MATEMATISKA INSTITUTIONEN
STOCKHOLMS UNIVERSITET
Avd. Beräkningsmatematik
Examinator: Lars Arvestad

Tentamensskrivning i
DA3018 Datalogi för matematiker
7.5 hp
2023-05-29

- **Write clearly.** Hard-to-read answers risk zero points.
- **Use one side of the paper**.
- Justify your answers (unless otherwise stated).
- **Grading thresholds:** E: 20, D: 26, C: 32, B: 38, A: 44

---

1. (a) What does a computer scientist mean with *space complexity*? (1p)

(b) What do we mean with "the algorithm runs in quadratic time"? (1p)

(c) What is *binary search*? (1p)

(d) What is the *heap property*? (1p)

(e) What is *divide-and-conquer* in Computer Science? Explain and give an example. (2p)

(f) Explain the concept *open addressing*. (2p)

2. Suggest an implementation of a *queue* using a linked list, supporting the following operations.

- `empty(Q)` returns `True` if Q is empty and `False` otherwise.
- `enqueue(Q, e)` puts the element e last in Q.
- `dequeue(Q)` returns and removes the element first in Q. Return null if there is no element in the queue Q.

If e_1 is "enqueued" before e_2, then dequeue should return e_1 before e_2. All operations should run in time $O(1)$. A linked list is chosen to avoid having to decide on a fixed capacity for the queue.

Describe the operations in pseudocode. It should be clear how the queue operations are implemented and how the linked list is maintained.

(a) Draw a diagram of a queue with three elements, as represented by a linked list. (1p)

(b) Show how `empty(Q)` is implemented. (1p)

(c) Show how `enqueue(Q, e)` is implemented. (3p)

(d) Show how `dequeue(Q)` is implemented. (3p)

3. Caleb Ration works as a party planner and he has decided to pay a programmer to get a computer program for seating guests. Caleb recently got a contract for a large dinner, but guest placement turns out to be tricky because his client wants to ensure that guests are happy and has therefore provided Caleb with details about the guests relationships (good, bad, divorcees, etc). Finding a good placement taking care of all those relationship details is simply too difficult to do manually.

(a) Help Caleb by formulating a *computational problem* (Swe: *beräkningsproblem*) to communicate to the programmer. The computational problem should be clearly formulated, but Caleb wants to leave implementation details and even feasibility to the programmer. (4p)

(b) Give a small example of an *instance* to the computational problem. (1p)

Note: we are not asking for an algorithm, just a formal statement of the problem.

4. Disa Ray implemented an algorithm in which sorting is a crucial part. Since the algorithm takes a full minute to run on 1000 elements and the comparisons seems to be clearly dominating the run time, Disa suspects that using Selection sort was a mistake.

(a) Help Disa estimate how much time each comparison takes. (2p)

(b) Estimate the run time if Disa switches to Heap sort or Merge sort instead. (3p)

Clarify your assumptions and make note of approximations.

5. Let a polynomial $p(x) = \sum_{i=0}^{n} c_i x^i$ be represented as a list of coefficients: $[c_0, c_1, \ldots, c_{n-1}, c_n]$. For example, $3x^2 - 2x + 1$ is stored in a computer as $[1, -2, 3]$ and x^3 as $[0, 0, 0, 1]$.

Figure 1 shows two methods for evaluating a polynomial in this list representation at some x. In (a) and (b) you have pseudocode for two versions of the algorithm that you probably learned in school, with (b) explicitly computing the exponentiation.

Figure 1c has pseudocode for Horner's method, which can be described as rewriting $c_0+c_1x+c_2x^2+c_3x^3$ to $c_0 + x(c_1 + x(c_2 + xc_3))$ and then evaluating the expression coefficient by coefficient.

(a) Analyze the time complexity for the algorithm in Figure 1a. (2p)

(b) Analyze the time complexity for the algorithm in Figure 1b. (3p)

(c) Analyze the time complexity for Horner's method in Figure 1c. (4p)

(d) Justify choosing Horner's method for evaluating polynomials. (2p)

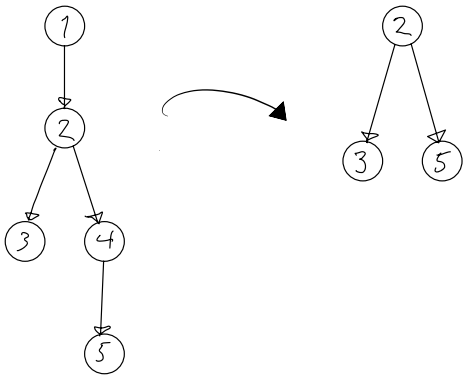
(e) Estimate how many bytes are used for a n-degree polynomial stored using a single-linked list. (2p)

6. Game developer Max Score implements game maps using graphs, where vertices are "points of interest" and edges are used to represent walkways between the points.

(a) Max would like to visualize the game map with a "fog of war": the point where the player stands is fully lit up, and the further away on the map you get the darker it gets. Discuss how to efficiently decide which degree of shading (say, from 0 to 1) should be used for each vertex. (3p)

(b) Max is also worried about maps having errors in them, such that certain points of interest might not be at all accessible. How can Max efficiently decide whether all points of interest are accessible from a map's start point? (3p)

7. You are working with trees and use a subroutine that returns binary trees such that vertices have 0, 1, or 2 children. The vertices with only a single child are redundant and should be removed before further processing. Suggest a *recursive* function that removes single-child vertices.
In the example to the right, the left tree contains two vertices with a single child (1 and 4) and removing them should "lift" vertex 2 to become a new root and vertex 5 to be the new child of 2. (5p)



(a)

```
1  def poly_eval(polynomial, x):
2      result = 0
3      x_pow = 1
4      for coeff in polynomial:
5          result = result + coeff * x_pow
6          x_pow = x_pow * x
7      return result
```

(b)

```
1  def poly_eval_alt(polynomial, x):
2      result = 0
3      degree = 0
4      for coeff in polynomial:
5          result = result + coeff * power(x, degree)
6          degree = degree + 1
7      return result
8
9  def power(x, i):    // compute $x^i$
10     result = 1
11     while i > 0:
12         result = result * x
13         i = i - 1
14     return result
```

(c)

```
1  def horners_eval(polynomial, x):
2      if len(polynomial) > 0:
3          result = polynomial[0] + x * horners_eval(polynomial[1:], x)
4          return result
5      else:
6          return 0
```

Figure 1: Three algorithms for evaluating a polynomial in list representation. (a) The schoolbook method where each term is evaluated separately, with x^i is accumulated through iterations. (b) The schoolbook version again, but we are computing x^i through iteration in each step. (c) Horner's method, in which we add the coefficient of lowest degree and continue recursively.