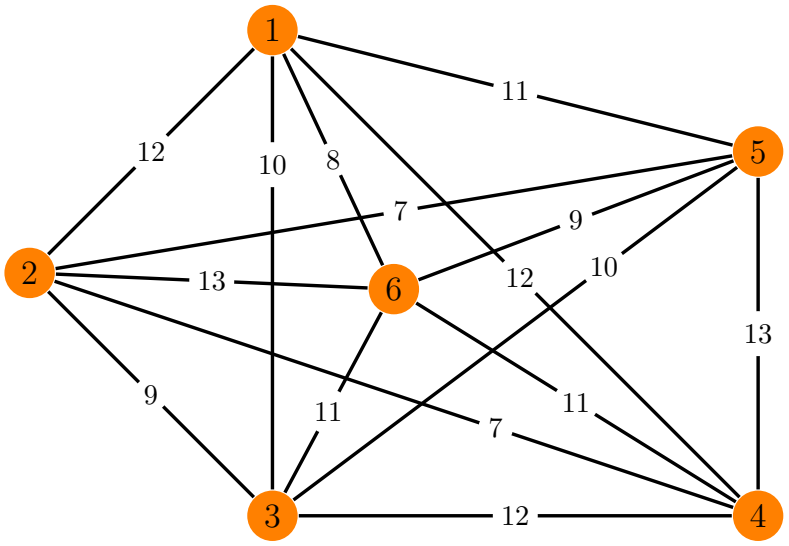
Aufgabe 1. Gegeben sei eine Instanz des symmetrischen Traveling Salesperson Problems (TSP) in Form des folgenden vollständigen Graphen G mit Knoten $V = \{1, 2, 3, 4, 5, 6\}$ und Kanten E, denen die eingezeichneten Gewichte zugeordnet sind:



(a) Wenden Sie auf diese Instanz *zweimal* die in der Vorlesung gezeigte Spanning-Tree-Heuristik an, wobei Sie unterschiedliche Eulerkreise verwenden, sodass zwei unterschiedliche Touren erzeugt werden. Beschreiben Sie die einzelnen Schritte im Detail.

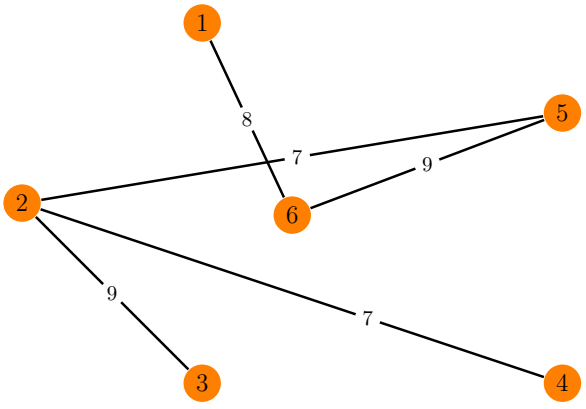
(b) Versuchen Sie eine optimale Tour durch „Hinsehen“ zu finden. Geben Sie die Tour und deren Länge an.

(c) Wie viele Touren müssten Sie im Allgemeinen mindestens für das symmetrische TSP mit n Städten und hier konkret in einem vollständigen Enumerations-Verfahren durchprobieren, um eine bewiesen optimale Lösung zu finden?

Hinweis: $n!$ kann durch Ausnutzung von Symmetrien verbessert werden.

(d) Ist die gegebene Instanz metrisch? Was bedeutet das im Allgemeinen für die Gütegarantie der Spanning-Tree-Heuristik? Welche konkreten Approximationsgüten haben Ihre beiden in Unteraufgabe (a) gefundenen Touren?

(a) 1. Bestimme MST von G:



2. Bestimme Eulertour auf Graph mit verdoppelten Kanten:
   i. $\{1, 6, 5, 2, 3, 2, 4, 2, 5, 6, 1\}$
   ii. $\{1, 6, 5, 2, 4, 2, 3, 2, 5, 6, 1\}$
3. Erzeuge aus Eulertour eine TSP Instanz:
   i. $\{1, 6, 5, 2, 3, 4, 1\}$
   Länge: $8 + 9 + 7 + 9 + 12 + 12 = 57$
   ii. $\{1, 6, 5, 2, 4, 3, 1\}$
   Länge: $8 + 9 + 7 + 7 + 12 + 10 = 53$

(b) Optimale Tour: $\{1, 3, 4, 2, 5, 6, 1\}$
Gesamtkosten: 53

(c) Wenn der Graph vollständig ist gibt es $\frac{(n-1)!}{2}$ Möglichkeiten für eine TSP-Tour.

(d) Der Graph ist metrisch. Das bedeutet, dass die Spanning-Tree-Heuristik eine Gütegarantie von 2 besitzt.
   i. $\frac{57}{53} \approx 1.075$
   ii. $\frac{53}{53} = 1$

Aufgabe 2. Betrachten Sie das folgende Optimierungsproblem:

> Sie wollen eine Arbeitsgruppe aus Experten bilden die eine Menge $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_m\}$ von Qualifikationen abdeckt und haben eine Menge an verfügbaren Experten $\mathcal{E} = \{E_1, E_2, \ldots, E_n\}$. Jeder dieser Experten $E \in \mathcal{E}$ hat eine nicht-leere Menge $Q_E \subseteq \mathcal{Q}$ von Qualifikationen. Für jede Qualifikation sind maximal 3 Experten verfügbar. Aus Kostengründen wollen Sie die Arbeitsgruppen so klein wie möglich halten.

Orientieren Sie sich an den Ideen des Algorithmus `Approx-Vertex-Cover` aus der Vorlesung und entwerfen Sie einen polynomiellen Approximationsalgorithmus mit Gütegarantie 3 für dieses Problem.

(a) Beschreiben Sie den Algorithmus.

(b) Erläutern Sie, weshalb die Approximationsgüte gilt.

(c) Geben Sie ein Beispiel an, bei dem sich die optimale Lösung und die Approximation um den Faktor 3 unterscheiden.

(d) Analysieren Sie die asymptotische Laufzeit Ihres Algorithmus.

Freiwillige Zusatzaufgabe: Können Sie die Approximationsgüte auch noch garantieren, wenn es eine Qualifikation (z.B. *Teamchef*) gibt, für die 9 159 993 Experten qualifiziert sind und es für alle anderen Qualifikation immer noch maximal 3 Experten gibt? Begründen Sie Ihre Antwort kurz.

(a) Überprüfe ob alle Qualifikationen von den Experten abgedeckt werden können (also ob die Vereinigung aller Q_E $\mathcal{Q}$ ergibt). Falls nein gib einen Fehler aus. Sonst: Initialisiere die Arbeitsgruppe als leere Menge. Füge für jede Qualifikation alle Experten hinzu, die diese abdecken. Lösche nun alle Qualifikationen die diese Experten insgesamt abdecken aus allen Mengen an Qualifikationen der restlichen Experten.

(b) Alle Qualifikationen müssen abgedeckt sein und für jede Qualifikation gibt es maximal 3 Experten. Daher wird für jede Qualifikation maximal das dreifache zu viel gewählt. Wenn ein Experte mehr als nur eine Qualifikation abdeckt kann die Güte nicht schlechter werden.

(c) $\mathcal{Q} = \{Q_1, Q_2, Q_3\}$
$\mathcal{E} = \{E_1, E_2, E_3, E_4\}$
$Q_{E_1} = \{Q_1\} \quad Q_{E_2} = \{Q_1, Q_2\} \quad Q_{E_3} = \{Q_1, Q_2, Q_3\} \quad Q_{E_4} = \{Q_3\}$
Arbeitsgruppe: $\{E_1, E_2, E_3\}$
$\frac{c_A}{c_{opt}} = \frac{3}{1} = 3$

(d) Für jede Qualifikation: $O(|Q|)$
Finde alle Experten die diese Abdecken: $O(|E|)$
Füge zur Arbeitsgruppe hinzu: $O(1)$
Lösche Qualifikationen:
Bilde Menge der abgedeckten Qualifikationen MQ: $O(3 \cdot |Q|) = O(|Q|)$
Für jeden Experten: $O(|E|)$
Für jede Qualifikation q des Experten: $O(|Q|)$
Überprüfe ob $q \in MQ$ und lösche falls ja: $O(|Q|)$

Gesamt: $O(|Q| \cdot (|E| + 1 + |Q| + |E| \cdot |Q| \cdot |Q|)) = O(|Q|^3 \cdot |E|)$

Zusatz: Nein, da in diesem Fall bei der Qualifikation Teamchef 9159993 Experten gewählt werden könnten und somit die Approximationsgüte 9159993 betragen könnte.

Aufgabe 3. Betrachten Sie das aus der Vorlesung bekannte MAX-CUT Problem, bei dem für einen gewichteten Graphen ein maximaler Schnitt berechnet werden soll.

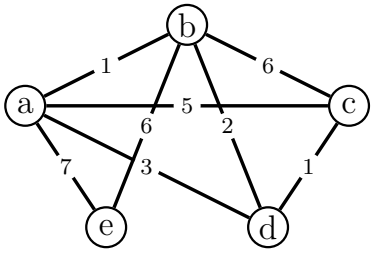
MAX-CUT: Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit positiven ganzzahligen Kantengewichten w_{uv} für alle Kanten $(u, v) \in E$. Finde einen Schnitt, d.h. eine Partition der Knoten (A, B), sodass das Gesamtgewicht $w(A, B)$ von Kanten, die Knoten in den unterschiedlichen Partitionen verbinden, maximiert wird.

$$w(A, B) := \sum_{u \in A,\ v \in B} w_{uv}$$

Betrachten Sie nun folgenden Greedy Algorithmus für MAX-CUT.

Function
GREEDY-MAX-CUT($G = (V, E)$)
  $A \leftarrow \emptyset$, $B \leftarrow \emptyset$
  for $v \in V$
    $a \leftarrow \sum_{u \in A} w_{uv}$
    $b \leftarrow \sum_{u \in B} w_{uv}$
    if $a \leq b$ **then**
      $A \leftarrow A \cup \{v\}$
    else
      $B \leftarrow B \cup \{v\}$
  return (A, B)

Beispiel Graph:



Führen Sie jetzt folgende Analyse durch:

(a) Probieren Sie den Algorithmus zunächst an dem gegebenen Beispiel aus. Betrachten Sie die Knoten in alphabetischer Reihenfolge.

(b) Geben Sie die Worst-Case Laufzeit in Θ-Notation in Abhängigkeit von $n = |V|$ und $m = |E|$ an.

(c) Zeigen Sie, dass GREEDY-MAX-CUT kein 4/5-Approximationsalgorithmus ist. Geben Sie dazu einen geeigneten Graphen, die optimale Lösung und die Lösung von GREEDY-MAX-CUT an. Halten Sie den Graphen dabei möglichst klein.

(d) Zeigen Sie, dass GREEDY-MAX-CUT ein 1/2-Approximationsalgorithmus ist.

Freiwillige Zusatzaufgabe: Zeigen Sie, dass 1/2 die beste Gütegarantie ist, die für diesen Algorithmus gilt. D.h. geben Sie für jede bessere Gütegarantie $1/2 + \varepsilon$ einen Graphen an, bei dem diese Gütegarantie nicht erfüllt ist.

(a)
- Node a: $a = 0 \quad b = 0 \quad A = \{a\} \quad B = \emptyset$
- Node b: $a = 1 \quad b = 0 \quad A = \{a\} \quad B = \{b\}$
- Node c: $a = 5 \quad b = 6 \quad A = \{a, c\} \quad B = \{b\}$
- Node d: $a = 4 \quad b = 2 \quad A = \{a, c\} \quad B = \{b, d\}$
- Node e: $a = 7 \quad b = 6 \quad A = \{a, c\} \quad B = \{b, d, e\}$

(b) for $v \in V$: $\Theta(n)$

Summe:

$0, 1, 2, ..., n-1$ Knoten $\Rightarrow$ Im Schnitt $\frac{n}{2}$ Knoten.

Annahme: Graph als Adjazenzmatrix mit 0 für „nicht verbunden“.

Alle Durchschnittlich $\frac{n}{2}$ Einträge der Adjazentmatrix müssen betrachtet werden $\Rightarrow \Theta(n)$

if: $\Theta(1)$

Add to set: $\Theta(1)$

Gesamt: $\Theta(n^2)$

(c) Siehe Beispielgraph aus der Angabe und Durchlauf des Algorithmus aus (a)
Optimale Lösung: $A = \{c, d, e\} \quad B = \{a, b\} \quad w(A, B) = 29$
Lösung des Algorithmus: $A = \{a, c\} \quad B = \{b, d, e\} \quad w(A, B) = 18$
$\frac{18}{29} < \frac{4}{5}$

(d)

Lokale Optimalität:

$$\forall u \in A : \sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$$

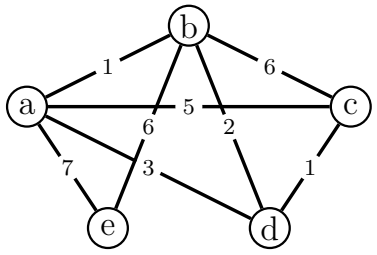
Ein Knoten u wird nur zu A hinzugefügt, wenn für alle bisherigen Knoten gilt:

$$\sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$$

Somit muss auch insgesamt gelten, dass $\forall u \in A : \sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$ und damit, dass der Greedy-Max-Cut Algorithmus eine lokal optimale Lösung liefert.

Da eine lokal optimale Lösung eine Approximationsgüte von $\frac{1}{2}$ besitzt (Folien Seite 37 bis 38) muss auch der Algorithmus eine Approximationsgüte von $\frac{1}{2}$ besitzen.

Aufgabe 4. Betrachten Sie das MAX-CUT Problem aus Aufgabe 3. Wir wenden nun lokale Suche für das MAX-CUT Problem an. Dazu verwenden wir die aus der Vorlesung bekannte Flip-Nachbarschaftsstruktur N für MAX-CUT. Es ist folgende Instanz von MAX-CUT gegeben:



(a) Bestimmen Sie $N((A, B))$ für den Schnitt $(A, B) = (\{a, c\}, \{b, d, e\})$.

(b) Wenden Sie lokale Suche mit der Flip-Nachbarschaftsstruktur N und Ausgangslösung $(A, B) = (\{a, c\}, \{b, d, e\})$ auf die obige Instanz an, bis ein lokales Optimum erreicht wird. Durchmustern Sie die aktuelle Nachbarschaft so, dass jedenfalls immer eine bessere Nachbarlösung gefunden wird, wenn eine solche in der Nachbarschaft existiert (Sie können frei zwischen First Improvement oder Best Improvement als Schrittfunktion wählen). Geben Sie alle Zwischenschritte an, also jeden Schnitt, welcher im Laufe der lokalen Suche ausgewählt wird.

(a) $N((A, B)) = \{$
$(\{c\}, \{a, b, d, e\}),$
$(\{a, b, c\}, \{d, e\}),$
$(\{a\}, \{b, c, d, e\}),$
$(\{a, c, d\}, \{b, e\}),$
$(\{a, c, e\}, \{b, d\})$
$\}$

(b) $w(A, B) = 18$

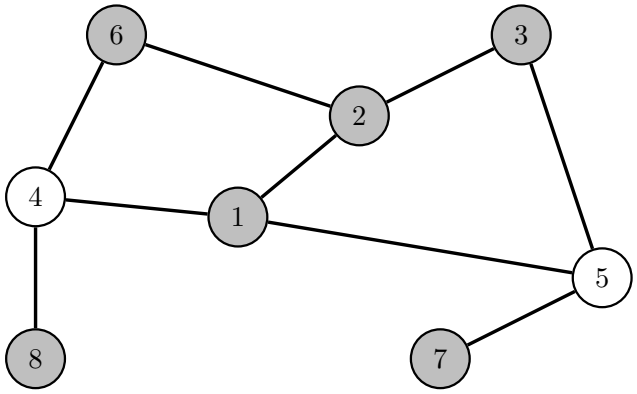
Es wird die „First Improvement“ Strategie verwendet.

Step 1: Betrachte Nachbar $(\{a, b, c\}, \{d, e\})$
$w(\{a, b, c\}, \{d, e\}) = 19$

Step 2: Betrachte Nachbar $(\{a, b\}, \{c, d, e\})$
$w(\{a, b\}, \{c, d, e\}) = 29$

Step 3: $N((\{a, b\}, \{c, d, e\})) = \{$
$(\{b\}, \{a, c, d, e\}),$
$(\{a\}, \{b, c, d, e\}),$
$(\{a, b, c\}, \{d, e\}),$
$(\{a, b, d\}, \{c, e\}),$
$(\{a, b, e\}, \{c, d\})$
$\} = \{V, W, X, Y, Z\}$
$w(V) = 15 \quad w(W) = 16 \quad w(X) = 19 \quad w(Y) = 25 \quad w(Z) = 17$
Termination

Aufgabe 5. Wir wenden lokale Suche für das VERTEX COVER Problem an. Dazu verwenden wir die zwei aus der Vorlesung bekannten Nachbarschafstrukturen N (simples löschen) und N' (alternative Nachbarschaft) für Vertex Cover. Gegeben ist nachfolgend ein Graph $G = (V, E)$ und eine Ausgangslösung $S = \{1, 2, 3, 6, 7, 8\}$ (Knoten mit grauem Hintergrund).



(a) Geben Sie $N(S)$ und $N'(S)$ an.

(b) Kann eine lokale Suche welche N verwendet, angewandt auf G, mit Ausgangslösung S ein minimales Vertex Cover finden? Falls ja, geben Sie einen Durchlauf der lokalen Suche an, der solch ein minimales Vertex Cover findet. Falls nein, zeigen Sie, dass die lokale Suche in keinem globalen Optimum enden kann.

(c) Beantworten Sie die selbe Frage wie bei Aufgabe 5(b), aber mit N' statt N.

(d) Überlegen Sie sich, welchen Einfluss die Wahl der Schrittfunktion (Best Improvement, First Improvement, Random Neighbor) bei der lokalen Suche für Vertex Cover mit Nachbarschafstruktur N bzw. N' hat. Vergleichen Sie dies mit dem Einfluss der Schrittfunktion in der lokalen Suche für das MAX-CUT Problem.

(a) $N(S) = \{\{1, 3, 6, 7, 8\}\}$
$N'(S) = \{\{1, 3, 6, 7, 8\}, \{1, 2, 3, 4, 7\}, \{1, 3, 4, 6, 7\}\}, \{1, 2, 5, 6, 8\}, \{1, 3, 5, 6, 8\}\}$

(b) Das minimale Vertex Cover beinhaltet die Knoten 2, 4 und 5. Da N niemals Knoten hinzufügt kann das minimale Vertex Cover nicht gefunden werden.

(c)
1. Betrachte Nachbar $\{1, 2, 3, 4, 7\}$
2. Betrachte Nachbar $\{1, 2, 4, 5\}$
3. Betrachte Nachbar $\{2, 4, 5\}$
4. $N'(\{2, 4, 5\}) = \emptyset$
   Termination

(d) Da die Verbesserung in jedem Schritt genau 1 beträgt macht die Wahl der Schrittfunktion keinen Unterschied. Bei MAX-CUT kann die Verbesserung einen beliebigen (auch negativen) Wert annehmen, wodurch dann auch die Wahl der Schrittfunktion einen Unterschied macht.