NTNU
Institutt for datateknikk
og informasjonsvitenskap

Examination paper for TDT4120 Algorithms and Data Structures

Academic contact during examination	Magnus Lie Hetland
Phone	91851949
Examination date	August 11, 2013
Examination time (from–to)	0900–1300
Permitted examination support material	D. No printed/handwritten materials permitted. Specific, simple calculator permitted.
Language	English
Number of pages	9
Number of pages enclosed	0
Checked by	Pål Sætrom

Date Signature

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 **in the given boxes**. 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, according to preference (unless the given problem states otherwise), as long as it is clear how the algorithm works. Short, abstract explanations may be just as good as extensive pseudocode, as long as they are precise enough. Algorithms that are constructed should in general be as efficient as possible, unless otherwise stated. Running times are to be given in asymptotic notation, as precisely as possible. Problems count equally toward the total score.

1. Write the function

$$T(n) = 492n^3 + 3\left(\frac{n}{4}\right)^8 + \sqrt{n} \cdot \log_2 n$$

using asymptotic notation.

Answer:

2. What is the running time of the following algorithm, expressed as a function of n?

$x = 0$
for $i = 1$ **to** n
  for $j = i$ **to** n
    $x = x + 1$
  end
end

Answer:

3. What is the point of using a randomly selected pivot in randomized Quicksort?

Answer:

4. Which assumptions are made about the input when using Floyd-Warshall?

Answer:

5. What is the relationship between Ford-Fulkerson and Edmonds-Karp?

Answer:

6. The nine nodes of a binary tree (not necessarily a binary search tree) have been given the labels **A**, **B**, **C**, … , **I** arbitrarily. A preorder tree walk prints the nodes in the following order:
   I A H G C F D E B
   A postorder tree walk prints the nodes in the following order:
   H G A D E F B C I
   Reconstruct the tree, and draw it below (with the left child nodes drawn to the left).

Answer:

7. Let **A** be the set of maximal acyclic undirected graphs and let **B** be the set of minimal connected acyclic undirected graphs, for arbitrary node sets, where “maximal” and “mimimal” refer to the number of edges. What is the relationship between **A** and **B**? Explain briefly.

Answer:

8. Solve the recurrence $T(n) = 3T(n/3) + n$. Give your answer in asymptotic notation.

Answer:

9. Your friend Lurvik has invented an algoritm. His algorithm can take a sequence of length n, where n is a multiple of k, and sort it into k equal-length segments, so that the elements in a given segment are not necessarily sorted with respect to each other, but all elements in any segment are greater than or equal those in all segments to the left, and less than or equal to those in segments to the right. Give a lower bound for the running time of the algorithm. Explain your reasoning briefly.

Answer:

10. A group of n knights have been jousting all week, and have now finally each had a match with each of the others, where each match had a winner, and you are asked to create a ranking list of the knights. You're pretty confident about this—you could just use a sorting algorithm, you think. Even if there had been ties, you could have used topological sorting. No problem! However, when you look at the data, you realize … there are cycles! For example, The Black Knight has bested The Red Knight, who in turn beat The Green Knight … who in turn defeated The Black Knight! What to do? You decide on the following solution: Create a ranking list where the knight in each position beat the knight in the next position down on the list. After some head-scratching you figure out that this must always be possible. You suspect that more efficient solutions are possible, but settle for a goal of a running time of $O(n^2)$. Describe an algorithm with this running time that solves the problem. Explain briefly why it is correct.

Answer:

11. You are given a flow network with an associated flow (that is, a flow function—not just the flow value). Briefly describe an algorithm for determining whether the flow is maximum. What would the running time be?

Answer:

12. You're playing around with Dijkstra's algorithm. Rather than putting all the nodes into the queue Q at the beginning, you use Q as a traversal queue, similar to in BFS, and add nodes as they are discovered, *if* they potentially provide shortcuts. (After all, a node must be visited if and only if it can provide shortcuts to other nodes.) In order to do this, you have rewritten RELAX so that it returns **true** if it modifies the distance and **false** otherwise. You also simply drop the set S, as in the following pseudocode:

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
Q = {s}
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    for each vertex v ∈ G.Adj[u]
        if RELAX(u, v, w)
            Q = Q ∪ {u}
```

(*Problem 12, contd.*)

You're pretty sure this should work, and as long as *Q* is a heap, you should be getting the same running time, too. You know that in the previous exam, one problem involved replacing *Q* with an unordered array, using that as a priority queue. You wonder if maybe your modified implementation of Dijkstra's algorithm doesn't *need* a priority queue at all! You get curious, and try to replace *Q* with a simple FIFO queue. What happens? When (i.e., for which instances) will this algorithm work as a single-source shortest path algorithm? What will the running time be? How does the algorithm relate to other algorithms in the curriculum? Discuss briefly.

Answer:

13. In an undirected graph, given three nodes *u*, *v* and *w*, you are to determine whether there is a path from *u* to *w* that passes through *v*. Briefly describe an algorithm that solves the problem.

Answer:

14. Given a connected, weighted undirected graph, you are to find some spanning tree (not necessarily a minimum spanning tree), whose heaviest edge is minimal. That is, there are no other spanning trees whose highest weight is lower. While you suspect there are more efficient ways of solving the problem, you decide to solve it by simply constructing a minimum spanning tree. Briefly explain why your solution will be correct.

Answer:

15. You are given two sequences **A** and **B** of respective lengths *k* and *n* and want to count how many times **A** occurs as a subsequence of **B**. In other words, if all the elements are identical, your answer should be the number of ways to select a set of *k* elements from a set of *n* elements, i.e., the binomial coefficient,

$$\binom{n}{k}.$$

Briefly describe an algorithm that solves the problem.

Answer: