Tentamen
TDDB56 DALGOPT-D Algoritmer och Optimering
8 december 2001, kl 8-13

Lokal: Garnisonen
Examinator: Christoph Kessler
Jourhavande lärare:
Christoph Kessler (070-3666687, 282406), visiting at 09:30 and 11:30,
Kaj Holmberg ( 0708-887246), visiting at 09:30 and 11:30
Visning: date/time to be announced on the course homepage (probably end of week 2/2002)

Hjälpmedel / Admitted material:
Miniräknare / pocket calculator
Engelsk-Svensk ordbok / Swedish-English dictionary
Kursböckerna DALG:

Årets kurslitteratur OPT:
- Papadimitriou, Steiglitz: Combinatorial Optimization,

Kurslitteratur för TAOP13 (2001):
- Linjär och ickelinjär optimering, Lundgren, Rönnqvist och Värbrand.
- Flöden i nätverk och kombinatorisk optimering, Holmberg.
- Kombinatorisk optimering II, Holmberg.
- Utdrag ur Kombinatorisk Optimering, problem och algoritmer, Migdalas and Göthe-Lundgren.

Kurslitteratur för TAOP13 (2000):
- Kombinatorisk Optimering, problem och algoritmer, Migdalas och Göthe-Lundgren.
General instructions

- This exam has seven assignments and six pages in total. Read all assignments carefully and completely before you begin.
- Use a separate sheet for each assignment. Mark each sheet on top with your name, personnummer, and the course code (TDDB56).
- You may answer in either English or Swedish.
- Write clearly. Unreadable text will be ignored.
- Be precise in your statements. Unprecise formulations may lead to a reduction of points.
- Motivate clearly all statements and reasoning.
- Explain calculations and solution procedures.
- The assignments are not ordered according to difficulty.
- If you use some result from the admitted literature that has not been part of the lectures, give a reference (book title, page) or give an explanation.

Betygsgränser (preliminary)

\[0p, 20p] \rightarrow U
\[20p, 28p] \rightarrow 3
\[28p, 34p] \rightarrow 4
\[34p, 40p] \rightarrow 5

Liten optimeringsordlista

Adjacent = närliggande
Arc = båge
Basic solution = baslösning
Branch-and-bound = trädsökning
Complementarity = komplementaritet
Constraints = bivillkor
Feasible set = tillåtet område
Integer = heltal
Minimal spanning tree = billigaste uppspännande träd
Objective function = målfunktion
Satisfied = uppfylld
Shortest path = billigaste väg
Starting from scratch = börja om från början
1. (1p) Which of the following three tasks has the asymptotically (i.e., for large $n$) fastest worst-case execution time?

(a) Sorting $n \cdot \log_2 n$ numbers with mergesort
(b) Sorting $n - \log_2 n$ numbers with insertion sort
(c) Sorting $n / \log_2 n$ numbers with selection sort

Solution proposal:
(a) takes time $O(n(\log n)^2)$ and is thus asymptotically fastest. (b) takes time $O(n^2)$. (c) takes time $O(n^2/(\log n)^2)$.

2. Consider the following LP-problem.

\[ \begin{align*}
\text{max } z &= 3x_1 + 2x_2 \\
2x_1 + 2x_2 &\leq 5 \\
2x_1 + 3x_2 &\leq 6 \\
x_1, x_2 &\geq 0 
\end{align*} \]

(a) (2p) Solve the problem with the simplex method. Give the primal and dual optimal solutions.
(b) (4p) Formulate the LP dual to the problem. Draw the feasible set for both the primal and dual problems, and mark the optimal solutions. Show that the complementarity conditions are satisfied. (Graphical motivation is allowed.) Decide how much the right-hand-side in the first (primal) constraint can be changed, without changing the optimal basic solution. Give a graphical motivation in the dual space.
(c) (2p) Add the requirement that $x_1$ and $x_2$ shall be integers. Solve the problem with Land-Doig-Dakin’s branch-and-bound method. (Here two-dimensional LP-problems may be solve graphically. Hint: Remember that $z$ will be integer.)

3. Selection:

(a) (2p) Give an algorithm (pseudocode) that finds the second-largest element in an unsorted array $a$ of floatingpoint numbers in time $O(n)$ in the worst case.
   Hint: Your algorithm will probably also compute the largest element as a byproduct.
(b) (2p) Give an algorithm (pseudocode) that finds both the largest and the smallest element of an unsorted array $a$ of $n$ floatingpoint numbers with $3\lceil n/2 \rceil$ (floatingpoint) comparisons. Determine formally the number of comparisons made by your algorithm (calculation). Assume for simplicity that all $n$ array elements $a_i$ are mutually different, i.e. $a_i \neq a_j$ for $i \neq j, 1 \leq i, j \leq n$.
   Hint: Imagine the array elements to represent the strengths of national soccer teams. A comparison of two numbers could be regarded as a match between two teams. How are the matches in the final rounds of the soccer world championship organized to find out the strongest team (for simplicity assuming an equal strength of playing of each team during the entire tournament)?
(c) (2p) Give an algorithm (pseudocode) that finds both the largest and the second-largest element in an unsorted array $a$ of floatingpoint numbers using only $n + \lceil \log_2 n \rceil - 2$ (floatingpoint) comparisons. Determine formally the number of comparisons made by your algorithm (calculation). Assume for simplicity that all $n$ array elements $a_i$ are mutually different, i.e. $a_i \neq a_j$ for $i \neq j, 1 \leq i, j \leq n$.
   Hint: In the soccer world championship, does the so-called “vice-champion”, i.e. the loser in the final match, always deserve this title? If not, which teams are the candidates for this title?
(d) (2p) What is the worst-case execution time of the QuickSelect algorithm, applied to an unsorted array of size $n$? Justify your answer (calculation).
4. A good understanding of a problem, and solution methods, can often give a possibility of deducing what happens to the optimal solution if input data is unexpectedly changed, and how one can, without solving the problem from scratch, calculate the new optimal solution.

Describe in detail for all the cases below (mathematical notation is encouraged) how one, in an algorithmic way, checks if the previous optimal solution is still optimal, and finds the new optimal solution, without starting from scratch.

Start from the previous optimal solution. You must motivate why the correct optimal solution is obtained. (Hint: Use standard methods as much as possible.)

(a) \((1.5p)\) Shortest path problem: A new arc is added.
(b) \((1.5p)\) Minimal spanning tree problem: A new node (and adjacent arcs) is added.
(c) \((2p)\) Minimal spanning tree problem: A node (and adjacent arcs) is removed.
(d) \((2p)\) Minimal spanning tree problem: A new arc is added.
(e) \((2p)\) Minimum cost network flow problem: The cost of an arc is changed.
(f) \((1p)\) Maximum flow problem: A new arc is added.

5. Trees:

(a) \((1.5p)\) For each of the following binary trees \(T_1, T_2, T_3\) storing integer keys, determine whether it is
   (i) a complete binary search tree?
   (ii) an AVL tree?
   Justify your answers!

Solution proposal:
(T1) some complete binary tree, not AVL (picture)
(T2) some non-complete binary tree, AVL, (picture)
(T3) some tree with AVL property, not binary search tree. (picture)

(b) \((1p)\) Draw an AVL tree of height 4 with a minimum number of nodes, where the balance of the root node is \(-1\).

(c) \((2p)\) Given the following AVL tree storing characters as keys (lexical ordering assumed):

\[
\text{T} \\
\text{g} \\
\text{b} \\
\text{a} \\
\text{e} \\
\text{c} \\
\text{f} \\
\text{h} \\
\text{i} \\
\text{d} \\
\text{c} \\
\text{o} \\
\text{d} \\
\text{n}
\]
Draw the tree resulting after insertion of $d$ and show the node balances for all inner nodes. Which node is the critical node? (1p)
Which operation(s) is/are needed to rebalance the tree? Show the resulting tree with node balances. (1p)

6. (2p) Give an undirected graph with 5 nodes and (at least) 6 arcs (no parallel) that does not contain any Euler cycle or Hamilton cycle. Explain why.

7. Matrix-matrix multiplication:

The product $C$ of two $n \times n$ matrices $A$ and $B$ is defined as

$$C = A \times B, \quad c_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n$$

(a) (1.5p) Give a pseudocode program that computes the matrix product according to this formula, assuming that $A$, $B$ and $C$ are represented by two-dimensional arrays of $n \times n$ elements. Analyze the (worst-case) run time of this program as a function of $n$ (calculation) and give it in terms of the big-Theta notation.

(b) (0.5p) What is the time complexity of elementwise adding ($C = A + B$) or subtracting ($C = A - B$) two $n \times n$ matrices?

(c) (3p) Apply a divide-and-conquer approach to the matrix-matrix multiplication problem as follows:
Assume for simplicity that $n$ is a power of 2. We split the $n \times n$ operand matrices $A$ and $B$ and the result matrix $C$ in four quarter submatrices of size $n/2 \times n/2$ each:

$$
\begin{pmatrix}
C^{0,0} & C^{0,1} \\
C^{1,0} & C^{1,1}
\end{pmatrix}
= 
\begin{pmatrix}
A^{0,0} & A^{0,1} \\
A^{1,0} & A^{1,1}
\end{pmatrix}
\times 
\begin{pmatrix}
B^{0,0} & B^{0,1} \\
B^{1,0} & B^{1,1}
\end{pmatrix}
$$

(1)
Assume for simplicity that splitting can be done in place and takes constant time. Obviously it is possible, after recursive application of the algorithm to these quarter matrices, to compute $C$ using eight $(n/2 \times n/2)$ matrix–matrix multiplications and four $(n/2 \times n/2)$ matrix additions:

$$C^{u,v} = A^{u,0} \times B^{0,v} + A^{u,1} \times B^{1,v}, \quad u = 0, 1, \ v = 0, 1$$

(2)
Note that the multiplication of $1 \times 1$ matrices is identical to the usual multiplication of two numbers.
Identify the divide, conquer, and combine phase of the algorithm and give a description of the algorithm using a recursive function (exact pseudocode is not required). (1p)
Analyze the worst-case time complexity of this algorithm (calculation). (2p)
Is it asymptotically faster than the standard method above? (0p)

(d) (1.5p) Strassen (1969) found out that, by spending some extra matrix additions and subtractions, the quarter matrices of $C$ can be computed from the quarter matrices of $A$ and $B$ by only seven $(n/2 \times n/2)$ matrix multiplications and a few matrix additions and subtractions:
Then, $C$ can be assembled from $Q^0, \ldots, Q^6$ by a few matrix additions and subtractions as follows:

$$
Q^0 \leftarrow (A^{0,0} + A^{1,1}) \times (B^{0,0} + B^{1,1}) \\
Q^1 \leftarrow (A^{1,0} + A^{1,1}) \times B^{0,0} \\
Q^2 \leftarrow A^{0,0} \times (B^{0,0} - B^{1,1}) \\
Q^3 \leftarrow A^{1,1} \times (B^{1,0} - B^{0,0}) \\
Q^4 \leftarrow (A^{0,0} + A^{0,1}) \times B^{1,1} \\
Q^5 \leftarrow (A^{1,0} - A^{0,0}) \times (B^{0,0} + B^{0,1}) \\
Q^6 \leftarrow (A^{0,1} - A^{1,1}) \times (B^{1,0} + B^{1,1})
$$

(3)

Now this method is applied recursively to the matrix multiplications needed to compute the $Q^i$, $0 \leq i \leq 6$ in the equations (3).

Analyze the worst-case complexity of this recursive algorithm (calculation).