CST0
COMPUTER SCIENCE TRIPOS Part IA

Tuesday 6 June 2023 09:00 to 12:00

COMPUTER SCIENCE Paper 1

*Answer **one** question from each of Sections A, B, C, D, and E.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

SECTION A

1 Foundations of Computer Science

Given the following unsorted lists of values, we need to sort them by the frequency with which individual values occur within the list.

```
let input1 = [4; 1; 3; 3; 2; 3; 1]
let input2 = ['a'; 'e'; 'i'; 'e'; 'o'; 'e'; 'i']
```

We will use a run-length encoding (RLE) to encode repeated elements in a more compact form. The RLE representation has type `('a * int) list`, where the `int` is the number of times the `'a` value is repeated.

(*a*) Define a `rev` function which can reverse an input list in $O(n)$ time complexity. [2 marks]

(*b*) Define a `sort` function for a list, using an algorithm of your choice. The first argument to `sort` is a comparator function with arguments `a` and `b` that returns `0` if `(a=b)`, a negative value if `(a<b)`, and a positive value if `(a>b)`.

```
val sort : ('a -> 'a -> int) -> 'a list -> 'a list
```

[6 marks]

(*c*) Define `rle_encode` that converts a sorted list of elements into an RLE encoded version, and `rle_decode` that converts them back to the original list.

```
val rle_encode : 'a list -> ('a * int) list
val rle_decode : ('a * int) list -> 'a list
```

[6 marks]

(*d*) Assume the comparator functions `int_cmp` and `char_cmp` are already defined:

```
val int_cmp : int -> int -> int
val char_cmp : char -> char -> int
```

Using all the earlier functions, define `freq_sort` that sorts the elements in ascending order of the total number of occurrences of each element within the list. When applied to the lists defined earlier, it should output:

```
# freq_sort int_cmp input1;;
- : int list = [2; 4; 1; 1; 3; 3; 3]
# freq_sort char_cmp input2;;
- : char list = ['a'; 'o'; 'i'; 'i'; 'e'; 'e'; 'e']
```

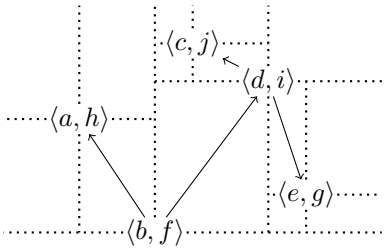
[6 marks]

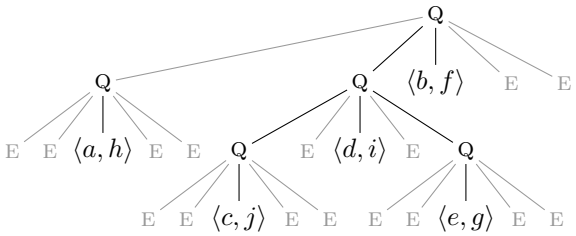
2 Foundations of Computer Science

This question considers the following type `qt` that represents *quadtrees* of points:

```
type point = int * int
type qt = Empty | Quad of qt * qt * point * qt * qt
```

A value `Quad(nw,ne,(x,y),sw,se)` contains points in the quadrants around `(x,y)`. `x` is the right bound of points in `nw` and `sw` and the left bound of points in `ne` and `se`. `y` is the upper bound of points in `sw` and `se` and lower bound of points in `nw` and `ne`.





(*a*) Write a function `compare_range` to find whether a number falls below, within or above a range:

```
type range = int * int
type rel = LT | IN | GT
val compare_range : int -> range -> rel
```

For example, `compare_range 3 (2,5)` should return `IN`, because $2 \leq 3 \leq 5$.

[2 marks]

(*b*) Write a function `has_point` to efficiently search a quadtree for a point:

```
val has_point : point -> qt -> bool
```

[8 marks]

(*c*) Write a function `has_point_in` to efficiently search a quadtree for a point within a rectangular region:

```
type rectangle = point * point
val has_point_in : rectangle -> qt -> bool
```

`has_point_in (p1,p2) qt` should return `true` if and only if `qt` contains a point in the rectangular region with lower-left corner `p1` and upper-right corner `p2`.

[10 marks]

SECTION B

3 Object-Oriented Programming

(*a*) This question covers the concept of variance in Java.

(*i*) Explain what is meant by Java arrays being covariant. [2 marks]

(*ii*) Provide a code example which type checks at compile time but yields a runtime exception due to covariant arrays. [2 marks]

(*iii*) Explain what is meant by Java generics being invariant and how it contrasts to the previous example. [2 marks]

(*b*) Explain the notion of *cohesion* in the context of classes. What is meant by high cohesion and low cohesion? Why is high cohesion desirable? [2 marks]

(*c*) Explain the notion of *coupling* in the context of classes. What is meant by high coupling and loose coupling? Why is low coupling desirable? [2 marks]

(*d*) Suppose we have a stock trading application that needs to notify users of changes in stock prices but allows users to choose how they want to be notified (e.g. via email or text message).

(*i*) Explain the intent and relevance of the observer design pattern for this problem. [1 mark]

(*ii*) Explain the intent and relevance of the strategy design pattern for this problem [1 mark]

(*iii*) Draw a UML diagram describing the key components of the observer design pattern. You do not need to recall the exact UML specification, instead you may provide a key explaining your notation. [2 marks]

(*iv*) Provide a skeleton of Java code for the subject as a class called `Stock`. Detail any methods required to notify and manage the users as well as an example of setting up at least two notification strategies for two different users. Explain any assumptions and tradeoffs of your approach. [6 marks]

4 Object-Oriented Programming

(*a*) Explain the concept of type erasure in Java. [2 marks]

(*b*) What do these types erase to?

(*i*) `List<List<Integer>>` [1 mark]

(*ii*) `List<String>[]` [1 mark]

(*iii*) `Map<String, List<Map<String, Integer>>>` [1 mark]

(*c*) This question covers the concept of immutability.

(*i*) Provide an example of a built-in immutable class in Java. [1 mark]

(*ii*) Explain what is required to declare an immutable class in Java. [2 marks]

(*iii*) Provide a code example of a declared immutable class called `ProductInfo` which contains two fields storing an `id` and a `description`. [2 marks]

(*d*) (*i*) Explain the meaning of the Liskov-Substitution principle. [1 mark]

(*ii*) Explain the meaning of the Single Responsibility principle. [1 mark]

(*iii*) Imagine that you are working in the insurance context. You can issue a Policy to insure a policy holder. There are two types of policies available: a life insurance policy and a car insurance policy. Both of these policies take into consideration the age of the policy holder. You need to calculate the premium (i.e. the cost) of each policy based on the following guideline:

- The life insurance takes 5 percent of the total sought coverage amount if the policy holder is under 35 of age or 10 percent otherwise
- For adults only, the car insurance takes 10 percent of the car value if the policy holder is over 30 of age or 20 percent otherwise

Define four classes `PolicyHolder`, `Policy`, `LifeInsurancePolicy` extends `Policy`, `CarInsurancePolicy` extends `Policy` such that they encapsulate this system and demonstrate a violation of the single Responsibility Principle in the `Policy` class and a Liskov-Substitution pre-condition violation in the `CarInsurancePolicy` class. [6 marks]

(*iv*) Describe what steps you would need to take to adhere to both the Liskov-Substitution and Single Responsibility principle. [2 marks]

SECTION C

5 Introduction to Probability

We consider the round of the last eight in the UEFA football playoffs, which involves a pairing of eight teams into four matches. Assume that these pairings are decided randomly by a lottery. Further, assume that four of the eight teams are considered *strong* and the other four are considered *weak*.

(*a*) What is the expected number of matches between a strong team and a weak team? [2 marks]

(*b*) What is the probability that all four matches are between a strong and a weak team? [3 marks]

Each match consists of a home game and an away game, and for each match the team which plays home at the first game is decided uniformly and independently at random.

(*c*) Let a random variable Z be the number of strong teams which play home at the first game.

(*i*) Determine the distribution of Z. [2 marks]

(*ii*) Compute the probability that $\mathbf{P}[Z \geq 1]$. [2 marks]

It turns out that matches between two strong teams are more likely to go into extra time (which can only occur in the second of the two games). Specifically, if both teams are strong, then this happens with probability 1/3; if exactly one team is weak, then this happens with probability 1/4; and finally, if both teams are weak, then this happens with probability 1/4, too.

(*d*) Consider one match, and let random variable $S \in \{0, 1, 2\}$ be the number of strong teams in that match, and let random variable $Y \in \{0, 1\}$ be an indicator which is 1 if and only if the second game of the match goes into extra time.

(*i*) Compute $\mathbf{E}[S]$ and $\mathbf{E}[Y]$. [2 marks]

(*ii*) Compute $\mathbf{Cov}[S, Y]$. [3 marks]

(*iii*) Are S and Y independent? Justify your answer. [2 marks]

(*e*) Not knowing the pairings, somebody tells you how many of the four matches went to extra time. How would you estimate the number of matches between two strong teams? [4 marks]

6 Introduction to Probability

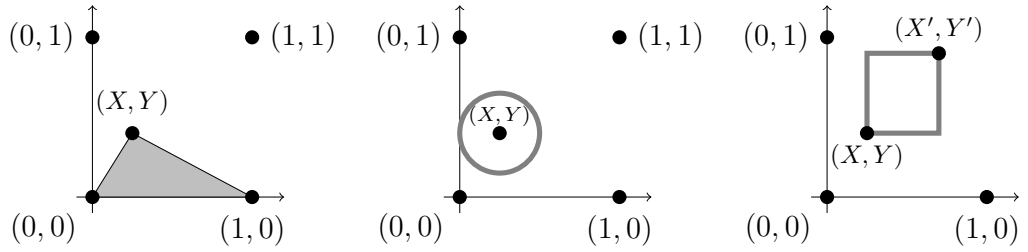
(*a*) Let $X \sim \mathsf{Uni}(0, 1/2)$ be a uniform continuous random variable. What are $\mathbf{E}[X]$ and $\mathbf{V}[X]$? [3 marks]

(*b*) Let $X \sim \mathsf{Uni}(0, 1/2)$ and $Y \sim \mathsf{Uni}(0, 1/2)$ be two independent uniform continuous random variables, and define $Z = \min(X, Y)$.

(*i*) What is the cumulative distribution function of Z? [2 marks]

(*ii*) What is $\mathbf{E}[Z]$? [3 marks]

(*c*) Let $X \sim \mathsf{Uni}(0, 1)$ and $Y \sim \mathsf{Uni}(0, 1)$ be two independent uniform continuous random variables.



(*i*) (*ii*) (*iv*)

(*i*) Consider a random triangle between the three points $(0, 0)$, $(1, 0)$ and (X, Y), as illustrated in the figure above. What is the expectation of the area? [2 marks]

(*ii*) Now consider a random circle with center (X, Y) such that the circumference is as large as possible but remains within the unit-square $[0, 1]^2$ (see figure). What is the expectation of the circumference? [4 marks]

(*iii*) Based on your answer from (*c*)(*ii*), what can you conclude about the expectation of the area of this circle? [2 marks]

(*iv*) Additionally, let $X' \sim \mathsf{Uni}(0, 1)$ and $Y' \sim \mathsf{Uni}(0, 1)$ be two uniform continuous random variables and assume X, Y, X', Y' are mutually independent. Consider a random rectangle with corner points (X, Y) and (X', Y'), which are diagonally opposite. What is the expectation of the circumference? [4 marks]

SECTION D

7 Algorithms 1

A DLL (doubly-linked list) may be trivially implemented as a chain of records, each record `r` having a pointer `r.p` to the previous record and a pointer `r.n` to the next. At the machine level, objects are identified by their memory address and pointers are memory addresses, so `r`, `r.p` and `r.n` are all pointers *and* memory addresses, as well as the handles by which we refer to those three records. At the logical level, there are two main classes: the record `r` and the DLL `d` itself—the latter being an object that contains, possibly among other things, a pointer `d.h` to the head record of the list.

An experienced machine code programmer suggests an improvement, the DLLx (Doubly-Linked List with XOR). By replacing the two `r.p` and `r.n` machine words with a single machine word holding their bitwise exclusive-or (`r.pn = r.p` $\oplus$ `r.n`), we save one word per record. The null pointer is represented by the word `0`. [*Hint:* For any three machine words `X, Y, Z`, whenever `X` $\oplus$ `Y == Z`, knowledge of any two of `X, Y, Z` lets you derive the third.]

(*a*) In a DLLx, how exactly do we move from one record to the next? Do we need any extra information to access the successor of record `r`, besides the address of `r` itself? If so, what? Do we need to store any extra information in the DLLx object besides the address of the head record `h`? If so, what? [4 marks]

(*b*) Let `a0`, `a1`, `a2`, ..., `a9` be the addresses of the records of a 10-element DLLx `d`, with `d.h == a0`. Express `a3` as a function of `a0`. [*Hint:* The final expression for `a3` will only use: parentheses, the two operators $\oplus$ and `.pn`, and the constant `a0`.] [4 marks]

(*c*) Give clear and well-commented pseudocode for an iterative function that returns the length of a DLLx `d`, where `d.h` is the address of its head record. [4 marks]

(*d*) Give clear and well-commented pseudocode for a function that inserts a new record `r` as the new head into a DLLx `d`, given `d` and `r`. [4 marks]

(*e*) What changes would you suggest to turn a DLLx into a CDLLx, meaning a *circular* doubly linked list with XOR, in which the successor of the last record is the head? Explain what fields the CDLLx object should contain. Assuming the CDLLx has at least three records, give a short sequence of algebraic expressions yielding the address of the third record from the end, going backwards from the head of the CDLLx. [*Note:* To avoid ambiguity with 0-based indexing: the first record from the end is the last record of the CDLLx. The 0-th record from the end would be the head.] [4 marks]

8 Algorithms 1

(*a*) Prove that, given any alphabet Σ, no lossless compression function $c : \Sigma^* \to \Sigma^*$ has the property that

$$\forall s \in \Sigma^* : |c(s)| < |s|$$

where the vertical bars around a string denote its length in symbols. [3 marks]

(*b*) Build the Huffman code for the following set of symbols and frequencies, drawing the corresponding tree. When combining two trees, put the tree with the smallest aggregate frequency on the 0 branch. List clearly, in order, the seven merge operations you performed. Finally, list the codeword for each symbol.

$$\{a : 0.15; \quad b : 0.25; \quad c : 0; \quad d : 0.1; \quad e : 0.12; \quad f : 0.18; \quad g : 0; \quad h : 0.2\}$$

[8 marks]

(*c*) Consider a string over the following set of symbols and frequencies, encoded with one byte per symbol using the ASCII codes for the lowercase letters.

$$\{a : 0.1; \quad b : 0.1; \quad c : 0.2; \quad d : 0.2; \quad e : 0.1; \quad f : 0.3\}$$

Imagine re-encoding it with the Huffman code shown below.

$$a : 010; \quad b : 011; \quad c : 01; \quad d : 100; \quad e : 101; \quad f : 11$$

What compression factor would you achieve? How much of this compression can be attributed specifically to Huffman coding? [3 marks]

(*d*) Consider strings over a set of 16 symbols. A trivial encoding, with no attempt at compression, will use codewords of 4 bits per symbol, without taking symbol frequencies into account. Prove that, when each symbol has a different frequency but the ratio between the largest and smallest frequency is strictly less than 2, Huffman coding of such strings cannot provide any compression compared to the trivial non-compressing encoding. [6 marks]

SECTION E

9 Algorithms 2

We are given a directed graph g with edge costs ≥ 0, and we wish to find the distance between two given vertices s and t. Your friend has the idea that we should waste less time exploring irrelevant parts of the graph, and suggests the following procedure:

"Run the standard version of Dijkstra's algorithm `dijkstra(`g`,`s`)` starting at s; and also run a variant `artskjid(`g`,`t`)` that starts at t and finds distances *to* t. Interleave these two by visiting one vertex with `dijkstra`, then one with `artskjid`, then one with `dijkstra`, and so on; terminate when one of them visits a vertex m that the other has already visited. Let $d = m$`.distance` be the distance computed by `dijkstra`, and let $e = m$`.ecnatsid` be the distance computed by `artskjid`; and return $d + e$."

(a) Explain how to implement `artskjid(`g`,`t`)` efficiently. What is the worst-case asymptotic running time of `artskjid`? [3 marks]

(b) Does your friend's procedure improve on the asymptotic worst-case time of simply running `dijkstra(`g`,`s`)`? Justify your answer. [8 marks]

(c) Your friend gives the following argument for correctness: "Since `dijkstra` visits vertices in order of increasing distance from s, and `artskjid` visits in order of increasing distance to t, the point where they meet must be on the shortest path from s to t."

Your friend's procedure can in fact give an incorrect answer. Demonstrate the problem with your friend's reasoning. [9 marks]

10 Algorithms 2

Consider a stack which supports `push` and `pop`, which we would like to back up periodically by copying it to disk. We can do this by keeping a counter x, initialized to 0 and incremented by 1 on every `push` or `pop`; if this increment results in $x = N$ then we copy the current stack to disk and reset x to 0. The cost of copying the stack to disk is proportional to the number of items on the stack, and the cost of pushing or popping an item (but not including the cost of copying to disk) is $O(1)$.

(*a*) Define *potential function*. Given a potential function, explain how one can use it to obtain amortized costs. [3 marks]

(*b*) Suppose that the stack size never exceeds N items. Using a potential function, or otherwise, show that the amortized costs of both `push` and `pop` are $O(1)$, asymptotic in N. [8 marks]

(*c*) Suppose there is no restriction on stack size. Show that the amortized costs cannot both be $O(1)$. [9 marks]

END OF PAPER