Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Kobitzsch, Dennis Schieferdecker

6. Übungsblatt zu Algorithmen II im WS 2011/2012

http://algo2.iti.kit.edu/AlgorithmenII.php
{kobitzsch,sanders,schieferdecker}@kit.edu

Aufgabe 1 (*Kleinaufgaben: Laufzeiten*)

a) 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.

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

b) Sei $f(n, k)$ die Laufzeit eines Algorithmus mit n der Eingabegröße des Problems und k ein beliebiger Parameter. Geben Sie an welche der folgenden Laufzeiten ein Problem *fixed-parameter-tractable* machen. Begründen Sie Ihre Antwort jeweils kurz.

- $f_1(n, k) = 3k^2n^2$
- $f_2(n, k) = n^k \cdot k^2 \cdot \sqrt{n^e}$
- $f_3(n, k) = 3n^2 + 2nk$
- $f_4(n, k) = e^k \cdot \sqrt{n}$
- $f_5(n, k) = e^n \cdot \sqrt{k}$
- $f_6(n, k) = k^3 \cdot \log n^2$

c) (*) Gegeben seien folgende Rekurrenzrelationen zur Beschreibung von Laufzeiten. Bestimmen Sie (z.B. mit Hilfe erzeugender Polynome) das asymptotische Wachstum dieser Laufzeiten.

- $T_1(n, k) = 2T_1(n, k - 3) + T_1(n, k - 6)$
- $T_2(n, k) = T_2(n, k - 1) + T_2(n, k - 4) + n$
- $T_3(n, k) = 9T_3(n, k - 3) - 4T_3(n, k - 1)$
- $T_4(n, k) = T_4(n, k - 4) + T_4(n, k - 2) + n^2$

Aufgabe 2 (*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.

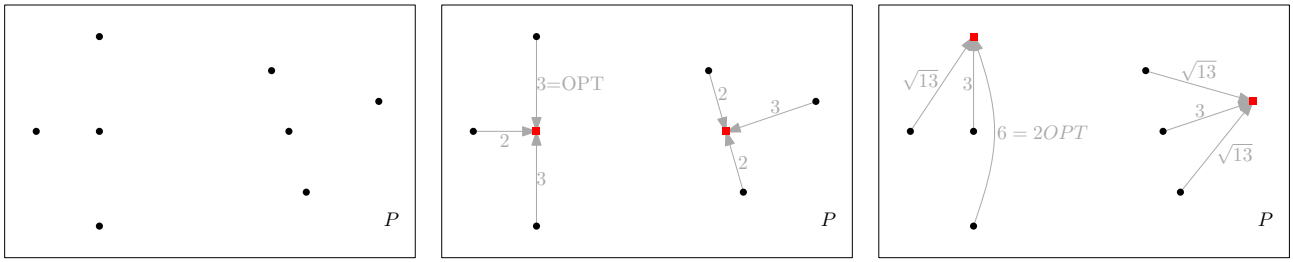
Aufgabe 3 (*Analyse: Metrisches k-Zentren Problem (*)*)

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?

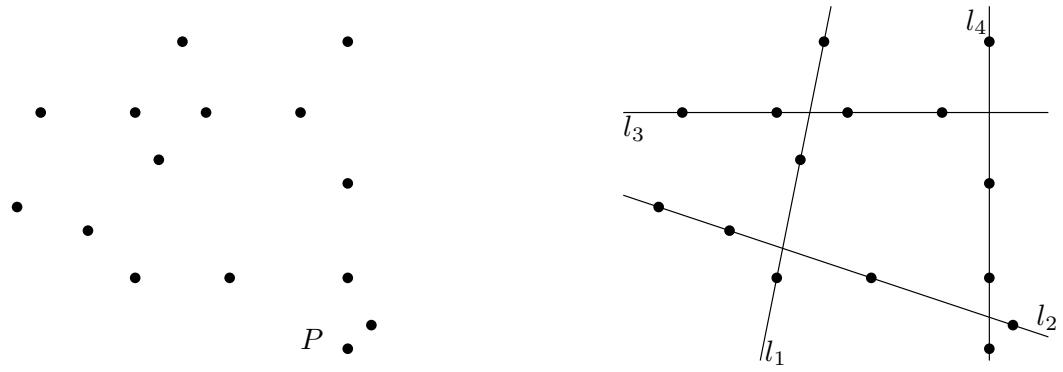
e) In einer Lösung des *greedy* Algorithmus sei der maximale Abstand eines Punktes $p \notin K$ zu seinem nächstgelegenen Zentrum $> l$. Zeigen Sie, dass l eine untere Schranke für den Abstand zwischen je zwei der Zentren $k_i,\ k_j \in K,\ i \neq j$ der Lösung darstellt.

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.

Aufgabe 4 (*Analyse:* line shooting *Problem*)

Gegeben seien n Punkte $P \subset \mathbb{R}^2$ in der Ebene sowie eine Zahl $k > 0$. Das *line shooting* Problem besteht darin zu bestimmen, ob es eine Menge L von k Geraden gibt, so dass jeder Punkt in P von mindestens einer dieser Geraden getroffen wird. Eine Probleminstanz wird durch das Tupel (P, k) charakterisiert.
In den Bildern sehen Sie links eine Punktmenge P und rechts eine mögliche Lösung für $(P, 4)$.



a) Begründen Sie kurz, warum eine Gerade l, die mehr als k Punkte trifft, Teil einer Lösung für die Probleminstanz (P, k) sein muss bei beliebigem P.

b) Begründen Sie kurz, warum die Instanz $(P, 3)$ des *line shooting* Problems keine Lösung besitzt mit P wie in obiger Abbildung.

c) Geben Sie einen Algorithmus an, der eine Instanz (P, k) des *line shooting* Problems exakt löst für beliebiges P. Bauen Sie dazu einen Suchbaum mit beschränkter Tiefe auf, der alle Kombination von k Geraden, die jeweils mindestens zwei Punkte treffen, generiert. Die Suchbaumtiefe und der Verzweigungsgrad sollen dabei polynomiell in k, der Aufwand pro Knoten polynomiell in $n = |P|$ sein.

   Hinweise: Verwalten Sie in jedem Knoten des Suchbaumes k Einträge, die jeweils bis zu zwei Punkte halten können (und damit eine Gerade definieren). Ein Suchbaum der Höhe $O(k)$ genügt.

d) Zeigen Sie, dass das *line shooting* Problem *fixed parameter tractable* bezüglich k ist. Geben Sie dazu die asymptotische Laufzeit Ihres Algorithmus in Abhängigkeit von n und k an.

Aufgabe 5 (*Analyse+Entwurf: Buffetplanung*)

Sie sind beauftragt worden, das Buffet für eine große Gala zu organisieren. Damit jeder geladene Gast auch eine seiner Lieblingsspeisen vorfindet, durften alle n Gäste eine Auswahl an d Gerichten angeben, die ihnen besonders schmecken. Der bisher für den Aufbau des Buffets vorgesehene Platz bietet allerdings nur Platz für k verschiedene Gerichte. Sie müssen nun entscheiden, ob der Platz ausreicht, so dass jeder Gast mindestens eines seiner angegebenen Gerichte vorfindet.

a) Formulieren Sie die Aufgabe als graphentheoretisches Problem (Stichwort: *Hypergraphen*).

b) Entwerfen Sie einen Algorithmus, der Ihr Entscheidungsproblem löst.

c) Stellen Sie eine Rekurrenzrelation für die Laufzeit Ihres Algorithmus auf und lösen Sie diese.

d) Ist Ihr Algorithmus *fixed parameter tractable*? Begründen Sie kurz.

Aufgabe 6 (*Analyse: ADAC Mitgliedschaft*)

Der "**A**utomobil **D**urch **A**lgorithmiker **C**lub" (ADAC) leistet auf Autobahnen Pannenhilfe. Ein Autofahrer hat in seiner Zeit als Verkehrsteilnehmer n Pannen, $n \in \mathbb{N}_{\geq 0}$, für die er die Hilfe des ADAC in Anspruch nehmen muss. Für jede geleistete Pannenhilfe verlangt der Club eine Aufwandsentschädigung abhängig von der Schwere der Panne. Mitglieder beim ADAC müssen lediglich ein Viertel dieser Kosten bezahlen. Eine lebenslange Mitgliedschaft kann man sich durch eine Einmalzahlung in Höhe von 1000 DM (**D**ijkstra **M**ark) sichern.
Da nicht schon mit Erwerb des Führerscheins klar ist, wie viele Pannen man in seinem Leben haben wird und wie schwerwiegend diese sein werden, stellt sich die Frage, ab wann es sich lohnt eine Mitgliedschaft beim ADAC zu erwerben.

a) Geben Sie eine Strategie an, die einen kompetitiven Faktor (competitive ratio) von ∞ erreicht. Begründen Sie kurz.

b) Wie gut ist die Strategie, sich nie eine Mitgliedschaft beim ADAC zu sichern? Begründen Sie.

c) Zeigen Sie, dass folgende Strategie einen kompetitiven Faktor $c = 3$ hat. Die Strategie ist, sich beim Pannenhelfer eine Mitgliedschaft zu kaufen, wenn die momentan von ihm bearbeitete Panne die Gesamtausgaben für Pannen (ohne Mitgliedschaft) auf über 500 DM anheben würde.

Hinweis: Verwenden Sie die summierten Gesamtkosten K über alle Pannen (ohne Mitgliederrabatt).

Aufgabe 7 (*Analyse: Online-gaming-Algorithmen*)

Angestellte in einem Rechenzentrum haben einen recht eintönigen Job. Ihnen stehen zwar die größten Rechner zur Verfügung, Sie dürfen diese aber nicht selbst verwenden. Stattdessen heisst es nur, die Maschinen möglichst gut auszulasten und Rechnungen zu schreiben. Ein ziemlich langweiliger Job möchte man meinen – zum Glück gibt es noch Computerspiele.
Ein Mitarbeiter hat zu Weihnachten ein neues Computerspiel erhalten, dass er liebsten andauernd spielen würde. Diesem Wunsch steht leider seine Arbeit im Wege. Eine neue Dienstanweisung besagt, dass die teuren Großrechner zu mindestens 50% ausgelastet sein müssen. Da das Scheduling in diesem Rechenzentren noch von Hand durchgeführt wird, muss der spielfreudige Mitarbeiter die laufenden Jobs überwachen und schnell neue Jobs auf leerlaufende Maschinen verteilen. So bleibt leider nur wenig Zeit zum Spielen am Arbeitsplatz.
Jeder Job hat eine Mindestlaufzeit von 5 Minuten. Die Zeit zum Verteilen der Jobs kann für die Bestimmung der Auslastung vernachlässigt werden. Der Mitarbeiter kann in dieser Zeit aber nicht spielen. Ein Job hat während seiner Ausführung die Maschine exklusiv. Es gibt keine automatischen Benachrichtigungen über das Ende eines Jobs. Der Mitarbeiter muss dies selbst periodisch überprüfen.

a) Entwerfen Sie einen *online scheduling* Algorithmus, der die freie Zeit des Mitarbeiters unter Einhaltung der Nebenbedingungen maximiert, wenn er einen Großrechner zu betreuen hat.

b) Zeigen Sie, dass Ihr Algorithmus optimal bzgl. der Freizeit des Mitarbeiters ist.

Ausgabe: 10.01.2012
Abgabe: keine Abgabe, keine Korrektur