Universität des Saarlandes
FR 6.2 Informatik

Prof. Dr. Kurt Mehlhorn, Dr. Konstantinos Panagiotou **WiSe 2011/2012**

Übungen zu Computational Thinking

http://www.mpi-inf.mpg.de/departments/d1/teaching/ws11/ct/

Blatt 7 Abgabeschluss: 19.12.11 16:00

Regeln: Bis zum Semesterende müssen mindestens 42% der maximal erreichbaren Punkte aller Übungszettel erworben werden.
Programmcode ist elektronisch per E-Mail abzugeben. Zusätzlich müssen die Ausgaben einer exemplarischen Programmausführung mitgeliefert werden.

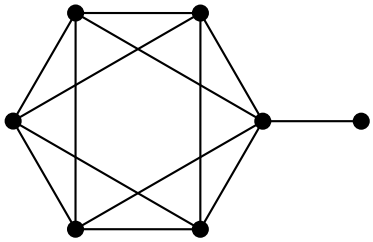
Bleistiftaufgaben

Aufgabe 1 (*15 Punkte*) Wir kennen mit Dijkstra's Algorithmus ein effizientes Verfahren um kürzeste Wege zu finden. Niemand kennt einen ähnlich effizienten Algorithmus um einen längsten Weg, also einen Weg der möglichst viele Knoten zwischen Start und Ziel hat und keinen Knoten zweimal besucht, zu berechnen.
Argumentieren Sie, warum man längste Wege nicht auf eine ähnliche Art wie kürzeste Wege finden kann. Was unterscheidet längste Wege fundamental von kürzesten?

Aufgabe 2 (*5 Punkte*) In der bekannten Science-Fiction Geschichte „The Nine Billion Names Of God" von Arthur Clarke (1953) haben tibetanische Mönche es sich zur Aufgabe gemacht, alle Namen Gottes aufzuschreiben. Ihrem Glauben nach geht mit Vollendung dieser Aufgabe das Ende der Welt einher. Ihr der Geschichte werden sie des Aufschreibens müde und bitten Computerexperten aus Amerika um mechanische Hilfe. Selbst in den Fünfzigern war es für Computer nicht so schwierig neun Milliarden Namen aufzulisten. Das Ende können Sie sich denken.
Nehmen Sie an, dass Gottes Namen in einem Alphabet mit 32 Buchstaben aufgeschrieben werden. Um die Namen aufzulisten wandeln wir die Erde in Computronium um und haben so einen Computer der etwa 10^{75} Namen pro Sekunde auflisten kann. Wie lang müssen Gottes Namen sein, damit wir bis zum Hitzetod des Universums in circa 10^{100} Jahren vor den Mönchen sicher sind?

Aufgabe 3 (*15 Punkte*) Beschreiben Sie informell das Programm einer Turingmaschine, die zu einer binärkodierten Zahl auf dem Band Eins dazuaddiert. Bei der Eingabe steht das wichtigste Bit ganz rechts, das Ende der Zahl wird durch ein `#` markiert. Die Zahl 6 ist folglich als `011#` kodiert. Ihre Beschreibung muss detailliert genug sein, um daraus ohne Weiteres das formale Programm der Turingmaschine ableiten zu können.

Aufgabe 4 (*10 Punkte*) Betrachten wir Graphen auf $n+1$ Knoten, die aus einem Kreis auf den Knoten 1 bis n bestehen, an dem noch ein weiterer Knoten hängt. Im Kreis hat Knoten i eine Kante zu den Knoten $i-2, i-1, i+1, i+2$, wobei $0-1=n$ definiert ist. Die Graphen sehen also alle etwa so aus:



Argumentieren Sie, dass der Backtracking-Algorithmus für das Hamiltonkreisproblem auf diesem Graphen mindestens $2^{n/2}$ Pfade betrachtet.

Python

Aufgabe 5 (*Variable Punkte*) Auf der Webseite verlinkt finden Sie eine Python-Datei und eine Reihe von Graphen, die mit Hilfe des Codes in der Python Datei eingelesen werden können (es steht Ihnen frei etwas selbstgeschriebenes zu benutzen). Die Knoten der Graphen sind Städte, sie sind also paarweise über eine gewichtete Kante (Luftlinie) verbunden (die Graphen sind *vollständig*).
Sie starten in Stadt 1 und wollen zwielichtige Geschäfte in allen anderen Städten betreiben, also jede Stadt besuchen. Um den Behörden zu entgehen, ist es Ihnen nicht möglich eine Stadt doppelt zu besuchen. Weil Zeit Geld ist, möchten eine möglichst kurze Tour finden, die jede Stadt genau einmal besucht und in Stadt 1 startet und endet.
Schreiben Sie eine Funktion, die eine kürzeste Rundtour in einem solchen Graphen findet und erläutern Sie kurz, wieso Ihr Algorithmus funktioniert. Das Programm soll die Länge sowie die Tour[1] ausgeben, also zum Beispiel für `22.tsp`

```
7013
[1, 8, 18, 4, 22, 17, 2, 3, 16, 21, 20, 19, 10, 9, 11, 5, 15, 6, 7, 12, 13, 14]
```

Ihr Programm muss nachweisbar für alle Graphen dieser Art eine kürzeste Tour finden und darf nicht schummeln, also zum Beispiel vorberechnete Lösungen verwenden.
Die Punkte für diese Aufgabe werden nach der Anzahl der Knoten im größten der Beispielgraphen bestimmt, für den Ihr Programm in weniger als 15 Minuten die korrekte Tour bestimmt. Sie bestimmen sich aus dieser Tabelle:

Knoten	10	11	12	13	14	16	22	96	137	202	229	431	535	666
Punkte	7	10	12	15	20	30	40	75	150	250	300	400	500	600

Sie erhalten eine Million Dollar, wenn Sie beweisen können, dass ihr Programm für jeden Graphen nur eine polynomielle Anzahl an Schritten braucht.

[1] Es kann mehrere Touren mit dem selben Gewicht geben, eine zu finden genügt.