Gedächtnisprotokoll

Algorithmentheorie - Sommersemester 2020

Klausur 20.07.2020

1.) Minimal Cost Flow (10 Punkte)

Matrix der Größe n x n gegeben. Es sollen Elemente mit maximaler Summe gewählt werden, so dass aus jeder Zeile und Spalte nur 1 Element gewählt wird.
Bsp:

```
    | (6) 4   2  |

M = |  3  4  (8) |

    |  2 (6)  5  |
```

Netzwerk bauen, Knoten, Kanten, Kapazitäten und Kosten angegben und erklären wie die Anwort abzulesen ist.

2.) (I) LP (10 Punkte ?)

a) Aufgabe (I)LP angeben

Es gibt m Drucker und n Veranstaltungen mit 1 Klausur, die gedruckt werden muss. Jede Veranstaltung i hat eine Druckzeit von t_i für alle Klausuren. Die Druckeraufträge können nicht auf mehrere Drucker aufgeteilt werden.
Jeder Drucker kann nur eine Klausur gleichzeitig drucken.
Ziel: Gesamtlaufzeit (C) (bis letzte Klausur gedruckt ist) soll minimal werden.
Hinweis: Verwenden Sie eine Variable C für die Gesamtlaufzeit.

3.) NP-schwere Probleme Spezialfall (8 Punkte ?)

Gegen ist Graph G = (V,E) und k Größe der zu suchenden Clique, Frage: Gibt es eine Clique der Größe k in G?. Das NP-schwere Problem kann in Spezialfällen in polynomieller Zeit gelöst werden. Zeigen Sie, dass dies in einem der folgenden Fälle geht:

3.1) Graph G hat n=m Kanten, d.h. Anzahl Knoten = Anzahl Kanten

3.2) Jede Zusammenhangskomponente hat maximal 7 Knoten

4.) 3-Colouring (10 Punkte ?)

3 Colouring kann in O(3^k (m+n)) gelöst werden, wenn k gegeben und die Anzahl der Knoten im minimalen Vertex Cover auf G entspricht. Zeige, dass das stimmt
(Hinweis: Nehmen Sie an, dass ein Vertex Cover der Größe k zur Eingabe gehört)

5.) Approximation (12 Punkte ?)

Knapsack Problem gegeben mit Beispiel
B = 10 (also das Fassungsvermögen des Rucksacks)

```
Objekt   | o1 | o2 | o3 | o4 | o5

Wert     |  2 |   5  |   3 |  8  | 4

Gewicht  | 4  |  6   |   2 |  7  |  5
```

5.1) Gib ein optimales Beispiel ohne Beweis an

5.2) Gegeben ist folgender Algorithmus

```
Lösche alle Objekte mit Gewicht > B

Sortiere Objekte absteigend nach Gewicht

Wähle k max, dass gilt sum_k Gewicht(j) <= B

Wenn sum_k Wert(j) >= Wert(k+1) gibt Objekt(k+1) zurück

sonst Objekte(1..k)
```

Zeige, dass dieser Algorithmus eine 3er Approximation ist (er ist auch eine 2er Approximation)