University of Glasgow

Tuesday 30 April 2024
2.00pm – 3.30pm BST
Duration: 1 hour 30 minutes

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

Note: in your solutions, if you use any standard algorithms that have been covered in lectures and/or in solutions to tutorial questions, you can simply refer to the relevant lecture or tutorial question solution without having to write out the algorithm in full.

1. Geometric algorithms [16 marks]

Let S be a set of n triangles in the plane, where each triangle has one side parallel to the x-axis and one side parallel to the y-axis. Assume that the length of the side parallel to the x-axis is equal to the length of the size parallel to the y-axis. Two triangles t_1 and t_2 are said to *intersect* if their sides have a non-empty locus of intersection, which includes two sides sharing one point. This means that t_1 and t_2 do not intersect if t_1 is completely contained in t_2, for example.

In Figure 1 below, parts (i), (ii) and (iii) each show different examples of two triangles intersecting, whilst part (iv) illustrates a case where two triangles do *not* intersect.

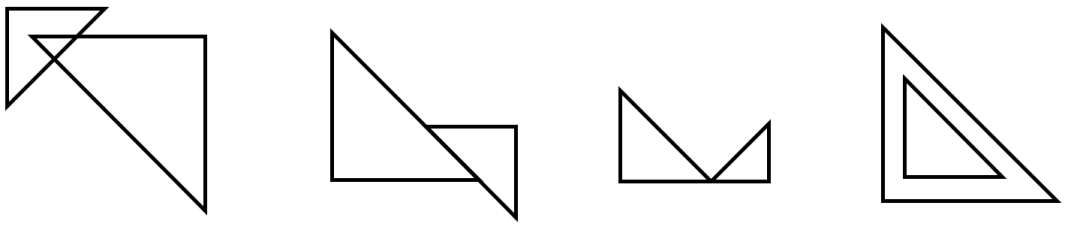


Figure 1: **(i)** **(ii)** **(iii)** **(iv)**

(a) Describe an O($n$log n) algorithm to determine whether any two triangles in S intersect. Your algorithm need only output "yes" or "no". You need not justify the complexity of your algorithm in this question part.

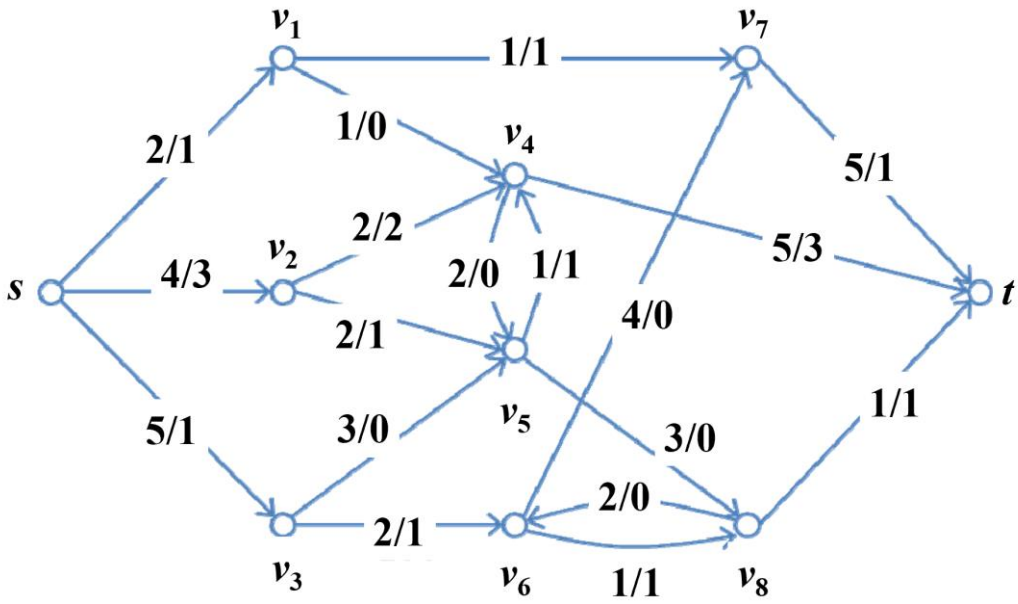
[12]

(b) Show briefly that your algorithm from Part (a) has O($n$log n) complexity.

[4]

2. Graph and matching algorithms [16 marks]

Let $G=(V,E)$ be the network with initial flow f illustrated below, where s is the source of G and t is the sink of G. Beside each edge (u,v) is a pair of integers a/b, where a denotes the capacity $c(u,v)$ of (u,v), and b denotes the initial flow $f(u,v)$ across (u,v).



(a) Using f as the initial flow, find a sequence of augmenting paths leading to a maximum flow in G. You should write out each augmenting path that you use, but you need not show the residual graph at each step. Each augmenting path can simply be written as a sequence of vertices, e.g., $s \rightarrow v_1 \rightarrow v_7 \rightarrow t$ (in the editor, $\rightarrow$ can be written as ->, for example).

[8]

(b) Prove that your final flow has maximum possible value by indicating an appropriate cut in G. The cut can be described as a set of edges, e.g., (s, v_1), (s, v_2), (s, v_3).

[3]

Now let $H=(V,E)$ be an arbitrary network, where s is the source of G and t is the sink of G, and $c(u,v)$ denotes the capacity of edge $(u,v) \in E$.

(c) Let $H'=(V',E')$ be the network obtained from H by increasing the capacity of one edge in E by 1 (i.e., by one unit). Let f be a maximum flow in H and let f' be a maximum flow in H'. Show that $\text{val}(f) \leq \text{val}(f') \leq \text{val}(f) + 1$.

[5]

3. String and text algorithms [15 marks]

Let *X* be the string `ACTGGC` and let *Y* be the string `CATGCG`. Suppose that the length of a Longest Common Subsequence (LCS) of *X* and *Y* is to be computed using dynamic programming combined with memoisation and virtual initialisation.

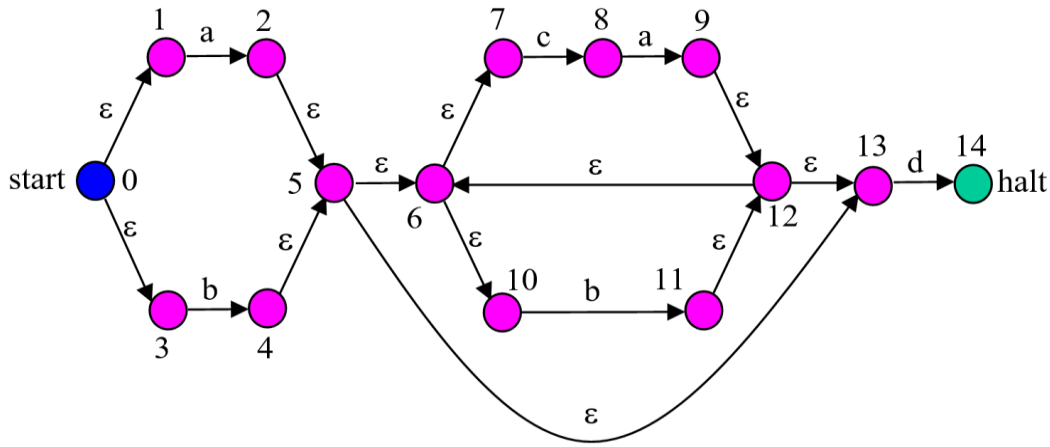
(a) By simulating the operation of the algorithm on a relevant dynamic programming table *T*, indicate which entries of *T* would ***not*** be computed during the algorithm's execution.

Note 1: it is not necessary to *compute* actual table entries, only to determine *which* entries would not be computed.

Note 2: the table entries that would *not* be computed can simply be written out as a set of pairs of the form (*i*, *j*), each corresponding to row *i* and column *j* of the dynamic programming table, e.g., (5,6), (4,3), (0,2).

[7]

Let Σ={`a`, `b`, `c`} be an alphabet and let *R* be the following regular expression over Σ: `(a|b)(ca|b)*d`. Consider the following non-deterministic finite-state automaton (NDFA) *A* for *R*:



(b) Let *T* be the string `bcdabcad`. Illustrate Algorithm `simulateNDFA` from Lecture 12 by showing the set of states of `x` that are reached, corresponding to each position of *T*, during the simulation of *A*.

[8]

4. Algorithms for hard problems [13 marks]

Consider the following instance of the Subset Sum Optimisation problem (SSO):

$$x_1=3,\ x_2=7,\ x_3=13,\ x_4=15,\ \ t=20$$

(a) Suppose $\varepsilon=2$ in the definition of the polynomial-time approximation scheme (ptas) for SSO given in Lecture 18. Describe the contents of the set `s` at the end of each iteration of the main loop involving variable `i`. Ensure that your solution clearly indicates the elements in the set that remain after any removals.

Hint: Given positive integers p, q, r, s, note that $p/q \geq r/s$ if and only if $ps \geq qr$.

[6]

(b) What is the measure value returned by the ptas, and what is the optimal measure value, for the above instance?

[2]

(c) Suppose that, in the ptas for SSO, the contents of the set `s` are stored in the variable `s[i]` after each iteration of the main loop involving variable `i`. Thus when the loop terminates, the contents of `s` that remain after each loop iteration are stored as sets in the variables `s[1]`, `s[2]`, …, `s[n]`. Assume that `s[0]={0}`. Show how to modify the ptas in order to output a set of elements from among $x_1, x_2, \ldots, x_n$ that, when summed together, achieve the measure value returned by the ptas. The elements do not have to be output in any particular order.

[5]