CST0
COMPUTER SCIENCE TRIPOS Part IA

Tuesday 10 June 2025 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

(*a*) Given the following incorrect code that implements a merge sort, identify five errors and suggest corrections with a brief explanation to make it work.

```
let length = function
  | [] -> 0
  | _ :: t -> 1 + length t

let rec merge l1 l2 =
  match l1, l2 with
  | [], l -> l
  | h1 :: t1, h2 :: t2 ->
      if h1 <= h2 then h1 :: merge t1 l2
      else h2 :: merge l1 t2

let rec split l l1 l2 n =
  match l, n with
  | [], _ -> (l1, l2)
  | h :: t, 0 -> split t l1 (h :: l2) n
  | h :: t, _ -> split t (l1 :: h) l2 n

let rec mergesort ls =
  match ls with
  | _ ->
      let n = length ls / 2 in
      let l, r = split ls [] [] n in
      merge (mergesort l) (mergesort r)
  | [] | [_] -> ls
```

[10 marks]

(*b*) Write a function `val check_sorted : 'a list -> bool` that returns `true` if the input list is already sorted, and define its time and space complexity. *(Hint: you can use the polymorphic* `<=` *operator here).* [4 marks]

(*c*) You find a library that implements `quicksort`, `bubblesort` and `insertsort` functions that all have the same type as `mergesort`. Write a function:

```
val checksort: ('a -> 'b list) list -> 'a -> unit
```

`checksort` should apply the input sorting functions and check that each result is sorted. Define a mechanism to signal if an unexpected mismatch is discovered. Give an example of using `checksort` to verify that `quicksort`, `bubblesort` and `insertsort` work for some sample integer list. [6 marks]

2 Foundations of Computer Science

The following type definition allows the representation of some mathematical expressions as an OCaml value:

```
type expr =
  | Add of expr * expr
  | Mul of expr * expr
  | Number of int
```

(*a*) Write the OCaml value that corresponds to the expression `(1+4)*(10+2)`. [2 marks]

(*b*) Write a function that will evaluate the numerical result of an `expr` argument. What is the OCaml type of your function? [4 marks]

(*c*) Another way to represent these expressions is to use "Polish notation". In this notation, the operator precedes its operands, and unlike the usual infix notation for expressions there is no need for parentheses. For example, the expression in part (*a*) would be written:

```
*  +  1  4  +  10  2
```

This can be reduced by the following steps:

```
*  5  +  10  2
*  5  12
60
```

Define a type `t` that could be used in a list to represent any expression that can be described with type `expr` above. [2 marks]

(*d*) Write a function `reduce` that performs one step of reduction. Give an example of applying `reduce` to the sample input above twice. *Hint: it should be able to reproduce the steps above.* [8 marks]

(*e*) Write a function `reduce_all` that reduces an argument of type `t list` until it cannot be simplified any further, and returns the simplest value. [4 marks]

SECTION B

3 Object-Oriented Programming

The Java Collections framework contains `PriorityQueue<E>`, which represents a priority queue based on a priority heap of objects of type `E`. Priority is specified implicitly by providing an ordering on `E` via either the `Comparable` or `Comparator` interfaces. Consider a task tracking app that uses the following class to represent a task:

```
public class WorkTask {
   private int priority;
   private String description;

   public WorkTask(String descriptor, int priority) {
      this.descriptor = descriptor;
      this.priority = priority;
   }

   public int getPriority() { return priority; }
   public int setPriority(int priority) {this.priority = priority; }
}
```

(*a*) (*i*) Compare and contrast Comparable and Comparator in Java's Collections framework. [3 marks]

(*ii*) Show how to make a `PriorityQueue<WorkTask>` maintain its contents highest priority first using:

(A) `Comparable` [2 marks]

(B) `Comparator` [2 marks]

(*iii*) Would you prefer Comparable or Comparator for this application? Explain your answer. [1 mark]

(*b*) PriorityQueue does not offer a method to change priorities. Instead, an object with a changed priority must be removed and reinserted. Using a design pattern that you should specify, write code for `AutoUpdatableQueue`, which extends `PriorityQueue` and automatically updates the queue when the priority of any object in the queue is updated. Make your solution flexible and demonstrate how to apply it to a queue of `WorkTask`s. [12 marks]

4 Object-Oriented Programming

(*a*) State the Open-Closed Principle and explain how it reduces the risk of regressions as software evolves [2 marks]

(*b*) Identify the four key principles of Object Oriented Programming and explain how each supports the open-closed principle [8 marks]

(*c*) Consider the Java method below. For each of the method bodies given, state whether they will compile and explain your reasoning.

```
public void wildcardMethod(List<? extends Number> numList,
                                   List<? Super Integer> intList)
{
  // Method body here
}
```

(*i*) `int sum = 0; for (Number n : numList) sum += n.intValue();` [2 marks]

(*ii*) `numList.add(6);` [2 marks]

(*iii*) `intList.add(6);` [2 marks]

(*iv*) `intList.add((Number)6);` [2 marks]

(*v*) `Integer i = intList.get(0);` [2 marks]

SECTION C

5 Introduction to Probability

For each of the three distributions, state the probability mass function and expectation:

(*a*) Binomial Distribution, [2 marks]

(*b*) Poisson Distribution, [2 marks]

(*c*) Hypergeometric Distribution. [2 marks]

Assume we want to load one container with 5 packets with weights X_i, $1 \leq i \leq 5$, which are independent exponential random variables with parameter $\lambda = 1/2$.

(*d*) Let $X := \sum_{i=1}^{5} X_i$ be the sum of weights. Compute the expectation $\mathbf{E}[X]$ and variance $\mathbf{V}[X]$. [3 marks]

We want to find a weight capacity w of the container such that with probability at least 0.95 we can load all 5 packets onto the container.

(*e*) Using Markov's inequality, find a solution for w. [3 marks]

(*f*) Using Chebyshev's inequality, find a solution for w. [4 marks]

(*g*) Using the Central Limit Theorem, find a solution for w.
Remark: You should **not** to give a numerical result, but your answer may involve the function $\Phi(.)$. [4 marks]

6 Introduction to Probability

Suppose that buses arrive at a bus stop randomly, so that at each minute at most one bus arrives. For example:

Minute	Bus arrives
0	yes
1	yes
2	no
3	no
4	yes
5	no
⋮	⋮

The waiting times (i.e., number of minutes) between two buses are independent and follow a geometric distribution which takes values in $\{1, 2, \ldots\}$ and has parameter $1/2$.

(a) What is the expectation of the waiting time and what is its variance? [2 marks]

(b) What is the probability that the waiting time is less than 5? [1 mark]

(c) Suppose that we know that the waiting time is at least 2. Conditional on this, what is the probability that the waiting time is at least 7? [2 marks]

(d) What is the distribution of buses that arrive in minutes $0, 1, \ldots, 10$, conditional on that a bus arrives at minute 0? What is the expectation? [3 marks]

Assume now that the waiting times are independent geometric random variables, but with *unknown parameter* $p \in (0, 1]$.

(e) Given you have n samples $X_1, X_2, ..., X_n$ of waiting times, construct an unbiased estimator for $1/p$. [2 marks]

(f) State the definition of the Mean-Squared Error. [2 marks]

(g) Analyse the Mean-Squared Error of your estimator in (e). [3 marks]

(h) Given you have n samples $X_1, X_2, ..., X_n$ of waiting times, construct an unbiased estimator for p. [5 marks]

SECTION D

7 Algorithms 1

(*a*) Find asymptotically tight lower and upper bounds for the following recurrence relation and explain your answer.

$$T(1) = 1$$

$$T(n) = 5T(n/5) + n^3$$

[7 marks]

(*b*) Find an asymptotically tight upper bound for the recurrence relation

$$T(1) = 1$$

$$T(n) = T(n-1) + \lg n$$

where lg denotes base-2 logarithms. Explain your answer. [3 marks]

(*c*) Show that the recurrence relation

$$T(1) = 1$$

$$T(n) = T(n/a) + \lg n$$

is in $\omega(\lg n)$ and $O(n)$, where $a > 1$ is a real-valued parameter and lg denotes base-2 logarithms. Is $T(n) \in o(\lg^2 n)$? Explain your answer. [7 marks]

(*d*) Let $T(n)$ be the recurrence relation defined by

$$T(1) = 1$$

$$T(n) = T(n/5) + T(3n/5) + kn$$

where $k > 0 \in \mathbb{R}$. Is $T(n) \in O(n)$? Justify your answer. [3 marks]

8 Algorithms 1

(a) Draw a diagram showing the final state of the data structure after inserting the following hash keys (in order) into an initially empty hash table of size 10. Resolve collisions by chaining, use the hash function $h(x) = x \bmod 10$, and use a method that ensures that, on average, searches for present and absent keys will take equal time.

$$3, 37, 16, 75, 27, 4, 8, 84$$

[5 marks]

(b) Draw a diagram showing the final state of the data structure after inserting the same keys (in order) into a table of size 16 using open addressing with the hash function $h(x) = x \bmod 16$, and the quadratic probe sequence given by

$$h(x, i) = \left(h(x) + \frac{i}{2} + \frac{i^2}{2} \right) \bmod 16$$

[6 marks]

(c) Suppose we need to support efficient deletion from a hash table of length `T.length` that uses open addressing and the same quadratic probe sequence as in part (b). You may assume that an appropriate hash function $h(x)$ has been implemented so you can call it where necessary.

(i) Provide pseudocode for the DELETE operation. [4 marks]

(ii) Provide pseudocode for the SEARCH operation. [3 marks]

(d) The probe sequence given in (b) hits every table position before revisiting any position, provided the table size is a power of 2. Explain why this is useful.

[2 marks]

SECTION E

9 Algorithms 2

(a) A connected, weighted, undirected graph $G = (V, E)$ is stored using an adjacency matrix to represent the edge set, E.

(i) Find asymptotically tight lower and upper bounds on the running time of breadth first search on G and explain your answers. [2 marks]

(ii) A minimum spanning tree, T, is found from G using Kruskal's algorithm. Devise an asymptotically optimal algorithm to find any two vertices in T with the greatest path cost between them. State and justify its asymptotic running time. [8 marks]

(b) Let $G = (V, E)$ be a weighted, undirected graph in which every edge weight is different. Let $d(u, v)$ be the edge weights, for all $(u, v) \in E$.

(i) Provide an algorithm that labels each vertex with the connected component containing it, running in $O(|V| + |E|)$ time. (The connected components of an undirected graph are connected subgraphs that are not part of any larger connected subgraph.) Justify that the running time of your algorithm is in $O(|V| + |E|)$. [6 marks]

(ii) Does the following greedy algorithm find a minimum spanning tree (MST) for G? If so, explain why and deduce the asymptotic running time. If not, provide a counterexample. [4 marks]

```
let MST = (V, E'={})      // A graph with G's vertices and
                          // initially empty edge set.
while (true)
  Label all vertices with their connected component in E'.
  Initialise the cheapest_edge for each component to NULL.

  For each (u,v) in E
    If u and v are in different components then
      If u.component.cheapest_edge != NULL
         && u.component.cheapest_edge.weight > d(u,v)
      then u.component.cheapest_edge = (u,v)
      If v.component.cheapest_edge != NULL
         && v.component.cheapest_edge.weight > d(u,v)
      then v.component.cheapest_edge = (u,v)

  If any component's cheapest edge != NULL, add it to MST.E'
  If all components' cheapest edges are NULL, return MST.E'
```

10 Algorithms 2

(*a*) True or false? Using the standard operations of a mergable priority queue, it is possible to construct a Fibonacci heap with the structure of a linked list, i.e. one node in the root list and it has one child, which has one child, ... to height n, where n is the number of nodes in the Fibonacci heap.

If true, outline the sequence of operations to achieve it; if false, prove why it cannot be achieved. [6 marks]

(*b*) Two programmers are worried about the difficulty of implementing a Fibonacci heap because the root list and every parent node's child list is cyclic.

(*i*) The first programmer says that they could use *acyclic* doubly linked lists instead of the usual cyclic lists, provided they make three adjustments:

- always move the MIN node to the head of the root list when the minimum changes;
- the pointer to the min node becomes a 3-tuple containing the key count and pointers to the first and last elements of the root list; and
- parents contain pointers to their first and last children instead of one pointer to an arbitrary child.

The programmer believes these changes will not alter the big-O costs for any of insert, extract-min, decrease-key, and destructive-union. Are they correct? Justify your answer by considering the work done for each operation in turn. [4 marks]

(*ii*) The second programmer says that, since they only need a Fibonacci heap implementation to speed up Dijkstra's algorithm on very large graphs, they could make the lists acyclic but not make the other changes. Are they correct? Justify your answer carefully. [4 marks]

(*c*) It is suggested that we can work out whether a directed graph G is acyclic by running Dijkstra's algorithm with a Fibonacci heap and checking whether any decrease-key operation causes a node to be cut out of the parent's child list. Either prove the correctness of this algorithm or provide a counterexample. [3 marks]

(*d*) You overhear suggestions to replace the Fibonacci heap used by Dijkstra's algorithm as a priority queue with a sorted linked list, a red-black tree, or a hash table. Deduce the big-O running times of Dijkstra's algorithm with these three alternative priority queues. [3 marks]

END OF PAPER