Aufgabe 1 – *Hamiltonkreise (In-Class Exercise)*

Sei $G = (V, E)$ ein Graph mit n Knoten, $V = \{v_1, \ldots, v_n\}$. Sei W die Menge aller geschlossenen Wege von v_1 nach v_1 der Länge n in G. Für $2 \leq i \leq n$ sei $A_i \subseteq W$ die Menge der Wege in W, die nicht den Knoten v_i besuchen. Ausserdem sei $A := \bigcup_{i=2}^{n} A_i$.

(a) Sei H die Menge der Hamiltonkreise in V mit Start-/Endpunkt v_1. In welcher Beziehung stehen die Mengen H, W, und A zueinander? Welche Gleichung erfüllen demzufolge die Grössen $|H|$, $|W|$, und $|A|$?

(b) Für jede Menge $S \subseteq V$ mit $v_1 \notin S$ definieren wir W_S als die Menge aller Wege in W, die keine Knoten in S besuchen. Insbesondere ist $A_i = W_{\{v_i\}}$. Zeigen Sie:

$$|A| = \sum_{\emptyset \neq S \subseteq V, v_1 \notin S} (-1)^{|S|+1} |W_S|.$$

(c) Geben Sie einen (effizienten) Algorithmus an, der als Input G, v_1 und S bekommt, und der $|W_S|$ berechnet. Welche Laufzeit hat Ihr Algorithmus?
Hinweis: Benutzen Sie Algorithmen als Subroutinen, die Sie aus diesem oder letztem Semester kennen. Von solchen aus den Vorlesungen bekannten Routinen brauchen Sie natürlich weder Implementierung noch Korrektheit neu zu diskutieren. Achten Sie jedoch wie immer darauf, dass Sie genau angeben, was Sie der Subroutine als Input geben.

(d) Entwerfen Sie einen Algorithmus, der als Input G und v_1 bekommt, und der $|A|$ berechnet. Der Algorithmus darf exponentielle Laufzeit haben (jedoch höchstens um einen polynomiellen Faktor schlechter als 2^n), aber er soll nur polynomiell viel Speicherplatz benötigen.

(e) Entwerfen Sie einen Algorithmus, der als Input G bekommt, und der die Zahl der Hamiltonkreise in G bestimmt. Der Algorithmus soll dieselben Laufzeit- und Speicheranforderungen erfüllen wie in (d).

(f) Entwerfen Sie einen Algorithmus, der entscheidet, ob es in einem Graphen G einen Hamiltonkreis gibt, und der dieselben Laufzeit- und Speicheranforderungen erfüllt wie in (d).

Lösung zu Aufgabe 1 – *Hamiltonkreise*

(a) Wir betrachten die Menge der Hamiltonkreise H als gerichtete Hamiltonkreise von v_1 nach v_1. Dann ist W die disjunkte Vereinigung von H und A. Insbesondere gilt also $|H| = |W| - |A|$.
Beweis: Da alle Wege $w \in A$ mindestens einen Knoten nicht besuchen, folgt sofort dass $H \cap A = \emptyset$. Desweiteren sind alle Elemente von $H \cup A$ geschlossene Wege (Hamiltonkreise sind auch Wege) von v_1 nach v_1 der Länge n, und somit ist $H \cup A \subseteq W$. Wir müssen also nur $W \subseteq H \cup A$ beweisen. Dazu betrachten wir einen beliebigen Weg $w \in W$. Entweder er besucht mindestens einen Knoten v_i nicht, dann gilt $w \in A_i \subseteq A$. Oder er besucht alle Knoten. Da w ein geschlossener Weg der Länge n ist, muss er in diesem Fall jeden Knoten sogar *genau einmal* besuchen. Also ist w ein Hamiltonkreis, und es gilt $w \in H$. Damit haben wir $W \subseteq H \cup A$ gezeigt, und zusammen mit der anderen Inklusion folgt $W = H \cup A$.

(b) Nach Inklusion-Exklusion-Prinzip haben wir

$$|A| = \Big|\bigcup_{i=2}^{n} A_i\Big| = \sum_{\emptyset \neq S \subseteq \{2,\ldots,n\}} (-1)^{|S|+1} \Big|\bigcap_{i \in S} A_i\Big|$$

Die Formel folgt also sofort, wenn wir zeigen, dass $W_S = \bigcap_{i \in S} A_i$.
Beweis von $W_S = \bigcap_{i \in S} A_i$*:* Die Wege $w \in \bigcap_{i \in S} A_i$ sind Wege der Länge n und besuchen keinen Knoten in S, also ist jeder Weg $w \in \bigcap_{i \in S} A_i$ auch in W_S und deshalb gilt $\bigcap_{i \in S} A_i \subseteq W_S$. Andererseits gilt für alle Wege $w \in W_S$, die S nicht besuchen, dass w insebesondere auch keinen einzelnen Knoten in S besucht. Das heisst, dass $w \in A_i$ für alle $i \in S$. Folglich gilt auch die Inklusion $W_S \subseteq \bigcap_{i \in S} A_i$ und somit folgt die Beahuptung $W_S = \bigcap_{i \in S} A_i$.

Bemerkung: Bei geeigneter Definition eines Schnittes über einer leeren Menge würde der Ausdruck $\bigcap_{i\in\emptyset} A_i$ der Gesamtmenge W entsprechen. Würden wir als Summand also auch $S = \emptyset$ zulassen, so ergäbe die obige Summe $|A| - |W| = -|H|$ und wir könnten somit $|H|$ direkt berechnen.

(c) Gesucht ist die Zahl der Wege der Länge n von v_1 nach v_1 im induzierten Graphen $G_S := G[V\backslash S]$. Aus dem letzten Semester wissen wir, dass wir diese Zahl aus dem Eintrag links oben von A_S^n ablesen können, wobei A_S die Adjazenzmatrix von G_S ist. Aus dem letzten Semester kennen wir den Algorithmus von Strassen, der die Matrizenmultiplikation in Zeit $O((n - |S|)^{2.807..})$ ausführt, und wir wissen ebenfalls aus dem letzten Semester, dass wir A_S^n durch $O(\log n)$ viele Matrizenmultiplikationen berechnen können (iteriertes Quadrieren). Also hat der Algorithmus Laufzeit $O((n-|S|)^{2.807..} \cdot \log n) \subseteq O(n^{2.807..} \cdot \log n)$. (Die Laufzeit zur Berechnung von G_S bzw. A_S ist $O((n - |S|)^2)$ und fällt daher nicht ins Gewicht.)

Anmerkung: Es sind auch Lösungen aus dem letzten Semester bekannt, die etwas weniger effizient, aber immer noch polynomielle Laufzeit haben (z.B. DP-Lösungen). Umgekehrt gibt es auch Verfahren zur Matrizenmulitplikation, die theoretisch schneller sind als Strassens Algorithmus, wenn auch nicht in der Praxis. Die beste theoretisch bekannte Laufzeit ist $O((n - |S|)^{2.37..})$.

(d) Für alle 2^n Teilmengen $S \subseteq [n]$ berechnen wir W_S und addieren es, multipliziert mit $(-1)^{|S|+1}$, zu einer laufenden Summe A'. Am Ende geben wir A' aus.
Korrektheit: Aus Teilaufgabe (b) wissen wir, dass $|A|$ genau gleich dieser Summe ist. Der Speicher, den wir brauchen, ist polynomiell, weil wir für das Berechnen eines einzelnen W_S nur Platz $O((n - |S|)^2) \subseteq O(n^2)$ brauchen (mit dem Algorithmus aus (c)). Nachdem wir einen Summanden aufaddiert haben, können wir den reservierten Speicher wieder freigeben oder neu benutzen.
(Genau genommen müssen wir auch prüfen, dass wir A' und die Zwischensummen mit polynomiellem Speicher abspeichern können. A' ist maximal die Summe aller W_S, also maximal $2^n \cdot n^n$. Also benötigt A' höchstens Speicherplatz $2n \cdot log(n)$)
Laufzeit: Wir durchlaufen in unserem Algorithmus jede Teilmenge S, wovon es höchstens 2^n gibt. In jeder Iteration durchlaufen wir den Algorithmus von (b), also nur eine polynomielle Laufzeit. Also hat der ganze Algorithmus insgesamt eine Laufzeit von polynomieller Faktor mal 2^n.

(e) Wir berechnen $|W|$ mit der Adjazenzmatrix A von G wie wir es in (c) beschreiben haben. Mit dem Algorithmus von (d) berechnen wir dann $|A|$, und wegen Teilaufgabe (a) können wir mit $|H| = |W| - |A|$ die Anzahl Hamiltonkreise in G bestimmen.
Korrektheit: Folgt direkt aus dem Algorithmus in (d) und der Aussage in (a).
Laufzeit: Wie in (d), da $|W|$ in polynomieller Zeit berechnet werden kann.
Speicher: Um $|W|$ (und die einzelnen $|W_S|$) zu berechnen, brauchen wir (wie in (d) beschrieben) einen Speicherplatz der Grösse $O(n^2)$.

(f) Um zu entscheiden, ob es einen Hamiltonkreis gibt, reicht es aus, den Algorithmus aus (e) zu nehmen und zu testen, ob die ausgegebene Zahl 0 ist. (Laufzeit/Speicherbedarf wie in (d).)