computers and Intractability. W.H. Freeman and Company.
- Hopcroft, Ullman: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Addison-Wesley.
- Papadimitriou: Computational Complexity. Addison-Wesley.

[1] Folien zu dem Buch gibt es hier: `https://www.cs.princeton.edu/~wayne/kleinberg-tardos`

[2] Eine gute Übersicht über schwere Optimierungsprobleme gibt es auf dieser Web-Seite: `https://www.csc.kth.se/tcs/compendium/`