Written Examination

Algorithms, Data Structures and Software Engineering for Media Technology

Medialogy 8th semester

1 June 2022, 09.00 – 13.00

Name:_____

Cpr.no.: _____

Study no.: _____

Aalborg University | MSc in Medialogy | Med8

Algorithms, Data Structures and Software Engineering for Media Technology

Ordinary Examination

June 2022

Instructions

- You have 4 hours to complete this examination.
- Neither electronic devices nor written material are allowed in the examination room.
- This examination consists of 10 questions. Each question is worth 10 marks. You must obtain at least 50 marks to pass.
- Do not write any answers on this question paper—answers written on the question paper will be ignored by the examiner. Write all your answers on the writing paper provided.
- Do not write your answers in pencil and do not use a pen with red or green ink. Use a pen with blue or black ink.
- Hand in no more than one answer to each question.
- Do not turn over until you are told to do so by the invigilator.

Question 1

For each of the following equations, state whether it is true or false.

a) $n^2 = O(n^3)$

b) $n^2 \log n = \Theta(n^2)$

c) $n^3 = \Omega(n^2 \log n)$

d) $\sqrt{n} = \omega(n^{0.3})$

e) $\frac{1}{n} = o(\log n)$

f) $n \log(n^2) = O\left(\left(\sqrt{n}\right)^3\right)$

g) $12\log(n^4) = \Theta(\log n)$

h) $n \log n + 3n^2 + \sqrt{n} = \Omega\left(\left(\sqrt{n}\right)^5\right)$

i) $5n^2 + 2n + 3 = \omega(n^2)$

j) $2n^2 \log n + 4n^3 = o(n^3)$

[1 mark for each correct part]

Question 2

The Master Theorem is stated as follows:

Theorem 4.1 (Master theorem)
Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

Given the Master Theorem, as stated above, write down the order of growth in terms of Θ notation for each of the following recurrences.

a) $T(n) = 25T(n/5) + 3n^2$
b) $T(n) = 3T(n/9) + \log n$
c) $T(n) = 8T(n/2) + 2n^4$
d) $T(n) = 2T(n/16) + 3\sqrt{(\sqrt{n})}$
e) $T(n) = T(n) + \sqrt{n} \log n$

[2 marks for each correct part]

Question 3

For each of the following sorting algorithms, state, using asymptotic notation, the best-case running time and the worst-case running time. Describe also the conditions under which the best case and worse case occur. If the algorithm requires the input data to satisfy certain constraints, then explain what these constraints are.

a) Merge sort
b) Heap sort
c) Quick sort
d) Counting sort
e) Bucket sort

[2 marks for each correct part]

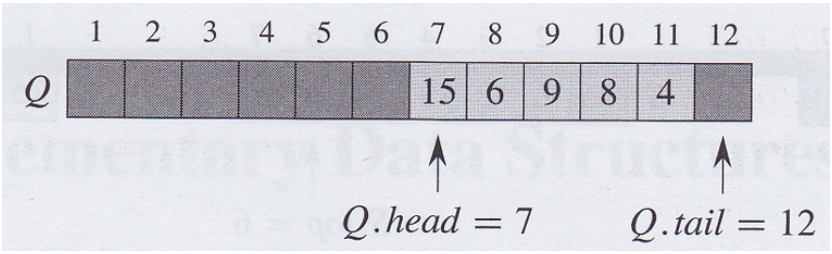
Question 4

The selection problem is as follows: given an array, A, containing n distinct numbers and an integer, i, such that $1 \leq i \leq n$, find the element x in A that is larger than exactly $i - 1$ other elements in A. We call the result, x, the *ith order statistic* of the set of numbers, A. Describe an algorithm that finds the ith order statistic of a set of n numbers in *expected* linear time. What is the worst-case time of this algorithm?

[10 marks]

Question 5

a) What are the three operations that must be supported by a *dictionary* data structure? [3 marks]
b) Which of the following three data structures is a LIFO data structure: stack, queue, binary tree? [1 mark]
c)



In the figure above, the array Q implements a queue data structure. The head and tail of the queue are at indices 7 and 12, respectively, as shown. Draw a diagram showing the state of the array Q, together with the positions of the head and tail, after the following operations have been executed in order:

```
ENQUEUE(Q,17);
ENQUEUE(Q,3);
ENQUEUE(Q,5);
DEQUEUE(Q);
```

[4 marks]

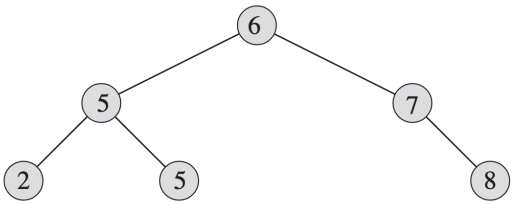
d) Under what conditions do we get "queue underflow"? [1 mark]
e) Express in asymptotic notation the time that it takes to find a given value in a doubly-linked, unsorted list of length n. [1 mark]

Question 6

a) Define what is meant by the *load factor* of a hash table. [1 mark]
b) Explain what is meant by *simple uniform hashing*. [1 mark]
c) If we have a hash table that satisfies the condition of simple uniform hashing, what is the expected length of a chain, assuming we are using chaining to resolve collisions? [1 mark]
d) Explain why it takes $\Theta(1)$ time on average to search for an element in a hash table, provided it satisfies the condition of simple uniform hashing. What other condition needs to be satisfied for searching to take constant time on average? [3 marks]
e) What is the worst-case time for deleting elements from and inserting elements into a hash table? [2 marks]
f) One popular simple scheme for designing a hash function is to use the "division method". If k is the key to be hashed and our hash table has m slots, write down the hash function that we use when applying the division method. When using the division method, why should we avoid using a power of 2 for the value of m? [2 marks]

Question 7

a) What is the expected height of a binary search tree containing n elements if the elements are inserted into the tree in a random order? [1 mark]
b) Study the following binary search tree and algorithm:




```
INORDER-TREE-WALK(x)
1  if x ≠ NIL
2      INORDER-TREE-WALK(x.left)
3      print x.key
4      INORDER-TREE-WALK(x.right)
```

If the root node of the tree (whose key is 6) is given as input, x, to the INORDER-TREE-WALK algorithm, what does this algorithm print out? [2 marks]
c) What is the running time of the Inorder-Tree-Walk algorithm shown in part (b)? [1 mark]
d) The following algorithm searches for a key, k, in a tree rooted on the node pointed to by x:

```
TREE-SEARCH(x, k)
1  if x == NIL or k == x.key
2      return x
3  if k < x.key
4      return TREE-SEARCH(x.left, k)
5  else return TREE-SEARCH(x.right, k)
```

If x is a pointer to the root node of the tree shown in part (b) and k is 8, how many

times in total is the Tree-Search algorithm called during the execution of the algorithm in this case and what does the algorithm return? [2 marks]

e) Write down an algorithm that, when given a pointer, *x*, to the root node of a binary search tree as its argument, returns a pointer to a node whose key is equal to the minimum key in the tree. [4 marks]

Question 8

```
CUT-ROD(p, n)
1  if n == 0
2       return 0
3  q = −∞
4  for i = 1 to n
5       q = max(q, p[i] + CUT-ROD(p, n − i))
6  return q
```

```
MEMOIZED-CUT-ROD(p, n)
1  let r[0..n] be a new array
2  for i = 0 to n
3       r[i] = −∞
4  return MEMOIZED-CUT-ROD-AUX(p, n, r)

MEMOIZED-CUT-ROD-AUX(p, n, r)
1  if r[n] ≥ 0
2       return r[n]
3  if n == 0
4       q = 0
5  else q = −∞
6       for i = 1 to n
7            q = max(q, p[i] + MEMOIZED-CUT-ROD-AUX(p, n − i, r))
8  r[n] = q
9  return q
```

```
BOTTOM-UP-CUT-ROD(p, n)
1  let r[0..n] be a new array
2  r[0] = 0
3  for j = 1 to n
4       q = −∞
5       for i = 1 to j
6            q = max(q, p[i] + r[j − i])
7       r[j] = q
8  return r[n]
```

The three boxes above show three different algorithms for solving the rod-cutting problem, with which you should be familiar. As input, each algorithm takes an array, *p*, of prices for rods of different lengths and the length, *n*, of the rod to be cut. The output of each algorithm is the maximum amount of revenue that can be obtained by cutting a rod of length *n*. (Question continued on next page.)

a) Which of the three algorithms shown use(s) dynamic programming? [2 marks]
b) For each algorithm state its worst-case running time using asymptotic notation. [3 marks]
c) Draw a subproblem graph for the rod-cutting problem when *n*, the original length of the rod to be cut, is equal to 4. [2 marks]
d) How does Bottom-Up-Cut-Rod need to be extended so that it outputs the actual lengths of the pieces into which a rod of length *n* needs to be cut in order to get the maximum revenue? [3 marks]

Question 9

The following is a CUDA C program that carries out parallel addition of vectors.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define N 3
4
5  __global__ void addVectors(float* dev_a, float* dev_b, float* dev_c) {
6      int i = blockIdx.x;
7      if (i < N) {
8          dev_c[i] = dev_a[i] + dev_b[i];
9      }
10 }
11
12 int main(void) {
13     float a[N] = {1,2,3}, b[N] = {4,5,6}, c[N];
14     float *dev_a, *dev_b, *dev_c;
15
16     cudaMalloc((void**)&dev_a, N*sizeof(float));
17     cudaMalloc((void**)&dev_b, N*sizeof(float));
18     cudaMalloc((void**)&dev_c, N*sizeof(float));
19
20     cudaMemcpy(dev_a, a, N*sizeof(float),cudaMemcpyHostToDevice);
21     cudaMemcpy(dev_b, b, N*sizeof(float),cudaMemcpyHostToDevice);
22
23     addVectors<<<N,1>>>(dev_a, dev_b, dev_c);
24
25     cudaMemcpy(c, dev_c, N*sizeof(float), cudaMemcpyDeviceToHost);
26
27     cudaFree(dev_a);
28     cudaFree(dev_b);
29     cudaFree(dev_c);
30
31     for(int i = 0; i < N; i++)
32         printf("%.0f ", c[i]);
33
34     return EXIT_SUCCESS;
35 }
```

Answer the following questions:

a) In which line or lines is memory allocated on the GPU? [1 mark]
b) In which lines is the kernel defined? [1 mark]
c) How many threads does the kernel run on? [1 mark]

d) How many threads are there in each block? [1 mark]
e) In which line or lines is memory copied from the GPU to the CPU? [1 mark]
f) Why does memory have to be copied between the GPU and the CPU? [2 marks]
g) Study the following CUDA C code and write down the output that it generates.

```
1  #include<stdio.h>
2  #include<stdlib.h>
3
4  #define N 4
5
6  __global__ void kernel(int *dev_a) {
7      unsigned i = threadIdx.x + blockIdx.x * blockDim.x;
8      dev_a[i] = i;
9  }
10
11 int main(void) {
12     int *dev_a;
13     cudaMalloc((void**)&dev_a, N*N*sizeof(int));
14     kernel<<<N,N>>>(dev_a);
15     int a[N][N];
16     cudaMemcpy(a, dev_a, N*N*sizeof(int), cudaMemcpyDeviceToHost);
17     cudaFree(dev_a);
18     for(int i = 0; i < N; i++) {
19         for(int j = 0; j < N; j++)
20             printf("%3d",a[i][j]);
21         printf("\n");
22     }
23     return EXIT_SUCCESS;
24 }
```

[3 marks]

Question 10

a) Two common methods of reusing functionality are *inheritance* and *composition*. Explain the difference between these two techniques. [2 marks]
b) What advantages does composition have over inheritance? [2 marks]
c) In Scrum, what is the *product backlog*? [2 marks]
d) In Scrum, who is responsible for the product backlog? [2 marks]
e) In Scrum, who is responsible for ensuring that the scrum framework is followed? [2 marks]

END OF EXAMINATION