

CS3 Algorithms and Data Structures 2011/12
Exercise Sheet 2

Issue date: 1st November 2011

The deadline for this coursework is *NOON* on Tuesday 22nd November, 2011 (**extended date**). Please submit your solutions to the ITO on level 4 of Appleton Tower, either to a member of staff, or into the box outside (you should not expect your coursework to be stamped immediately, I was wrong about that for Exercise 1). Your coursework should be submitted in clear handwriting.

Remember that the School’s policy is that late coursework will not be accepted without good reason. If you have good reason, this goes through your DoS.

This exercise is worth 50% of the coursework for A&DS. Questions are worth varying numbers of marks - Q1 (20), Q2 (15) and Q3(15). The “marks” given in the margins of this sheet may be interpreted as percentages of the overall coursework credit for A&DS.

- In this question we consider the problem of constructing *Minimum Link Spanning Trees (MLSTs)* and their relationship to *Minimum Spanning Trees (MSTs)*.

We are given an undirected graph $G = (V, E)$ and a positively-valued weight function $W : E \rightarrow \mathbb{R}^+$ as usual. We assume the graph is connected, as usual. For any particular *Spanning Tree* T (whether an MST, or not an MST), we define the *link-weight* $\ell(T)$ of T as follows:

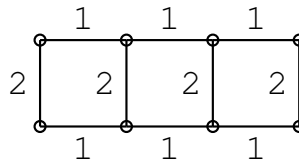
$$\ell(T) = \max_{e \in T} W(e).$$

A spanning tree T of the graph (G, W) is said to be a *Minimum Link Spanning Tree (MLST)* if there is no spanning tree T^* of G such that $\ell(T^*) < \ell(T)$.

- Prove that every MST T of a connected, positively-weighted graph (G, W) is *also* [10 marks] a MLST for (G, W) .

Hint: This proof will use ideas very similar to the proof of Prim’s algorithm. If you understand the proof of Prim’s algorithm well, you will be able to come up with a proof (using the proof by contradiction technique) of this result.

- Give an example of a connected positively-weighted graph on 4 vertices, which has [2 marks] some MLSTs which are *not* MSTs.
- Consider the following grid with 8 vertices:



- How many MSTs does this grid have? Justify your answer. [2 marks]
- How many MLSTs does this grid have? Justify your answer in detail. [6 marks]

2. Consider the problem of neatly formatting a paragraph of words on the screen. Suppose that L characters fit into one line, and our paragraph consists of the words x_1, \dots, x_n (in this order). Suppose that the length of each word is at most L so that it fits into a single line. When printing, we always put an empty space between two successive words. We want to print the paragraph in such a way that the sum of the squares of the number of empty spaces at the end all lines except for the last line is minimised.

Give a $O(n^3)$ time *dynamic programming (DP)* algorithm solving this problem efficiently. Justify the correctness of your algorithm and also justify its running time.

Hint: Suppose we split the paragraph into m lines as follows:

$$\begin{array}{c} x_1 x_2 \dots x_{i_1} \\ x_{i_1+1} \dots x_{i_2} \\ \vdots \\ x_{i_{m-1}+1} \dots x_n \end{array}$$

Define $i_0 = 0$ and $i_m = n$. Then we have $0 = i_0 < \dots < i_m = n$. For $1 \leq p \leq m$, the empty space e_p at the end of line p is

$$L - \left(\sum_{j=i_{p-1}+1}^{i_p} |x_j| + i_p - i_{p-1} - 1 \right),$$

where $|x_j|$ denotes the length of word x_j . The term $i_p - i_{p-1} - 1$ accounts for the empty spaces between the words. Of course i_1, \dots, i_{m-1} must be chosen in such a way that $e_p \geq 0$ for $1 \leq p \leq m$.

The overall “cost” (in terms of space) of printing the paragraph this way is

$$\sum_{p=1}^{m-1} e_p^2.$$

Note that we only sum up to $m - 1$ because the empty space at the end of the last line does not count. Your algorithm is supposed to minimise this cost.

Make sure you cover all the following points:

- Describe the tables used by your dynamic programming algorithm and explain what the entries of the tables represent. *[5 marks]*
- Describe the algorithm (either using pseudocode, or giving a detailed explanation in your own words) that you use to iteratively update your dynamic programming tables and eventually compute the solution. Pay particular attention to the order in which a table is filled. Explain why your algorithm is correct. Make sure you describe (and justify) any recurrences that are used by your algorithm. *[5 marks]*
- Show that the running time of your algorithm is $\Theta(n^3)$ (or, if you have a better algorithm, show its $\Theta(\cdot)$ expression). *[3 marks]*
- Give an algorithm to print out the paragraph in its optimal format (accessing the dynamic programming tables to do this). *[2 marks]*

3. Throughout this question, $G = (V, E)$ is a flow network with capacity function c , source vertex s and sink vertex t .

(a) Consider the specific flow network G with vertex set $\{s, \alpha, \beta, \gamma, \delta, t\}$, and with capacities and flows as given in the following tables:

c	s	α	β	γ	δ	t
s	0	9	3	0	0	0
α	0	0	2	0	7	0
β	0	0	0	6	0	0
γ	0	0	0	0	3	6
δ	0	0	0	0	0	5
t	0	0	0	0	0	0

f	s	α	β	γ	δ	t
s	0	5	3	0	0	0
α	-5	0	2	0	3	0
β	-3	-2	0	5	0	0
γ	0	0	-5	0	2	3
δ	0	-3	0	-2	0	5
t	0	0	0	-3	-5	0

Draw the *residual network* associated with this network and flow, explaining how you arrived at it. [4 marks]

(b) By referring to the residual network, write down the (unique) augmenting path in G for the given flow. What is the maximum extra flow that can be routed along this augmenting path? [3 marks]

(c) Draw the flow in G that results from routing the maximum possible additional flow through the augmenting path found in part (b). Identify a cut in G that proves (via the Min-cut/max-flow Theorem) that the flow is maximal. [3 marks]

(d) In general a network has many maximum flows. In this case, however, there is just one. Demonstrate, using an ad-hoc (though clear and concise) argument, that the flow constructed in part (c) is unique. (Certain edges must carry exactly the flows they do. Now fix the flows in these edges and argue that some further edges must carry the flows they do, and so on.) [2 marks]

(e) It can be shown that a maximum flow f is unique if and only if the associated residual network has no non-trivial simple cycles. (A cycle is *simple* if it has no repeated vertices; it is “non-trivial” if its length is greater than two.) Construct the residual network for the maximum flow found in part (c) and verify that it has no non-trivial simple cycles. (There is a simple observation that allows one to see almost immediately that there are no such cycles.) [3 marks]

Mary Cryan