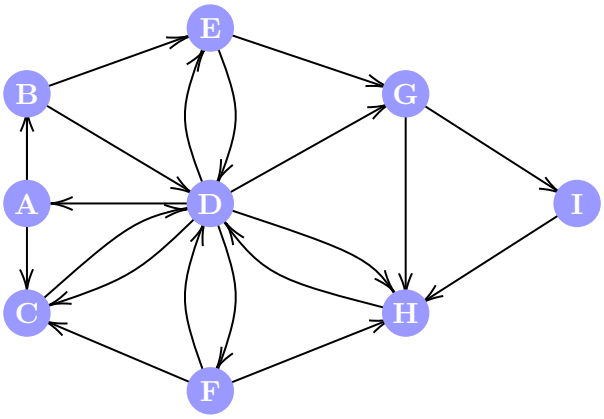
Übungsblatt 9

Abgabe via Moodle.
Deadline Fr. 7ter July

Aufgabe 1 (*Breitensuche,* $1+1+1+1$ *Punkte*)

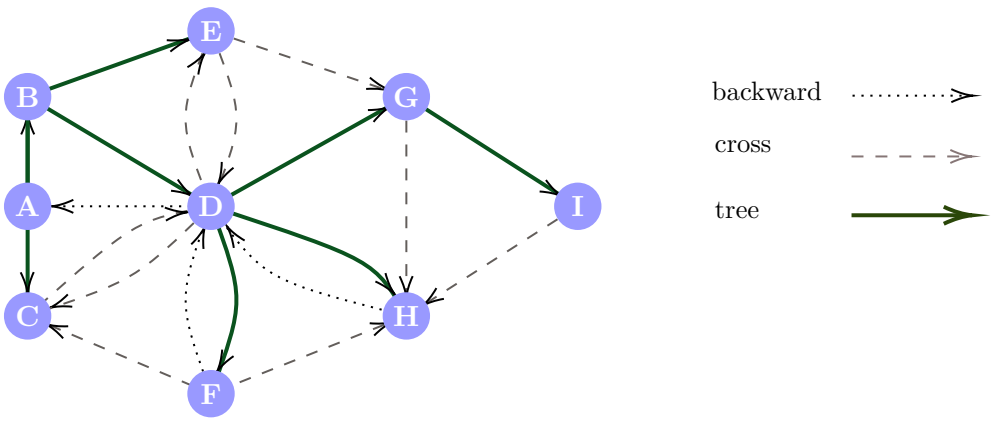
Gegeben sei folgender gerichteter Graph mit der Knotenmenge $\{A, B, \ldots, I\}$:



In diesem Graph werde nun eine Breitensuche durchgeführt, und zwar ausgehend vom Knoten A.

1. Zählen Sie die Knoten des Graphen in einer Reihenfolge auf, in der diese von einer Breitensuche jeweils zum ersten mal berührt werden. Heben Sie im Graph außerdem alle Kanten (z.B. farbig) hervor, die Bezüglich dieser Breitensuche *tree* Kanten sind.
2. Sind im Graph bzgl. dieser Breitensuche irgendwelche *backward* Kanten vorhanden? Wenn ja, heben Sie diese hervor.
3. Sind im Graph bzgl. dieser Breitensuche irgendwelche *cross* Kanten vorhanden? Wenn ja, heben Sie diese hervor.
4. Sind im Graph bzgl. dieser Breitensuche irgendwelche *forward* Kanten vorhanden? Wenn ja, heben Sie diese hervor.

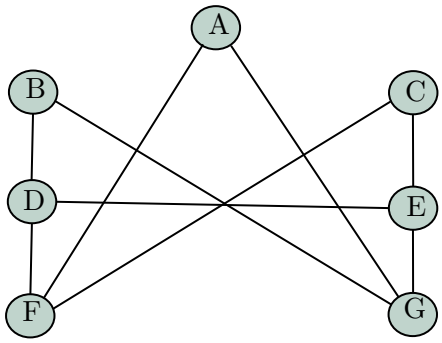
Lösung:
Aufzählung in Reihenfolge der ersten Berührung durch Breitensuche: $A, B, C, D, E, F, G, H, I$. Kanten vom Typ *forward* können bei Breitensuche nicht auftreten. Die anderen Kantenarten sind wie in der Legende angegeben hervorgehoben:

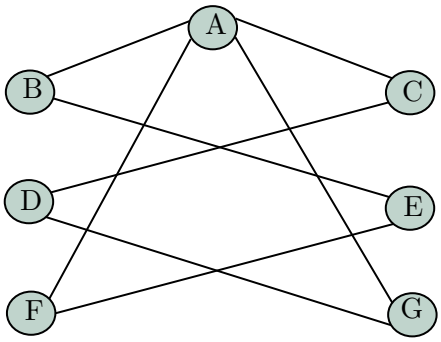


Aufgabe 2 (*Bipartite Graphen*, $2 + 6 + 2$ *Punkte*)

Ein ungerichteter zusammenhängender Graph $G = (V, E)$ mit $|V| \geq 1$ heißt **bipartit**, wenn die Knotenmenge V so in zwei Mengen V_1 und V_2 aufgeteilt werden kann, dass für jede Kante $(u, v) \in E$ gilt $u \in V_1$ und $v \in V_2$ oder $v \in V_1$ und $u \in V_2$. Mit anderen Worten existiert eine Zerlegung von V, so dass Kanten nur zwischen Knoten aus V_1 und V_2 bestehen, nicht aber zwischen Knoten der V_i selbst.

1. Zeigen oder widerlegen Sie jeweils die Bipartitheit der folgenden beiden Graphen!





2. Entwickeln Sie einen Algorithmus, der in $\mathrm{O}(n + m)$ entscheiden, ob ein ungerichteter zusammenhängender Graph bipartit ist. Im Fall der Bipartitheit soll der Algorithmus eine mögliche Unterteilung V_1, V_2 ausgeben, sonst *nicht bipartit*. Geben Sie **Pseudocode** an.

3. Erweitern Sie ihren Algorithmus, so dass im Falle der Nichtbipartitheit ein Zeuge ausgegeben wird. **Hinweis:** Pseudocode ist nicht zwingend erforderlich.

Lösung:

1. - Der linke Graph ist nicht bipartit: es existiert ein Kreis ungerade Länge $K = \langle A, F, C, E, G, A \rangle$ und ein ungerichteter zusammenhängender Graph ist genau dann bipartit, wenn er keinen Kreis ungerader Länge enthält.
   - Der rechte Graph ist bipartit: Definiere $V_1 := \{A, D, E\}, V_2 := \{B, C, F, G\}$.

2. Beim Entwurf nutzen wir die Eigenschaft aus, dass es zwischen zwei nicht aufeinander folgenden Leveln bei einer Breitensuche keine Kanten geben kann, da diese sonst von der Breitensuche schon vorher gefunden worden wären. D.h. man kann die Knoten der Level der Breitensuche jeweils abwechselnd in die V_1 und V_2 packen. Sollte die Breitensuche eine Kreuzkante finden, so ist der Graph nicht bipartit, da der Graph dann einen Kreis ungerader Länge als Teilgraph enthält. Kreise ungerader Länge sind nicht bipartit.

```
1: function isBipartit(G = (V[1..n], E[1..m])
2: (parent, d) := bfs(V[1], G) // BFS in G from first node
3: V_1, V_2: Queue of Vertices
4: f[1..n]: Array of {weiß, schwarz}
5: for i := 1 to n do
6:   if d[i] mod 2 = 0 then
7:     V_1.push_back(i)
8:     f[i] = white
9:   else
10:    V_2.push_back(i)
11:    f[i] = black
12: end for
13: for i := 1 to m do
```

14: $\{u, v\} := E[i]$
15: **if** $f[u] = f[v]$ **then return** "nicht bipartit"
16: **end for**
17: **return** (V_1, V_2)

3. Am einfachsten ist es einen Kreis ungerader Länge auszugeben. Man kann dazu den Algorithmus aus Aufgabenstellung b) leicht modifizieren. Wenn man $f[u] = f[v]$ gefunden hat, so startet man bei u eine Rückverfolgung der Parentzeiger bis man bei $V[1]$ angekommen ist und hängt die dabei besuchten Knoten einschließlich u an eine Liste K an. Wir nehmen dabei zunächst vereinfachend an, dass $V[1]$ der kleinste gemeinsame Vorgänger von u und v im BFS Baum ist. Man beachte, dass es sich bei der Kante {u,v} um eine Kreuzkante handeln muss, da G ungerichtet ist und somit in dem durch die BFS erzeugten Baum keine Vorwärts- und Rückwärtskanten existieren können.
   An die Liste K werden nun in umgedrehter Reihenfolge die Knoten vom Pfad $v \to V[1]$, die man wieder über das parent Array der Breitensuche erhält, angehängt. Schließlich wird noch u an K angehängt. Nun haben wir einen Kreis $K = u \to V[1] \to v \to u$. Dieser hat ungerade Länge, da die Längen von $u \to V[1]$,$V[1] \to v$ entweder beide gerade oder beide ungerade sind und genau eine Kante extra nämlich die von v nach u in K enthalten ist.
   Sollte $V[1]$ nun nicht der kleinste gemeinsame Vorgänger von u und v sein, so wird es noch ein wenig komplizierter, da der Pfad von $V[1]$ zum kleinsten gemeinsamen Vorgänger von u und v nicht im Kreis enthalten sein soll. Da die beiden Knoten auf dem gleichen Level sind, kann man zum Beispiel die Parentzeiger von u und v wechselseitig verfolgen bis man den kleinsten gemeinsamen Vorgänger gefunden hat und dann entsprechend wie oben den Kreis K zusammenbauen.

Aufgabe 3 (*Dichte und spärliche Graphen,* $2+2+2+2+2$ *Punkte*)

Eine **Familie von Graphen** ist eine unendliche Menge von Graphen, mit der Eigenschaft dass sich für jedes $n_0 \in \mathbb{N}_{\geq 0}$ stets ein Graph mit $n \geq n_0$ Knoten in der Menge befindet.

1. Zeigen Sie: Vollständige ungerichtete Graphen mit n Knoten haben $\Theta(n^2)$ Kanten.

Eine Familie von ungerichteten Graphen heiße eine Familie **spärlicher** Graphen, wenn jeder Graph mit n Knoten $\mathrm{O}(n)$ Kanten hat.

2. Zeigen Sie: In einer Familie spärlicher Graphen liegt der maximale Knotengrad nicht notwendigerweise in $\mathrm{O}(1)$.

Eine Familie von ungerichteten Graphen heiße eine Familie **gleichmäßig spärlicher** Graphen, wenn jeder Knoten jedes Graphen mit n Knoten $\mathrm{O}(1)$ Kanten hat.

3. Zeigen Sie: Eine Familie gleichmäßig spärlicher Graphen ist auch eine Familie spärlicher Graphen.

Eine Familie von ungerichteten Graphen heiße eine Familie **dichter** Graphen, wenn jeder Graph mit n Knoten $\Omega(n^2)$ Kanten hat.

4. Zeigen Sie: In einer Familie dichter Graphen hat nicht unbedingt jeder Knoten eines Graphen mit n Knoten $\Omega(n)$ Kanten.

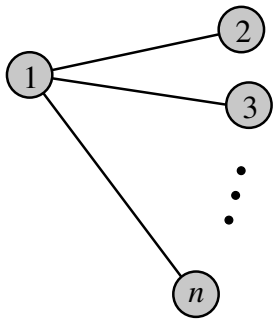
Eine Familie von ungerichteten Graphen heiße eine Familie **gleichmäßig dichter** Graphen, wenn jeder Knoten jedes Graphen mit n Knoten $\Omega(n)$ Kanten hat.

5. Zeigen Sie: Eine Familie gleichmäßig dichter Graphen ist auch eine Familie dichter Graphen.

Lösung:

1. Ein vollständiger ungerichteter Graph mit n Knoten hat $n(n-1)/2$ Kanten. Dies liegt daran, dass jeder der n Knoten jeweils genau eine Kante hat zu jedem der $n-1$ anderen Knoten, aber keine zu sich selbst. Allerdings zählt jede Kante für zwei Knoten, weshalb halbiert werden muss. Weiter ist
$$\frac{1}{2}n(n-1) \leq n(n-1) = n^2 - n \leq n^2$$
und
$$\frac{1}{2}n(n-1) \geq \frac{1}{2}\left(n^2 - \frac{1}{2}n^2\right) = \frac{1}{4}n^2$$
für genügend große n. Es gilt also $n(n-1)/2 = \Theta(n^2)$.

2. Als Gegenbeispiel geben wir eine Familie von spärlichen Graphen an: Zu jedem n enthalte die Familie genau einen ungerichteten Graph mit $n \geq 2$ Knoten. Dieser habe folgende Form:



Offenbar hat dieser Graph, wie von der Definition einer Familie spärlicher Graphen gefordert, $\mathrm{O}(n)$ Kanten. Jedoch hat Knoten 1 immer $n-1$ Kanten für alle $n \in \mathbb{N}_{>0}$. Es ist aber $n-1 \notin \mathrm{O}(1)$.

Wichtig: Ein Graph mit einer *festen* Anzahl Knoten ist *kein* ausreichendes Gegenbeispiel. Da es hier um asymptotische Aussagen ($\mathcal{O}$-Kalkül) geht, muss ein Gegenbeispiel von n abhängen.

3. Man betrachte eine Familie gleichmäßig spärlicher Graphen. Nach Definition gibt es Konstanten $c > 0$ und $n_0 > 0$, so dass alle Graphen dieser Familie mit $n \geq n_0$ Knoten höchstens c Kanten haben an jedem Knoten. Insgesamt hat jeder Graph der Familie mit $n \geq n_0$ Knoten also höchstens cn Kanten. Also haben die Graphen der Familie $\mathrm{O}(n)$ Kanten bei n Knoten. Somit gilt die Behauptung.

4. Als Gegenbeispiel geben wir eine Familie von dichten Graphen an: Zu jedem n enthalte die Familie genau einen ungerichteten Graph mit n Knoten. Dieser bestehe aus einem vollständigen Graph mit $n-1$ Knoten und einem weiteren isolierten Knoten (d.i. ein Knoten ohne Kanten). Der vollständige Teilgraph mit $n-1$ Knoten hat $(n-1)(n-2)/2 = \Omega(n^2)$ Kanten, also gilt das auch für den Gesamtgraph, da dieser allenfalls mehr Kanten hat. Somit ist die Definition einer Familie spärlicher Graphen erfüllt. Der isolierte Knoten hat nun 0 Kanten für alle n. Es ist aber $0 \notin \Omega(n)$, also gilt die Behauptung.

Wichtig: Genau wie in Teilaufgabe b) ist ein Graph mit einer festen Anzahl Knoten kein ausreichendes Gegenbeispiel.

5. Man betrachte eine Familie gleichmäßig dichter Grap en. Nach Definition gibt es Konstanten $c > 0$ und $n_0 > 0$, so dass alle Graphen dieser Familie mit $n \geq n_0$ Knoten mindestens cn Kanten haben an jedem Knoten. Insgesamt hat jeder Graph der Familie mit $n \geq n_0$ Knoten also mindestens cn^2 Kanten. Also haben die Graphen der Familie $\Omega(n^2)$ Kanten bei n Knoten. Somit gilt die Behauptung.

Aufgabe P9 (*Maximum Independent Set, optional*)

Für die praktischen Übungen verwenden wir die Plattform `www.hackerrank.com`. Hier müssen Sie sich registrieren um an den Übungen teilzunehmen. Unter dem Link

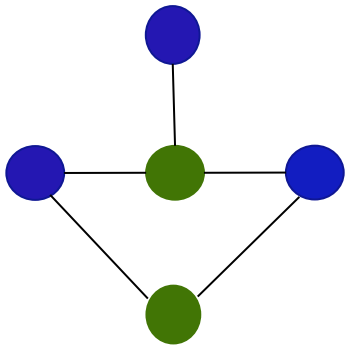
`https://www.hackerrank.com/adsi-2023`

finden die praktischen Übungen in der Form eines Programmierwettbewerbs statt.

In der neunten Challenge geht es um das Problem, eine möglichst große, maximale stabile Menge zu berechnen. Für einen gegebenen Graphen $G = (V, E)$ bezeichnet man eine Teilmenge S von V als stabile Menge (*engl. independent set*), falls die Knoten in S nicht zueinander adjazent sind. Eine maximale stabile Menge ist eine stabile Menge S', sodass für jeden Knoten $u \in V \setminus S'$, $S' \cup \{u\}$ keine stabile Menge mehr ist.

Ihnen wird ein ungerichteter, ungewichteter Graph G als Eingabe gegeben. Ihre Aufgabe ist es, eine Heuristik zu entwerfen, welche eine möglichst große, maximale stabile Menge findet.

Beachten sie, es gibt mehrere maximale stabile Mengen, welche unterschiedlich groß sein können, wie das folgende Beispiel zeigt:



In diesem Graphen bilden sowohl die grünen, als auch die blauen Knoten maximale stabile Mengen.

Auf Moodle finden Sie wieder ein Framework für diese Aufgabe: *framework9.cpp*. Der Score für diese Aufgabe wird nicht mittels der Laufzeit Ihres Algorithmus berechnet, sondern wird über die Größe der maximalen stabilen Menge berechnet, welche Sie finden.