Final Examination—Session II

Algorithms & Data Structures
NTNU IDATA 2302

May 2024

1 Basic Knowledge

Question 1.1 (1 pt.). *In the worst-case, Algorithm X can search through a collection of n items in $O(\log_2 n)$ seconds. How big is a collection it can search through in 10 seconds? What well-known search algorithm has a similar runtime complexity?*

Question 1.2 (1 pt). *Consider the binary search tree (BST) shown below. What is the runtime complexity of searching in such binary search tree? Why? Describe a insertion scenario that would lead to such a configuration. Explain your reasoning.*

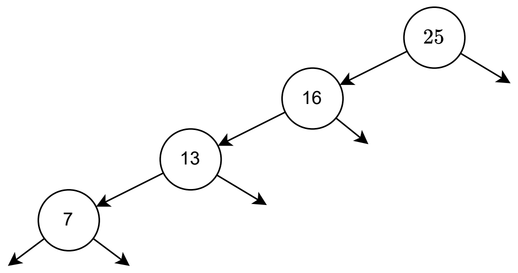


Figure 1: A sample binary search tree (BST)

Question 1.3 (1 pt.). *Consider a dynamic array, implemented using a static array with a capacity of 5. The dynamic array double in size whenever it gets full. Provided the length of the dynamic array is currently 5, what happens when we insert another item? Justify your answer.*

Question 1.4 (1 pt.). *Why is recursion considered less efficient than iteration (i.e. loops)? Justify your answer.*

Question 1.5. *Consider the directed graph shown below. How many paths connect Node D to Node G? Explain your reasoning?*

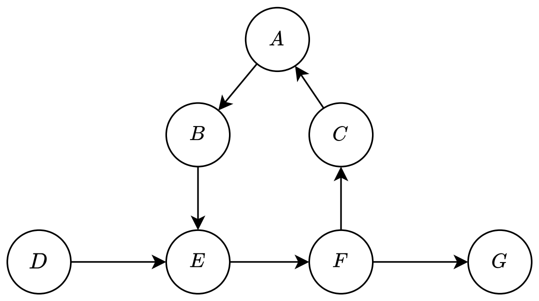


Figure 2: A directed graph

2 Nested-loop Join

Let's look at the *nested-loop join* algorithm: A simple yet naive solution to join two tables in databases. Consider the two tables shown below: The Customers table contains the identifier, the name and the country of the customers, whereas the *Order* table contains the date and amount (price) of their purchases.

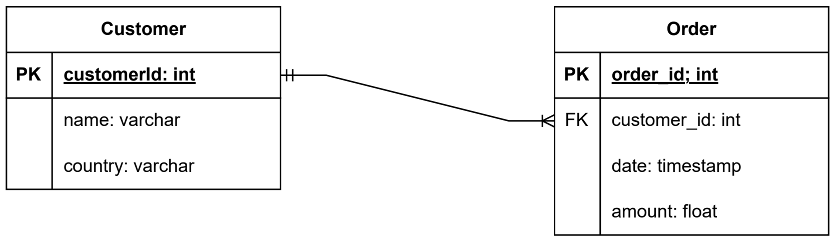
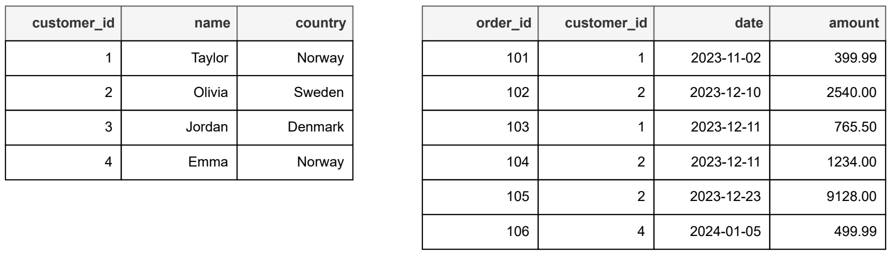


Figure 3: An Entity-Relationship diagram to illustrate the customer-order relationship

The figure below shows an example of data stored in these two tables.

Say we need to find all the order placed by customers from Norway. We could use the *nested-loop join* as follows:

```
List<Order> selectNorwegianOrders() {
    var selectedOrders = new ArrayList<Order>();
    for(Customer customer : customers) {
        if(customer.getCountry().equals("Norway")) {
            for(Order order : orders) {
                if(order.getCustomerId() == customer.getCustomerId()) {
                    selectedOrders.add(order)
                }
            }
```



customer_id	name	country
1	Taylor	Norway
2	Olivia	Sweden
3	Jordan	Denmark
4	Emma	Norway

order_id	customer_id	date	amount
101	1	2023-11-02	399.99
102	2	2023-12-10	2540.00
103	1	2023-12-11	765.50
104	2	2023-12-11	1234.00
105	2	2023-12-23	9128.00
106	4	2024-01-05	499.99

Figure 4: Example of data stored in the Customer and Order tables

```
        }
    }
    return selectedOrders
}
```

Question 2.1 (1 pt)**.** *Consider the sample tables shown above. With these very tables, how many comparisons will be performed by the* nested-loop join*? Note that comparisons only include the* `.equals()` *and the* `==` *operators. Explain you calculation.*

Question 2.2 (1 pt)**.** *Consider the nested-loop join algorithm below. Describe the best and worst-case scenarios for its runtime performance. Under which circumstances do they occur? Explain your reasoning.*

Question 2.3 (2 pt)**.** *In a nested-loop join operation, given n customers and m orders, calculate the number of comparisons needed in the worst case to find all orders from a specific country c. Express your answer as a function of n and m, and provide a detailed step-by-step explanation of your calculations.*

Question 2.4 (1 pt.)**.** *Is it correct to say that, in the worst case, the nested-loop join runs in $\Theta(n \times m)$ (Big-Theta)? Justify your reasoning.*

3 Book Cipher

A book cipher is a type of encryption where a book, so called the key, is used to encode messages. In this method, each letter of the message is replaced by a number representing the position or 'rank' of a word in the book that starts with that letter. To decode the message, the recipient refers to the same book, using the numbers to identify the starting letters of the corresponding words. Both sender and recipient must use the same text as the key to ensure that the message can be accurately decoded.

In the example below, we have encoded the message "HELLO PEOPLE" with the using the first paragraph of the book "Harry Potter and the Philosopher's Stone" as a key. We have first numbered each word of the paragraph, such that

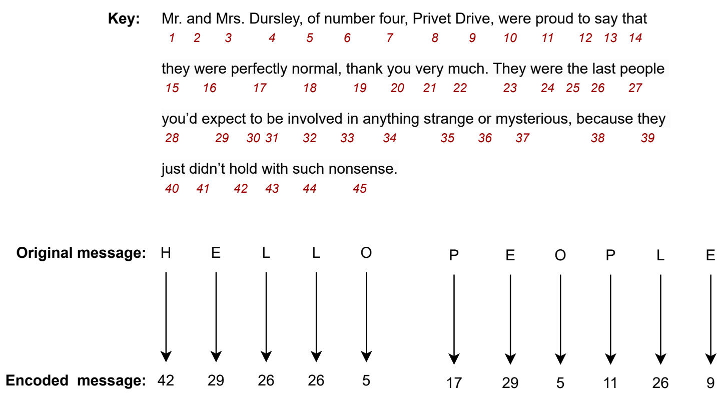

Key:
Mr. and Mrs. Dursley, of number four, Privet Drive, were proud to say that
1 2 3 4 5 6 7 8 9 10 11 12 13 14
they were perfectly normal, thank you very much. They were the last people
15 16 17 18 19 20 21 22 23 24 25 26 27
you'd expect to be involved in anything strange or mysterious, because they
28 29 30 31 32 33 34 35 36 37 38 39
just didn't hold with such nonsense.
40 41 42 43 44 45
Original message:
H E L L O P E O P L E
Encoded message:
42 29 26 26 5 17 29 5 11 26 9

Figure 5: Encoding "HELLO PEOPLE" using "Harry Potter"

- "H" becomes 42 because the 42th word is "hold", which starts with an "H".
- "E" becomes 29 because the 29th word is "expect", which starts with an "E"
- "L" becomes 26 because the 26th word is "last", which starts with a "L"
- "O" becomes 5 because the 5th word is "of", which starts with a "O"
- etc.

Note that several numbers can represent the same letter. In our Harry Potter example, "P" can be encoded by either of 8 (Privet), 11 (proud), 17 (perfectly), or 27 (people)

In this exercise, we assume we are given the following `Key` interface, with operation to encode and decode individual letters.

```
class Key {
    char decodeNumber(int number);

    int encodeLetter(char letter);
}
```

Question 3.1 (1 pt.). *Propose an algorithm to decode a given encoded text using a given key text. The text comes as a array of numbers, and the key as an object instance of the Key class. Assume the following:*

- *The Key class is already implemented: You do not need to implement its operations, but you should use them in your algorithm.*
- *The given text does not contains punctuation, space, or any other symbols besides the 26 letters of the English alphabet.*

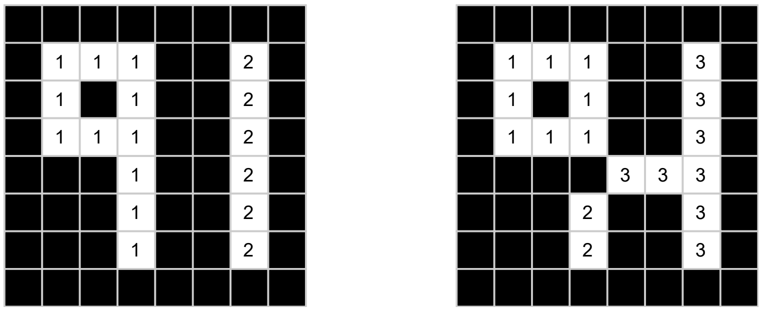


Figure 6: Two binary images and their connected components. On the left, there are only two connected components, whereas there are three on the right because pixel do not connect in diagonal

```
char[] decode(int[] encodedText, Key key) {
    // Provide some pseudo-code here.
}
```

Question 3.2 (1 pt). *Assume the* `decodeNumber` *method of the Key interface runs in $O(n)$ time (worst-case), where n is the number of words in the "key" text. What is the worst-case runtime complexity of decoding a text that contains m characters? Justify your answer based on the number of calls to* `decodeNumber`.

Question 3.3 (3 pts.). *Given that the* `decodeNumber` *method runs in $O(n)$ time, how would you improve its runtime efficiency. What pre-processing steps and what data structure would you use to achieve faster encoding of a letter? Explain how your proposed data structure and processing steps enhance performance.*

4 Connected Components Analysis

We now turn to computer vision and the detection of *connected components* in a binary image. A binary image is a grid of pixel that are either black or white.

The figures below illustrates the idea of binary images and connected components. A connected component is a group of contiguous white pixels. For instance on the left image, there are two connected components, labelled 1 and 2, respectively. By contrast, on the right image, there are three connected components, because their pixel are not contiguous (they are connected by a corner and not by an edge).

In this exercise, we are looking for an algorithm to detect such connected components in binary images, using graph algorithms.

Question 4.1 (1 pt.). *How can a depth-first search (DFS) help us find connected components in a graph?*

Question 4.2 (1 pt.). *How would you model a binary images such as those* *shown above using a graph so that a DFS can be used? What would be a node, what would be an edge?*

Question 4.3 (2 pt.)**.** *Propose an algorithm to count the connected components in a given image.*

Question 4.4 (2 pt.)**.** *In the worst case, what is the runtime complexity of your algorithm? Explain your reasoning?*