COMPUTER SCIENCE TRIPOS Part IA

NATURAL SCIENCES TRIPOS Part IA (Paper CS/1)
PSYCHOL. AND BEHAVIOURAL SCIENCES TRIPOS Part I (Paper CS 1)

Monday 4 June 2018 1.30 to 4.30

COMPUTER SCIENCE Paper 1

*Answer **one** question from each of Sections A, B and C, and **two** questions from Section D.*

*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) Sets containing integers can be represented as `int list` values. Consider two such representations called *unordered* and *ordered*. In the former elements can appear in any order; in the latter elements are required to be in ascending order. In both representations elements must not be repeated.

(i) Using the *unordered* representation give ML functions corresponding to set intersection and set union. [3 marks]

(ii) For your answers to Part (a)(i) give the associated time complexities, assuming both input sets have at most n elements. [2 marks]

(iii) Now, using the *ordered* representation, give ML functions corresponding to set intersection and set union along with their time complexities, noting reasons for any differences in complexity compared to those for the *unordered* representations. [4 marks]

(iv) Without giving any ML code, suggest a technique whereby set intersection for *unordered* can be implemented in $O(n \log n)$ time. [1 mark]

(b) One often hears "in ML, all functions take exactly one argument". Explain *two* techniques which enable us to circumvent this rule, illustrating your answer by giving ML definitions for the standard `map` and a variant which "takes the same arguments in the same order".
Call the variant `map'`. [3 marks]

(c) For each of the five following ML expressions give definitions of `f`, `g`, `h`, `xs` `ys` and `zs` (as appropriate) which cause the expression to evaluate to `true`, or explain, giving reasons, why this is impossible. (The functions `map` and `map'` are as discussed in Part (b).)

(i)
```
map f [1,2] = [[1,2],[3,4]]
```

(ii)
```
map g [1,2,3,4] = [[1,2],[3,4]]
```

(iii)
```
map (map h) xs = [[1,2],[3,4]]
```

(iv)
```
map map' ys = [[1,2],[3,4]]
```

(v)
```
map map zs = [[1,2],[3,4]]
```

[7 marks]

2 Foundations of Computer Science

We have a 2-player game in which players **A** and **B** take turns to remove either the leftmost or rightmost coin from of a row of coins of varying values. When no coins are left, the player with the higher total value wins.

Example: For a row of coins with values given by the list [20, 40, 30, 15], player **A** must select the rightmost coin (with value 15) in order to win with a total amount of 55, leaving player **B** with a total amount of 50.

(*a*) You are given three helper functions. The first function, `poplast`, takes a list and returns the list without its last element. The second function, `last`, takes a list of integers and returns the last value of that list. The third function, `max`, takes two integers and returns the larger of the two values.

Using these helper functions, write a recursive function `winning_diff` that takes a list of integers (representing the row of coins), and that returns the final difference of amounts between players **A** and **B**, assuming that player **A** goes first, and that player **B** plays optimally. If the difference is positive, player **A** wins, if it is negative, **B** wins. [8 marks]

(*b*) We are interested in implementing a *functional deque* that computes `poplast` and `last` in amortised constant time and that also enables access to the first element in amortised constant time. Write the code for the data type, and functions `poplast` and `last`. You may also need to code a function `norm` that guarantees amortised constant time in all circumstances. [8 marks]

(*c*) Consider the complexity of your algorithm for `winning_diff` for Part (*a*). For this question, assume that the three helper functions compute in constant time.

(*i*) Give the recurrence relation $T(n)$ for the running time of your algorithm, where n is the number of coins in the row.

(*ii*) State the complexity of the algorithm in O-notation.

[4 marks]

SECTION B

3 Object-Oriented Programming

Consider the following Java class.

```
 1  public class Child extends Parent {
 2
 3      public int x=0;
 4
 5      public boolean equals(Object obj) {
 6              if (this == obj) return true;
 7              if (!super.equals(obj)) return false;
 8              if (getClass() != obj.getClass())
 9                      return false;
10              Child other = (Child) obj;
11              if (x != other.x) return false;
12              return true;
13      }
14  }
```

(*a*) Explain why it is bad practice for member variables such as `x` to be `public`. [2 marks]

(*b*) For each of the lines 6–12, explain its purpose. Illustrate your answer with examples. [7 marks]

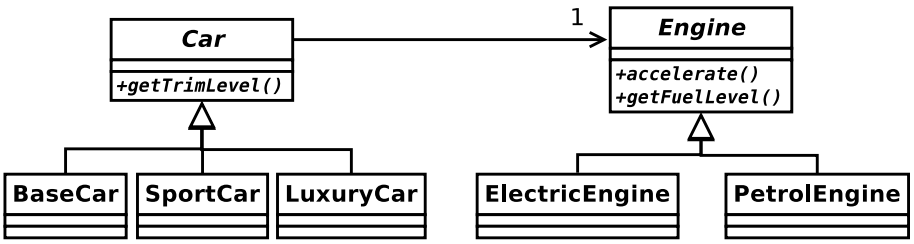
(*c*) What could result from a call to `c.equals(null)`, assuming `c` is a reference to a `Child` object? [3 marks]

(*d*) What would be the consequences of replacing the `equals` signature with `public boolean equals(Child obj)`? [4 marks]

(*e*) Explain why `Child` should also override the `hashCode()` method. [4 marks]

4 Object-Oriented Programming

A car manufacturer uses Java software to track current vehicles being built. The UML diagram below shows an excerpt of the current software structure. You should assume the presence of other appropriate fields and methods.



(*a*) Each car can be built to one of three trim levels: *base*, *luxury* or *sport*. They can also be configured with an *electric* or *petrol* engine. At various points in the manufacturing process the customer can choose to change the trim level.

(*i*) Explain in detail why the current structure does not support this. [3 marks]

(*ii*) Show how to refactor the structure to allow trim-level change using a standard design pattern, which you should identify. Explain how it addresses the problems you identified in Part (*a*)(*i*) and show how you would implement `getTrimLevel()` in `Car`. [5 marks]

(*iii*) Compare and contrast your solution to Part (*a*)(*ii*) with an alternative approach that stores the current trim level as a private `String` field within `Car`. [2 marks]

(*b*) The manufacturer decides to offer a vehicle with a hybrid engine that is both an electric engine and a petrol engine.

(*i*) Some languages support multiple inheritance of type, method implementation and state. Identify the problems each introduces and discuss the extent to which Java supports them. [5 marks]

(*ii*) Refactor the software structure to achieve the effect of multiple inheritance of type and method implementation for `HybridEngine` in Java without using `default` methods. Show how your solution would support an `accelerate()` method for `HybridEngine` that uses the electric engine if the battery level is above some threshold or the petrol engine otherwise. [5 marks]

SECTION C

5 Numerical Methods

(*a*) Consider implementing the natural logarithm function $\ln(t)$ for floating-point numbers using the McLaurin series:

$$\begin{aligned}\ln(1+x) &= \sum_{n=1}^{\infty}(-1)^{n-1}\frac{x^n}{n} \\ &= x \;-\; \frac{x^2}{2} \;+\; \frac{x^3}{3} \;-\; \frac{x^4}{4} \;+\; \cdots\end{aligned}$$

(*i*) List all special behaviours the natural logarithm function should have in different parts of its range and when t takes the special values NaN and $\pm\infty$. [2 marks]

(*ii*) The function must accept a broad range of numerical values but the series only converges when the absolute value of x is less than one, $|x| < 1$. Describe a range-reduction procedure that pre-processes the argument and post-processes the result so that the series always acts on small values of x. [6 marks]

(*iii*) State the two precision requirements normally expected for mathematical libraries. Considering the worst-case value(s) of x after range reduction, approximately how many terms are needed to meet one of these requirements for a single-precision implementation? Do you expect the other requirement to be met? [6 marks]

(*b*) The Trapezoidal Rule for numerical definite integration returns the area of the trapezium-shaped strips formed by each pair of adjacent points. The area under each such strip is:

$$\int_a^b f(x)\,dx \approx \tfrac{b-a}{2}\,[f(a) + f(b)]$$

(*i*) A program computes the area between two points A and B using N strips of width h. What should be taken into account when choosing h? Suggest a good value for h. [3 marks]

(*ii*) Assuming the best choice for h, what characteristics of $f()$ will affect the accuracy achieved? [3 marks]

6 Numerical Methods

(*a*) A programmer plans to replace single-precision floating-point arithmetic in a subroutine with a fixed-point implementation that is guaranteed to have at least the same range and precision. Roughly how many bits must the fixed-point representation have? [5 marks]

(*b*) The designers of a new computer architecture provide an instruction that uses the fixed-point implementation of Part (*a*) to sum long lists of single-precision floating-point numbers. This implements the rounding and re-normalisation only once at the end of the operation.

What are the benefits of such an instruction compared with folding the standard two-argument addition operator over the list? [4 marks]

(*c*) A tri-diagonal square matrix has all entries zero except for the leading diagonal and the two diagonals on either side of the leading diagonal.

(*i*) What is the execution cost, in terms of the number of operations, of Gaussian Elimination without pivoting, both for 'school method' and for L/U decomposition? [3 marks]

(*ii*) Without any pivoting, the L/U decomposition of a tri-diagonal matrix results in L having ones on its leading diagonal and another partial diagonal of non-zero coefficients and U also being largely all zeros. Write out the equations or pseudocode that determine L and U. What is the cost of solving a tri-diagonal set of simultaneous equations? [5 marks]

(*iii*) What happens to the tri-diagonal structure when pivoting is used? [3 marks]

SECTION D

7 Algorithms

This question considers sorting arrays of numbers.

(*a*) Mergesort can be implemented as a conventional two-way mergesort or a three-way mergesort. The latter splits the input into three and applies three-way mergesort recursively to each segment.

(*i*) Derive an expression for the worst-case number of comparisons needed to merge *two* sorted arrays of length $\frac{n}{2}$. [2 marks]

(*ii*) Derive an expression for the worst-case number of comparisons needed to merge *three* sorted arrays of length $\frac{n}{3}$. Consider merging three arrays at once as well as merging in pairs. [5 marks]

(*iii*) Using your answers for Parts (*a*)(*i*) and (*a*)(*ii*) and solving suitable recurrence relations, find expressions for the total number of comparisons needed for two-way and three-way mergesort. [6 marks]

(*iv*) Assuming comparisons to be the dominant cost, would you expect two-way or three-way mergesort to perform faster on an arbitrary array? [2 marks]

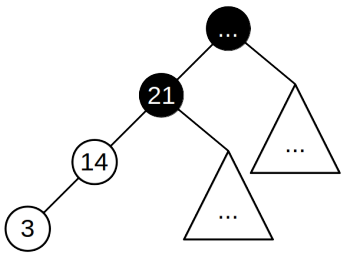
(*b*) Consider a large dataset of numbers between 0.0 and 1.0 drawn from some known non-uniform distribution. Describe a sorting algorithm that would be expected to perform better than the above mergesort on this data. What complexity would your algorithm have in terms of space and time in the worst and average cases? [5 marks]

8 Algorithms

A 2-3 tree is analogous to a 2-3-4 tree but has only 2-nodes and 3-nodes.

(*a*) Show in detail the steps to build a 2-3 tree from the sequence {7,3,9,8,11,10}. Highlight any procedural differences to building a 2-3-4 tree. [7 marks]

(*b*) A red-black tree can be based on a 2-3 tree. An example *red violation* for such a structure is sketched below, with red nodes represented using unfilled circles.



Sketch examples of the remaining red-violation cases, providing example values within the nodes. For each case, sketch its resolution, assuming each case occurs as a sub-tree of a larger tree. [5 marks]

(*c*) Consider restricting the 2-3 variant of a red-black tree so that red nodes may only lie on the left of a parent.

(*i*) Discuss the effect this has on the search and insert performance. How does it impact the implementation? [5 marks]

(*ii*) How does it affect the worst-case costs of finding the minimum and maximum values in the tree? [3 marks]

9 Algorithms

The *chaining* collision-resolution scheme for hash tables uses an array where each element is a linked list.

(*a*) For a hash table that uses chaining to resolve collisions with an array of length m and n keys inserted, the average asymptotic complexity for search and insert is $O(1 + \frac{n}{m})$.

(*i*) Explain why search and insert are considered to be $O(1)$ and not $O(n)$ in practice. [2 marks]

(*ii*) Give the worst-case complexities for search and insert. State the factors that determine the performance observed in practice. [5 marks]

(*b*) An alternative hash-table implementation replaces the linked lists in each slot with red-black trees. Discuss the advantages and disadvantages of this change. [4 marks]

(*c*) The linked lists may also be replaced by dynamically sized arrays. When full, an array is expanded by a factor of k.

(*i*) Derive a *potential* to compute the amortised cost of adding an element to such an array using the potential method. On inserting an element that triggers an expansion, your potential should be zero just after the expansion but before the insertion of the new element. [3 marks]

(*ii*) Use your potential to compute the amortised cost of adding an element to the array. [3 marks]

(*iii*) What factors would influence your choice of k when using the arrays in a hash table? [3 marks]

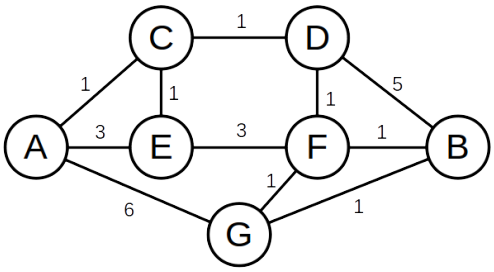
10 Algorithms

(*a*) Let `dijkstra_path`(g, a, b) be an implementation of Dijkstra's shortest path algorithm that returns the shortest path from node a to node b in a graph g. Prove that the implementation can safely terminate when it first encounters node b. [5 marks]

(*b*) Consider all paths in a graph from a to b, ordered from shortest to longest. Assuming $p =$ `dijkstra_path`(g, a, b) is the first path in this collection, an algorithm to find the second path considers deviations from the vertices of p. An algorithm to do this is given below.

```
function second_path(Graph g, Vertex a, Vertex b):
   p = dijkstra_path(g,a,b)
   best_so_far = []
   for i = 1 to len(p)-1:
      t = p[:i]    # First i elements of p
      c = g.get_edge_weight(p[i], p[i+1])
      g.set_edge_weight(p[i], p[i+1], infinity)
      t.append(dijkstra_path(g,p[i],b))
      if (len(best_so_far) == 0 or
          cost(t) < cost(best_so_far)):
        best_so_far = t
      g.set_edge_weight(p[i], p[i+1], c)
   return best_so_far
```

(*i*) Show the steps of this algorithm on the following graph, from A to B.



[5 marks]

(*ii*) What is the asymptotic complexity of this algorithm in terms of the number of edges, E, and the number of vertices, V? Assume the implementation of Dijkstra's algorithm uses a priority queue based on a Fibonacci heap. [4 marks]

(*iii*) Show how to adapt this algorithm to find the top-k shortest paths in the collection. State the complexity of the adapted algorithm. [6 marks]

END OF PAPER