Assignment 0 — 2012

Engi-6892. Theodore S. Norvell

Revised Sept 10. Due Sept 13, 2012, 10:30am

Parts (a) and (b) are an exercise in carefully explaining algorithms. Use pseudocode and clear English as appropriate. You can look in the text book for examples of pseudo code or at a document on my notation that I will post. You will by no means be marked on the efficiency of your algorithms, however you should make an effort to make your answer to part (b) more efficient than your answer to part (a).

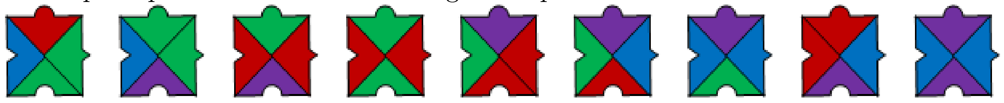
(a) Design an algorithm to solve the following problem. Describe the algorithm as clearly and carefully as you can.

Problem: Unrestricted Jigsaw
Input: A set of n^2 square pieces p_0, p_1, $\cdots$, p_{n^2-1}. Each piece is coloured with four colours: p_i.north, p_i.west, p_i.south, p_i.east.
Output: An arrangement of the pieces into an n by n grid so that:

- if two pieces are horizontally adjacent, the one to the left has a east colour equal to the west colour of the one on the right; and
- if two pieces are vertically adjacent, the one above has a south colour equal to the north colour of the one below;

If no such arrangement is possible, a message to that effect.

An example input would be the following set of pieces.[0]



(b) Design an algorithm to solve the following problem. Describe the algorithm as clearly and carefully as you can.

Problem: Restricted Jigsaw.
Input: A set of n^2 square pieces p_0, p_1, $\cdots$, p_{n^2-1}. Each piece is coloured with four colours: p_i.north, p_i.west, p_i.south, p_i.east.
Precondition: You may assume that each colour is used at most twice as an east or west colour and at most twice as a north or south colour.[1] Furthermore there are n colours that appear only once as a north colour and never as a south colour, n colours that appear only once as a south colour and never as a north colour; n colours that appear once as an east colour and

[0]For no extra credit you can try putting these pieces into a 3 by 3 grid. I'll post a solution.
[1]Using set notation we can express the restriction by saying that for each colour c, $|E_c \cup W_c| \leq 2$ and $|N_c \cup S_c|$,

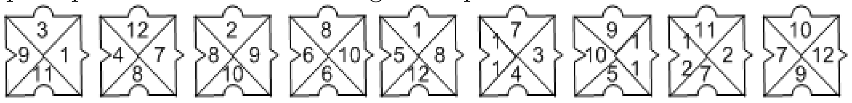
never as a west colour, and n colours that appear once as a west colour and never as an east colour.[2]

Output: Either an arrangement of the pieces into an n by n grid so that:

- if two pieces are horizontally adjacent, the one to the left has a east colour equal to the west colour of the one on the right; and
- if two pieces are vertically adjacent, the one above has a south colour equal to the north colour of the one below;

or, if no such arrangement is possible, a message to that effect.

An example input would be the following set of pieces.



(Note. Clearly your algorithm for part (a) will also work for part (b). Try to design an algorithm for part (b) that takes advantage of the restriction on the input.)

Briefly answer each of the following questions as best you can. Keep in mind that my point in asking you the following questions is to get you thinking about the kinds of issues that this course is about. At this point I have *no* expectation that you will come up with "correct" answers, although that would be a happy outcome. My expectation is that you will give these questions some careful thought and write something brief and cogent.

(c) We need quick algorithms. But how can we determine whether an algorithm is quick or not without implementing it and running it on all possible inputs?[3]

(d) Assuming each "basic operation"[4] takes 1ns, *estimate* how long each of your algorithms could take (in the worst case) for an input of 100 pieces. Is there something profoundly different about the two algorithms?

(e) The programs have to be correct. How can we determine whether an algorithm is correct for all possible inputs without implementing it and testing it on all possible inputs?[5]

(f) Do your algorithms embody patterns that we might be able to apply to other problems?

---

where

$$\begin{aligned} E_c &= \{i \in \{0, ..n^2\} \mid c = p_i.\text{east}\} \\ W_c &= \{i \in \{0, ..n^2\} \mid c = p_i.\text{west}\} \\ N_c &= \{i \in \{0, ..n^2\} \mid c = p_i.\text{north}\} \\ S_c &= \{i \in \{0, ..n^2\} \mid c = p_i.\text{south}\}. \end{aligned}$$

[2] Let's call these colours, respectively, north edge colours, south edge colours, etc. This condition makes it easy to identify edge and corner pieces. The north west corner (if there is one) will have a north edge colour as its north colour and a west edge colour as its west colour, and so on.

[3] And note that implementing these algorithms and running them on all possible inputs is hardly feasible, given that there are an infinite number of possible inputs.

[4] As part of your answer, you may want to clarify which operations you consider basic operations.

[5] Again this strategy is not possible in principle, owing to the infinite number of possible inputs. Furthermore, even if we give a finite selection of inputs, there is also the compounding problem of having to determine whether the algorithm has output the correct answer in each case.