The Role of Algorithms in Computing

Algorithms

- **Definition:**
  - Any well-defined computational procedure that takes some value, or set of values, as *input* and produces some value or set of values, as *output*.
  - *A sequence of computational steps* that transform the input into the output.
  - A tool for solving a well-specified *computational problem*.
- **Statement of the problem**
  - Specification of the desired input/output relationship in general terms.

Algorithms

- ***Sorting problem***
  - ***Input*: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.**
    - $\langle 31, 41, 59, 26, 41, 58 \rangle$
    - **An *instance* of the sorting problem.**
  - ***Output*: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$.**
    - $\langle 26, 31, 41, 41, 58, 59 \rangle$
- **An *instance of a problem***
- ***Correct* algorithm**
  - **For every input instance, the algorithm halts with the correct output.**

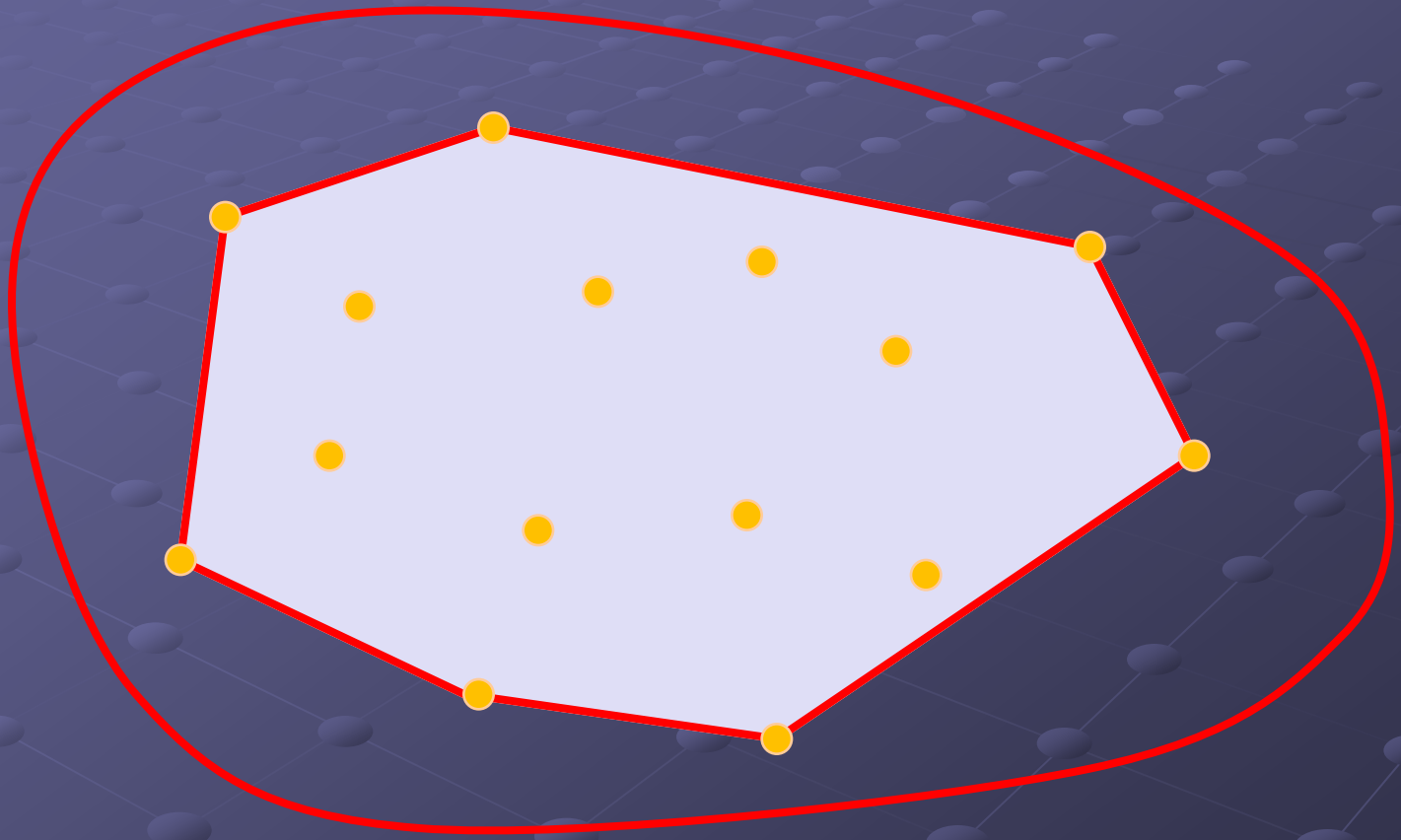
Practical Applications of Algorithms

- Human Genome Project
  - Identifying all the 100,000 genes in human DNA.
  - Determining the sequences of the 3 billion chemical base pairs that make up human DNA.
- Information retrieval on Internet
- Electronic commerce
  - Public-key cryptography
  - Digital signature
- Resources allocation
  - Linear programming

Typical Problems

- The shortest path problem
- Longest common subsequence of two ordered sequences of symbols
- Topological sorting problem
  - List the parts of a mechanical design in order so that each part appears before any part that uses it.
- Convex hull problem

Convex Hull

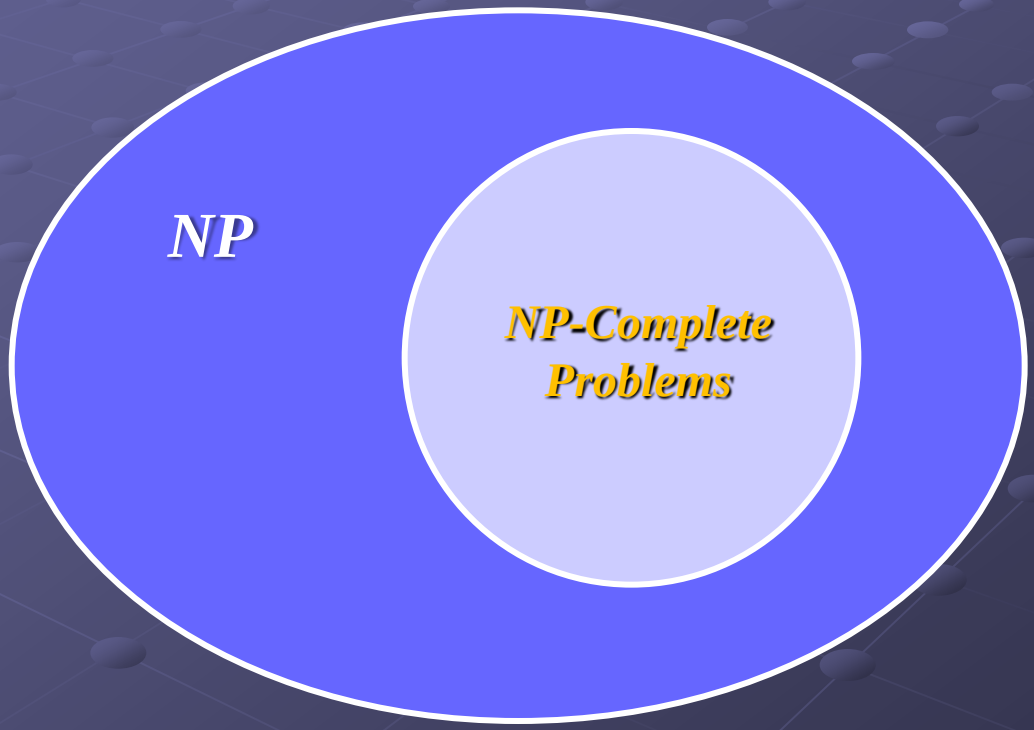


Hard Problems

- Usual measure of efficiency – ***Speed***.
- ***NP***[註1]***-complete*** problems
  - No efficient algorithm has ever been found.
  - It is unknown whether or not efficient algorithms exist for *NP*-complete problems.
  - If an efficient algorithm exists for any one of them, then efficient algorithms exist for all of them.
- If we can show a problem is *NP*-complete
  - Develop an efficient algorithm which gives a good solution instead of the best solution.

註1. ***NP*** (***Nondeterministic Polynomial***)

Hard Problems



Algorithms As A Technology

- Suppose
  - Computers were infinitely fast
  - Computer memory was free
- Would you have any reason to study algorithm?
- Yes! Why?
  - Within the bounds of good software engineering practice.
    - Your implementation should be well designed and documented.
  - Use whichever method was the easiest to implement.
- ***But computers are not infinitely fast, memory is not free.***

Efficiency

- Two computers A and B
  - A is 1000 times faster than B.
- Insertion sort running on A – $c_1 n^2$

$$\frac{2 \cdot (10^7)^2 \ instructions}{10^{10} instructions/second} = 20{,}000 \ seconds \ (more \ than \ 5.5 \ hours)$$

- Merge sort running on B – $c_2 \ n \ lg \ n$

$$\frac{50 \cdot 10^7 \ lg \ 10^7 \ instructions}{10^7 \ instructions/second} \approx 1163 \ seconds \ (less \ than \ 20 \ minutes)$$

- No matter how much smaller c_1 is than c_2
  - There will always be a crossover point beyond which merge sort is faster

Algorithms and Other Technologies

- Algorithms v.s. other advanced technologies
  - Advanced computer architectures and fabrication technologies
  - Easy-to-use, intuitive, graphical user interface
  - Object-oriented systems
  - Integrated Web technologies
  - Fast networking, both wired and wireless
- Algorithms are at the core of most technologies used in contemporary computers.
- At ***larger problem sizes*** that the differences in efficiency between algorithms become particularly prominent.