What Types of Problems Do We Have to Solve

1 Problems in physical sciences

- All scientific disciplines have goals as the solutions of problems
- Physics seeks to explain the universe as it is
- Chemistry does the same
- Maths the same etc etc
- However in CS the process to the solution is as important as the solution, making it different.

CS problems

- The study of computers and what they can do
- The inherent powers and limitations of abstract computers
- The design, characteristics and use of real computers
- The applications of computers to solve problems
-
- How can we write programs that don't have bugs?
- How do we store things and find things?
- Is P equal to NP
- How do we make computers easily programmable

2 Essential notions

Key to problems in CS are:

- Repeated applications of solutions
- This makes CS unique within the physical sciences

2.1 More essential notions

- TSP
  - Finding the shortest tour of a given collection of nodes
  - Is the route the best route?
  - Lots of things to optimise for

Notions of

- Computation
- Resource
- Correctness

Additional Complication:

- CS is concerned with real world problems, but where the problem is being solved is not the real problem, but an abstraction of the problem

3 Abstraction

- Remove detail not needed for computation
- Just keep enough detail so that the conclusions reached are valid in the real world
- Model - A simulation of what would happen, Abstraction - Cover a whole set of problems, a generalisation of the problem

3.1 More on abstraction

- We might abstract the internet as a collection of nodes interconnected by channels, along which they can send and receive messages
- However, if you want to find different things about a real world thing you might not be able to with the current abstraction, so the abstraction would need to be changed to solve the problem

3.2 Abstracting the TSP

- Core info
  - Number of cities
  - Start node
  - Distance between cities
- As you go along you might learn more information, changing the situation, perhaps asked to visit extra node half way through
- Deal with not being able to get between two nodes
-

3.3 Decision Problems:Map Colouring

- Colour a map with as few colours as possible, without two of the same colour touching
- For any pair of regions either:
  - They share no part of a border
  - Exactly one point of a border, when they tough
  - They share a segment of some border when they tough

3.4 Decision Problems: Draughts

- From a given position, is there a way in which you can always win?
- Some games, like chess have a huge state space (the given combinations of possibilities)

3.5 Decision Problems

- Have a set of instances, from the set of instances, decide yes or no

A Decision problem consists of

- A set of instances I
- A subset $Y \subseteq I$ of yes instances
  A set of no instances

4 Abstracting map colouring

- Repeat an instance of the problem as a graph
  - V is a set of verticies, one for each region
  - E is a set of edges (a set of pairs of distinct vertices)
  - This has changed to a graph problem

4.1 Method

- Draw a graph between the centres of each area, draw a line between them if they share a border
- Remember that this method is destructive, it will be impossible to reconstruct the map from the graph
- You never need more than 4 colours
- Never overlapping edges - if there are overlapping edges it can't be from a map and may require more than 4 colours
- 2 colours is the most optimal scenario for any connected graph

4.1.1 Different approaches to hand colouring

- Work through the vertices in order and try to use the lowest colour (give them hierarchy)
- Can't use the same colour if connected by an edge
- This is a **greedy algorithm**
- This is non optimal as it gives a 5 colouring for some 4 colourable graphs

5 Sorting lists

- You are given a list and can order them in some way
- There can be repetitions and potentially available items might not appear in the list. Even though there are repetitions, the order of those elements matters
- The desired output is the sorted list

6 Finding routes

- Want to get from A to B, but following rules (for example, don't use motorways)
- Draw cities as vertices, roads as edges
- When it comes to the abstraction, your decision whether to include things excluded by the rules. Leaving them in makes it easier to solve multiple problems with the same map but different rules

7 Search Problems

- Numerous acceptable solutions, but only one should be returned or
- There is no acceptable solution, and this should be signalled

A search problem consists of

- A set of instances, i (for example any map you could think of)
- A set of solutions, j
- A binary search relation $R \subseteq I \times J$

In order to solve a search problem, for instance any $x \in J$ we need to find a solution $y \in J$ such that $(x, y) \in R$ if there is one, or answer no if there is no solution

Sometimes our search problem might be such that for every $x \in I$, there is exactly one $y \in J$ such that $(x, y) \in R$

8 Decision vs Search

- Corresponding to any decision, there is usually a search problem
- For example with map colouring decision problem, the equivalent search problem is to give the colouring

There is always a decision problem corresponding to a search problem

- The decision problem has the same set of instances I
- If you can find a solution in the search problem, then the answer is yes to the decision problem
- The yes instances are those instances $x \in J$ for which there exists some $y \in J$ for which $(x, y) \in R$

9 TSP

- This is a different type of problem as there are lots of solutions, but some are better than others

9.1 Finding tours

- Pick the shortest distance
-

10 Sports Day

- Job to create the schedule for a school sports day
- Every event takes 30 mins
- Each time slot 30 mins
- It is possible to have different events occurring at the same time, providing people are available for it (not doing 2 events at once)
- During the first slot, have as many events as possible
- Try to finish as early as possible
- Also with this, not all schedules are equal

10.1 Abstracting

- This can be abstracted as a graph $G = (V, E)$
  - Verticies of V consist of the different events
  - Edge $(u, v) \in E$ if there is an athlete that wants to participate in sports u and v
- Want to maximise the number of non connected vertices
  1. Choose an event at random
  2. Rule out the connected events
  3. Choose a random unconnected event
  4. Rule out the connected events to that new event
  5. Keep going until no more can be chosen
  6. Remove all from graph and do again for the next slot
- It is then simple to make the first slot have the most events
- However the total problem is difficult to solve

10.1.1 Subtleties

- The random choosing can lead to more or less optimal solutions

11 Generalised colouring

- What are the smallest number of colours required to colour a graph
- Lemma: A planar graph can be coloured in 4 colours
- May be asked to minimise/maximise the number of nodes of a certain colour

Proof of 4 colour theorem

- Start with assumption the graph needs at least 5 colours
- Take a set of graphs and prove that this assumption can't exist

12 Optimisation Problems

- Set of problems, instances and solutions
- Need information about the quality of a solution
- Looking for the best solution

An optimisation problem is defines:

- There is a set of instances I
- For every instance $x \in I$ there is a set of **feasible solutions f(x)**
- For every $x \in I$ and for every $y \in f(x)$ there is a value $v(x, y) \in N$ giving the **measure** of the feasible solution y for the instance x
- There is a goal which is either min or max

To solve an optimisation problem, given $x \in I$

- We need to find a feasible solution of the max/min measure

There may be

- No feasible solutions for an instance or
- For a maximisation problem, feasible solutions of increasingly large measure

13 Application of independent sets

- Given a communications channel where message bits get randomly flipped
- When encode the characters, make sure that every pair of characters has at least 2 bits different (2 bits need to change to change between characters)
- This allows you to know (if only 1 bit has been flipped) that an error has occurred
- These character encodings can be represented in a graph (all distance 2 apart)
- This cannot be used to correct errors, just to know that one has occurred

14 Finding independent sets

- Can use a greedy algorithm to find them
  - Find a vertex of smallest degree and put it in the independent set
  - Remove it and its neighbours from the graph
  - Iterate the remaining graph until there is nothing left, output I
- This is reasonably good, but does not guarantee an optimal solution

15 Finding tours in the TSP

- Use a greedy algorithm
- Proceeds as follows:
  1. Start at some city
  2. Go to the closest unvisited neighbour
  3. Output the resulting tour
- This doesn't necessarily result in an optimal tour
- Moving a node further away will result in it being missed out when visiting near it, resulting in a longer route

16 Recap

- Implementation is massively important
- Greedy algorithms are easy to implement and give pretty good answers

17 Practical applications

17.1 Algorithmic Graph Theory

- The study of the algorithmic and structural properties of graphs
- Look for better and better algorithms to solve the various problems in graphs
- Lots of ”hard” problems in graph theory - try to solve by finding ”easy” sub problems