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	December 7, 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	6
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. All subtasks count equally toward the total score

1. What is the minimum and maximum number of edges in a connected, undirected graph with *n* nodes?

Minimum:
Maximum:

2. What is the minimum and maximum number of nodes in a rooted binary tree of depth *n*? (The depth is the maximum number of edges in any path from the root.)

Minimum:
Maximum:

3. What is the expected (average case) asymptotic running time for an unsuccessful search in a hash table with *m* slots and *n* elements? Express the answer as a function if *n* and *m*. (You may assume simple, uniform hashing and that collisions are reseolved by chaining.)

Answer:

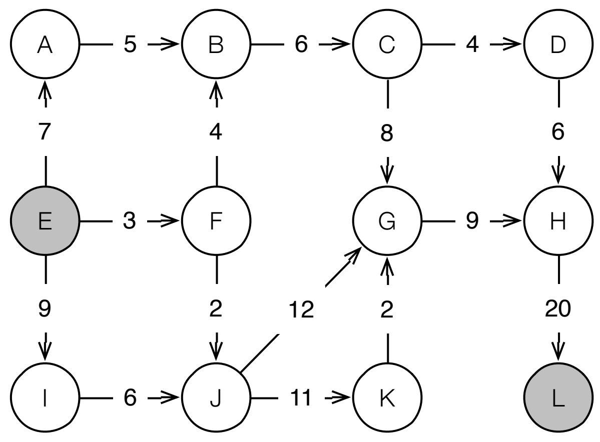
4. A problem can often be decomposed into subproblems, where the individual subproblems depend on other subproblems. To solve the problem, the subproblems must be solved in an order that satisfies these dependencies. If you view the subproblems as nodes and the dependencies as edges in a directed graph, what is such an order called?

Answer:

5. Solve the recurrence $T(n) = T(n+1) - 2^n$; $T(1) = 1$. Give the answer exactly (i.e., not asymptotically).

Answer:

6. The following directed graph represents a flow network with the given capacities. What is the maximum flow from node E to node L?



Answer:

7. What is the max-heap property?

Answer:

8. Sorting of *n* elements can, in general, not be performed in *O*(*n*) time. Even so, counting sort, for example, is able to do this. How can this be?

Answer:

9. Your friend claims to have invented an algorithm that solves the traveling salesman problem in *O*(*n* log(log(*n*))) time, where *n* is the number of vertices in the graph. If your friend's claim were correct, would there be any important consequences of his or her discovery? Explain briefly.

Answer:

10. You have *n* friends. You know that some of your friends hate each other. Hatred is always mutual. You know exactly which pairs of friends that hate each other. You wish to unfriend some of your friends, so that none of your remaining friends hate each other. Your task is to determine if you can solve this problem by unfriending *k* friends, where *k* is a an input parameter. You can assume that this decision problem is in NP. Show that it is NP-complete. Explain your reasoning clearly, and make sure you include all required parts of such an argument.

Answer:

11. Student Lurvik claims to have found a way of making Dijkstra's algorithm deal with negative edges: Find the shortest edge, i.e., the "most negative," and call its length *w*. Add $|w| + 1$ to all the edge lenghts, so that all the edges have positive lengths. The run DIJKSTRA on the modified graph; take the answer you get, and use the corresponding edges in the original graph. Prove that this will always give the correct answer, or find a counter-example that shows that Lurvik's algorithm can yield wrong answers.

Answer:

12. Dijkstra's algorithm may be implemented by using a binary heap as the priority queue, or by simply using the unsorted node table as priority queue. This leads to different running times. Given the following pseudocode for DIJKSTRA, give the lines that are affected by this choice, and explain why the corresponding running times are $O(E \lg V)$ og $O(V^2)$. When should you use which queue?

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  $S = \emptyset$
3  $Q = G.V$
4  while $Q \neq \emptyset$
5      $u$ = EXTRACT-MIN($Q$)
6      $S = S \cup \{u\}$
7      for each vertex $v \in G.Adj[u]$
8          RELAX($u$, $v$, $w$)
```

Lines that are affected (line numbers):
Explanation of running times:

When one should use an unsorted list/table:

13. Multiplication of a matrix with n rows and m columns by a matrix with m rows and k columns has a running time of $O(nmk)$. The algorithm for solving the 0/1 knapsack problem where one is to steal n objects using a knapsack with a capacity of k runs in $O(nk)$. The first algorithm has a polynomial running time, while the second one has an exponential running time. How do you explain this?

Answer:

14. Your friend Smartnes thinks he has read in the textbook that the worst-case running time of comparison-based sorting is Ω(*n* lg *n*), but he doesn't quite understand what this means. Explain briefly what it means, or explain briefly that it makes no sense.

Answer:

15. Two persons are to visit a sequence of cities. You are given an ordered sequence of *n* cities, and the distances between all pairs of cities. You are to partition the cities into two subsequences (where the cities are in their original order, but not necessarily contiguous in the original sequence) so that person A visits the cities in the first sequence (in order) but none of the others, person B visits the cities in the second sequence (in order) but none of the others, and such that the sum of distances traveled by A and B in total is minimized. (In other words, each city is to be visited by exactly one of A and B.) Assume that A and B start in the first cities of their respective subsequences. Describe an efficient algorithm that solves the problem. What is the running time?

Answer: