University of Glasgow

Thursday 08 May 2025
09:30-11:00 BST
Duration: 1 hour 30 minutes
Timed exam – fixed start time

DEGREES OF MSc, MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

ALGORITHMICS II (H)
COMPSCI4003

(Answer all 4 questions)

This examination paper is a digital on-campus assessment and is worth a total of 60 marks

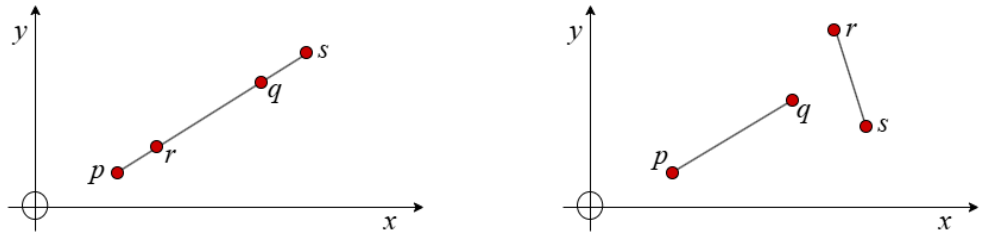
1. Geometric algorithms [16 marks]

(a) Let *p*, *q*, *r* and *s* be four distinct points in the plane. Recall that the line segment intersection algorithm for determining (without using division) whether the line segments **p—q** and **r—s** intersect employs two tests, namely the "On Opposite Sides" test (used twice) and the "Bounding Box" test. These tests are as follows:

```
onOppositeSides(p,q,—r—s—)
onOppositeSides(r,s,—p—q—)
boundingBox(p—q,r—s)
```

Give the Boolean results of each of the above three tests in each of the following two cases, and justify your answer briefly in each case:

(i) (points are collinear) **(ii)**



[6]

(b) Let *P* be a set of *n* distinct points in the plane, where $n \geq 4$. Suppose a subset *R* of points in *P* are coloured red, and a subset *B* of points in *P* are coloured blue, where $R \cup B = P$ and $R \cap B = \varnothing$. Assume that $|R| \geq 2$ and $|B| \geq 2$.

Describe in outline an $O(n \log n)$ algorithm for computing the distance between a closest pair of points in *P* having the same colour. You can use (without further explanation) the fact that there is an $O(n \log n)$ algorithm for computing the distance between a closest pair of points in *P*.

[3]

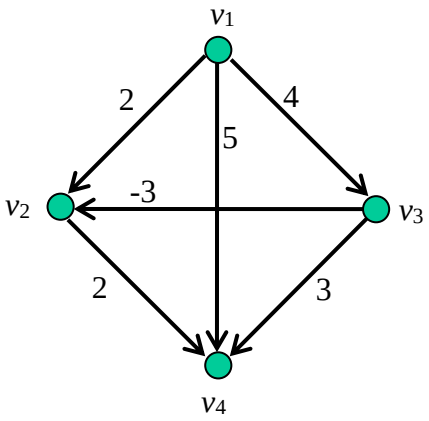
(c) Let *S* be a set of *n* distinct points in the plane. Describe an $O(n^2 \log n)$ algorithm that determines whether any three points in *S* form the corners of an equilateral triangle, one of whose sides is either horizontal or vertical. You need not justify the complexity of your algorithm.

You may use the fact that an equilateral triangle with a horizontal side of length *x* has height $\sqrt{3}x/2$.

[7]

2. Graph and matching algorithms [15 marks]

(a) Let $G=(V,E)$ be the weighted directed graph shown below, where the weight of each edge is shown beside the edge. Using the Floyd-Warshall algorithm, compute the length of a shortest path between each pair of vertices in G. Illustrate your working by showing each of matrices D_0, D_1, D_2, D_3, D_4. Make sure that you show how to find the length of a shortest path between each pair of vertices.



In your answer, you can represent an $n\times n$ matrix by giving each row followed by "\\", for example (1,0,0,0) \\ (0,1,0,0) \\ (0,0,1,0) \\ (0,0,0,1) is a representation of the following matrix:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[8]

(b) Let R be a finite binary relation on a set X. The *transitive closure* R^* of R is the smallest relation on X that contains R and is transitive (that is, for any three elements x, y and z in X, if $R^*(x, y)$ and $R^*(y, z)$ then $R^*(x, z)$). (Here, "smallest" means that $|R^*|$ is as small as possible subject to the required properties.)

For example, suppose that X is a set of airports, and R is the binary relation "there is a direct flight from x to y" for x, y in X (note that R is not necessarily symmetric). Then R^* is a binary relation, where $R^*(x, y)$ means that it is possible to fly from x to y using one or more flights, for x, y in X.

Show how the Floyd-Warshall algorithm can be used to compute the transitive closure R^* of R in $O(n^3)$ time, where $n=|X|$.

Notes: (1) you need not modify the version of the Floyd-Warshall algorithm as described in the lectures;

(2) you need not justify the complexity of your approach.

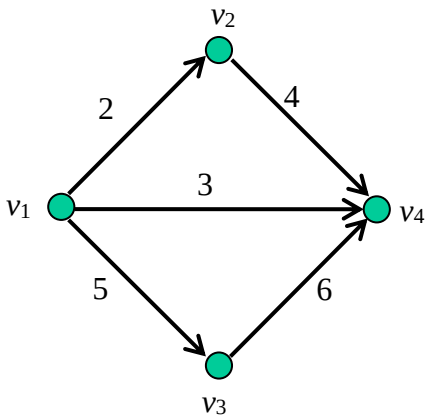
[4]

[*Question 2 is continued overleaf*]

[*Question 2 is continued from the previous page*]

(c) Let $G=(V,E)$ be a weighted directed acyclic graph, where all edge weights are positive integers and represent edge capacities. The *Widest Path* problem is defined as follows. Given two vertices u and v in V, the *capacity* of a path P between u and v is defined to be the minimum weight (capacity) of an edge on P, or $-\infty$ if no path between u and v exists. The problem is to compute the maximum capacity of a path between u and v, for any two vertices u and v.

For example, consider the following weighted directed graph:



There are three paths between v_1 and v_4. The path via v_2 has capacity 2 (edge (v_1,v_2) has the smallest weight among the two edges on this path from v_1 to v_4), the path along edge (v_1,v_4) has capacity 3, whilst the path via v_3 has capacity 5. Hence the maximum capacity of a path between v_1 and v_4 is 5.

The Floyd-Warshall algorithm can be modified to solve the Widest Path problem in $O(n^3)$ time. Recall that the Floyd-Warshall algorithm as covered in the lectures involves the computation of distance matrices D_k, for k=0, 1, 2,…, n, where $n=|V|$.

For the Widest Path problem, D_0 can be defined as follows:

$$D_0(i,j) = \begin{cases} 0, \text{ if } i = j \\ wt(i,j), \text{ if } i \neq j \text{ and } (i,j) \in E \\ -\infty, \text{ otherwise} \end{cases}$$

Show how to define D_k for k=1, 2,…, n.

[3]

3. String and text algorithms [15 marks]

Let S be the string `barbarian`.

(a) Construct the suffix tree T for S, using short edge labels. You may use the fact that the last character of this string does not appear elsewhere in the string, and thus there is no need to append `$` before constructing the suffix tree.

Give your answer using the following representation of a tree. Assume that r is the root of the tree. Label remaining nodes in the tree as appropriate. That is, a leaf node can be labelled by the suffix number that it corresponds to, whilst a branch node other than the root can be given a letter (other than r) as a label. Ensure that you give the short edge labels on the edges. For example, $r(2,4)v$ represents an edge with label (2,4) from the root node r to branch node v, whilst $v(5,9)2$ represents an edge with label (5,9) from branch node v to the leaf node with suffix number 2.

[11]

Now let X be a string of length n over an alphabet $\sum$.

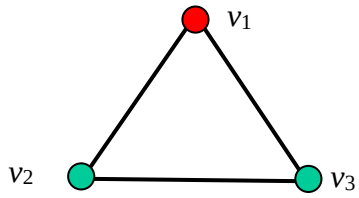
(b) Describe in outline an O(n) algorithm to find a longest repeated substring of X that occurs at least r times in X, for a given integer $r \geq 2$ (overlapping embeddings are permitted). You need not justify the complexity of your algorithm.

[4]

4. Algorithms for hard problems [14 marks]

Let $G=(V,E)$ be a connected undirected graph where $|V| \geq 2$. A *dominating set* is a set of vertices S such that, for each vertex $v \in V$, either $v \in S$ or v is adjacent to some $w \in S$.

For example, if G comprises vertices v_1, v_2, v_3 and edges $\{v_1,v_2\}$, $\{v_2,v_3\}$, $\{v_1,v_3\}$ (i.e., G is a triangle), then $S=\{v_1\}$ is a dominating set. This is illustrated in the diagram below.



Here, $v_1 \in S$, whilst each of v_2 and v_3 are adjacent to $v_1 \in S$.

The Minimum Dominating Set problem (MDS) is defined as follows:
Instances: the set of all graphs G
Feasible solutions for a given graph G: the set of all dominating sets in G
Measure of a feasible solution S: $|S|$ (i.e., the cardinality of S)
GOAL: min (to minimise)

It is known that MDS is NP-hard.

Consider the following approximation algorithm A for MDS:

```
X = V;
S = ∅;
while ( X ≠ ∅ )
{  choose v∈X;
   S = S ∪ {v};
   X = X \ {v};
   for ( each w adjacent to v )
      X = X \ {w};
}
return S;
```

(a) Show that A returns a dominating set for a given graph $G=(V,E)$.

[3]

(b) Show that A does not have performance guarantee c, for any $c > 1$. *Hint*: give an example of a graph where A might perform poorly. You should not need to draw the graph; it should be small enough so that you can just write down the set of vertices and the set of edges. Demonstrate what sequence of choices might lead A to perform poorly on this input. By giving an optimal solution, show that A does not have a performance guarantee of c on this input.

[6]

[*Question 4 is continued overleaf*]

[*Question 4 is continued from the previous page*]

Recall that a *vertex cover* is a set of vertices S such that, for each edge $\{u, v\} \in E$, either $u \in S$ or $v \in S$. A vertex cover in G is also a dominating set in G. Recall the following 2-approximation algorithm B for the Minimum Vertex Cover problem:

```
F = E;
S = ∅;
while ( F ≠ ∅ )
{   choose e={u,v}∈F;
    S = S ∪ {u, v};
    F = F \ {e};
    for ( each f adjacent to e in F )
        F = F \ {f};
}
return S;
```

(c) Construct a graph H to show that that B does not have performance guarantee c (as an approximation algorithm for MDS), for any $c > 1$. You need not show that B fails to achieve a performance guarantee of c on H. *Hint*: as in Part (b), H should give an example of a graph where A might perform poorly. You should not need to draw H; it should be small enough so that it is enough just to write down the set of vertices and the set of edges in H.

[5]