NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for datateknikk og informasjonsvitenskap

ENGLISH

FINAL EKSAM IN

TDT4120/IT1105

ALGORITHMS AND DATA STRUCTURES

Friday December 8th, 2006
09.00 – 13.00

Contact during the exam:

Magnus Lie Hetland, ph. 918 51 949

Tools:

Any printed and handwritten tools allowed. Specific, simple calculator allowed.

Grading date:

January 8th. Results will be available from http://studweb.ntnu.no and the grade phone 815 48 014.

Important:

Read the entire exam before you start, plan your use of time, and prepare any questions by the time the lecturer does his rounds. Read the problems thoroughly. The percentages indicate how much weight each problem is given when grading the exams. Make assumptions where needed. Keep your answers short and precise. Please fill in your answers in the answer form (i.e., please don't submit extra sheets, unless absolutely necessary). Feel free to use a pencil (and eraser). Long explanations that don't directly answer the questions may be partially or completely ignored.

Answer form

Please fill in your answers in the spaces below. The problem texts begin on page 4.

1a (6%):
1b (6%):
1c (6%):
1d (6%):
1e (6%):
2a (10%): $D^{(0)}$ … $D^{(4)}$: $\begin{pmatrix} 0 & & \\ & 0 & \\ & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & \\ & 0 & \\ & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & \\ & 0 & \\ & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & \\ & 0 & \\ & & 0 \end{pmatrix}, \begin{pmatrix} 0 & & \\ & 0 & \\ & & 0 \end{pmatrix}$
3a (5%):
3b (5%):
3c (5%): $T(n) = \Theta($ $)$.

4a (5%):	
4b (3%): Θ( )	
4c (7%):	
4d (5%): Θ( ).	
5a (5%):	
5b (5%):	
6a (2%):	6b (3%):
6c (10%): Running time: Θ( )	

Problem 1 (30%)

Keep your answers to the following questions as short and precise and possible.

a) If the input numbers are not uniformly distributed, BUCKET-SORT may not have a linear running time. Why is this?

b) If c is the table used in LCS-LENGTH(X, Y), what does the number $c[i, j]$, represent, where $1 \leq i \leq length[X]$ and $1 \leq j \leq length[Y]$?

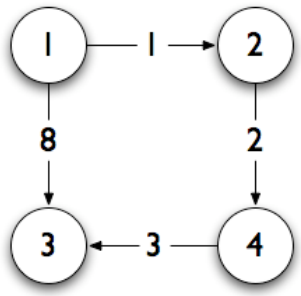
c) Why is it pointless to use a (binary) heap as the priority queue in DIJKSTRA if the graph is complete, i.e., the number of edges is $\Theta(V^2)$?

d) Assume that you have a linear program that contains a variable x that may be negative. What would you do with this variable if the program is to be converted to standard form?

e) Give a brief argument that HAM-CYCLE may be seen as a special case of TSP.

Problem 2 (10%)

Consider the following graph $G = (V, E)$:



You are to simulate/execute FLOYD-WARSHALL on the graph G.

a) Fill in the arrays/matrices $D^{(0)}$ … $D^{(4)}$ in the answer form.

Problem 3 (15%)

Assume that the statement "Algorithm A has a running time of $O(n^2)$" gives a tight upper bound for the running time of A.

a) Does this entail that A has a running time of $\Theta(n^2)$? Explain your reasoning.

b) Does this entail that A has a *worst-case* running time of $\Omega(n^2)$? Explain your reasoning.

c) Solve the recurrence $T(n) = 16T(n/4) + n^2$. Give your answer in Θ notation. Explain your answer very briefly.

Problem 4 (20%)

Consider the following algorithm:

```
WHATZIT(A, B):
for i ← 1 … n
      B[i] ← 0
for i ← 1 … n
      B[A[i]] ← B[A[i]] + 1
```

Assume that A and B are arrays with length n that can contain integers in the value range $1 \ldots n$. Assume that the first elements of both A and B have index 1.

a) What does the algorithm WHATZIT do? (I.e., what is its function/purpose? A direct reformulation of the pseudocode in your own words will not necessarily be given any credit.)

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

Consider the following algorithm:

```
WAZZUP(A, a, b, c):
if c – b < 5
        return b, c
x = b + 5
y = c – 5
if a ≥ A[x]
        return WAZZUP(A, a, x, c)
else
        return WAZZUP(A, a, b, y)
```

Assume that A is a sorted table with n integers, and that a, b and c are integer parameters.

c) What does the algorithm WAZZUP do if it is called as WAZZUP(A, a, 1, n)? (I.e., what is its function/purpose? A direct reformulation of the pseudocode in your own words will not necessarily be given any credit.) Assume that a is a number in A.

d) What is the running time of the algorithm WAZZUP if it is called as WAZZUP(A, a, 1, n)? Assume that a is a number in A. Give your answer in Θ notation. Briefly explain your answer.

Problem 5 (10%)

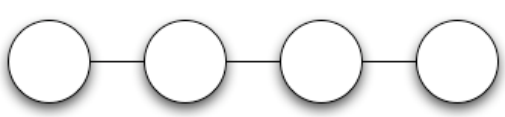
Please keep your answers to the following questions as short and precise as possible.

a) Professor Znurrebarth has designed an algorithm to compute square roots (with limited precision). The running time to compute the square root of the integer n with his algorithm is $\Theta(\lg n)$. He claims that this running time is sublinear (asymptotically faster than linear). Do you agree or disagree? Explain your reasoning.

b) You have observed that the computer game *Slurm Invaders* is very similar to the NP complete problem VERTEX-COVER, and you intend to use this similarity to show that *Slurm Invaders* is NP complete. How would you proceed? You may assume that you already have shown that *Slurm Invaders* is in the set NP. (Please keep your answer brief.)

Problem 6 (15%)

En *independent set* of nodes in a graph $G = (V, E)$ is a subset I of V such that there are no edges in E between any of the nodes in I. Normally, finding an independent set is hard (NP complete) — but assume here that the graph G has a particular structure: The nodes may be ordered as a sequence $[v_1, v_2, \ldots, v_n]$, and there are edges only between neighbors in this sequence, as illustrated in the following figure:



Assume that each node v_i has a weight w_i. You wish to find an independent set S in the graph G with maximum weight (i.e., such that the sum of its node weights is maximum).

An intuitively appealing (greedy) solution we may call "Heaviest first" is to first find the heaviest node v, add this to the set S, and then remove v and its neighbors from G. We repeat this until there are no nodes left in the graph.

a) Give an example of a graph G of the type described earlier, where the algorithm "Heaviest first" does not return the heaviest independent subset. Give your answer as a sequence of node weights.

Another intuitively appealing solution we may call "Every other" is to partition the nodes into two sets, S_1 and S_2, where S_1 is the set of nodes in odd positions, and S_2 is the set of nodes at even positions. We then let the set S be the one of S_1 og S_2 that has the greatest weight.

b) Give an example of a graph G of the type described earlier, where the algorithm "Every other" does not return the heaviest independent subset. Give your answer as a sequence of node weights.

c) Describe (with text or pseudocode) an algorithm that finds the weight of the heaviest independent subset in a graph G (with weights w_i) with a structure as described earlier. The algorithm should be as efficient as possible. Give the running time in Θ notation, as a function of the number of nodes, n. Please keep your answer brief.
Note: The algorithm is only to find/return *the total weight* of the heaviest independent set — not which nodes it contains.