Universitas Gadjah Mada Yogyakarta
Fakultas Matematika dan Ilmu Pengetahuan Alam
Program Studi Ilmu Komputer

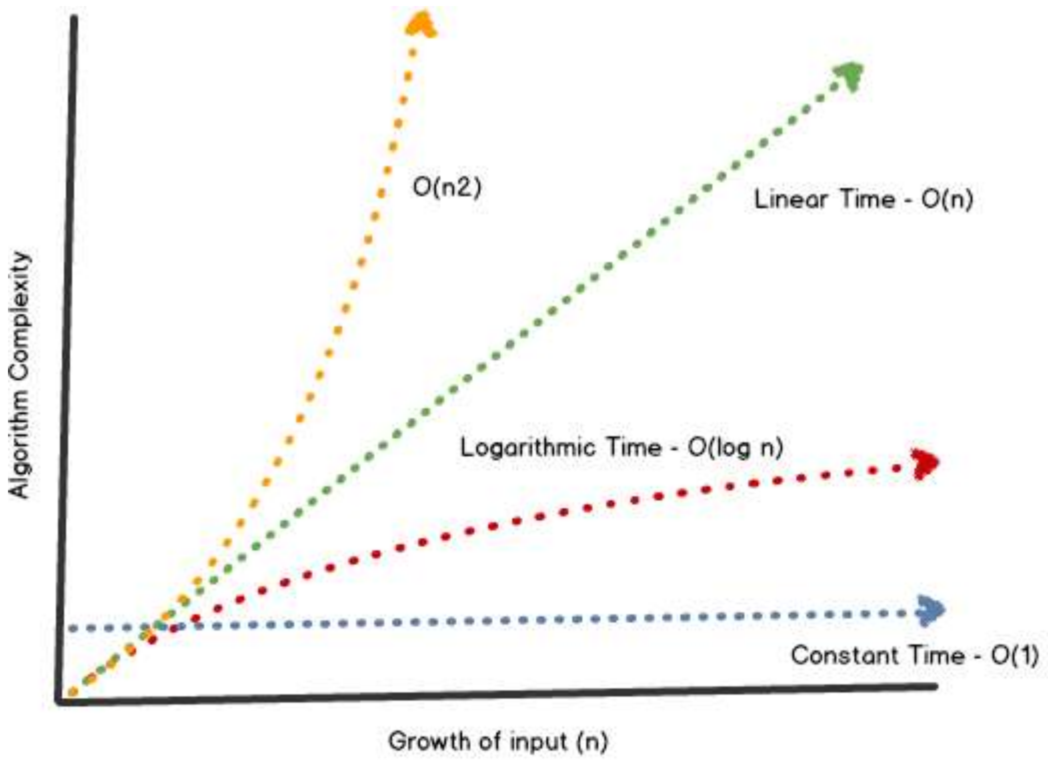
Teori

P, NP, NP-Hard dan NP-Complete

Tenia Wahyuningrum
16/405316/SPA/00573

Introducing…

The time requirement of the algorithm varies



Good!

It's **Polynomial** solution!

What is P?

- **P**olynomial **T**ime
- The worst-case time complexity is **limited** by the polynomial function of its input size.
- The class of problems where the solution can ***discovered*** **“quickly”**
  - In time polynomial in the size of the input

What is P? (cont.)

Example

Sorting → $T(n) = O(n^2)$, $T(n) = O(n \log n)$
Searching → $T(n) = O(n)$, $T(n) = lO(\log n)$
Matrix multiplication → $T(n) = O(n^3)$, $T(n) = O(n^{2.83})$

P problems

- P Problems are the set of all decision issues **that can be solved** by algorithms with polynomial time requirements.
- More specifically, they are problems **that can be solved** in time $O(n^k)$ for some constant k, where n is the size of the input to the problem.
- The key is that n is the **size of input.**

What is the complexity of **primality testing**?

```
public static boolean isPrime(int n){
    boolean answer = (n>1)? true: false;

    for(int i = 2; i*i <= n; ++i)
    {
        System.out.printf("%d\n", i);
        if(n%i == 0)
        {
            answer = false;
            break;
        }
    }
    return answer;
}
```

This loops until the square root of n
So this should be $O(\sqrt{n})$

But what is the input size?
How many bits does it take to represent the number n?
log(n) = k

What is $\sqrt{n}$

$$\sqrt{n} = \sqrt{2^{log(n)}} = (2^k)^{0.5}$$

Naïve primality testing is exponential!!

Intractable and tractable

- An issue is said to be **intractable** if it is not possible to be solved by a polynomial-time algorithm.

- Conversely, the problem is said to be **tractable** if a polynomial algorithm can solve it.

Is this **tractable?**

Multiplication of the chain matrix

$A_1 \times A_2 \times \ldots \times A_n$

By means of a brute force attack and divide conquer

→T(n), Non Polynom

But, by using dynamic programming

→ $T(n) = O(n^3)$, Polynom

*Multiplication of the chain matrix is tractable

Is there **an intractable** issue?

It’s a **halting problem**

by Alan Turing, 1963

Example

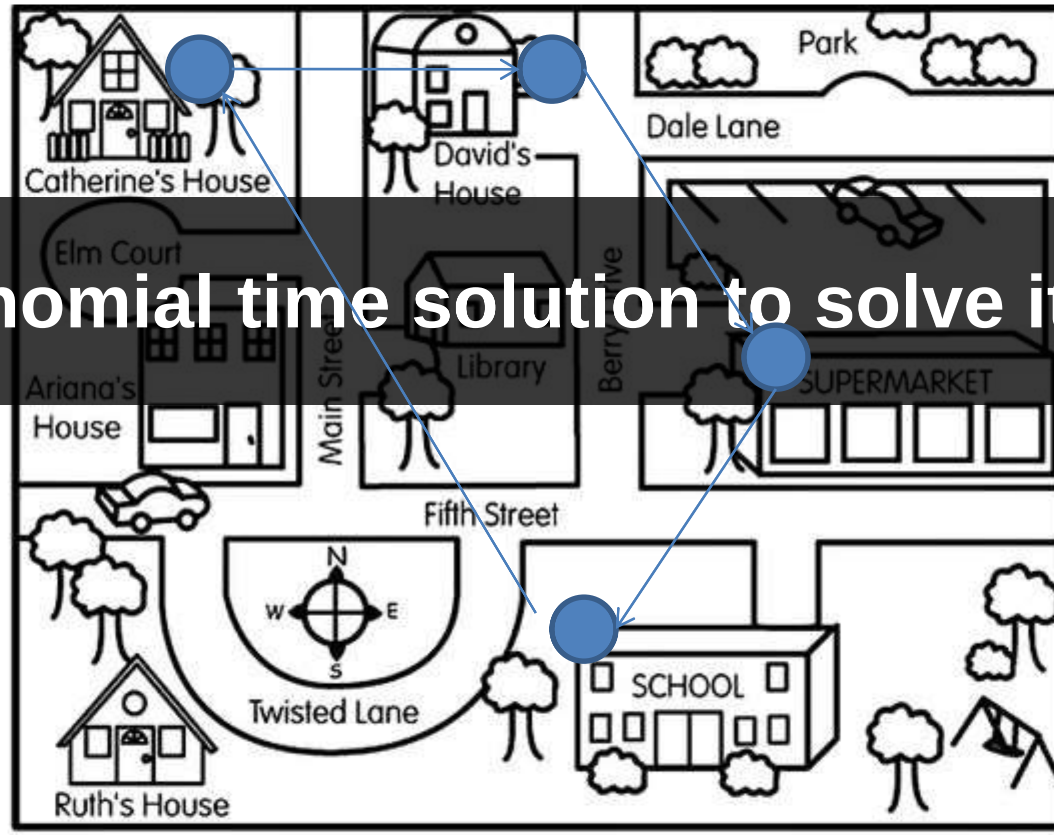
- Take x integers less than ten multiply hold x by itself until the result is more than 100
  - If x= 9, then the program **will stop in step 2**
  - If x=4, then the program **will stop in step 3**
  - But, if x=1, then the program **will never stop**

* Halting problem

How about

Travelling Salesman Problem?

There is no polynomial time solution to solve it!



Time complexity = O(*n*!).

What is NP?

- **N**ondeterministic **P**olynomial time
- The class of problems where the solution can ***verified*** "**quickly**"
  - In time polynomial in the size of the input

What is NP? (cont.)

Example

- TSP,
- integer knapsack problem,
- graph coloring,
- Hamilton Circuit,
- partition problem,
- integer linear programming

NP theory

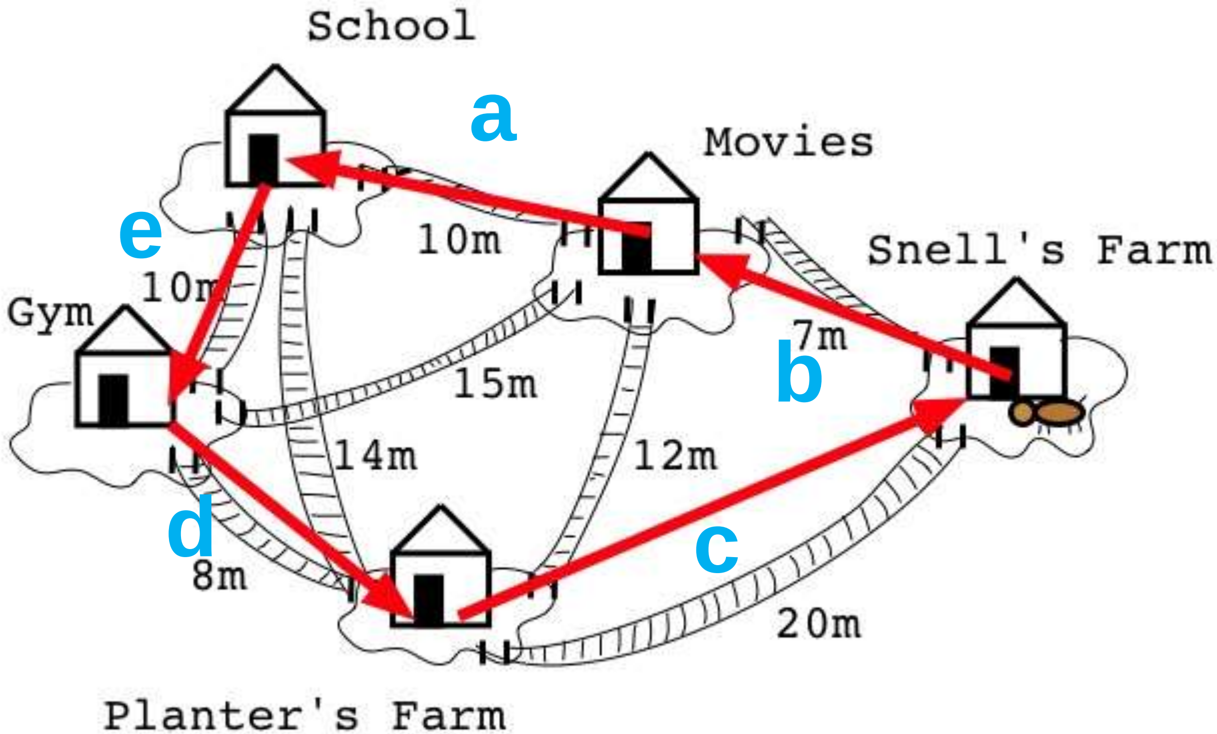
- In discussing the theory of NP, we only limit the **decision problem**.

- Decision issue is a problem whose solution is only "**yes**" or "**no**" answer.

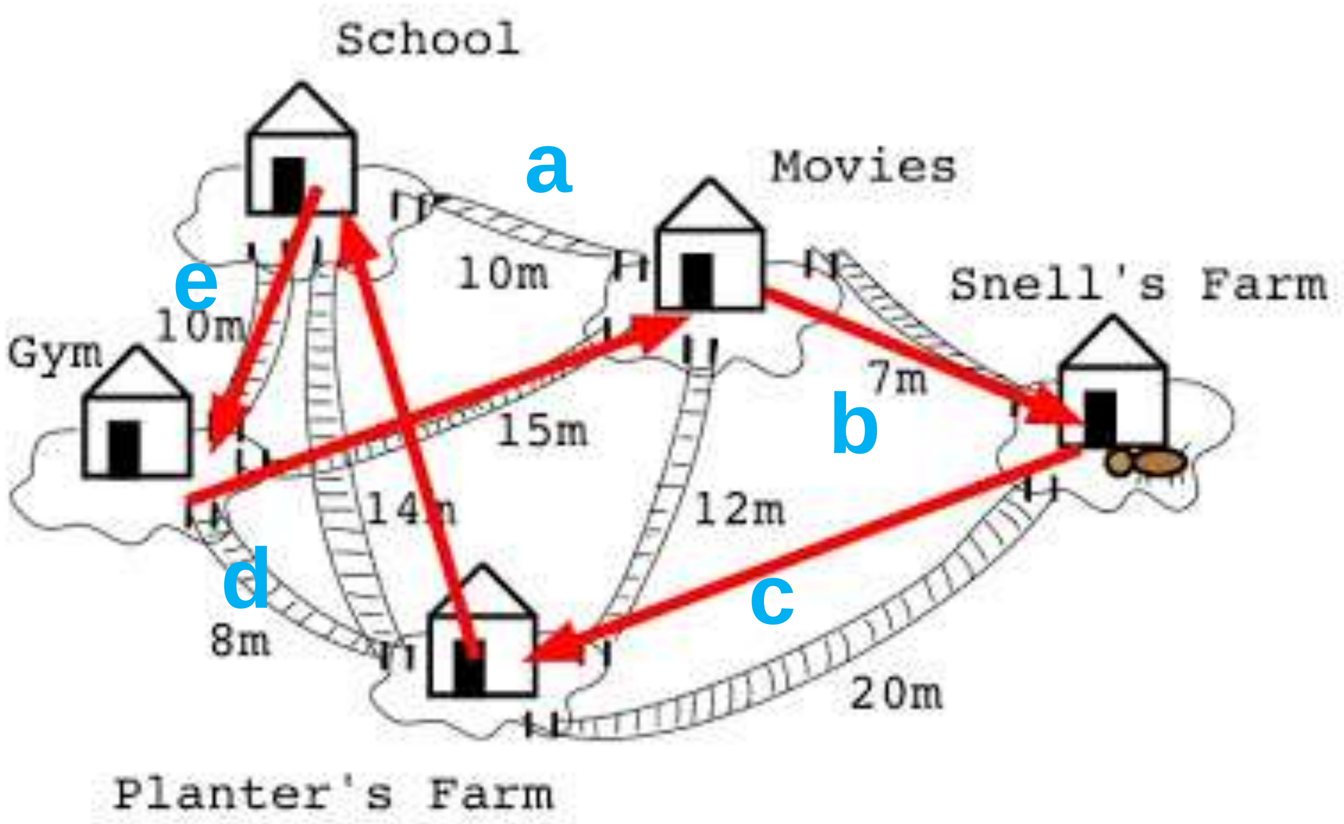
Example:

- Given an integer x. Determine whether **the element x is in the table**?
- Given an integer x. Determine whether x **is prime number**?

NP problems


School
Movies
Snell's Farm
Gym
Planter's Farm
a
b
c
d
e
10m
10m
7m
15m
14m
12m
8m
20m

NP problems



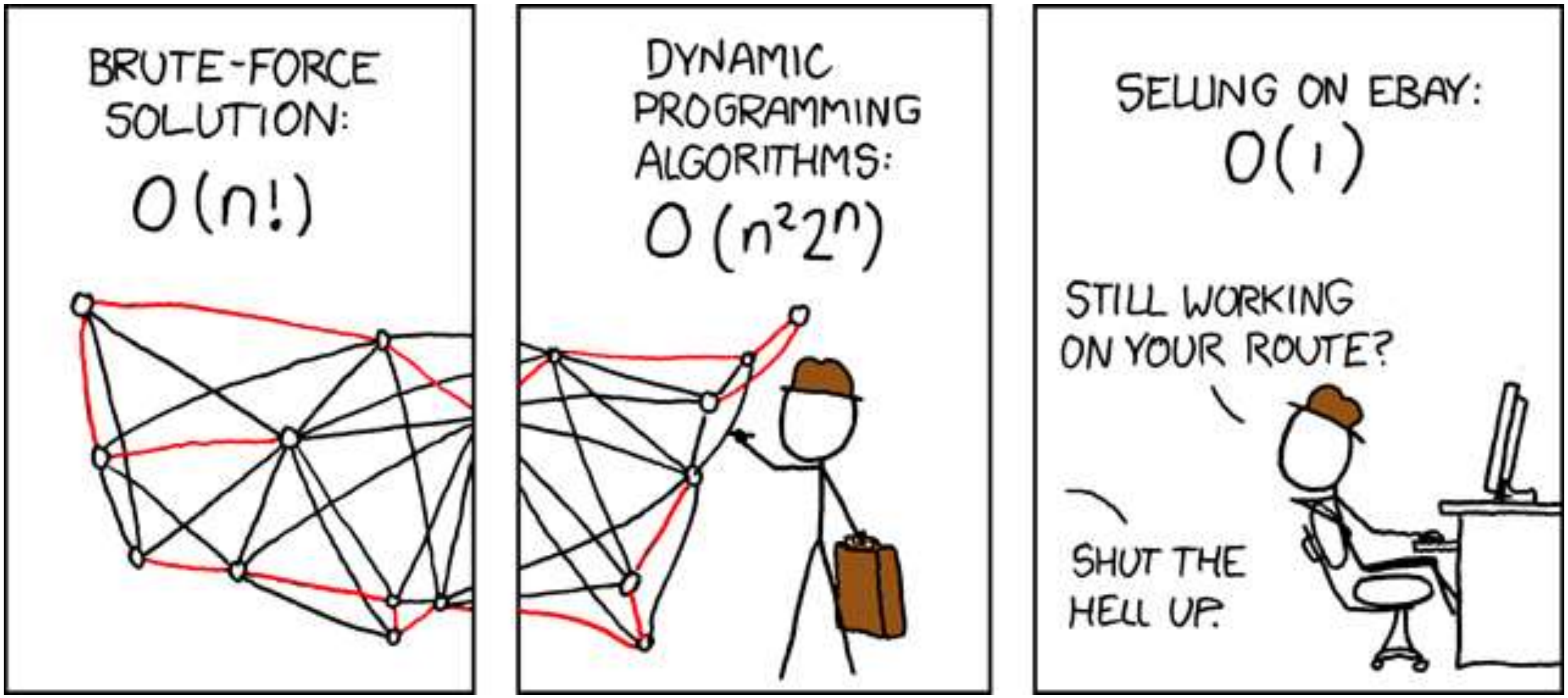
a+ b+c+d+e = minimum weight
a+ b+c+d+e < x

TSOP = minimum weight
TSDP = total weight<x

- **Traveling Salesman Optimization Problem (TSOP)** is a matter of determining a tour with a minimum total of minimum weights. (*TSP is commonly known).
- **Traveling Salesman Decision Problem (TSDP)** is a matter of determining whether there is a tour with the total weight of the sides not exceeding the value x.

- Example: if the problem Traveling Salesman Optimization Problem (TSOP) minimum tour is 20,

- Then the answer to the problem of Traveling Salesman Decision Problem (TSDP) **is "yes" if x >= 20**, and **"no" if x <20**.

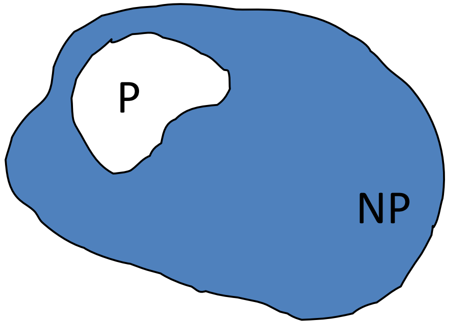

BRUTE-FORCE SOLUTION:
O(n!)
DYNAMIC PROGRAMMING ALGORITHMS:
O(n²2ⁿ)
SELLING ON EBAY:
O(1)
STILL WORKING ON YOUR ROUTE?
SHUT THE HELL UP.

*jokes

NP hard problem

- Class of decision **problems which are at least as hard** as the hardest problems in NP.
- Problems that are NP-hard **do not have to be elements of NP**; indeed, they may not even be decidable.

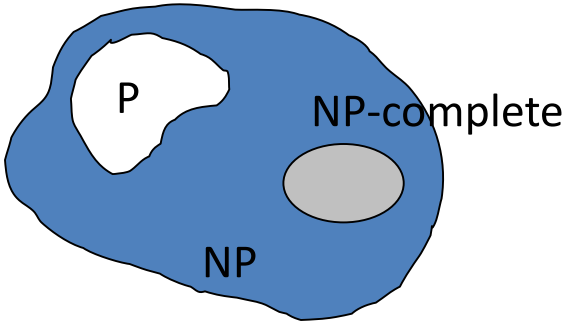
Is **P = NP?**



- Any problem in P is also in NP:
- P $\subseteq$ NP
- The big (and **open question**) is whether NP $\subseteq$ P or P = NP
  - i.e., if it is always easy to check a solution, should it also be easy to find a solution?
- Most computer scientists believe that this is false but we do not have a proof …

NP-Completeness **(informally)**

- **NP-complete** problems are

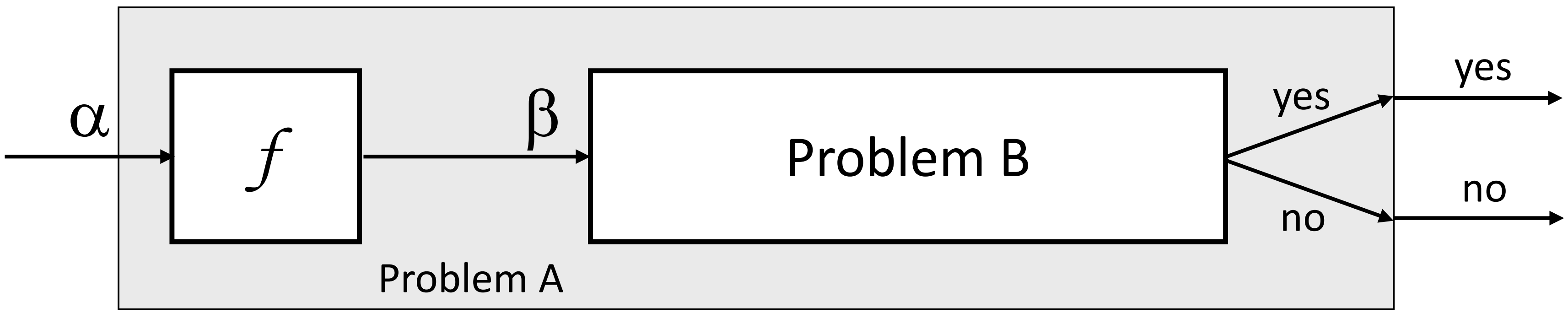


- defined as the hardest problems in NP
- Most practical problems turn out to be either P or NP-complete.
- Study NP-complete problems …

Reductions

- Reduction is a way of saying that one problem is **"easier"** than another.
- We say that problem A is easier than problem B, (i.e., we write **"A ≤ B"**)

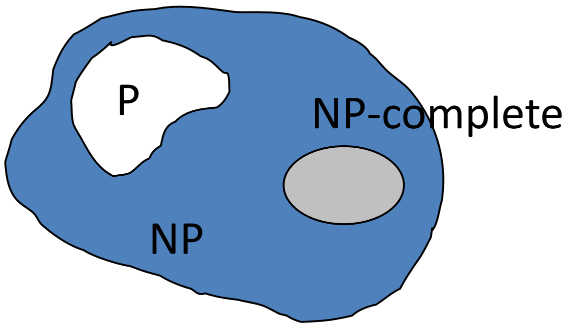
  if we can solve A using the algorithm that solves B.
- **Idea:** transform the inputs of A to inputs of B



Polynomial Reductions

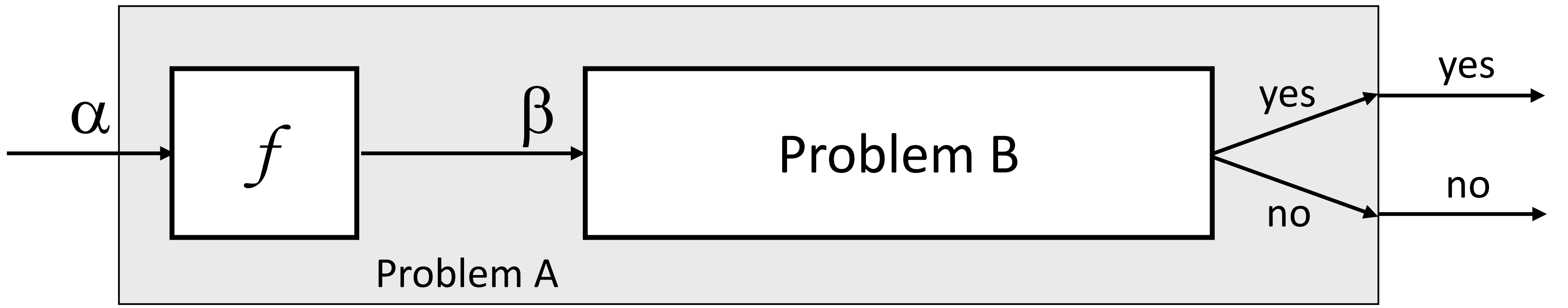
- Given two problems A, B, we say that A is polynomially **reducible** to B ($A \leq_p B$) if:
  - There exists a function f that converts the input of A to inputs of B in polynomial time
  - $A(i) = YES \Leftrightarrow B(f(i)) = YES$

NP-Completeness (formally)



- A problem B is **NP-complete** if:
- (1) B $\in$ **NP**
- (2) A $\leq_p$ B for all A $\in$ **NP**
- If B satisfies only property (2) we say that B is **NP-hard**
- No polynomial time algorithm has been discovered for an **NP-Complete** problem
- No one has ever proven that no polynomial time algorithm can exist for any **NP-Complete** problem

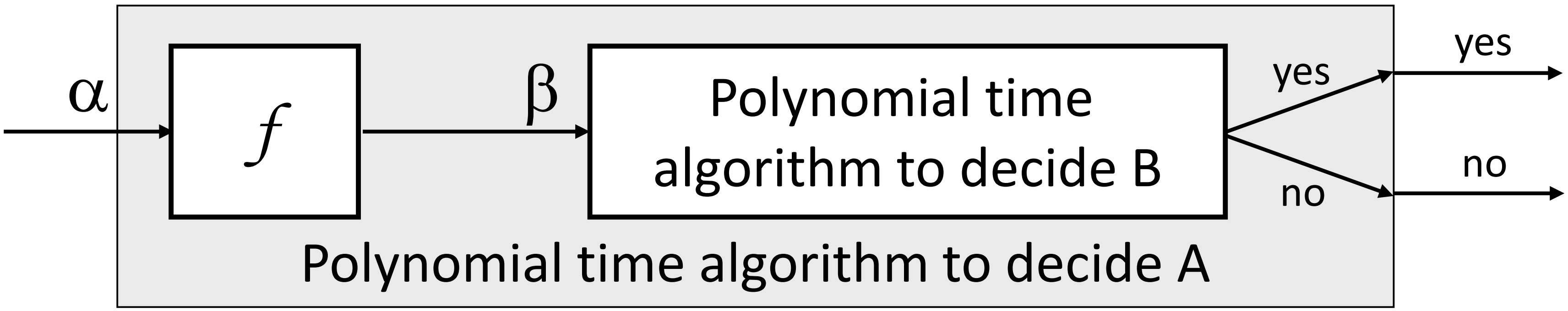
Implications of Reduction



- If $A \leq_p B$ and $B \in P$, then $A \in P$

- if $A \leq_p B$ and $A \notin P$, then $B \notin P$

Proving Polynomial Time

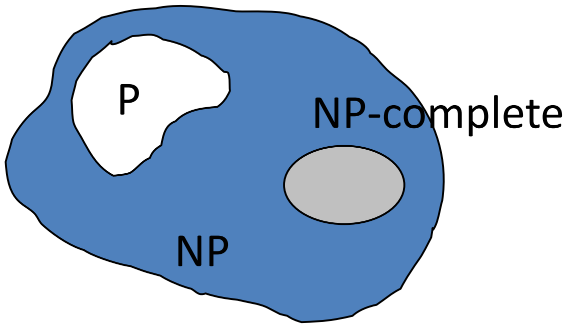


1. Use a **polynomial time** reduction algorithm to transform A into B
2. Run a known **polynomial time** algorithm for B
3. Use the answer for B as the answer for A

Proving NP-Completeness In Practice

- Prove that the problem B is in NP
  - A randomly generated string can be checked in polynomial time to determine if it represents a solution
- Show that **one known** NP-Complete problem can be transformed to B in polynomial time
  - No need to check that **all** <u>NP-Complete</u> problems are reducible to B

Revisit "Is P = NP?"



Theorem: If any NP-Complete problem can be solved in polynomial time $\Rightarrow$ then P = NP.

P & NP-Complete Problems

- **Shortest simple path**
  - Given a graph G = (V, E) find a **shortest** path from a source to all other vertices
  - <u>Polynomial solution:</u> O(VE)
- **Longest simple path**
  - Given a graph G = (V, E) find a **longest** path from a source to all other vertices
  - <u>NP-complete</u>

P & NP-Complete Problems

- **Euler tour**
  - G = (V, E) a connected, directed graph find a cycle that traverses each edge of G exactly once (may visit a vertex multiple times)
  - Polynomial solution O(E)
- **Hamiltonian cycle**
  - G = (V, E) a connected, directed graph find a cycle that visits each vertex of G exactly once
  - NP-complete

NP-naming convention

- **NP-complete -** means problems that are 'complete' in NP, i.e. the most difficult to solve in NP
- **NP-hard** - stands for 'at least' as hard as NP (but not necessarily **in** NP);
- **NP-easy** - stands for 'at most' as hard as NP (but not necessarily **in** NP);
- **NP-equivalent** - means equally difficult as NP, (but not necessarily **in** NP);

Examples NP-complete and NP-hard problems

Hamiltonian Paths

NP-complete

Optimization Problem: Given a graph, find a path that passes through every vertex exactly once

Decision Problem: Does a given graph have a Hamiltonian Path ?

Traveling Salesman

NP-hard

Optimization Problem: Find a minimum weight Hamiltonian Path

Decision Problem: Given a graph and an integer k, is there a Hamiltonian Path with a total weight at most k ?

Thank you