Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Laupichler
Nikolai Maas

3. Übungsblatt zu Algorithmen II im WS 2023/2024

https://algo2.iti.kit.edu/AlgorithmenII_WS23.php
{sanders, moritz.laupichler, nikolai.maas}@kit.edu

Aufgabe 1 (*Analyse:* preflow-push *Algorithmus (Wiederholung)*)

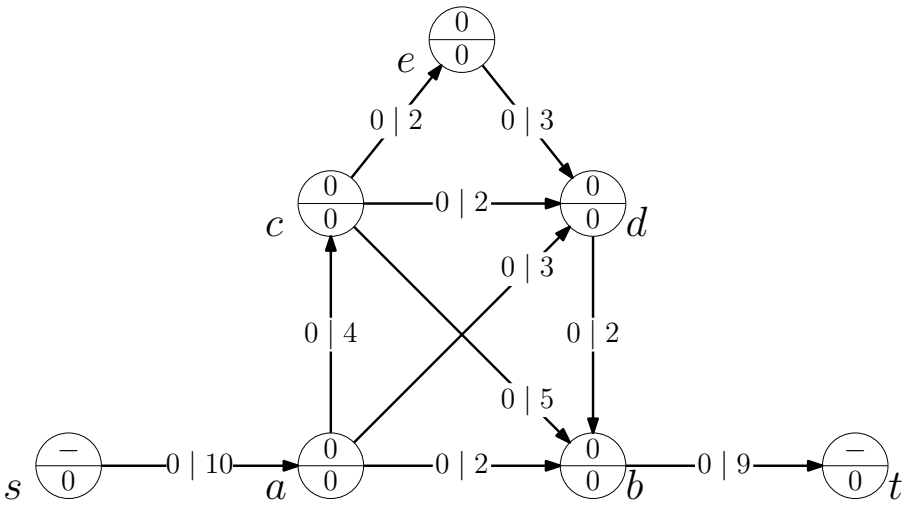
Sei durch S, T ein minimaler (s, t) Schnitt gegeben. Zeigen oder widerlegen Sie folgende Eigenschaften der Distanzfunktion (Label) nach Ausführung des *preflow-push*-Algorithmus:

a) $\forall v \in T : d(v) < n$

b) $\forall v \in S : d(v) \geq n$

Aufgabe 2 (*Rechnen: preflow-push Algorithmus*)

Gegeben sei folgender Flussgraph:



(das Haus vom Nikolaus mit Vorgarten und Hinterhof)

Knotenbeschriftung: Level (unten), Überschuss (oben)
Kantenbeschriftung: Fluss (vorne), Kapazität (hinten)

Bestimmen Sie den maximalen Fluss von s nach t mit dem generischen *preflow-push* Algorithmus.

Aufgabe 3 (*Analyse: Matchings*)

a) Zeigen oder widerlegen Sie. Existiert in einem bipartiten Matching ein maximaler alternierender Pfad, dessen erste und letzte Kante nicht Teil des Matchings sind, so hat das zugehörige Flussnetzwerk einen augmentierenden Pfad.

b) Gegeben ein bipartiter Graph $G = (L \cup R, E)$ mit $|L| = |R| = n$. Bezeichne $neigh(S) \subseteq R$ die Nachbarn der Knoten aus $S \subseteq L$. Zeigen Sie, gdw. ein perfektes Matching für G existiert, gilt für alle $S \subseteq L$: $|neigh(S)| \geq |S|$.

c) An einem Tanzkurs wollen n Männer und n Frauen teilnehmen. Jeder Teilnehmer kennt genau k Teilnehmer des anderen Geschlechts (alle Bekanntschaften sind beidseitig). Ist es möglich eine Paarung zu finden, in der jeder mit einem Bekannten tanzt?

Der Tanzkurs dauert genau k Wochen. Ist es möglich für jede Woche Paarungen zu bilden, so dass immer zwei Bekannte zusammen tanzen, sich aber keine Paarung wiederholt?

Aufgabe 4 (*Schlechter Zufall*)

Gegeben sei ein randomisierter Algorithmus badBit, der keine Eingabe liest und als Ausgabe zufällig mit Wahrscheinlichkeit p die Zahl 0 und mit Wahrscheinlichkeit $q = 1 - p$ die Zahl 1 liefert. Es sei $0 < p < 1$; der konkrete Wert von p sei aber unbekannt. Der Algorithmus gebe bei mehrfachem Aufruf unabhängige Ergebnisse aus.
Entwerfen Sie einen randomisierten Algorithmus fairBit, der keine Eingabe liest und als Ausgabe immer zufällig eine der Zahlen 0 und 1 mit Wahrscheinlichkeit 1/2 liefert.
Was können Sie über die Laufzeit Ihres Algorithmus sagen?

Aufgabe 5 (*Kleinaufgaben: Laufzeiten*)

Sei $T(n, \varepsilon)$ die Laufzeit eines Approximationsalgorithmus und $g(n, \varepsilon)$ seine Approximationsgarantie. Geben Sie für die folgenden Fälle an, ob der Algorithmus ein PTAS, FPTAS oder keines von beiden ist. Begründen Sie Ihre Antwort jeweils kurz.

- $T_1(n, \varepsilon) = \frac{1}{\varepsilon} \cdot (4n^3 + n^2)$, $\quad g_1(n, \varepsilon) = (1 - \varepsilon)$
- $T_2(n, \varepsilon) = \frac{1}{\varepsilon} \cdot n^2$, $\quad g_2(n, \varepsilon) = (1 + 2\varepsilon)$
- $T_3(n, \varepsilon) = \sqrt{n} + n^{\frac{3}{2}}$, $\quad g_3(n, \varepsilon) = 2 + \frac{1}{n}$
- $T_4(n, \varepsilon) = n \cdot \log \frac{1}{\varepsilon}$, $\quad g_4(n, \varepsilon) = (1 - \varepsilon)$
- $T_5(n, \varepsilon) = \varepsilon + e^{\log n} + n^5$, $\quad g_5(n, \varepsilon) = (1 + \varepsilon)$
- $T_6(n, \varepsilon) = n^{\frac{1}{\varepsilon}} + n^5$, $\quad g_6(n, \varepsilon) = (2 + \varepsilon)$

Anmerkung: Für $g_6(n, \epsilon)$ können Sie annehmen, dass der Approximationsalgorithmus mit beliebigem $\epsilon \in (-1, 0)$ spezifiziert werden kann.

Aufgabe 6 (*Analyse+Rechnen: Vertex-Cover*)

Gegeben sei folgender Algorithmus zur Berechnung eines *vertex cover* C für einen Graph $G = (V, E)$:

1. Initialisiere die Ergebnismenge $C = \varnothing$ als leere Menge.
2. Wähle Kante $(u, v) \in E$ beliebig.
3. Füge u, v zu C hinzu.
4. Entferne u, v und alle inzidenten Kanten aus G.
5. Wiederhole solange G noch Kanten hat

Nach Abschluss des Algorithmus ist $C \subseteq V$ ein *vertex cover*, d.h. für jede Kante $(u, v) \in E$ ist einer ihrer beiden Knoten in C. Falls o.b.d.A. $u \in C$ sagt man auch Knoten u *überdeckt* Kante (u, v).

a) Zeigen Sie, dass der angegebene Algorithmus ein korrektes *vertex cover* berechnet.

b) Geben Sie ein Beispiel an, in dem der Algorithmus ein minimales *vertex cover* liefert.

c) Geben Sie ein Beispiel an, in dem der Algorithmus kein minimales *vertex cover* liefert.

d) Zeigen oder widerlegen Sie, dass der Algorithmus eine 2-Approximation für *vertex cover* berechnet, d.h. dass er höchstens doppelt soviele Knoten auswählt als minimal nötig.

Betrachten Sie abschließend diesen alternativen Algorithmus zur Bestimmung einer 2-Approximation von *vertex cover*:

1. Initialisiere die Ergebnismenge $C = \varnothing$ als leere Menge.
2. Wähle Knoten $u \in V$ mit minimalem Grad.
3. Füge u zu C hinzu.
4. Entferne u und alle inzidenten Kanten aus G
5. Wiederhole solange G noch Kanten hat

Der Algorithmus berechnet offenbar –mit ähnlichen Argumenten wie in Teilaufgabe (a)– ein *vertex cover*. Es bleibt folgende Frage zu beantworten:

e) Zeigen oder widerlegen Sie, dass der Algorithmus eine 2-Approximation für *vertex cover* berechnet, d.h. dass er höchstens doppelt soviele Knoten auswählt als minimal nötig.

Aufgabe 7 (*Analyse: Metrisches k-Zentren Problem (*)*)

Gegeben sei eine Menge an Punkten $P \subset \mathbb{R}^2$ in der Ebene sowie eine Zahl $k > 0$. Gesucht ist eine k-elementige Teilmenge $K \subset P$ dieser Punkte, genannt Zentren, so dass für jeden Punkt $p \in P$ der maximale Abstand zu seinem nächstgelegenen Zentrum minimal ist.

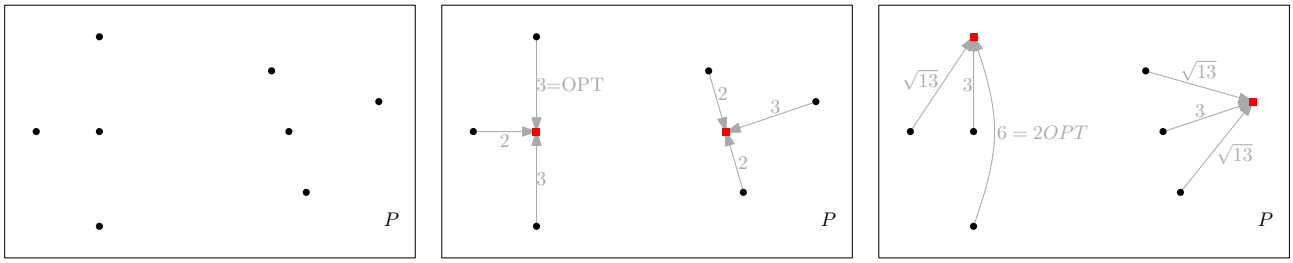
Es existiert folgender *greedy* Algorithmus, der eine 2-Approximation des Problems berechnet:

1. Wähle beliebigen Punkt aus P als erstes Zentrum

2. Wähle Punkt aus P als nächstes Zentrum mit größter Entfernung zu allen bisherigen Zentren (d.h. der den maximalen kürzesten Abstand zu einem Zentrum besitzt)

3. Wiederhole bis k Zentren gewählt worden sind

Das folgende Beispiel veranschaulicht die Problemstellung für $k = 2$:



Links ist eine Punktmenge P abgebildet. In der Mitte ist eine optimale Lösung zu sehen. Die roten Quadrate sind die ausgewählten Zentren. Die Kanten geben das nächstgelegene Zentrum für jeden Knoten sowie den Abstand an. Rechts ist eine weitere aber suboptimale Lösung aufgezeigt.

Zunächst einige allgemeine Fragen zu diesem Algorithmus:

a) Beschreiben Sie in Worten, welche Bedeutung OPT sowie die Aussage eine Lösung sei eine 2-Approximation des metrischen k-Zentren Problems, haben.

b) Handelt es sich bei dem angegebenen Algorithmus um ein PTAS, ein FPTAS oder um keines von beiden? Begründen Sie kurz.

Im Folgenden soll gezeigt werdem, dass der Algorithmus tatsächlich eine 2-Approximation berechnet. Dafür sind zunächst einige Vorüberlegungen nötig.

c) Zeigen Sie, bei einer Auswahl von $k + 1$ Punkten aus P existieren immer mindestens 2 Punkte, die das gleiche nächstgelegene Zentrum haben.

d) Gegeben eine optimale Lösung, wie groß kann der Abstand zwischen zwei Punkten maximal sein, wenn diese das gleiche nächstgelegene Zentrum besitzen?

e) In einer Lösung des *greedy* Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> l$. Zeigen Sie, dass l eine untere Schranke für den Abstand zwischen je zwei der Zentren $k_i,\ k_j \in K,\ i \neq j$ der Lösung darstellt.

f) Zeigen Sie mit obigen Aussagen, dass der angegebene *greedy* Algorithmus eine 2-Approximation für das Problem berechnet. Nehmen Sie dazu an, in der Lösung des Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> 2 \cdot OPT$, und führen Sie diese Aussage zum Widerspruch.

Hinweis: Machen Sie zunächst eine Aussage über die paarweisen Abstände von $k + 1$ speziell gewählten Punkten. Verwenden Sie anschließend einen Vergleich zu Abständen in der optimalen Lösung, um zum Widerspruch zu gelangen.

Ausgabe: 28.11.2023
Besprechung: 12.12.2023