Student number _____

Exam in
IT1105/TDT4120 Algorithms and Data Structures
December 13, 2004 0900-1300

Contact during exam: Arne Halaas (*ph* 416 61 982),
Magnus Lie Hetland (*ph* 918 51 949)
Tools: Simple calculator allowed. No printed or handwritten materials allowed.
Note: Please write your answers where indicated.

Problem 1 (20%)

For each algorithm below, please describe *briefly* (*i*) *which problem* the algorithm solves, (*ii*) *which requirements*, if any, are placed on the problem instances, and (*iii*) roughly *how the algorithm works*. Please answer each question with *no more than 50 words*. Feel free to use keywords and partial sentences.

4% *a*) QuickSort

Answer:

4% *b*) Dijkstra's algorithm

Answer:

4% *c*) Binary search

Answer:

4% *d*) Prim's algorithm

Answer:

4% *e*) Warshall's algorithm

Answer:

Problem 2 (20%)

In the following questions, check either "yes" or "no" and give a brief account of your reasoning.

Note: Answers with wrong or missing reasoning will not receive any points.

4% *a*) A poor algorithm A, which checks whether a number n is a prime, has a running time of $\Theta(n)$. Is this running time exponential?
Hint: How is problem size defined?

Answer: [ ] Yes [ ] No

Reasoning:

4% *b*) Do balanced, directed trees have fewer possible topological orderings than other connected, directed acyclic graphs with the same number of vertices?

Answer: [ ] Yes [ ] No

Reasoning:

4% *c*) Steinar Hopp wants to get over a wide river. His jumping distance is, of course, limited, so he plans to jump from stone to stone. He wants to find the sequence of stones that gives the lowest number of jumps, and thinks that depth-first-search is more suited for this than breadth-first-search. Do you agree? In your reasoning, describe how the algorithms may be used and why you prefer one over the other.

Answer: [ ] Yes [ ] No

Reasoning:

4% *d*) In asymptotic notation, O and Ω describe different things. Is it correct to say that O is a measure of the worst-case running time of an algorithm and that Ω is a measure of the best-case running time?

Answer: [ ] Yes [ ] No

Reasoning:

4% *e*) We can say that a problem *A* is more difficult than a problem *B* if all instances of problem *B* can also be seen as instances of problem *A*, but not vice versa. Is topological sorting a more difficult problem than regular sorting?

Answer: [ ] Yes [ ] No

Reasoning:

Problem 3 (20%)

4% *a*) Here is a list of running time classes – please place them in an order of increasing complexity: *constant*, *exponential*, *linear*, *cubic*, *quadratic*, *factorial*, *logarithmic*, *n-log-n*. No explanation is necessary.

Answer:

4% *b*) What does the following little program do, and what is its running time? Use Θ notation and explain your reasoning. Please be brief.

Algorithm *Azathoth*($A[1\ldots n]$)

```
1. i ← 1
2. j ← n
3. while i < j
4.     if A[i] > A[j]
5.         i ← i + 1
6.     else
7.         j ← j – 1
8. return A[i]
```

Answer:

Reasoning:

6% *c*) What does the following little program do, and what is its running time? The function *floor*() rounds down to the nearest integer. Use Θ notation and explain your reasoning. Please be brief.

Algorithm *Astaroth*(*i*, *j*, *A*[1…*n*], *B*[1…*n*])

```
1. if i = j
2.     if A[j] > B[j]
3.         return A[j] – B[j]
4.     else
5.         return B[j] – A[j]
6. k ← floor((i + j) / 2)
7. return Astaroth(i, k, A, B) + Astaroth(k + 1, j, A, B)
```

Answer:

Reasoning:

6% *d*) An algorithm has a fixed set of instructions that are always executed, a set of instructions where the number of steps is a fixed percentage of the problem size, and *k* recursive calls on one *k*th (that is, 1/*k*) of the problem. What is the running time of the algorithm? Use Θ notation and explain the answer by giving a short sketch of your calculation.

Answer:

Reasoning:

Problem 4 (10%)

If we colour the nodes of a graph and two neighbours have the same colour we call this a conflict. A graph is k-colourable if it can be coloured with k colours without any conflicts arising.

4% *a*) Is there a known polynomial algorithm that determines whether a graph is k-colourable, for an arbitrary k? If so, how does it work, and what is its running time (in Θ notation)? If not, why?

Answer:

2% *b*) How would you solve the problem if $k = 2$? Give upper and lower asymptotic limits for the running time.

Answer:

4% *c*) Assume that you have a graph that is *not* 2-colourable. You wish to colour it with two colours in such a manner that the number of conflicts is minimised. Give a brief description of a simple solution to this problem using the branch and bound strategy.

Answer:

Problem 5 (15%)

For each of the algorithms below, which of the following asymptotic running time complexity classes can be used to describe the algorithm? You can not assume anything about the problem instances.

$\Omega(n)$, $\Omega(n \log n)$, $\Omega(n^2)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $\Theta(n)$, $\Theta(n \log n)$, $\Theta(n^2)$

Give the relevant complexity classes (for example, "$\Omega(n^2)$, $O(n)$") or write "None" if none applies. If several classes apply, all of the must be given in the answer. Give a brief explanation of your reasoning for each question.

5% *a*) Inserting n values in an empty binary search tree.

Answer:

Reasoning:

5% *b*) Insertion sort.

Answer:

Reasoning:

5% *c*) MergeSort.

Answer:

Reasoning:

Problem 6 (15%)

Student Lurvik is going to sum n positiv floating point numbers, $x[1]…x[n]$, in such a way as to minimise the rounding error. Depending on how this is done, the number of summations the numbers participate in may differ. For example, $x[1]$, $x[2]$, $x[3]$ og $x[5]$ will participate in more summations (three) than $x[6]$ (one) in the following summation order:

$$(((x[1] + x[3]) + (x[2] + x[5])) + x[6])$$

Note that both the order of the numbers and the placement of parentheses is chosen freely.

How great the rounding error for a sum of two floating point numbers is depends on how great the sum itself is (that is, a greater sum gives a greater error). Lurvik realises that it may be useful to sum the numbers in a way such that the small numbers participate in as many sums as possible (that is, they end up "deeply" nested in parentheses) while the big numbers participate in as few as possible. If $p(i)$ is the "parenthesis-depth" of floating point number $x[i]$ (that is, the number of parentheses surrounding the number) Lurvik defines the *error* of a possible solution as

$$p(1)\cdot x[1] + p(2)\cdot x[2] + \ldots + p(n)\cdot x[n].$$

Lurvik now wishes to find a parenthesis-setting (and order for the numbers) that minimises this error expression. Sketch out a solution for this problem. Which design technique do you use? Show, with support in the curriculum, that the solution is correct.

Answer: