DA4002 (HT11) Halmstad University
Introduction to Algorithms, Data Structures, and Problem Solving

Written Exam

Monday, October 24, 2011
15h30 – 19h30, room Kåren

Examiner: Roland Philippsen (phone 7249)

Student Name:

Rules

Aside from the obvious rules of conduct exams (e.g. no chatting):

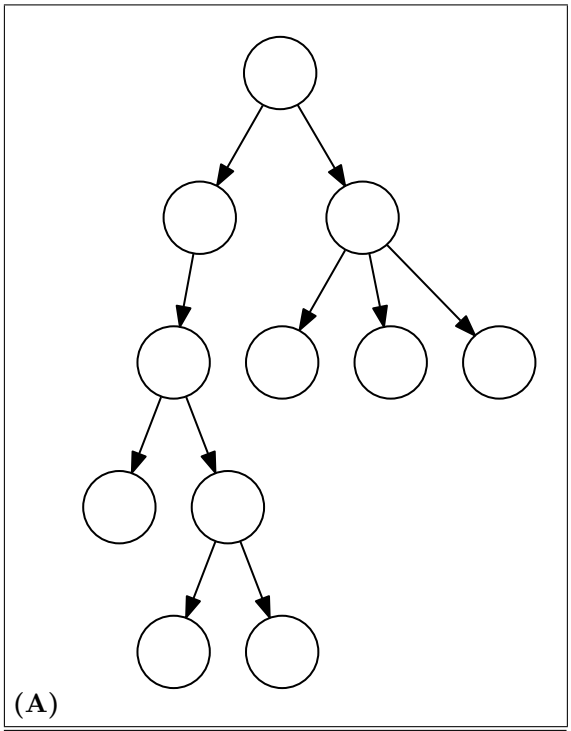
- **No computing devices** (laptops, phones, calculators, *etc*).
- **No books or printouts**.
- **Allowed self-written notes**: two sheets of A4 paper (front and back).

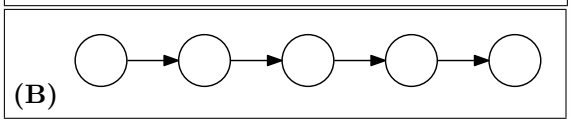
General Guidelines

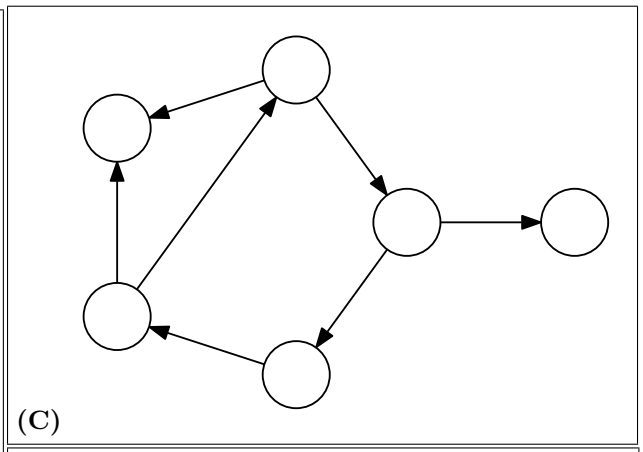
- **Read carefully** and pace yourself. You can solve the problems in any order you want, but later problems may be easier to solve after you have answered the preceding questions.
- **Write clearly** and draw clear diagrams. If you need to correct a mistake, then cleanly cross out the wrong answer and clearly indicate where the correction can be found.
- **Indicate the question number** for each of your answers. If a question has sub-questions, indicate the sub-question number after the main question number, separated by a dot. For example, question 3 has 4 sub-questions, and their answers should be numbered 3.1, 3.2, 3.3, and 3.4.

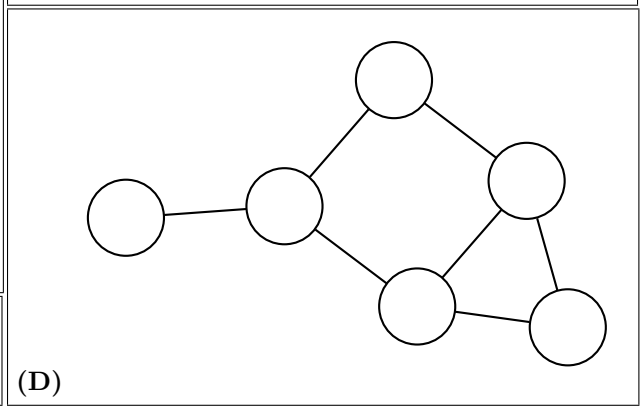
Question 1

The following four figures show four examples of data structures. They are labeled with the letters **(A)**, **(B)**, **(C)**, and **(D)**. The table below them lists names of data structure types. For each type listed in the table, determine whether it is represented by one of the figures. If there is a figure for a given type, write the corresponding letter into the second column of the table. Mark structure types which are not shown in any diagram with a big **X**.









container type	*A, B, C, D, or X*
simply linked list	
doubly linked list	
binary tree	
k-ary tree	
undirected graph	
directed **acyclic** graph	

Question 2

In question 1, three of the types listed in the table are missing from the figures. Draw a diagram for each missing type. Clearly indicate which diagram is for which container type. Make each diagram as similar as possible to one of the figures from question 1.

Question 3

The following Java code implements two data structures called `ContainerOne` and `ContainerTwo`. Answer the following questions. You can write the short answers as annotations on the source code, and use a separate sheet of paper for the ones that require more space.

1. Which container types are implemented by `ContainerOne` and `ContainerTwo`?
2. Draw a diagram of what each container looks like after `insert` has been called with the following sequence of values: 5, 3, 13, -7, 3, 7, 42, 3, 6, 12, 11.
3. In `ContainerOne`: what is the role of `ContainerOne.handle`, `Node.foo`, and `Node.bar`? Suggest better names for these three fields.
4. In `ContainerTwo`: what is the role of `ContainerTwo.alpha`, `ContainerTwo.beta`, `Node.foo`, and `Node.bar`? Suggest better names for these four fields.

```
class Node {
    long value;
    Node foo, bar;

    Node(long _value) {
        value = _value;
    }
}

class ContainerOne {
    Node handle;

    static Node helper(long value, Node node) {
        if (null == node) {
            return new Node(value);
        }
        if (value < node.value) {
            node.foo = helper(value, node.foo);
        }
        else if (value > node.value) {
            node.bar = helper(value, node.bar);
        }
        return node;
    }

    void insert(long value) {
        handle = helper(value, handle);
    }
}

class ContainerTwo {
    Node alpha;
    Node beta;

    void insert(long value) {
        Node node = new Node(value);
        if (null == beta) {
            alpha = node;
            beta = node;
            return;
        }
        beta.foo = node;
        node.bar = beta;
        beta = node;
    }
}
```

Question 4

A company sends their vice president for marketing (VPM) and their chief technology officer (CTO) to a big trade show. Each of them collects business cards from the many people they meet. Sitting down at the desks in their hotel rooms in the evening, they sort the received business cards alphabetically. However, they employ different strategies:

The VPM proceeds as follows:

- He places the pile of unsorted cards onto the desk to his right.
- He picks the top card from the unsorted pile and places it to his left.
- Then, he picks the next card from the unsorted pile, and places it either underneath or on top of the card on the left depending on whether the name on the new card comes before or after the name on the old card.
- He then repeatedly does the following, until the unsorted pile has vanished: he picks one card after another from the right, scans for its place in sorted pile, and inserts it there.

The CTO does it differently:

- He also starts by placing the unsorted pile on the right, but then he picks up groups of five cards, sorts each group in his hand, and places the resulting small sorted pile on the left. He repeats this until the unsorted pile is gone and he has many small sorted piles on the desk.
- Then, he proceeds to merge pairs of these small piles: he chooses a pair, sequentially picks the topmost card from either pile, whichever card comes first in alphabetical order, and thus creates a bigger sorted pile.
- When he runs out of small piles, he repeats the previous step with the freshly created bigger piles, merging them into even bigger piles.
- He repeats the pairwise merging until he has only one pile left.

Assume that the VPM needs 0.4 seconds to compare two individual cards. The CTO needs 5 seconds to sort 5 cards "in hand," and 0.5 seconds for each card during the merging of two piles. Assume that all other delays can be ignored. Answer the following questions:

1. Develop a formula for $T_{\mathrm{VPM}}(N)$ which allows to estimate the amount of time required by the VPM to sort N business cards.
2. Give the *Big-Oh* complexity for $T_{\mathrm{VPM}}(N)$.
3. Develop a formula for $T_{\mathrm{CTO}}(N)$ which allows to estimate the amount of time required by the CTO to sort N business cards.
4. Give the *Big-Oh* complexity for $T_{\mathrm{CTO}}(N)$.
5. Who will take longer to sort small amounts of business cards ($N < 20$ or so)? Who will take longer to sort large amounts of cards ($N > 40$ or so)? Explain your answers!

Question 5

The number sequence $\{G_n\}$ is defined for all natural numbers $n > 0$ by the following recursion. In order to compute G_n, one of your friends implemented the `gRec` Java method shown in the box below the equation. There are three `return` statements in that method, and they are labeled with the letters **(A)**, **(B)**, and **(C)**.

$$G_n = \begin{cases} 0 & \text{for } n \leq 2 \\ 1 & \text{for } n = 3 \\ 2G_{n-3} - G_{n-2} + G_{n-1} & \text{for } n \geq 4 \end{cases}$$

```
class GRec {
    static long gRec (long n) {
        if (n <= 2) {
            return 0;           // (A)
        }
        if (n == 3) {
            return 1;           // (B)
        }
        return 2 * gRec(n-3) - gRec(n-2) + gRec(n-1); // (C)
    }
}
```

In order to validate that the `gRec` method works correctly, your friend has traced which `return` statements are encountered when `gRec` is called with `n=4`. This execution trace is given in the table shown below: the first column lists the value of the method argument `n`, the second column lists the label of the `return` statement which gets executed, and the third column shows which value gets returned. For the statement marked **(C)**, your friend helpfully also wrote down how the value gets computed.

n	label	value
1	A	0
2	A	0
3	B	1
4	C	2* 0 - 0 + 1 = 1

Answer the following questions:

1. Trace the `gRec` method call for `n=6`. Write it down as a table that follows the same format as the table above.

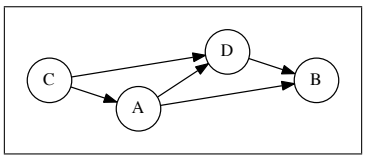
2. Your friend notices that his implementation is really slow for `n` larger than 20 or so: with `n=20` it takes 0.3 seconds, but with `n=40` it takes almost a minute! Explain why this happens. If you can draw a diagram to illustrate the root cause, even better!

3. There are two problem solving techniques which can avoid the rapid growth of execution time in this case. Briefly explain each technique *(you should not need more than two or three sentences for each).*

4. Apply both of these problem solving techniques to solve your friend's problem. Write separate code segments for each of them. You can use Java syntax or pseudo-code, as long as it is clear what is done in which order.

Question 6

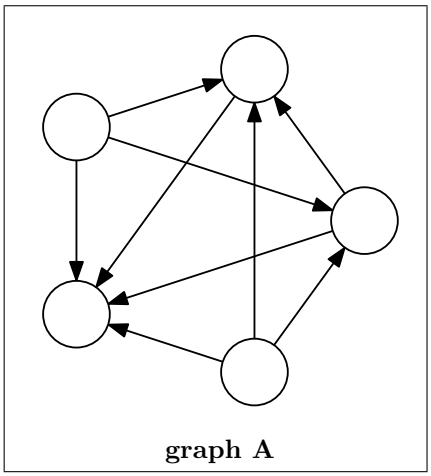
Definition: A **topological ordering** of a directed graph is a sequence of its vertices such that, for every *edge* (U, V), U appears before V in the sequence. Note that for any *path* from U to V, this also implies that U comes before V in the ordering. In the example graph shown below, the sequence (C, A, D, B) is a topological ordering.



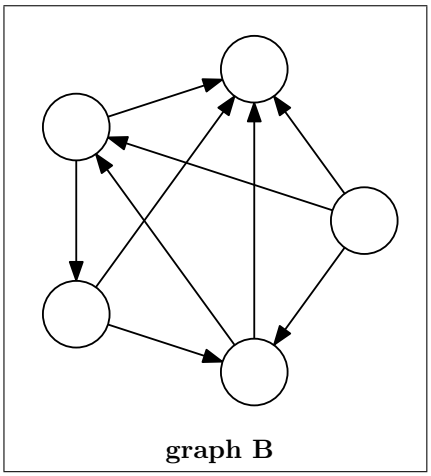
Theorem: A topological ordering is possible if and only if the graph has no cycles. In other words, a graph has to be a **directed acyclic graph** (DAG) in order to have a topological ordering. And also, every DAG has at least one topological ordering.

Answer the following questions:

1. Prove the theorem.
2. Develop and describe an algorithm to detect whether a given directed graph has a cycle, based on the above theorem.
   Hints:
   - The *indegree* of a vertex V is the number of incoming edges. In the above example, the indegree of D is two.
   - If V has an indegree equal to zero, then it can only have *outgoing* paths. This means that V cannot be on a cycle. In the above example, C is such a vertex.
   - If we remove an edge (V, W), then the indegree of W is decreased by one. If V is *not* on a cycle, and the removal of (V, W) makes the indegree of W drop to zero, then W also cannot be on a cycle.
3. Apply your algorithm to the two graphs shown below. Illustrate each step of your algorithm with a little diagram. Annotate each diagram with a short sentence about what has changed since the previous diagram. At the end of each sequence of diagrams, clearly state whether the graph has a cycle, and how this is determined by looking at the last diagram.



graph A



graph B