NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

ENGLISH

FINAL EKSAM IN

TDT4120

ALGORITHMS AND DATA STRUCTURES

Saturday August 12th, 2006
Kl. 09.00 – 13.00

Contakt during exam:

Magnus Lie Hetland, ph. 918 51 949

Tools:

No printed or handwritten materials allowed. Specified, simple calculator allowed.

Grading date:

September 2nd, 2006. Results will be available from http://studweb.ntnu.no og and the grade phone 815 48 014.

Important:

Please read the entire exam text before you begin. Read the problems thoroughly. Points are given for each subproblem. Make your assumptions clear when necessary. Please write short and concise answers in the boxes on the exam. Long explanations that do not directly answer the questions will count only marginally.

Problem 1 (20%)

Each of the following subproblem consists of 5 statements. By each of these you are to check the box marked “yes” or “no”. Check “yes” if you believe the statement is true, and “no” if you believe the statement is false or incorrect (or does not make sense). Do not check “no” if you *agree* with a statement that uses the word “not”. Then you should check “yes”, meaning that the statement is *correct*.

Each statement where the checkmark is placed correctly gives 1 point, a question without a checkmark gives 0 points, and a statement where the checkmark is placed incorrectly gives –1 point. A negative sum for a subproblem (a–e) is rounded up to 0.

Please read the problems thoroughly. The text is constructed so that the answers should not be obvious. Answer only if you are certain that you have understood the problem and that you know the answer.

a) Consider the following claims about the sorting algorithm MERGESORT.

[ ] [ ] Yes No 1. It exploits overlapping subproblems.

[ ] [ ] Yes No 2. It is asymptotically more efficient than HEAPSORT.

[ ] [ ] Yes No 3. It has a worst-case running time of $\Theta(n^2)$.

[ ] [ ] Yes No 4. It is well suited for recursive implementation.

b) Consider the recurrence $T(n) = 2T(\lfloor n/2 \rfloor) + n$.

[ ] [ ] Yes No 1. The recurrence is typical for dynamic programming.

[ ] [ ] Yes No 2. The recurrence is typical for *divide and conquer* algorithms.

[ ] [ ] Yes No 3. $T(n) \in \Omega(n \log n)$.

[ ] [ ] Yes No 4. The recurrence describes, among other things, the best-case running time of QUICKSORT.

c) Consider the following statements about the *selection* problem.

[ ] [ ] Yes No 1. It can be solved in $O(n)$ time in the worst case.

[ ] [ ] Yes No 2. It requires sorting the data set.

[ ] [ ] Yes No 3. RANDOMIZEDSELECT is a greedy algorithm.

[ ] [ ] Yes No 4. RANDOMIZEDSELECT uses memoization.

d) Consider a problem P that can be solved with a greedy algorithm.

[ ] [ ] 1. It will be advantageous to use dynamic programming to solve *P*.
Yes No

[ ] [ ] 2. Locally optimal choices will give an optimal solution to *P*.
Yes No

[ ] [ ] 3. The problem *P* has optimal substructure.
Yes No

[ ] [ ] 4. The running time to solve *P* can be more than linenar.
Yes No

e) Consider the problem VERTEX-COVER — finding a *vertex cover* of size *k* on a graph $G = (V, E)$.

[ ] [ ] 1. The size of a cover is the number of nodes it consists of.
Yes No

[ ] [ ] 2. There is a known reduction from VERTEX-COVER to HAM-CYCLE.
Yes No

[ ] [ ] 3. There is a known reduction from VERTEX-COVER to MINIMAL-SPANNING-TREE.
Yes No

[ ] [ ] 4. The parameter *k* cannot be greater than $|E|/2$.
Yes No

Problem 2 (35%)

a) Describe FLOYD-WARSHALL with pseudocode.

Answer (10%):

b) Consider an undirected, connected, weighted graph *G*. Assume that all the edges of *G* have different weights. Explain why the lightest edge (the one with the lowest cost) in any cut in an arbitrary cut in *G* must be part of a minimal spanning tree over *G*.

Answer (8%):

FORD-FULKERSON is a general method for finding maximum flow in a flow network. The general method does not have a guaranteed polynomial running time. EDMONDS-KARP is a variation of FORD-FULKERSON that has a polynomial running time because of the way it selects augmenting paths.

c) How does EDMONDS-KARP select its augmenting paths? (Please answer briefly.)

Answer (8%):

Consider the following table, *A*: [2, 2, 4, 0, 3, 4, 3]. This table is to be sorted with COUNTING-SORT. Assume that you know beforehand that the values in the table only can fall in the range from 0 to 5, inclusive. Assume that your counting table is called *C* and the result table is called *B* (as in the textbook).

e) Report the contents of tables *B* og *C* for each number inserted into *B* (that is, *right after* the number has been inserted, and *C* has been updated). Only report the first 3 steps. Use a dash ("–") for any unused slots in the tables. Note that a number already has been added to *C*.

Answer (9%):

Trinn 1:

B: [, , , , , ,]

C: [, , 3, , ,]

Trinn 2:

B: [, , , , , ,]

C: [, , , , ,]

Trinn 3:

B: [, , , , , ,]

C: [, , , , ,]

Problem 3 (20%)

Consider the following pseudocode:

PLOFFSKIN(*gromboolian*, *chankly*, *bore*)
if *chankly* ≥ *bore*
    return 42
for *pelican* ← *chankly* … *bore*
    print *gromboolian*[*pelican*]
jee ← ⌈(*bore* – *chankly*) / 4⌉
PLOFFSKIN(*gromboolian*, *chankly*+*jee*, *bore*–*jee*)

a) Assume that A is a 1-indexed table with size n. What is the running time of PLOFFSKIN(A, 1, n)? Give the running time in Θ notation. Show the main steps in your calculation.

Answer (10%):

The following is a variation of Dijkstra's algorithm that is a bit different from the algorithm as presented in the textbook. Assume that the queue Q is implemented as an unordered, linked list.

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
    d_min ← ∞
    for each vertex v ∈ Q
        if d[v] < d_min
            d_min ← d[v]
            u ← v
    Q ← Q − {u}
    S ← S ∪ {u}
    for each vertex v ∈ Adj[u]
        RELAX(u, v, w)
```

b) What is the running time of the algorithm? Give your answer in Θ notation. Briefly describe your reaoning.

Answer (10%):

Reasoning:

Oppgave 4 (25%)

The store chain “8–12” is building stores along Main Street, which runs east–west through Metropolis city center. They have been offered to buy several buildings, with positions $x[1] \ldots x[n]$ (in meters from the beginning of the street, in increasing order from the west end). The cost of buying these buildings is ignored; only the long-term profit is to be considered. The yearly profit for a store depends on its placement, and the profit for position $x[i]$ is given by $f[i] > 0$.

The city council has put its foot down, and decided that there must be at least 500 meters between the stores. You have been assigned the task of calculating the legal selection of store positions that gives the greatest total yearly profit.

Preliminary analyses from the store chain’s own computer people indicates that you should calculate a table $y[1] \ldots y[n]$, where $x[y[i]]$ is the position of the eastmost house west of $x[i]$ that has a legal distance to $x[i]$. If $y[i] = 0$, this means there are no legal positions west of $x[i]$.

a) Describe briefly an algorithm for calculating the table y as efficiently as possible. What is the running time? (Give your answer in Θ notation.)

Answer (10%):

Running time:

b) Assume that the table *y* from the previous subproblem is available (that is, that it has already been calculated). Describe briefly an algorithm which, as efficiently as possible, finds the optimal profit with legally placed stores. What is the running time? (Give your answer in Θ notation.)

Answer (15%):

Running time: