Aufgabensammlung ADS-Repetitorium WS 25/26

Graphen – Graphenalgorithmen – Greedy-Algorithmen

Aufgabe 1: Modellierung als Graphen

Viele algorithmische Fragestellungen können als Graphen modelliert werden. Bearbeiten Sie folgende Punkte für jede Fragestellung:

1. Geben Sie einen Algorithmus in Worten an, der die Fragestellung als Graph modelliert. Achten Sie auf eine präzise Definition der Knoten- und Kantenmenge.
2. Mit welchem Graph-Algorithmus kann die Fragestellung auf dem Graph gelöst werden? Wie müssen die Algorithmen gegebenfalls modifiziert werden?
3. Von welchen Parametern hängt die Laufzeit ab? Können Sie die Laufzeit genau angeben?

Einige der Probleme lassen sich eventuell auch einfacher ohne Graph lösen. In dieser Aufgabe sollen sie aber explizit den Umgang mit Graphen üben.

(a) Sie arbeiten im Marketing einer Firma, die Kameras herstellt. Auf einer Veranstaltung, die von n Menschen besucht wird, bieten Sie folgende Werbekampagne an:
Sie verteilen m Einwegkameras, mit denen man je genau ein Bild schießen kann. Die Kameras sollen dazu benutzt werden, Twofies (= Bilder, auf denen genau zwei Personen abgebildet sind) zu schießen. Die Person, die am Ende der Veranstaltung mit den meisten anderen Menschen auf Twofies abgebildet ist, hat gewonnen.
Wie können Sie die Person ermitteln, die gewinnt?

(b) Das Kommunalunternehmen eines Landkreises mit n Gemeinden möchte jede Gemeinde an ein Radwegnetz anbinden. Um Kosten zu sparen, sollen die Radwege nur an den *breitesten* Straßen im Landkreis angelegt werden und Rundfahrten zwischen den Gemeinden sollen nicht möglich sein.

(c) Ein Software-Projekt besteht aus n Modulen. Jedes Modul kann von anderen Modulen abhängig sein. Ein Modul kann nur gestartet werden, falls alle Module, von denen das Modul abhängt, bereits gestartet wurden. Gesucht ist also die Reihenfolge, in der die Module gestartet werden können.

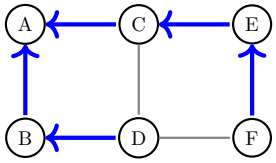
Aufgabe 2: Graphen-Cliquen

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $C \subseteq V$ heißt *Clique*, wenn jedes Knotenpaar $e, d \in E$ durch eine Kante verbunden ist.
Schreiben Sie einen Algorithmus, der für einen Graphen $G = (V, E)$ und eine Teilmenge $C \subseteq E$ prüft, ob C eine Clique in G ist.

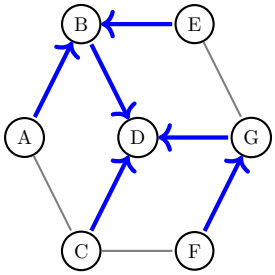
Aufgabe 3: Korrektheit von Breitensuchen erkennen

In folgenden Graphen sind die π-Zeiger der Knoten nach einer Breitensuche in blau eingezeichnet. Geben Sie an, ob die folgenden Graphen durch eine Breitensuche entstanden sein können. Begründen Sie Ihre Antworten.

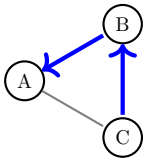
(a)



(b)



(c)



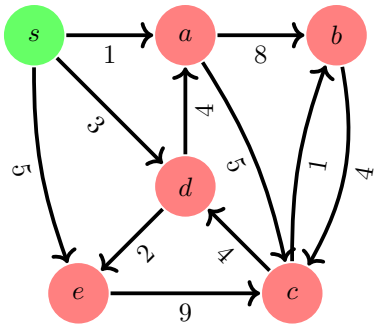


Abbildung 1: Beispiel-Graph für Dijkstras-Algorithmus.

Aufgabe 4: Tiefensuche iterativ

Sie haben in der Vorlesung zwei Durchlaufstrategien für Graphen kennengelernt: die Breitensuche und die Tiefensuche. Während die Breitensuche „iterativ“ mit einer Schlange funktioniert, benutzt die Tiefensuche Rekursion, um den Graphen zu explorieren. Da rekursive Algorithmen manchmal langsamer sind und außerdem durch die Größe der maximalen Rekursionstiefe limitiert ist, möchte man gelegentlich die Tiefensuche ebenfalls „iterativ“ implementieren.
Hinweis: Sie müssen die Farben nicht setzen und können eine einzige flag $u.visited$ für einen Knoten u benutzen.

(a) Implementieren Sie die Tiefensuche mit Hilfe eines Stapels in Pseudocode. Ignorieren Sie hierfür die *time stamps* $u.d$ und $u.f$ eines Knotens u. Die Entdeckungsreihenfolge muss nicht identisch mit der rekursiven Version der Tiefensuche sein!

(b) Können Sie Ihren Pseudocode so zu modifizieren, dass Sie auch die *time stamps* korrekt setzen?

Aufgabe 5: Dijkstra's Algorithmus

Gegeben sei der Graph in Abbildung 1. Bearbeiten Sie auf diesem Graph die folgenden Aufgaben.

(a) Führen Sie Dijkstra's Algorithmus mit Startknoten s aus. Erstellen Sie dazu eine Tabelle mit drei Spalten: *Iteration*, *Schwarze Knoten*, *Graue Knoten*, zu der sie nach jeder Iteration der While-Schleife eine Zeile hinzufügen. Schreiben Sie hinter jeden Knoten in Klammern die aktuelle Distanz und den Vorgänger-Knoten.

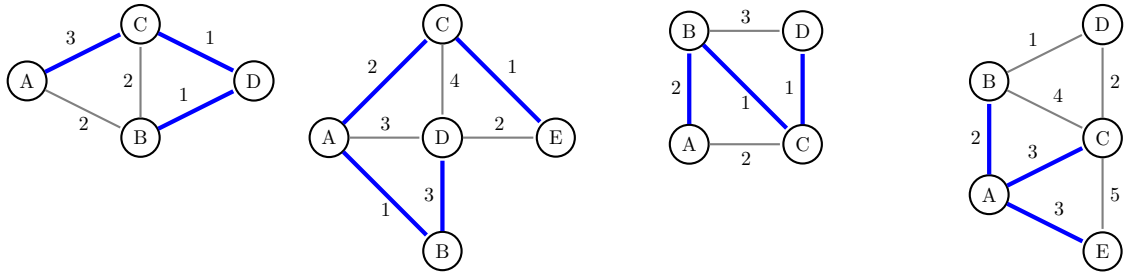
(b) Zeichnen Sie den entstandenen Kürzeste-Wege-Baum.

(c) Seien nun negative Kantengewichte erlaubt. Verändern Sie den gegeben Graphen, sodass Dijkstra nach Beendigung kein richtiges Ergebnis liefert, obwohl der kürzeste Weg definiert ist. Warum kann dies nicht behoben werden, indem wir das minimale Kantengewicht $g < 0$ identifizieren und alle Gewichte um $|g|$ erhöhen? Dann wären alle Kantengewichte wieder positiv und wir könnten Dijkstra verwenden. Begründen Sie allgemein, warum diese Vorgehensweise nicht korrekt ist.
Vorschau: Morgen werden wir uns im Rahmen der Dynamischen Programmierung mit einem Algorithmus beschäftigen, der mit negativen Kanten zurecht kommt.

Aufgabe 6: Erkennen von minimalen Spannbäumen

Entscheiden Sie für die folgenden blau markierten Teilgraphen, ob diese minimale Spannbäume sind. Geben Sie für jeden Graphen, dessen blau markierter Teilgraph kein minimaler Spannbaum ist, einen minimalen Spannbaum an und begründen Sie Ihre Antworten.

(a) (b) (c) (d)



Geben Sie für jedes der folgenden Probleme einen Greedy-Algorithmus an. Denken Sie daran, die folgenden Punkte zu beweisen:

Aufgabe 7: Greedy-Algorithmen

- Beweisen Sie, dass die Lösung des Greedy-Algorithmus zulässig ist.
- Beweisen Sie, dass die Lösung des Greedy-Algorithmus optimal ist, oder geben Sie ein Beispiel an, in dem der Greedy-Algorithmus nicht die optimale Lösung findet.
- Geben Sie die Laufzeit an.

(a) Sei $G = (V, E)$ ein ungerichteter Graph. Gesucht ist eine möglichst große Teilmenge U der Knoten V, sodass für keine zwei Knoten $u, v \in U$ die Kante uv in E ist.

(b) Gegeben ein Baum $T = (V, E)$, in dem jeder Knoten v einen positiven Wert $v.w$ hat. Gesucht ist ein Pfad von der Wurzel zu einem Blatt, sodass die Summe der auf dem Weg liegenden Knoten möglichst groß ist.

(c) Gegeben sind n positive Zahlen in einem Feld. Selektieren Sie $n/2$ Zahlen so, dass dass deren Summe möglichst klein ist.

(d) Gegeben eine Liste von Jobs $j_1, \ldots, j_n$ sowie die Zeiten t_i, die zum Abarbeiten des i. Jobs benötigt wird. Es stehen zwei Maschinen zur Verfügung. Verteile die Jobs so auf beide Maschinen, dass alle Jobs möglichst schnell bearbeitet wurden.

(e) Sie wollen eine Wüste durchqueren. In der Wüste sind $n + 1$ Oasen, Sie starten in Oase 0 und Ihr Ziel ist Oase n. Dazwischen sind $n - 1$ Oasen, in denen Sie Ihre Wasserflasche auffüllen können. Ihre Wasserflasche reicht für r Kilometer. Oase i und Oase $i + 1$ sind d_i Kilometer voneinander entfernt. Finden Sie eine Folge von Oasen, sodass die möglichst selten nachfüllen müssen. Sie können davon ausgehen, dass alle Distanzen zwischen den Oasen kleiner als r sind.

Aufgabe 8: Definition eines Graphen

Die Definition eines Graphen $G = (V, E, w)$ ist gegeben durch die Knotenmenge V und die Kantenmenge E:

$$V = \{(x, y) \mid x, y \in \mathbb{N} \wedge x < n \wedge y < n\}, \ n \in \mathbb{N}^+$$

$$E = \{((x_1, y_1), (x_2, y_2)) \in V \mid w((x_1, y_1), (x_2, y_2)) < a\}, \ a \in \mathbb{R}^+$$

Die Funktion w ist durch die Zuordnungsvorschrift $(x_1, y_1), (x_2, y_2) \mapsto \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ gegeben. Falls $((x_1, y_1), (x_2, y_2)) \in E$, so repräsentiert w die Gewichtsfunktion der Kanten.

(a) Kann der Graph als ungerichteter Graph aufgefasst werden? Begründen Sie Ihre Antwort.

(b) Welches reale Szneario könnte durch diesen Graphen mit einem festen n und a beschrieben werden?

(c) Zeichnen Sie den Graph so, dass sich keine Kanten überkreuzen, für den Fall, dass $n = 5$ und $a = \sqrt{2}$.

(d) Geben Sie für $a = \sqrt{2}$ eine äquivalente Definition der Kantenmenge E an. In Ihrer Definition soll die Funktion w nicht vorkommen.

(e) Geben Sie ein $0 < a < 3\pi$ an, sodass die Berechnung der *Länge* des kürzesten Weges zwischen zwei Knoten bei beliebigem n in konstanter Zeit möglich ist. Wie funktioniert dann die Berechnung der Länge des kürzesten Weges?

Aufgabe 9: DNS-Sequenzen und k-mere

Eine DNS-Sequenz ist ein String aus den vier Zeichen `G`, `T`, `C` und `A`. In der Biologie vergleicht man die DNS-Sequenzen verschiedener Organismen. Dazu muss der DNS-String zunächst aus der Zelle extrahiert werden, wozu Sequenzierautomaten verwendet werden. Technisch bedingt ist die Länge k der Ausgabe der Automaten beschränkt, sodass eine DNS-Sequenz nicht komplett sequenziert wird. Stattdessen werden alle Substrings der Länge k in beliebiger Reihenfolge ausgegeben, wobei auch doppelte Substrings enhalten sind. Diese Substrings der Länge k werden als k-mere bezeichnet. Gesucht ist nun nach einer Möglichkeit, aus den k-meren die ursprüngliche DNS-Sequenz herzustellen. Ein *Präfix* eines k-mers ist der String des k-mers ohne das letzte Zeichen. Äquivalent dazu ist der *Suffix* eines k-mers der String des k-mers ohne das erste Zeichen.

(a) Ein Automat mit $k = 3$ sequenziert ein Genom mit der Sequenz `GTCCAGTCCAC`. Was ist die Ausgabe?

(b) Wie viele k-mere gibt es in einer Sequenz der Länge $n > k$?

(c) Gegeben ist der Graph G_H in Abbildung 2, der aus den 3-meren `ACG`, `ACT`, `CGT`, `CTA`, `GTA`, `GTT`, `TAC` und `TAC` entstanden ist. Leiten Sie die Vorgehensweise ab, mit der aus beliebigen k-meren ein Graph G_H gebildet wird.

(d) Beschreiben Sie in einem Satz, wie man mit G_H eine Sequenz findet, die die gegebenen k-mere aufweist. In der Bioinformatik ist die Anzahl der k-mere häufig in der Größenordnung einiger Millionen. Warum können Sie die Methode mit dem Graphen G_H nicht empfehlen?

(e) Geben Sie eine Sequenz s an, deren Sammlung der k-mere identisch zu denen aus c) ist.

(f) Glücklicherweise gibt es eine andere Möglichkeit, einen Graphen G_E aufzubauen, um eine Sequenz aus gegebenen k-meren zu berechnen: Die Kantenmenge besteht aus allen k-meren der Eingabemenge, wobei Duplikate erlaubt sind. Diese Kanten schreiben wir untereinander, beschriften sie mit dem jeweiligen k-mer und richten sie nach rechts. An die linke Seite jeder Kante fügen wir nun einen Knoten mit dem Präfix der Kante an und an der rechten Seite einen Knoten mit dem Suffix der Kante. Nun werden die Kanten analog zu einem Dominospiel an passenden Knoten aneinandergefügt.

Zeichnen Sie diesen Graph für die 3-mere aus c).

(g) Beschreiben Sie in einem Satz, wie man mit G_E eine Sequenz findet, die die gegebenen k-mere aufweist.

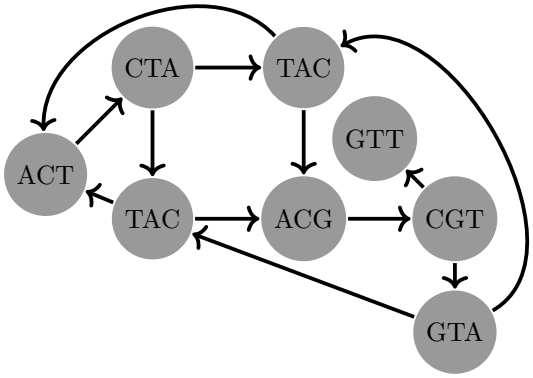


Abbildung 2: 3-mere-Graph für die Sequenz-Aufgabe.

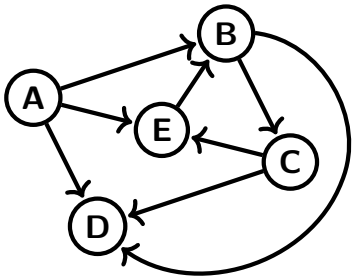


Abbildung 3: Ein gerichteter Graph für Aufgabe 10

Aufgabe 10: Repräsentation von Graphen
Gegeben sei der Graph in Abbildung 3.

(a) Enthält der Graph Kreise? Wenn ja, geben Sie einen Kreis an.

(b) Repräsentieren Sie diesen Graphen mit einer Adjazenzliste.

(c) Repräsentieren Sie diesen Graphen mit einer Adjazenzmatrix.

(d) Sei nun ein beliebiger simpler Graph G gegeben, dessen Maximalgrad $\Delta(G) = \lfloor\sqrt{|V|}\rfloor$ ist. In welcher Zeit kann man den Grad eines Knotens bestimmen oder testen, ob zwei Knoten benachbart sind, wenn G mit einer Adjazenzliste oder eine Adjazenzmatrix dargestellt ist?

(e) Beweisen oder widerlegen Sie folgende Aussage: Für einen beliebigen zusammenhängenden, einfachen Graphen $G = (V, E)$ mit $|V| \geq 2$ gilt: $\log(|E|) \in \Theta(\log|V|)$.

Aufgabe 11: Matchings in Graphen
Sei $G = (V, E)$ ein ungerichteter Graph. Eine Teilmenge $M \subseteq E$ der Kantenmenge heißt *Matching*, wenn keine zwei Kanten in M einen Knoten gemeinsam haben. Ein Matching heißt *nicht erweiterbar*, wenn es keine Kante e in $E \setminus M$ gibt, sodass $e \cup M$ ein Matching ist.

(a) Überlegen Sie sich ein Szenario, das man als Graph modellieren und mit einem Algorithmus, der ein maximales Matching findet, lösen kann.

(b) Schreiben Sie einen Algorithmus in Pseudocode, der für einen gegebenen Graphen $G = (V, E)$ und eine Teilmenge $M \subseteq E$ bestimmt, ob M ein Matching ist.

(c) Entwickeln Sie einen Algorithmus in Pseudocode, der für einen gegebenen Graphen $G = (V, E)$ ein gültiges, nicht erweiterbares Matching berechnet.

Aufgabe 12: Fehlende Kanten

Auf folgendem Graph in Abbildung 4 wurde eine Breitensuche ausgeführt. Dabei steht die Zahl in den Knoten für den Zeitpunkt, zu dem sie entdeckt wurden. Einige Kanten in dem Graph sind hier nicht dargestellt. Zeichnen Sie diese ein. Dabei darf jeder Knoten maximal den Grad 3 haben.

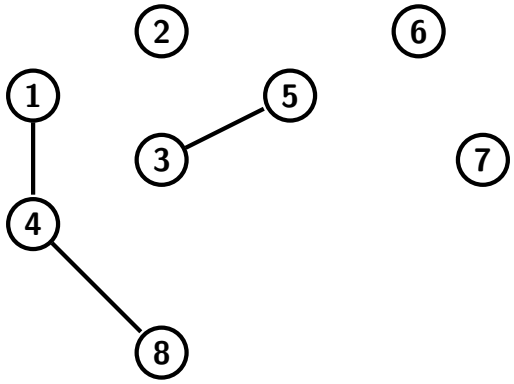


Abbildung 4: Abbildung Graphens für Aufgabe 12

Aufgabe 13: Graphen-Theorie

Als *bipartit* wird ein Graph $G = (V, E)$ bezeichnet, falls sich seine Knotenmenge V in zwei Mengen S_1 und S_2 aufteilen lässt, sodass alle Kanten einen Knoten in S_1 mit einem Knoten in S_2 verbinden.

(a) Vervollständigen Sie die Definition, sodass sie äquivalent zur obigen textuellen Definition ist. Füllen Sie in jede Lücke genau ein Zeichen.

Graph G ist bipartit $\Leftrightarrow$

$$\exists \_\_ \subseteq V \; \forall \; (\_\_, v) \in \_\_ : (\_\_ \in S_1 \; \wedge \; \_\_ \in V \setminus \_\_) \vee (\_\_ \in V \setminus S_1 \_\_ u \in S_1)$$

(b) Beweisen Sie folgende Aussage: Graph G ist bipartit $\Leftrightarrow$ Graph G ist zweifärbbar.

(c) Der folgende Algorithmus überprüft, ob ein gegebener Graph bipartit ist. Ergänzen Sie die Lücken. Zu Beginn des Algorithmus hat das marker-Attribut aller Knoten den Wert 0.

Algorithmus 9: boolean checkIfBipartite(Graph G, Vertex u = **null**)

```
1  if u = null then
2      u = beliebiger Knoten
3      u.marker = _____
4  foreach v ∈ Adj[u] do
5      if v.marker ≠ 0 then
6          if _____ then return false
7      else
8          if u.marker = ___ then v.marker = ___
9          if u.marker = ___ then v.marker = ___
10         flag = checkIfBipartite(___,___)
11         if flag = false then _____
12 return _____
```

(d) Welcher Algorithmus aus der Vorlesung liegt checkIfBipartite(G,v) zu Grunde?

Aufgabe 14: Terminale verbinden

Sie sind Betreiber eines Routernetzwerkes, wobei R die Menge Ihrer Router darstellt. Die Router sind untereinander verbunden, sodass Ihr gesamtes Netzwerk zusammenhängend ist. Dabei muss nicht notwendigerweise jeder Router mit jedem anderen Router verbunden sein. Damit die Verbindungen funktionieren, muss jede Verbindung mit Strom versorgt werden.

Im Falle eines Stromausfalls kann jede Verbindung mit je einem teurem Notstromaggregat betrieben werden. Die Kosten für dieses Notstromaggregat sind je Verbindung verschieden.

In Ihrem Netzwerk befinden sich einige besonders wichtige Knotenrouter K. Falls der Strom ausfällt, sollen weiterhin alle Router in K miteinander verbunden sein. Alle anderen Router $R \setminus K$ dürfen, aber müssen nicht unbedingt, ans Netzwerk angeschlossen sein. Sie sind nun an einer Auswahl an Verbindungen interessiert, die möglichst günstig mit Notstromaggregaten betrieben werden können und alle Knotenrouter K verbindet.

(a) Modellieren Sie das Problem als Graph-Problem. *Tipp:* Machen Sie sich mit einer Skizze die Aufgabenstellung klar.

(b) Angenommen $|K| = 2$. Geben Sie einen effizienten Algorithmus an, der das Problem löst.

(c) Angenommen $K = R$, mit anderen Worten: Alle Router sind wichtig. Geben Sie einen effizienten Algorithmus an, der das Problem löst.

(d) Das Software-Unternehmen PISNP bietet einen Algorithmus an, der das Problem für alle K löst. Dieser Algorithmus berechnet einen Graph T, der angeblich die oben beschrieben Anforderungen erfüllt. Geben Sie einen effizienten Algorithmus an, der überprüft, ob T tatsächlich gültig ist. Ihr Algorithmus erhält als Eingabe ihr Routernetzwerk, wie in a) modelliert, sowie K und T.

Aufgabe 15: Kürzeste-Wege-Bäume und minimale Spannbäume

Die Algorithmen von Dijkstra und Jarnik-Prim gehen ähnlich vor. Beide berechnen, ausgehend von einem Startknoten s, einen Baum. Allerdings brechnet der Algorithmus von Dijkstra einen Kürzesten-Wege-Baum, während der Algorithmus von Jarnik-Prim einen minimalen Spannbaum berechnet.

(a) Geben Sie einen ungerichteten gewichteten Graphen $G = (V, E)$ mit höchstens 5 Knoten und einen Startknoten $s \in V$ an, sodass Dijkstra und Jarnik-Prim ausgehend von s verschiedene Bäume in G liefern. Geben Sie beide Bäume an.

(b) Geben Sie eine Familie von Graphen an, auf denen der Algorithmus von Jarnik-Prim asymptotisch schneller als der Algorithmus von Kruskal ist.

(c) Angenommen Sie haben einen ungerichteten, gewichteten Graphen $G = (V, E)$ gegeben und alle Gewichte sind natürliche Zahlen zwischen 1 und $|V|$. Wie könnten Sie die Laufzeit von Kruskal's Algorithmus beschleunigen?

Aufgabe 16: MaxCut

Gegeben sei ein ungewichteter ungerichteter Graph $G = (V, E)$. Im MaxCut Problem ist ein Schnitt $(C, V \setminus C)$ gesucht, sodass die Anzahl der Kanten, die diesen Schnitt schneiden maximal ist.

Algorithmus 11: `RandomizedMaxCut`(Graph $G = (V, E)$)

```
1 C ← ∅
2 foreach v ∈ V do
3 |   m = throwCoin()          // Gibt mit Wahrscheinlichkeit 1/2 Kopf oder Zahl in O(1)
4 |   if m == Kopf then
5 |   |   C ← C ∪ {v}
6 return (C, V \ C)
```

(a) Um welche Art von Algorithmus handelt es sich? Ist die Lösung immer valide?

(b) Mit welcher Datenstruktur könnte man die Menge C darstellen?

(c) Welche Laufzeit hat dieser Algorithmus?

(d) Sei $\{u, v\} \in E$. Mit welcher Wahrscheinlichkeit liegt diese Kante auf dem Schnitt?

(e) Sei $(C, V \setminus C)$ ein Schnitt, den `RandomizedMaxCut` zurückgegeben hat. Wie viele Kanten liegen erwartet auf diesem Schnitt?

(f) Wie viele Kanten können maximal auf einem Schnitt liegen?

(g) Welchen erwarteten Approximationsfaktor hat `RandomizedMaxCut`?