Algorithms and Abstract Data Types (in scheme)

18 lectures (or there abouts)

24 hours of labs (or there abouts)

Newsgroup cs.ugrad.alg

Me: Dr. Patrick Prosser (pat@cs.strath.ac.uk)

No printed notes or handouts

What is an Algorithm?

ALGORISM the Arabic system of numeration

Abu Jafar Mohammed ibn Musa Al Khwarizmi (A.D. 825) first suggested a mechanical method for adding two numbers represented in the Hindu numeral system

Al Khwarizmi ... became ... algorithm

Algorithms are ancient

Euclid's algorithm (3d century BC)
an iterative method (algorithm) of finding the highest common factor (HCF) of two integers by ...

dividing the larger by the smaller then

dividing the smaller by the remainder of the first division then

dividing the remainder of the first division by the remainder of the second division

and so on, until the process terminates with remainder of zero

Properties of an algorithm

FINITENESS: algorithm terminates after a finite number of steps

DEFINITENESS: each step of algorithm is precisely defined, rigorously and unambiguously defined

GENERALITY (completeness): it can solve all problems of a type (for any input data) for which it is designed

EFFECTIVENESS: all operations used are basic and are capable of being performed mechanically (ie. no intuitive step)

INPUT/OUTPUT: has precise inputs or data, and the outputs (results) are generated in the intermediate as well as the final steps

Classification of Algorithms

DETERMINISTIC: control logic forces it one way or another

NON-DETERMINISTIC: capable of exploring large number of alternatives simultaneously, for example in searching for an optimal reult

RANDOM/STOCHASTIC: throws a dice to make a decision

DIRECT: number of iterations is known in advance, example might be array multiplication

INDIRECT: do not know in advance number of iterations example gcd

INFINITE: an algorithm that gives a better and better result (estimate) as we continue to run it example Newton-Raphson method

GREEDY: never retracts a decision (aka heuristic)

Greatest Common Divisor (aka HCF)

gcd of two integers, p and q

let r be the remainder of p divided by q

if r is 0 then q is the greatest common divisor

Otherwise set p to q, set q to r and repeat the process

INPUT p and q, where both are integers

OUTPUT will be gcd

1. r is remainder of p divided by q

2. IF r is 0 then gcd is q, STOP

3. IF r > 0
   (a) p becomes q
   (b) q becomes r

4. go to step 1

```
(define (gcd p q)
  (let ((r (remainder p q)))
    (cond ((= 0 r) q)
          (t (gcd q r)))))
```

Demo using p = 30 q = 45
p = 45 q = 30
p = 23 q = 5

Adlai Stevenson’s Recipe for carp:

1. **Take a 1 to 2 pound carp and allow it to swim in clear water for 24 hours.**

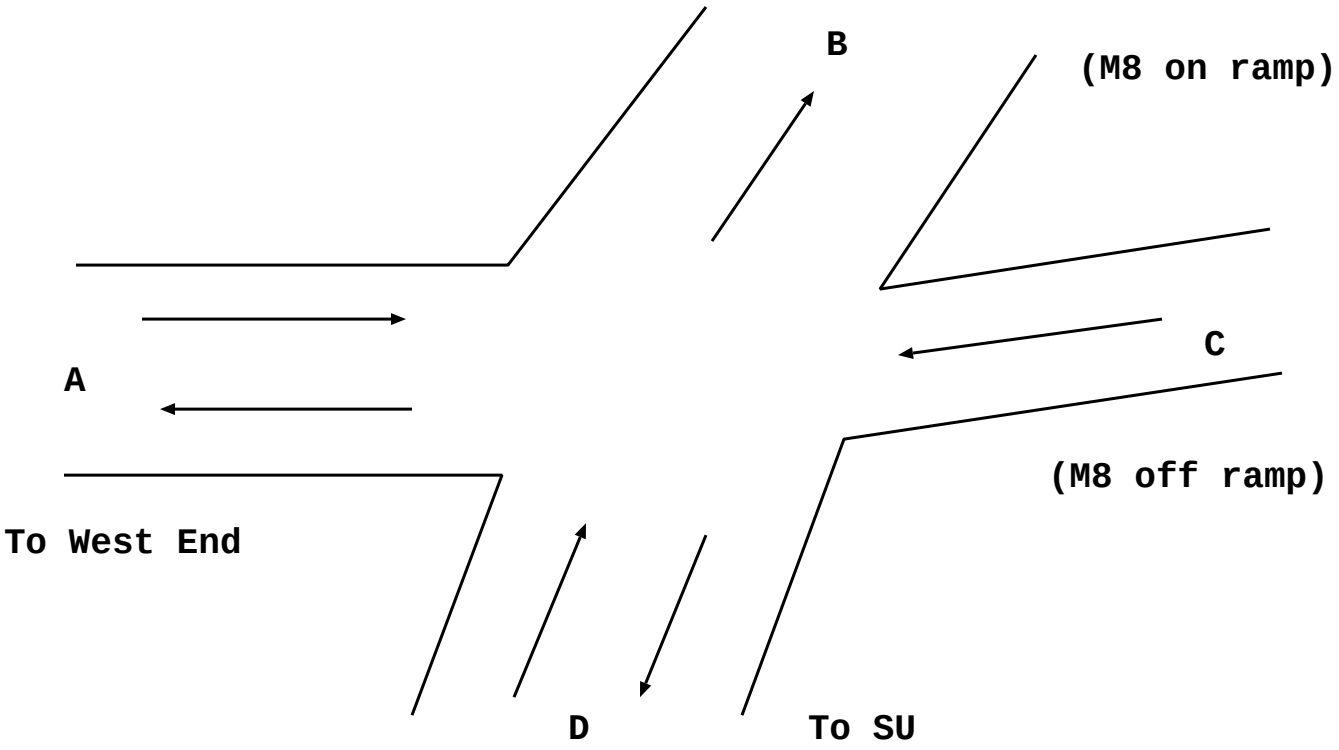
2. **Kill the carp, then scale and fillet**

3. **Rub fillets with butter and season with salt and pepper**

4. **Place on a board and bake in a moderate oven for 20 minutes**

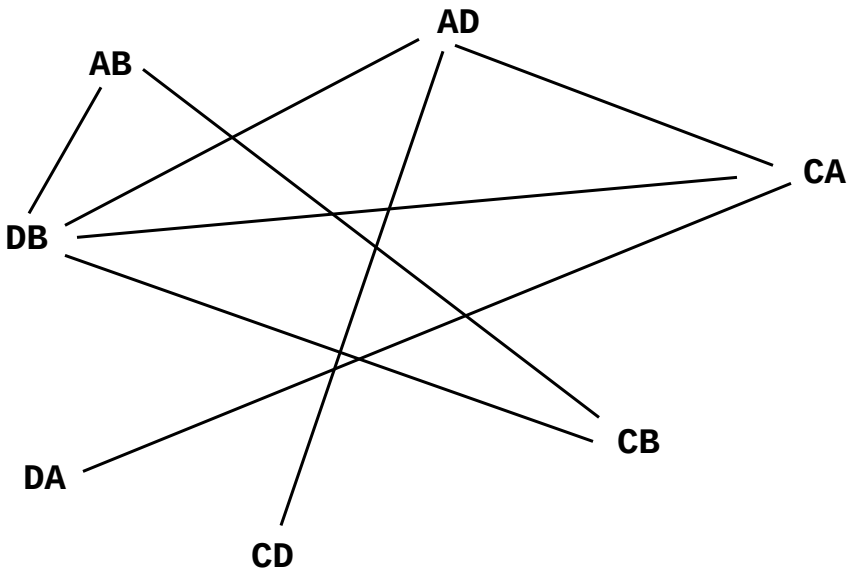
5. **Throw away the carp and eat the board**

A problem with traffic lights



Maximise traffic flow & do not allow lines of traffic to cross

Consider the set of legal turns AB, AD, CA, CB, CD, DA, DB

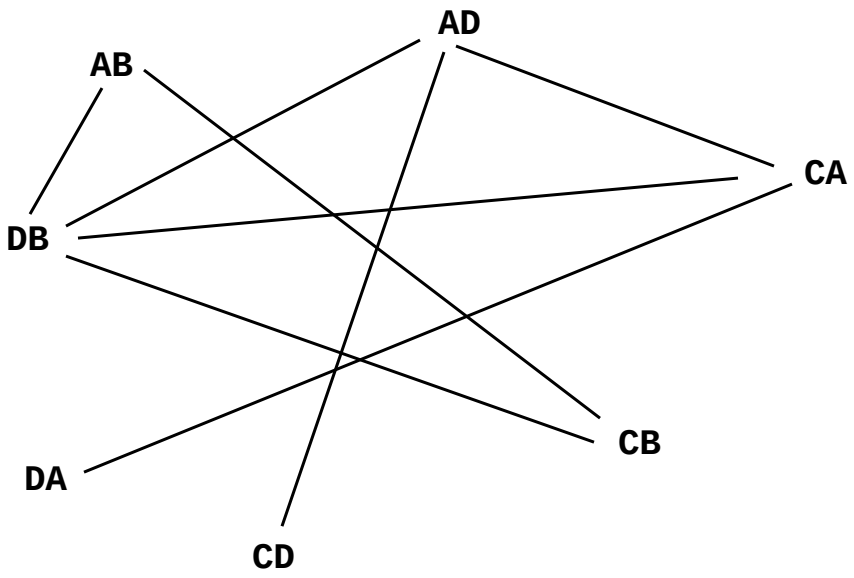


Represent as a graph, nodes are "turns", edges are 2 turns not allowed at same time

Now colour, using minimum colours

GRAPH COLOURING ... MANY ALGORITHMS but NP hard
ie. worst case is consider all possibilities

A problem with traffic lights



A "greedy" algorithm

1. select an uncloured node

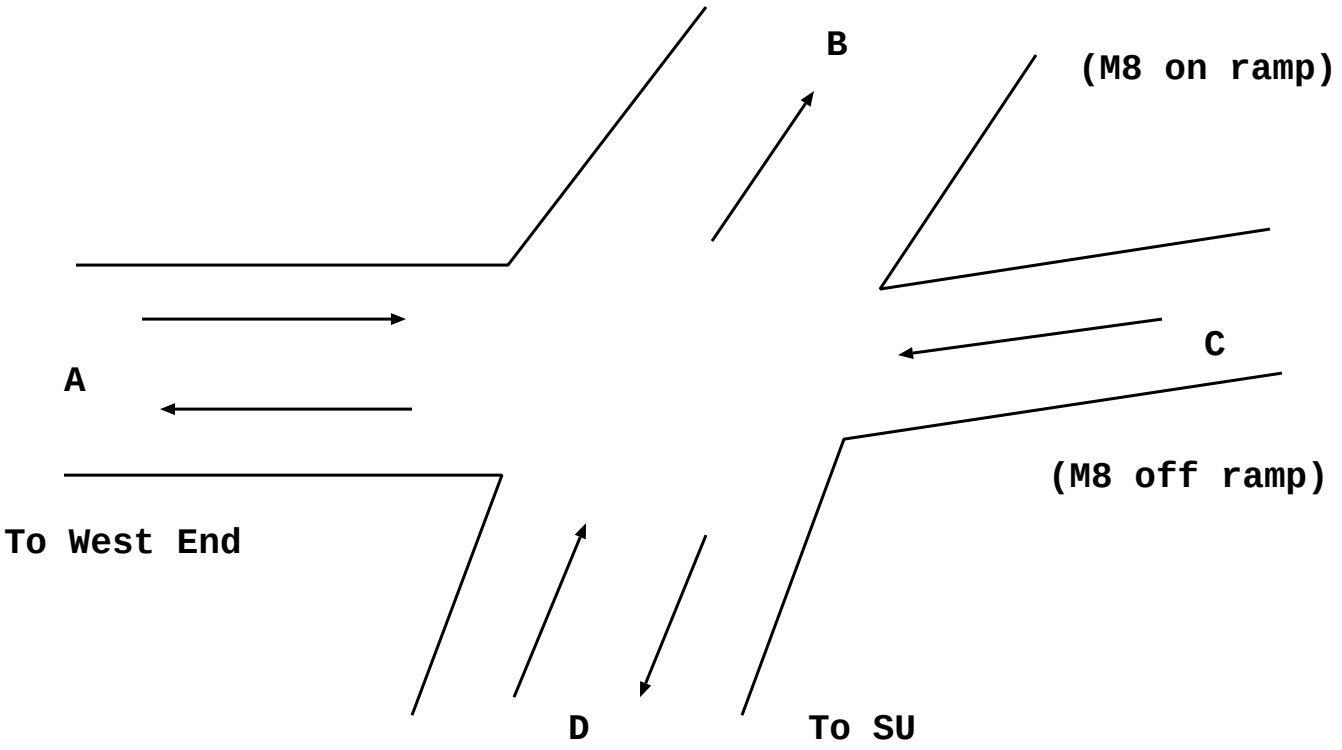
2. Colour it with the "lowest" possible colours
   ie. if it is adjacent to some coloured nodes
   then give it the lowest possible colours that is
   different from neighbours

3. If there are uncoloured nodes remaining, go to 1

AB = 1, AD = 1, DB = 2, CB = 3, CA = 3,
DA = 1 or 2, CD = 2 or 3

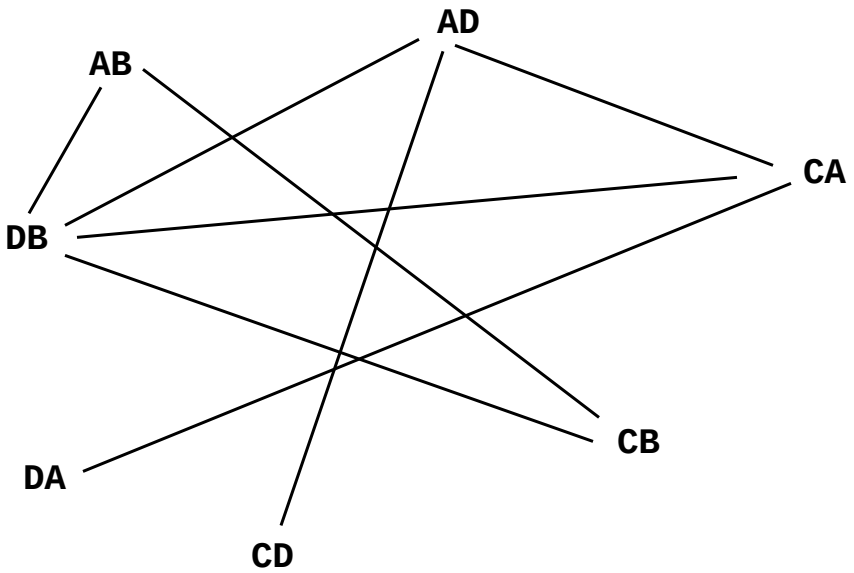
IS THIS OPTIMAL?

A problem with traffic lights



Sequence is as follows

1. [AB AD DA], 2. [CD DA DB], 3. [CA CB CD]



Morals from the lesson

1. Algorithms predate computers by thousands of years

2. A computer program aint necessarily an algorithm (and vice versa)

3. I can describe algorithms in many different ways

4. There are diffrent classes of algorithms

5. Sometimes you can look at a real world problem, recognise the "essence" of that problem, know something about how difficult it may be to solve that problem (ie. worst case run time), and pick an algorithm off the shelf.