"More Advanced" Topics

- Can computers solve all problems? NO!

  - Example: Can you write a program that will check if a given program will stop on input "0".

    * No such program exists!

    * This is an example that something is not computable.

  - Example: King Aruthur's Problem

    * There are problems that, although computable, require too much time to compute. They are practically non-computable.

- King Arthur's Problem
  - 150 knights.
  - Some pairs of knights hate each other (so they can't sit side by side).
  - King Arthur wants to have a round-table meeting with all his 150 knights. (Actually 149, 1 was absent in history).
  - Problem: How does the King arrange the knights around the table so that no pair of knights who "hate each other" sit side by side?

*trivial solution: try all possible permutations of 150 knights. But there are 150! of them.

* King does not have time or patience to go through 150! permutations.

* In fact: Neither do all modern computers!

Let's see: assume King Arthur had a Cray Computer computing 10^{10} steps/second:

- In 1 minute: $60 * 10^{10}$ steps
  In 1 hour: $36 * 10^{12}$ steps
  In 1 day: $\approx 10^{15}$ steps
  In 1 year: $\leq 5 * 10^{17}$ steps
  In 1000 years: $< 5 * 10^{20}$ steps.

- However: $150! > 10^{100}$ (calculate this by Stirling formula) $\Rightarrow$ The King would still be waiting!

Let's formalize the King Arthur's Problem:

- Each knight is a vertex.
- If two knights do not "hate" each other, add an edge between the two.

So we got a graph G.

Now the King Arthur's Problem becomes: Find a cycle in G such that the cycle passes each node precisely once and include all nodes. This problem also had a different name: the **Hamiltonian cycle problem**.

- Important Remark: What we have just done is a "**Reduction**": we have reduced the King Arthur's problem to "Hamiltonian Cycle Problem".

- The Hamiltonian problem is a so-called NP-Complete problem. And it is not likely to have a polynomial time solution.

- There are thousands of problems that are equivalent to the above two problems.

  - They are all NP-Complete problems.

  - If any one of them does not have a polynomial time solution (hence can't b e solved practically), then all do not!

- What is a NP-Complete problem: It can be solved by guessing, in polynomial time. (Nondeterministic Polynomial time), and it is "hardest" such problem.

- King Arthur's Problem can be solved non-deterministically.

  Remember: There is a magician, Merlin, who had an oracle. King Arthur would ask Merlin for help. Merlin just asked his oracle, the oracle (guessed) comes up with a correct answer. Then it only takes Arthur $O(n)$ time to check the correctness of such answer. This is a nondeterministic alg. and it is $O(n)$ time only.

Some other examples of NP-complete problem:

- Satisfiability problem: Given $f = (x_1 + \bar{x_2} + x_3)(x_1 + x_5 + \bar{x_8})(x_1 + x_2 + x_{10})(x_3 + \bar{x_7} + x_8)$. Does there exist an 0,1 assignment to $x_1 \ldots x_n$ such that $f(x_1, \ldots, x_n) = 1$?

- Superstring Problem: Given strings $a_1, \ldots, a_n$, $a_i \in \{0, 1\}^*$, find a *shortest common superstring* s s.t. for all i, a_i is substring of s.

Approximation. NP-complete problems take too muh time to solve *precisely*, we an try to solve them *approximately*.

Example. To solve the superstring problem, biologists are using the following “Greedy” algorithm.

```
Repeat
   merge 2 strings with maximum overlap
until 1 string left.
```

If the optimal length is n, the above guarantees $3n$.

Example: ba^n, a^{n+1}, $a^n b$, so this greedy algorithm is no better than $2n$.