P, NP, NP-COMPLETE, NP-HARD

M. Elif Karslıgil, Yıldız Technical University, Istanbul,

What is an efficient algorithm?

- Algorithm Efficiency:

  An algorithm is efficient iff it runs in polynomial time on a serial computer.

- Runtimes of efficient algorithms:

  $O(n^2)$, $O(n^3)$, $O(1)$, $O(n \lg n)$, $O(n^{10,000,000,000})$

- Runtimes of inefficient algorithms:

  $O(2^n)$, $O(n^n)$, $O(n!)$

Runtimes of some problems

Input Size vs. Complexity	10	20	30	40	50	60
n	.00001 s	.00002 s	.00003 s	.00004 s	.00005 s	.00006 s
n^2	.0001 s	.0004 s	.0009 s	.0016 s	.0025 s	.0036 s
n^3	.001 s	.008 s	.027 s	.064 s	.125 s	.216 s
n^5	.1 s	3.2 s	24.3 s	1.7 min	5.2 min	13.0 min
2^n	.001 s	1.0 s	17.9 min	12.7 days	35.7 years	366 centuries
3^n	.059 s	58 min	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^{13} centuries

Tractable vs. Intractable Problems

- A problem that can be solved in polynomial time are called **tractable.**
  - Most searching and sorting algorithms $O(n^2)$, $O(n*\lg n)$
  - Problems that can **not** be solved in polynomial time are called **intractable or unsolvable.** Complexity may be described by exponential functions.
  - Traveling salesperson $(O(n^2 2^n))$, knapsack $(O(2^{n/2}))$
- Are non-polynomial algorithms always worst than polynomial algorithms?
  - $n^{1,000,000}$ is *technically* tractable, but really impossible
  - $n^{\log n}$ is *technically* intractable, but easy

Intractable Problems

- Can be classified in various categories based on their degree of difficulty, e.g.,
  - NP
  - NP-complete
  - NP-hard

Intractable Problems

- Can be classified in various categories based on their degree of difficulty, e.g.,
  - NP(Non deterministic Polynomial)
  - NP-complete
  - NP-hard

Decision vs. Optimization Problems

- Decision problems :
  - Given an input and a question regarding a problem, determine if the answer is yes or no
- Optimization problems:
  - Find a solution with the best value

Decision vs. Optimization Problems

- **Hamiltonian Circuit :** Given a directed graph, we want to decide whether or not there is a Hamiltonian cycle in this graph.
- **The Traveling Salesman :** Given a complete graph and an assignment of weights to the edges, find a Hamiltonian cycle of minimum weight.

Decision vs. Optimization Problems

- Each optimization poblem has a corresponding decision problem.
- The optimization version of the TSP :
  - input : weighted complete graph
- the decision version of the TSP :
  - input : weighted complete graph and a real number c. whether or not there exists a Hamiltonian cycle whose combined weight of edges does not exceed c

Deterministic / Non-deterministic Algorithms

- **Deterministic algorithms :** the outcome of every operation is uniquely defined.
- **Nondeterministic algorithms:** the result of an operation is not uniquely defined, but restricted to a specific set of possibilities.

Deterministic / Non-deterministic Algorithms

- Deterministic algorithms : the outcome of every operation is uniquely defined.
- Nondeterministic algorithms: the result of an operation is not uniquely defined, but restricted to a specific set of possibilities.

Nondeterministic Algorithms

- **Nondeterministic algorithm** has two phases and an output step

1) **Nondeterministic (“guessing”) stage:** generates randomly an arbitrary string that can be thought of as a candidate solution (“certificate”)

2) **Deterministic (“verification”) stage:** takes the certificate and the instance to the problem and returns YES if the certificate represents a solution

3) **Output :** If the verfying stage returned **true** the output is **success**. Otherwise it is **failure**.

Nondeterministic Algorithms

- The nondeterministic algorithm uses three basic procedures;
  - **CHOICE(1,n) or CHOICE(s) :** chooses and returns an arbitrary element, from the closed interval [1,n] or from the set s.
  - **SUCCESS :** successful completion of the algorithm.
  - **FAILURE :** an unsuccessful termination of the algorithm.

A Searching Example

// **the problem is to search for an element X in array A[1..n]**//

//output : j such that A(j) = x; or j = 0 if x is not in A//

```
NSearch (A,x)
j = choice (1. . . n)
if A[j] = x then
        print(j)
        success
endif
print('0')
failure
```

Complexity O(1) : Since A is not ordered, every deterministic search algorithm is of complexity $\Omega(n)$, whereas the non-deterministic algorithm has the complexity as O(1)

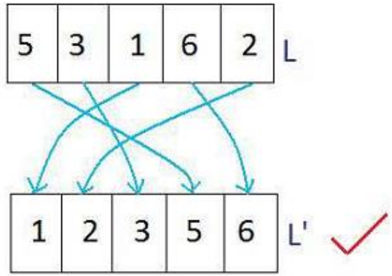
Nondeterministic decision algorithms generate a zero or 1 as their output.

A Sorting Example

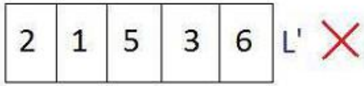
// **sort the array A[1..n] into and arrayB[1..n]**//

```
NSort(A, B)
for i=1 to n do
        B[i] = 0
for i=1 to n do
        j=choice(1... n)
        if B[j] != 0  then failure
        B[j]=A[i]
endfor
for i=1 to  n−1do
        if B[i]<B[i+ 1]then failure
endfor
 print(B)
 Success
```

Complexity $\Theta(n)$:



A path resulting in this permutation is always followed.



A path resulting in this permutation is never followed.

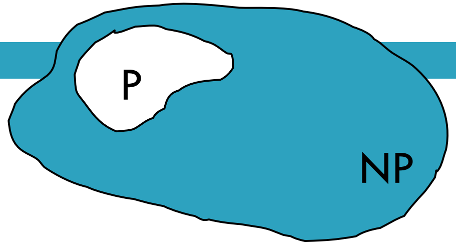
NP(Nondeterministic Polynomial) Algorithms

- NP: the class of decision problems that are solvable in polynomial time on a non deterministic machine (or with a non-deterministic algorithm)
- NP Algorithms : Verification stage is polynomial

Class P and Class NP

- P is the set of all decision problems solvable by a deterministic algorithm in polynomial time.
- NP is the set of all decision problems solvable by a nondeterministic algorithm in polynomial time.
- Since a deterministic algorithm is just a special case of a nondeterministic one, it is clear from the definitions that **P $\subseteq$NP.**

Is P = NP?



- Any problem in P is also in NP:

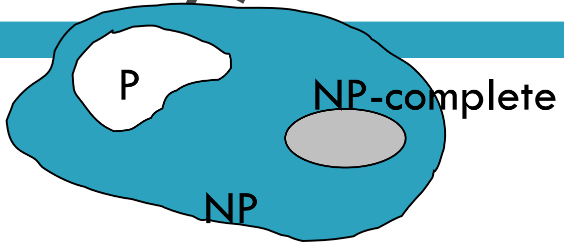
$$P \subseteq NP$$

- 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

P is a subset of NP

- Since it takes polynomial time to run the program, just run the program and get a solution
- But is NP a subset of P?
- No one knows if P = NP or not
- Solve for a million dollars!
  - http://www.claymath.org/millennium-problems
  - The Poincare conjecture is solved today

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.

NP-Complete

- A decision problem D is NP-complete iff
  1. $D \in NP$
  2. every problem in NP is polynomial-time reducible to D

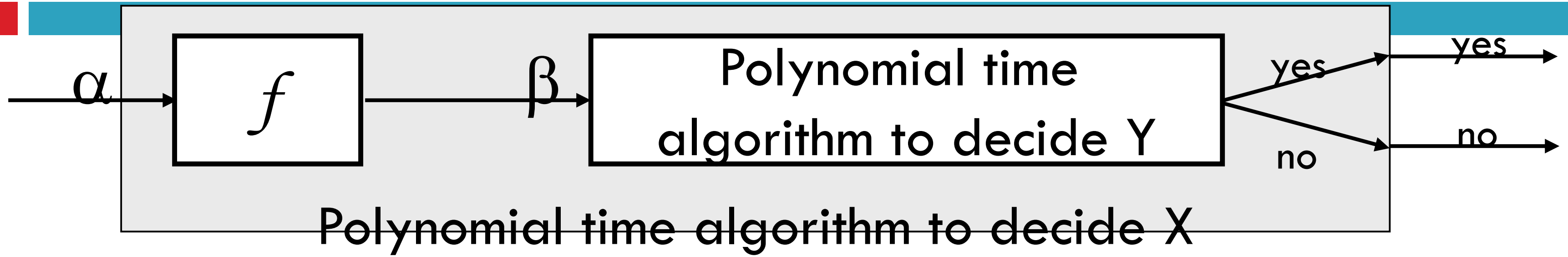
NP-complete Algorithms

- Directed Hamiltonian Cycle Decision Problem (DHC)
- Travelling Salesperson Decision Problem (TSP)
- Multiprocessor Scheduling Problem
- 0/1 Knapsack Decision Problem (KNP)

Reduction

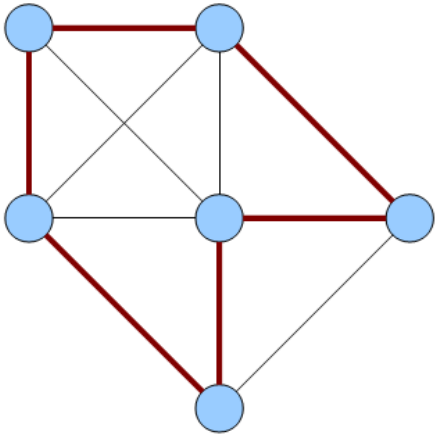
- Many problems for which there are no known polynomial time algorithms are computationally related.
- Problem X is easier than problem Y
- Problem X reduces to problem Y : if you can use an algorithm that solves Y to help solve X.
- **Idea:** transform the inputs of X to inputs of Y
- Cost of solving X = M * (ost of solving Y) + cost of reduction
- $X \leq_p Y$

Proving Polynomial Time



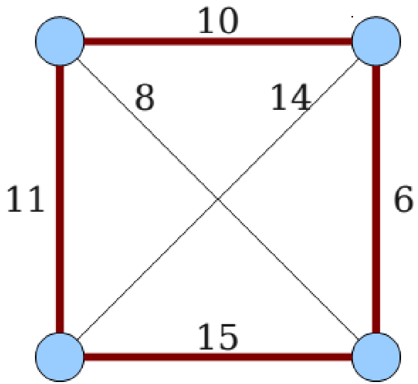
1. Use a **polynomial time** reduction algorithm to transform X into Y
2. Run a known **polynomial time** algorithm for Y
3. Use the answer for Y as the answer for X

Reduction Example



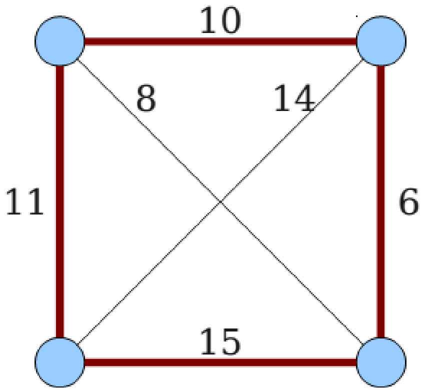
A **Hamiltonian cycle** in an undirected graph G is a simple cycle that visits every node in G.

Reduction Example



Given a complete, undirected, weighted graph G, the **traveling salesperson problem** (**TSP**) is to find a Hamiltonian cycle in G of least total cost.

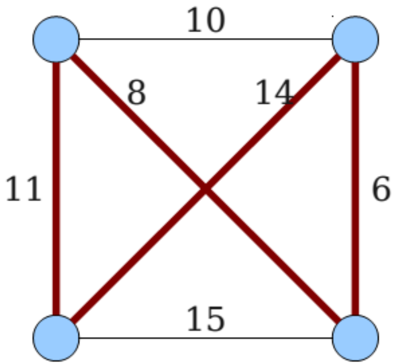
Reduction Example



Cost: **42**

Given a complete, undirected, weighted graph G, the **traveling salesperson problem** (**TSP**) is to find a Hamiltonian cycle in G of least total cost.

Reduction Example



Cost: **39**

(This is the optimal solution)

Given a complete, undirected, weighted graph G, the **traveling salesperson problem** (**TSP**) is to find a Hamiltonian cycle in G of least total cost.

Reduction Example

- Given as input
  - A complete, undirected graph G, and
  - a set of edge weights, which are positive integers,
- the TSPis to find a Hamiltonian cycle in G with least total weight.
- Since G is complete, there has to be at least one Hamiltonian cycle. The challenge is finding the least-cost cycle.

Naïve Solution

- Try all possible Hamiltonian cycles in the graph.
- How many Hamiltonian cycles are there?
  - Answer: $(n-1)!\,/\,2$
- Spend $O(n)$ time processing each cycle.
  - Total time: $\Theta(n!)$.
- This is completely impractical!

Reduction Example

- Hamiltonian Cycle : Given a undirected graph G=(V,E), does there exists a simple cycle that visits every node?
- Traveling Salesman : Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤D? (Decision problem)

Hamiltonian cycles

- Determining whether a directed graph has a Hamiltonian cycle does not have a polynomial time algorithm (yet!)
- However if someone was to give you a sequence of vertices, determining whether or not that sequence forms a Hamiltonian cycle can be done in polynomial time
- Therefore Hamiltonian cycles are in NP

Polynomial Time Reduction

- **Reduction:** Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, for reduction
  - One call to function for Y.
  - Notation : $X \leq_p Y$

Polynomial Time Reduction

- Show Hamiltonian Cycle $\leq_P$ TSP:
  - For every instance of Hamiltonian Cycle create an instance of TSP such that the TSP instance has tour of length $\leq n$ if and only if G has a Hamiltonian cycle

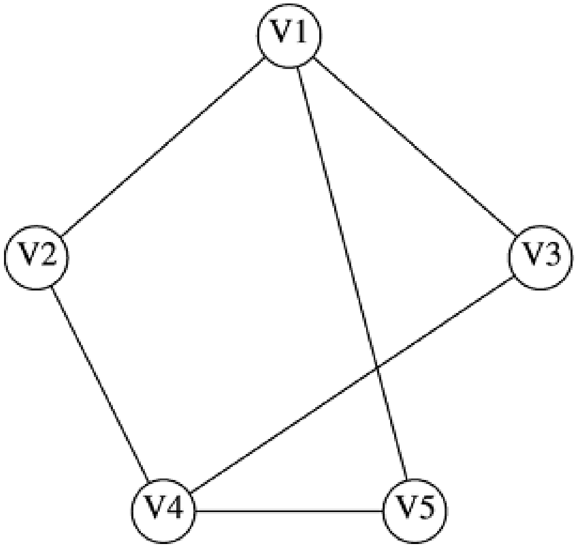
Polynomial Time Reduction

- Reduction:
  - Given instance G=(V,E) of Hamiltonian Cycle, create n cities with distance function

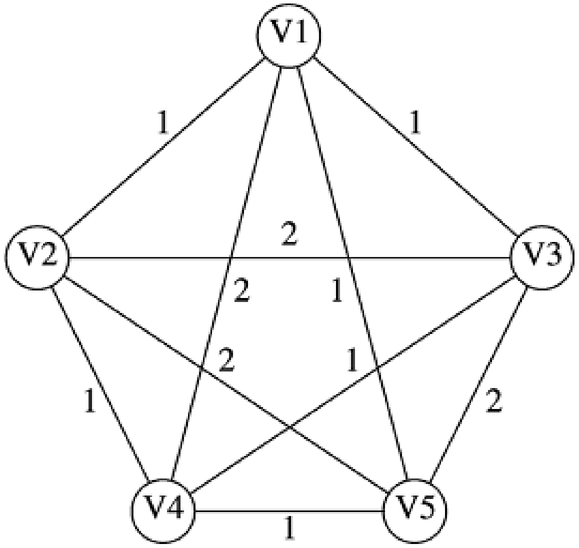
$$d(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 2 & \text{if } (u,v) \notin E \end{cases}$$

  - TSP instance has tour of length ≤n if and only if G has a Hamiltonian cycle.

Reduction of Ham-Circuit to TSP



G



G'

Reduction of Ham-Circuit to TSP

- Reduce Ham-Cycle to TSP
- Given Ham-Cycle input graph G=(V,E)
- Create complete, weighted graph G'=(V,E')
  - Each edge in E is in E' with weight of 1
  - Each edge not in E is in E' with weight of 2
- Solve TSP with k = |V|
- If such a TSP tour exists in G', the same tour is a Hamiltonian Cycle in G

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 ?

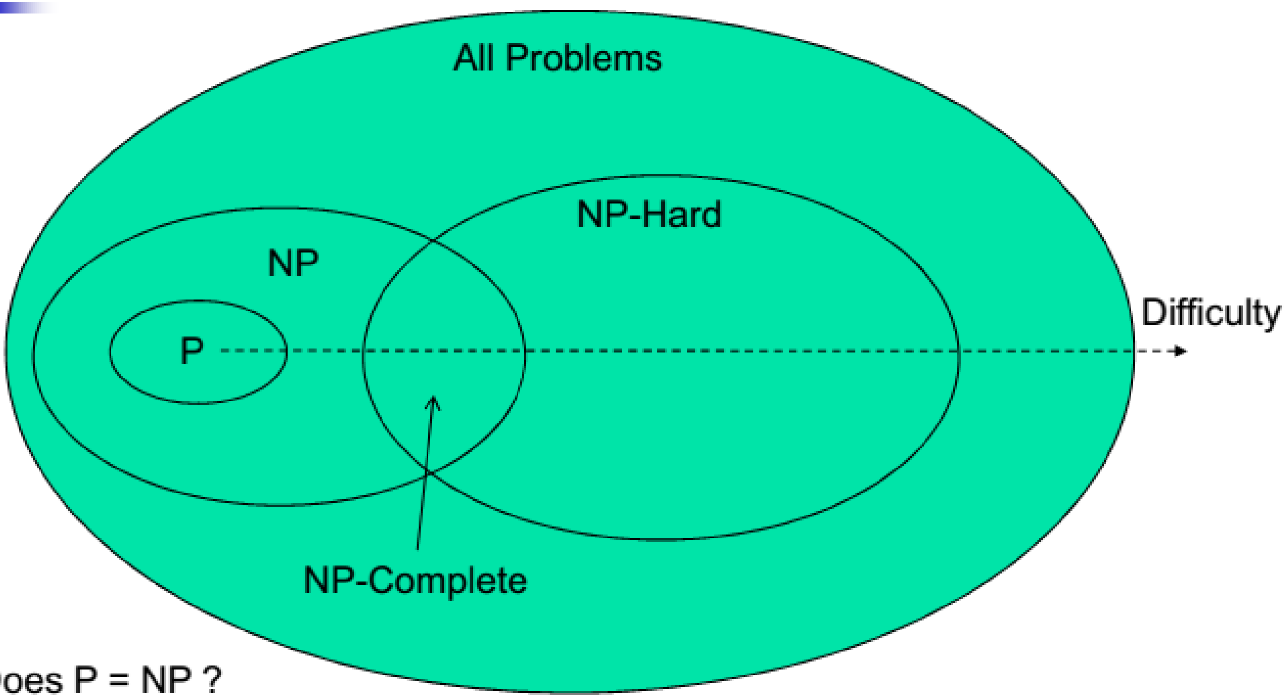
Traveling Salesman

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 ?

P, NP, NP-complete, NP-hard

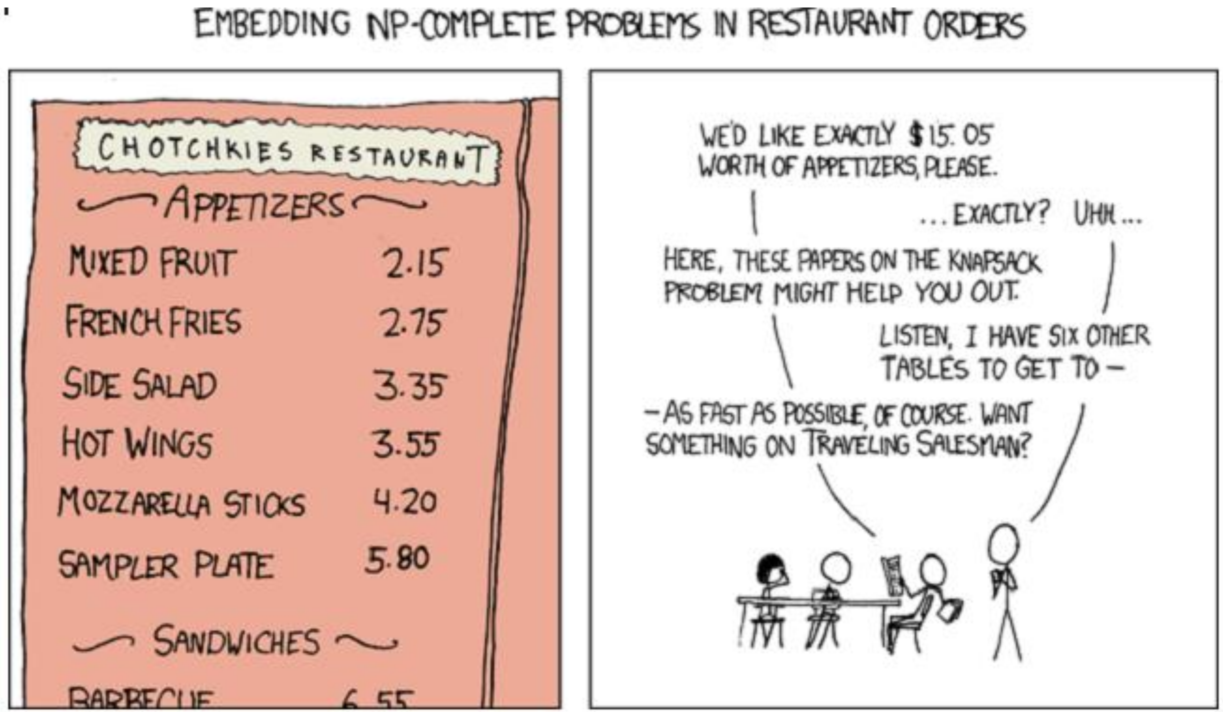


Does P = NP ?

Amusing analogy (thanks to lecture notes at University of Utah)

- Students believe that every problem assigned to them is NP-complete in difficulty level, as they have to find the solutions.
- Teaching Assistants, on the other hand, find that their job is only as hard as NP, as they only have to verify the student's answers.
- When some students confound the TAs, even verification becomes hard

Fun with NP-Complete



Examples of Intractable Problems

Hamiltonian Paths

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

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 ?