**PLEASE REMEMBER TO NOT COPY EXACTLY AS THESE ARE HOW MY NOTES LOOK , DR JANSEN WILL READ THIS AND KNOW IF THEY ALL LOOK THE SAME**

Monday 11th April

**What computers can do.**

chess computer, robot vacuum cleaner, autonomous car, high frequency trading, medical diagnosis, drone control

**What computers can not do!**

With predictions about what computers can and cannot do. AI researchers have routinely overestimated whar computers are able to do.  
Famous misconceptions are threat computers would been a human in chess in 1967 and this was later done in 1997.

AI critics have routinely used vague and non-falsifiable statements about what computers can do e. g., doubting ‘true consciousness’ without precisely defining what this is and how it can be determined Consider computers making decisions with ethical component and ask can computers be responsible in an ethical sense? Take note increasing number of people see AI as existential thread for mankind.

**Deep learning.**

**Our Goal** is for us to restrict ourselves to computational problems determine if those can be solved by computers.

**A computational problem**has some finite input and is solved by some finite, correct output and they are solved by some correct output.

The **definition of a computational problem** is a problem defined by a set of finite inputs over a finite input alphabet and for each input a set of correct finite outputs over a finite output alphabet. Example input directed graph with edge weights, nodes A and B output shortest path from A to B.

An **optimisation problem** is a computational problem where the output is the value of an optimal solution.

In the image above, Dr J drew this on the board and explained the example input directed graph with edge weights, nodes A and B output shortest path from A to B.

When programming, we are able to use functions as if they are commands. This is the simple idea of function calls.

**Taking a closer look**  
  
If we assume we have an algorithm that if we give it a graph, it calculates the length of the shortest path. In this case, we have no idea how it its.  
  
The question is assuming we have such an algorithm, does it help us solve the optimisation problem.

**Solving a Opt. Problem with Help of the Decision Problem**

e efficient algorithm for solving decision variant Algorithm to solve the computational problem

Compute S := sum of all edge weights.

If solution with length ≤ S exists then

{ Set l := 0 and u := S. while l < u { Set m := dl + (u − l)/2e.

If solution with length ≤ m exists then set l := m else set u := m − 1.

}

Output l

}

else

Output ‘no path from A to B’

It can be said that we have reduced the computational variant to the optimisation variant.

**Reductions**

Fact Solving some problem P using an algorithm for some other problem Q is called ‘reducing P to Q’ and implies in some sense P is not much harder than Q Special case solving decision problem P using an algorithm for some other decision problem Q is called ‘reducing P to Q’ and written as P ≤ Q and implies in some sense P is not much harder than Q

The observations are as follows

Reductions can help to find more problems that computers cannot solve If we know that P cannot be solved by computer and if we know that P ≤ Q then Q cannot be solved by computer because we can solve P with the help of Q and the ‘≤’-algorithm

The most important takeaway is that there are things that computers can do and there are things they cannot do. the next section looks at what is a computer.  
  
**What is a computer?**

huge number of memory cells (to avoid problems with lack of memory, let’s say infinitely many) the CPU contains little memory.  TheCPU operates in steps  and in each step, the CPU can read from one cell of memory, write to one cell of memory, can change its state 276 Introduction and Motivation Computational Problems Turi

**The Turing Machine**

(Exact definition below as copied from slide)

A Turing machine has a finite set of possible states Q, an initial state q0 ∈ Q, a finite memory alphabet Γ that contains the blank character , a finite input alphabet Σ that does not contain , an infinite memory that is linearly organised, and a current position in the memory. Initially the input is in the memory, the current position is the first position of the input and all unused memory cells contain .

That’s pretty much it, next week we will summarise!