Final exam in
TDT4120 Algorithms og data structures

Exam date	3 December 2012
Exam time	0900–1300
Grading date	3 January 2013
Language	English
Concact during the exam	Magnus Lie Hetland (ph. 91851949)
Aids	No printed/handwritten; specific, simple calculator

Please read the entire exam before you start, plan your time, and prepare any questions for when the teacher comes to the exam room. Make assumptions where necessary. Keep your answers short and concise. Long explanations that do not directly answer the questions are given little or no weight.

You may describe your algorithms in text, pseudocode or program code, as long as it is clear how the algorithm works. Short, abstract explanations can be just as good as extensive pseudocode, as long as they are precise enough. Running times are to be given in asymptotic notation, as precisely as possible.

Important: See the appendix at the back for information that may be used to solve the problems.

a) Show, by finding the constants of the definition, that $\sin(n)$ is $O(1)$.

Answer (6%):

b) One way to avoid consistent worst-case behavior in hashing is to choose the hash function randomly. What is this called?

Answer (6%):

c) One technique used in dynamic programming is to store the return value of a function the first time it is called with a given set of parameters, and then simply return/use this value directly if the function is called with these parameters again. What is this called?

Answer (6%):

d) Draw an example of an undirected, connected graph with a marked starting node, where breadth-first search will find the shortest paths to all other nodes, while depth-first search is guaranteed to fail, no matter which order the neighbors are examined in. (That is, depth-first search must find *at least one* wrong path.) Use as few nodes as possible.

Answer (6%):

e) In the textbook (Cormen et al.) it is argued that it is pointless to permit negative cycles in shortest paths. If we limit ourselves to search only for paths *without cycles* (that is, simple paths), why is the presence of negative cycles (that can be reached from the start node) in our graph still problematic?

Answer (6%):

f) In the 0-1 knapsack probolem, let $c[i, w]$ be the optimal value for the first i objects, with a capacity of w. Let v_i and w_i be the value and weight of object i, respectively. Complete the recurrence for $c[i, w]$ for the case where $i > 0$ and $w_i \leq w$.

Answer (6%): $c[i, w] = \max\{$, $\}$.

g) Edmonds-Karp is an implementation of Ford-Fulkerson that guarantees a polynomial running time. What is it about Edmonds-Karp that yields this guarantee?

Answer (6%):

h) Construct a Huffman-tree for the characters A, B, C, D and E with frequencies of 10, 11, 23, 12 and 13, respectively. Always let the smallest of two subtrees be placed to the left. Draw the tree below.

Answer (6%):

A friend of yours has a problem that consists of determining whether a graph G contains a cycle of length k (as measured by the number of edges), for a given integer k. Your friend wants to show that the problem is NP-complete, and intends to do so by letting the graph H be a cycle of length k, and then solving the cycle problem by means of the subgraph isomorphism problem.

i) What is wrong with your friend's reasoning?

Answer (6%):

j) You wish to sort the nodes of a directed, acyclic graph so that all edges point "from left to right" (that is, to nodes later in the ordering). If the graph has n nodes and m edges, what is the running time for this sorting?

Answer (6%):

Consider the following pseudocode. Assume that A is an array of length $n \geq 1$, where n is an integer power of 2. Assume that PARSUM initially is called with $i = 0$ and $j = n - 1$. The function floor(x) returns the largest integer y such that $y \leq x$.

```
PARSUM(A, i, j):
if i < j
    k = floor((i + j) / 2)
    return PARSUM(A, i, k) + PARSUM(A, k + 1, j)
else
    return A[i]
```

k) What is the parallelism of the algorithm PARSUM, as a function of n? Give your answer in asymptotic notation.

Answer (7%):

l) Solve the recurrence $T(n) = T(2n)/2 - n$, $T(1) = 1$. Give your answer in asymptotic notation.

Answer (7%):

You are to find the shortest/cheapest paths in a directed graph where the weights are placed on the *nodes*, rather than on the edges. That is, each node v has a weight $w(v)$. You represent this as an ordinary shortest-path problem by giving all edges into a given node v a weight of $w(v)$, ignoring the weight in the node itself. You now use Dijkstra's algorithm to find the shortest paths from a given start node to all other nodes. The graph has n nodes and m edges.

m) In the worst case, how many times will you need to make changes in the table of distance estimates, d? (That is, updates of the estimate *after* the initialization.) Give your answer in asymptotic notation. Briefly explain your reasoning.

Answer (8%):

Consider the following pseudocode.

```
GLYMPHID(anx, gerb):
for bith in 701, 301, 132, 57, 23, 10, 4, 1
    for chiss = bith to gerb–1
        ewok = anx[chiss]
        while chiss ≥ bith and anx[chiss – bith] < ewok
            anx[chiss] = anx[chiss – bith]
            chiss = chiss – bith
        anx[chiss] = ewok
```

n) Which problem does the algorithm GLYMPHID solve?

Answer (8%):

Assume that you are given two integer arrays A = $[a_0, a_1, \ldots, a_{n-1}]$ and B = $[b_0, b_1, \ldots, b_{m-1}]$. You want to decide whether the array A can be partitioned into contiguous, non-empty, non-overlapping segments such that each segment sums to one of the numbers in B. All numbers in A are to be part of exactly one segment. Several segments may sum to the same number in B and not all numbers in B need be used.

Example: A = [2, 4, 5, 1, 2, 2, 9, –5], B = [6, 8].
The answer here is yes. (Possible partition: sum(2, 4) = 6, sum(5, 1, 2) = 8, sum(2, 9, –5) = 6.)

Note: You are not required to find the partition itself. You only need to decide whether such a partition is possible or not.

o) Describe an algorithm that solves the problem as efficiently as possible. What is the running time in the worst case? (Give your answer in asymptotic notation.)

Answer (10%):

Running time:

Appendix

If all keys are known when a hash table is built, it is possible to improve the worst-case running time for search to be the same as the average-case running time. This is called *perfect hashing*. Assume that such a perfect (collision-free) hash table over n elements can be built in $O(n)$ time.

Two graphs are isomorphic if they have the same structure—that is, they are identical if you ignore the node labels. The subgraph isomorphism problem consists of deciding whether a given graph H is isomorphic to some subgraph in another given graph G. This problem is NP-complete.