University of Konstanz
Department of Computer & Information Science
Sabine Cornelsen / Jan Christoph Athenstädt

Combinatorial Optimization
Summer 2014

Assignment 0

Post Date: 25 April 2013 **Due Date:** – none – **Tutorial:** 30 April 2013
You are permitted and encouraged to work in groups of two.

Problem 1: Matrix Multiplication

Let $\mathbf{A}$ be an $n \times m$ matrix and $\mathbf{B}$ an $m \times p$ matrix. The matrix product $\mathbf{A} \cdot \mathbf{B}$ or simply $\mathbf{AB}$ is an $n \times p$ matrix where the entry i, j is defined as:

$$(\mathbf{AB})_{ij} := \sum_{k=1}^{m} \mathbf{A}_{ik}\mathbf{B}_{kj}$$

(a) Calculate $\begin{pmatrix} 1 & 2 & 5 & 8 \\ 1 & 6 & 4 & 3 \end{pmatrix} \cdot \begin{pmatrix} 1 & 5 & 8 \\ 6 & 4 & 3 \\ 4 & 8 & 2 \\ 3 & 9 & 3 \end{pmatrix}$.

(b) Which of the following properties of scalar multiplication hold for matrix multiplication as well?

- commutative
- distributive (on matrix addition)
- associative

Let $\mathbf{A}$ be a $n \times m$ matrix. The *transpose* $\mathbf{A}^T$ of $\mathbf{A}$ is defined as the $m \times n$ matrix in which each entry i, j of $\mathbf{A}$ becomes the entry j, i in $\mathbf{A}^T$. (Rows and columns are "swapped".)

(c) Show that $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$.

Problem 2: Determinants

A very useful number associated with a square matrix $\mathbf{A}$ is its *determinant* $|\mathbf{A}|$. For 2×2 and 3×3 matrices, we can use the following formulas:

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} = aei + bfg + cdh - ceg - bdi - afh$$

The development of the above formula is an example of the *Laplace expansion* along the first row:

Let $\mathbf{A}$ be an $n \times n$ matrix and $\mathbf{S}_{ik}$ be the $(n-1) \times (n-1)$ matrix that results from deleting the ith row and the kth column. We can use an arbitrary row i or column k to perform the Laplace expansion:

$$|\mathbf{A}| = \sum_{i=1}^{n} a_{ik} \cdot (-1)^{i+k} |\mathbf{S}_{ik}| = \sum_{k=1}^{n} a_{ik} \cdot (-1)^{i+k} |\mathbf{S}_{ik}|$$

(a) Calculate $\begin{vmatrix} 3 & 4 & 1 \\ 6 & 2 & 4 \\ 6 & 3 & 4 \end{vmatrix}$.

(b) Calculate $\begin{vmatrix} 3 & 4 & 0 & 1 \\ 7 & 8 & 1 & 2 \\ 6 & 2 & 2 & 4 \\ 6 & 3 & 0 & 4 \end{vmatrix}$.

Problem 3: Asymptotic Running Time

Let $f(n)$ and $g(n)$ be two functions. We say that $f(n) \in \mathcal{O}(g(n))$ if and only if there exists a constant $C > 0$ such that $|f(n)| \leq C|g(n)|$ for all sufficiently large values of n.

(a) Which of the following statements are true?

- $7n^4 + 3n^{1.5} + 100^{100} \in \mathcal{O}(n^5)$
- $3n^2 \cdot \log n \in \mathcal{O}(n^2)$
- $3n^2 \cdot \log n \in \mathcal{O}(n^3)$
- $3n^2 + \log n \in \mathcal{O}(n^2)$
- $n \in \mathcal{O}(n!)$

(b) Calculate the asymptotic running time of Algorithm 1.

(c) What is the asymptotic running time of computing the determinant of an $n \times n$ matrix using the Laplace expansion? Do you know a better way?

Algorithm 1: A sorting algorithm

```
Data: Array A
repeat
    swapped ← false;
    for i ← 1 to n − 1 do
        if A[i] > A[i − 1] then
            swap A[i] and A[i − 1];
            swapped ← true;
until swapped = false;
```

Problem 4: Shortest Paths

Let $G = (V, E)$ be a directed graph with non-negative edge-lengths. Given a node $a \in V$, how can we find the shortest path from the node to all other nodes?

Dijkstra's Algorithm solves this problem. It is one of the classical and most famous algorithms in computer science. Here is a textual description of the algorithm:

- Assign the distance-value 0 to a and ∞ to all other nodes
- Initialize a queue with all nodes and set their previous-attribute to undefined
- As long as there are still nodes with non-infinite distance-value in the queue: select the node u with the smallest distance-value
    - Take every neighbor v of u and compare its distance-value to the distance-value of u plus the length of the edge (v, u). If the distance-value of v is greater, update it to u plus the length of the edge (v, u) and set the previous-attribute of v to u.
    - Remove u from the queue.
- The shortest path from any node b to a can be found by following the chain of nodes stored in the previous-attributes starting at b.

(a) Transform the textual description of the algorithm into pseudo-code (see Problem 3 for an example).

(b) What is the running time of the algorithm (in $\mathcal{O}$-notation) when using a simple list as queue ($\mathcal{O}(n)$ to get the element with the smallest distance-value)?

(c) Let $d(a, b)$ denote the distance of the shortest path between nodes a and b. If $d(u, v) = 9$ and $d(u, w) = 12$, what is the minimum length of the edge (v, w)?

(d) Show that every subpath of a shortest path must be a shortest path itself.