Foundations of Computer Science : Projects

These are optional projects that you may try after you have completed at least 6 past exam questions. Each project covers a particular area of Foundations of Computer Science that will be useful in some other courses in the tripos. Since these questions are not part of the course, you are allowed (and sometimes even encouraged) to search online or in books. You may also email me for hints.

If you take any of these projects, then you will present your findings during the revision session.

Project 0: Read and present a functional pearl

The "functional pearls" column contains several challenging problems, presented in the form of a dialogue. You can find a number of them online or in the "Pearls of Functional Algorithm Design" book by Richard D. Bird. Unfortunately, most of these pearls are written in Haskell, but with some Googling you should be able to map the syntax to the OCaml equivalents.

Here is is a list of pearls that are relevant to the course (or to the upcoming Algorithms course):

1. "How to mingle streams": This concerns enumeration of infinite list of infinite lists. We briefly touched upon this during supervisions.
2. "The smallest free number": This is the first pearl in the book. It is about finding the smallest natural number not in a given list. You can find here an OCaml version of the article.
3. "A surpassing problem": This is the second pearl in the book. It is about couting the number of pairs of values, where the indices are in reverse order of their values. You can find here an OCaml version of the article.
4. "Saddleback Search": This is the third pearl in the book. It is a common interview question where the goal is to find an element in a row-wise and column-wise sorted array. You can find here an OCaml version of the article.
5. "A selection problem": This is the fourth peral in the book. You can find here an OCaml version of the article.
6. "Not the maximum segment sum"
7. "Removing duplicates"

Note: Do not worry if you do not understand all of the details. You can present only parts of it.

Project 1: Binary arithmetic

In this project, you will implement Big Number binary arithmetic for integers in OCaml.

1. Decide on a representation for arbitrarily large numbers in OCaml. How would you represent negative numbers and zero?
2. Write an OCaml function to add these large numbers.
3. Write an OCaml function to negate these large numbers.
4. Write an OCaml function to multiply these large numbers.
5. Write an OCaml function to print these in decimal format.

Project 2: Cycles of permutations

This is good practice for the Algorithms, Discrete Maths (and also for interviews).

- A *permutation cycle* $(a_1 \; \ldots \; a_n)$ is a permutation that maps $a_1 \mapsto a_2$, $a_2 \mapsto a_3$, $\ldots$, $a_n \mapsto a_1$.
- Prove that each permutation can be decomposed into permutation cycles.
- Prove that the inverse of a cycle if the cycle written in reverse.
- [Java] Write an algorithm to find the cycles of a permutation. How fast is your algorithm?
- [Java] Given an array and a permutation, write a program to permute the elements of the array according to the permutation. Can you do it using $\mathcal{O}(1)$ extra memory space?
- (Optional Discrete Maths question) How many times do you have to apply a permutation to reach the original configuration?

Project 3: Gaussian elimination

1. Remind yourself of the Gaussian elimination algorithm and how it is used to solve systems of linear equations.
2. Write an OCaml function to solve $Ax = b$ for given matrix A and vector b. (**Note:** You do not need to handle the cases when there are infinitely many solutions or when there are no solutions)
3. What is the time complexity of your implementation?

Project 4: Nim's game

Read pages 9-11 from Feruson's book and answer the following questions.

1. Define the game of Nim.
2. How would you play optimally Nim's game for one pile?
3. State the strategy for the three pile Nim game and prove that this is optimal.
4. Write an OCaml function that given the current state of the piles it returns the next move according to the optimal strategy.
5. (Optional) How would you play misere Nim?

Project 5: Polynomials

Read lecture XI from the 2018/19 lecture notes.

1. What are the two possible ways to represent polynomials?
2. Write an OCaml function to add polynomials for each case. What is the time complexity of your algorithm?
3. Write an OCaml function to multiply polynomials for each case. What is the time complexity of your algorithm?
4. Write an OCaml function to divide polynomials for one of the cases. What is the time complexity of your algorithm?

You may want to try the following exercises for warm-up:

Exercise [Polynomial addition] Add the following polynomials:

1. $P_1 = 3 \cdot x^2 - 2x + 5$ and $P_2 = 6 \cdot x^5 - 4x^3 + 5x$
2. $P_1 = 2 \cdot x^4 - 4x + 5$ and $P_2 = 6 \cdot x^5 - 4x^4 + 4x + 2$

Exercise [Polynomial multiplication] Multiply the following polynomials:

1. $P_1 = 3 \cdot x^2 - 2x + 5$ and $P_2 = 6 \cdot x^5 - 4x^3 + 5x$
2. $P_1 = 2 \cdot x^4 - 4x + 5$ and $P_2 = 6 \cdot x^5 - 4x^4 + 4x + 2$

Exercise [Polynomial division] Divide the following polynomials (P_1 / P_2) and find the remainder:

1. $P_1 = 2x^3 + 7x^2 + 2x + 9$ and $P_2 = 2x + 3$
2. $P_1 = x^3 + 3x^2 - 4x - 12$ and $x^2 - x + 6$

Project 6: Sum of squares

1. Write an ML function that given r finds two positive integers x and y such that $x^2 + y^2 = r$.
2. What is the time complexity of your algorithm?
3. Write an ML function that returns a lazy list of all solutions.

Project 7: Taylor series represented using infinite lists

(This exercise is good practice for infinite lists)

Complete the 2001p1q6 question.

Project 8: Binomial Heaps

(This exercise asks you to read and implement Binomial heaps, a data structure that you will learn more about in the Algorithms course)

Read this article about the binomial heap and answer the following questions.

1. What problem do binomial heaps solve?
2. What are the properties of binomial trees?
3. How does the binomial heap make use of the binomial trees?
4. Implement the insert function for Binomial heaps. What is the time complexity of your implementation?
5. Implement the get-min function for Binomial heaps. What is the time complexity of your implementation?

Project 9: 2-3 Trees

(This exercise asks you to read and implement 2-3 trees, a data structure that you will learn more about in the Algorithms course)

Read the Functional pearl “On Constructing 2-3 trees” (article, slides) and provide an implementation of insert and find for 2-3 trees.

Project 10: Palindromes

(This exercise will be good practice for infinite lists)

1. Write an OCaml function to check if a given list is palindromic. A list is palindromic if it reads the same forwards as backwards.

2. Write an OCaml function to generate a lazy list of all possible binary sequences. (Follow the steps in the 2003p1q5 (a-d)).
3. Write an OCaml function to generate a lazy list of all possible palindromic binary lists. (Follow 2003p1q5 (e))
4. Can you extend this to no-binary palindromes?

Project 11: Othello player

(This exercise is good practice for search strategies and OOP)

1. Familiarise yourself with the Othello game.
2. Describe the interaction between `OtherlloBoard`, `OtheloPlayer`, `ComputerPlayer`, `HumanPlayer` and `GameManager` classes.
3. Implement the `OthelloBoard` so that you can retrieve the game state, display the game state and also make a move. Your class should throw an exception if any of the moves is invalid.
4. Implement the `GameManager` and `HumanPlayer`, so that two human players can play the game. (Be careful for the case where one user does not have a move).
5. (Optional) Implement a `ComputerPlayer` that looks ahead a few moves and has a function that evaluates how good a board is. Based on this it should make a move.

Note: This is a big project. A high-level approach to the first two/three questions will be considered a descent attempt.