Chapter 2

Basics of Algorithm Analysis
2.1 Computational Tractability

As soon as an Analytic Engine exists, it will necessarily guide the future course of the science. Whenever any result is sought by its aid, the question will arise - By what course of calculation can these results be arrived at by the machine in the shortest time? - Charles Babbage (1791—1871)
Computer Science Approach to problem solving

• If my boss / supervisor / teacher formulates a problem to be solved urgently, can I write a program to efficiently solve this problem ??
Computer Science Approach to problem solving

• Are there some problems that cannot be solved at all? and, are there problems that cannot be solved efficiently??
A few Computability Classes

ALL PROBLEMS

COMPUTABLE
Alan Turing

In 1934, he formalized the notion of decidability of a language by a computer.
In 1936, he proved that certain languages cannot be decided by any algorithm whatsoever...
Emil Post

In 1946, he gave a very natural example of an undecidable language...
(PCP) Post Correspondence Problem

An instance of PCP with 6 tiles.

A solution to PCP

```
  aaa   a   bbb   aa   bb   b
    bb   bb   a    a     b
```
Given $n$ tiles, $u_1/v_1 \ldots u_n/v_n$
where each $u_i$ or $v_i$ is a sequence of letters.

Is there a $k$ and a sequence $<i_1,i_2,i_3,\ldots,i_k>$
(with each $1 \leq i_j \leq n$) such that

$$u_{i_1} | u_{i_2} | u_{i_3} | \ldots | u_{i_k} = v_{i_1} | v_{i_2} | v_{i_3} | \ldots | v_{i_k}$$
A Solution to Post Correspondence Problem

A solution is of this form:

\[
\begin{array}{cccccc}
u_1 & u_2 & u_3 & \ldots & u_n \\
v_1 & v_2 & v_3 & \ldots & v_n \\
\end{array}
\]

◊ A solution is of this form:

\[
\begin{array}{cccccc}
u_{i1} & u_{i2} & u_{i3} & u_{i4} & u_{i5} & \ldots & u_{ik} \\
v_{i1} & v_{i2} & v_{i3} & v_{i4} & v_{i5} & \ldots & v_{ik} \\
\end{array}
\]

with the top and bottom strings identical when we concatenate all the substrings.
Post Correspondence Problem

Theorem:
The Post Correspondence Problem cannot be decided by any algorithm (or computer program). In particular, no algorithm can identify in a finite amount of time some instances that have a No outcome. However, if a solution exists, we can always find it.
Post Correspondence Problem

**Proof:**

Reduction technique – if PCP was **decidable** then another undecidable problem would be **decidable**.
The Halting Problem

- Notice that an algorithm is a piece of text.
- An algorithm can receive text as input.
- An algorithm can receive the text description of an algorithm as input.

**The Halting Problem:**
Given two texts A, B, consider A as an algorithm and B as an input. Will algorithm A halt (as opposed to loop forever) on input B?
The Halting Problem and PCP

- Any algorithm to decide PCP can be converted to an algorithm to decide the Halting Problem.

Conclusion: PCP cannot be decided either.
A few Computability/Complexity Classes

ALL PROBLEMS

COMPUTABLE

P-Space

NP

P
Chapter 1

Introduction: Some Representative Problems
1.2 Five Representative Problems
**Interval Scheduling**

**Input.** Set of jobs with start times and finish times.

**Goal.** Find maximum cardinality subset of mutually compatible jobs.

jobs don’t overlap
Weighted Interval Scheduling

**Input.** Set of jobs with start times, finish times, and weights.

**Goal.** Find maximum weight subset of mutually compatible jobs.
Bipartite Matching

**Input.** Bipartite graph.

**Goal.** Find maximum cardinality matching.
Independent Set

**Input.** Graph.

**Goal.** Find **maximum cardinality independent set.**

subset of nodes such that no two joined by an edge
**Competitive Facility Location**

**Input.** Graph with weight on each node.

**Game.** Two competing players alternate in selecting nodes. Not allowed to select a node if any of its neighbours have been selected.

**Goal.** Select a maximum weight subset of nodes.

Second player can guarantee 20, but not 25.
Five Representative Problems

Variations on a theme: independent set.

Interval scheduling: $n \log n$ greedy algorithm.

Weighted interval scheduling: $n \log n$ dynamic programming algorithm.

Bipartite matching: $n^k$ max-flow based algorithm.

Independent set: $NP$-complete.

Competitive facility location: $PSPACE$-complete.
A few low order Complexity Classes

P = NP ?
NP = P-Space ?
Chapter 1

Introduction:
Some Representative Problems
Computer Science Approach to problem solving

- If my boss / supervisor / teacher formulates a problem to be solved urgently, can I write a program to efficiently solve this problem???
Brute force. For many non-trivial problems, there is a natural brute force search algorithm that checks every possible solution.
- Typically takes $2^N$ time or worse for inputs of size $N$.
- Unacceptable in practice.

Desirable scaling property. When the input size doubles, the algorithm should only slow down by some constant factor $C$.

There exists constants $a > 0$ and $d > 0$ such that on every input of size $N$, its running time is bounded by $a \cdot N^d$ steps.

Def. An algorithm is poly-time if the above scaling property holds.

Property: poly-time is invariant over all (non-quantum) computer models.
Chapter 2

Basics of Algorithm Analysis