Formula Satisfiability

Formula Satisfiability

Input: boolean formula with n variables, with $\vee, \wedge, \neg$ operators.

Output: whether the boolean formula is satisfiable

- Example: $\neg((\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3) \vee x_1 \vee (\neg x_2 \wedge x_3))$ is not satisfiable
- Trivial algorithm: enumerate all possible assignments, and check if each assignment satisfies the formula. The algorithm runs in exponential time.
- Formula Satisfiablity is NP-hard

Outline

Decision Problem Vs Optimization Problem

Def. A problem X is called a decision problem if the output is either 0 or 1 (yes/no).

Decision Problem Vs Optimization Problem

Def. A problem X is called a decision problem if the output is either 0 or 1 (yes/no).

- When we define the P and NP, we only consider decision problems.

Decision Problem Vs Optimization Problem

Def. A problem X is called a decision problem if the output is either 0 or 1 (yes/no).

- When we define the P and NP, we only consider decision problems.

Fact For each optimization problem X, there is a decision version X' of the problem. If we have a polynomial time algorithm for the decision version X', we can solve the original problem X in polynomial time.

Optimization to Decision

Shortest Path

Input: graph $G = (V, E)$, weight w, s, t and a bound L

Output: whether there is a path from s to t of length at most L

Optimization to Decision

Shortest Path

Input: graph $G = (V, E)$, weight w, s, t and a bound L

Output: whether there is a path from s to t of length at most L

Maximum Independent Set

Input: a graph G and a bound k

Output: whether there is an independent set of size at least k

Encoding

The input of a problem will be encoded as a binary string.

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String:

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String: 11/

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String: 11/110/

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String: 11/110/1100100/

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String: 11/110/1100100/ 1001/

Encoding

The input of a problem will be encoded as a binary string.

Example: Sorting problem

- Input: (3, 6, 100, 9, 60)
- Binary: (11, 110, 1100100, 1001, 111100)
- String: 11/110/1100100/ 1001/111100/

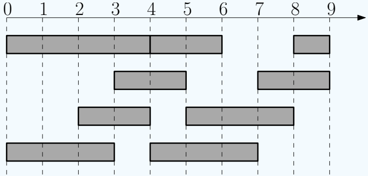
Encoding

The input of an problem will be encoded as a binary string.

Encoding

The input of an problem will be encoded as a binary string.

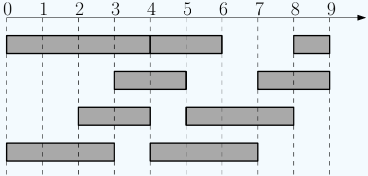
Example: Interval Scheduling Problem



Encoding

The input of an problem will be encoded as a binary string.

Example: Interval Scheduling Problem

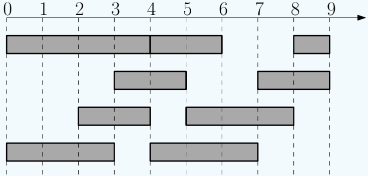


- $(0, 3, 0, 4, 2, 4, 3, 5, 4, 6, 4, 7, 5, 8, 7, 9, 8, 9)$

Encoding

The input of an problem will be encoded as a binary string.

Example: Interval Scheduling Problem



- $(0, 3, 0, 4, 2, 4, 3, 5, 4, 6, 4, 7, 5, 8, 7, 9, 8, 9)$
- Encode the sequence into a binary string as before

Encoding

Def. The size of an input is the length of the encoded string s for the input, denoted as $|s|$.

Q: Does it matter how we encode the input instances?

Encoding

Def. The size of an input is the length of the encoded string s for the input, denoted as $|s|$.

Q: Does it matter how we encode the input instances?

A: No! As long as we are using a "natural" encoding. We only care whether the running time is polynomial or not

Define Problem as a Function $X : \{0,1\}^* \to \{0,1\}$

Def. A decision problem X is a function mapping $\{0,1\}^*$ to $\{0,1\}$ such that for any $s \in \{0,1\}^*$, $X(s)$ is the correct output for input s.

- $\{0,1\}^*$: the set of all binary strings of any length.

Define Problem as a Function $X : \{0,1\}^* \to \{0,1\}$

Def. A decision problem X is a function mapping $\{0,1\}^*$ to $\{0,1\}$ such that for any $s \in \{0,1\}^*$, $X(s)$ is the correct output for input s.

- $\{0,1\}^*$: the set of all binary strings of any length.

Def. An algorithm A solves a problem X if, $A(s) = X(s)$ for any binary string s

Define Problem as a Function $X : \{0,1\}^* \to \{0,1\}$

Def. A decision problem X is a function mapping $\{0,1\}^*$ to $\{0,1\}$ such that for any $s \in \{0,1\}^*$, $X(s)$ is the correct output for input s.

- $\{0,1\}^*$: the set of all binary strings of any length.

Def. An algorithm A solves a problem X if, $A(s) = X(s)$ for any binary string s

Def. A has a polynomial running time if there is a polynomial function $p(\cdot)$ so that for every string s, the algorithm A terminates on s in at most $p(|s|)$ steps.

Complexity Class P

Def. The complexity class P is the set of decision problems X that can be solved in polynomial time.

Complexity Class P

Def. The complexity class P is the set of decision problems X that can be solved in polynomial time.

- The decision versions of interval scheduling, shortest path and minimum spanning tree all in P.

Certifier for Hamiltonian Cycle (HC)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for HC

Certifier for Hamiltonian Cycle (HC)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for HC
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Certifier for Hamiltonian Cycle (HC)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for HC
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given a graph $G = (V, E)$ with a HC, how can Alice convince Bob that G contains a Hamiltonian cycle?

Certifier for Hamiltonian Cycle (HC)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for HC
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given a graph $G = (V, E)$ with a HC, how can Alice convince Bob that G contains a Hamiltonian cycle?

A: Alice gives a Hamiltonian cycle to Bob, and Bob checks if it is really a Hamiltonian cycle of G

Certifier for Hamiltonian Cycle (HC)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for HC
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given a graph $G = (V, E)$ with a HC, how can Alice convince Bob that G contains a Hamiltonian cycle?

A: Alice gives a Hamiltonian cycle to Bob, and Bob checks if it is really a Hamiltonian cycle of G

Def. The message Alice sends to Bob is called a certificate, and the algorithm Bob runs is called a certifier.

Certifier for Independent Set (Ind-Set)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for Ind-Set
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Certifier for Independent Set (Ind-Set)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for Ind-Set
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given graph $G = (V, E)$ and integer k, such that there is an independent set of size k in G, how can Alice convince Bob that there is such a set?

Certifier for Independent Set (Ind-Set)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for Ind-Set
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given graph $G = (V, E)$ and integer k, such that there is an independent set of size k in G, how can Alice convince Bob that there is such a set?

A: Alice gives a set of size k to Bob and Bob checks if it is really a independent set in G.

Certifier for Independent Set (Ind-Set)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for Ind-Set
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given graph $G = (V, E)$ and integer k, such that there is an independent set of size k in G, how can Alice convince Bob that there is such a set?

A: Alice gives a set of size k to Bob and Bob checks if it is really a independent set in G.

- Certificate: a set of size k

Certifier for Independent Set (Ind-Set)

- Alice has a supercomputer, fast enough to run the $2^{O(n)}$ time algorithm for Ind-Set
- Bob has a slow computer, which can only run an $O(n^3)$-time algorithm

Q: Given graph $G = (V, E)$ and integer k, such that there is an independent set of size k in G, how can Alice convince Bob that there is such a set?

A: Alice gives a set of size k to Bob and Bob checks if it is really a independent set in G.

- Certificate: a set of size k
- Certifier: check if the given set is really an independent set

The Complexity Class NP

Def. B is an efficient certifier for a problem X if

- B is a polynomial-time algorithm that takes two input strings s and t, and outputs 0 or 1.
- there is a polynomial function p such that, $X(s) = 1$ if and only if there is string t such that $|t| \le p(|s|)$ and $B(s,t) = 1$.

The string t such that $B(s,t) = 1$ is called a certificate.

The Complexity Class NP

Def. B is an efficient certifier for a problem X if

- B is a polynomial-time algorithm that takes two input strings s and t, and outputs 0 or 1.
- there is a polynomial function p such that, $X(s) = 1$ if and only if there is string t such that $|t| \leq p(|s|)$ and $B(s,t) = 1$.

The string t such that $B(s,t) = 1$ is called a certificate.

Def. The complexity class NP is the set of all problems for which there exists an efficient certifier.

HC (Hamiltonian Cycle) $\in$ NP

- Input: Graph G

HC (Hamiltonian Cycle) $\in$ NP

- Input: Graph G
- Certificate: a permutation S of V that forms a Hamiltonian Cycle
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G)|)$ for some polynomial function p

HC (Hamiltonian Cycle) $\in$ NP

- Input: Graph G
- Certificate: a permutation S of V that forms a Hamiltonian Cycle
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G)|)$ for some polynomial function p
- Certifier B: $B(G, S) = 1$ if and only if S gives an HC in G
- Clearly, B runs in polynomial time

HC (Hamiltonian Cycle) $\in$ NP

- Input: Graph G
- Certificate: a permutation S of V that forms a Hamiltonian Cycle
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G)|)$ for some polynomial function p
- Certifier B: $B(G, S) = 1$ if and only if S gives an HC in G
- Clearly, B runs in polynomial time
- $\text{HC}(G) = 1 \quad \Longleftrightarrow \quad \exists S,\ B(G, S) = 1$

MIS (Maximum Independent Set) $\in$ NP

- Input: graph $G = (V, E)$ and integer k

MIS (Maximum Independent Set) $\in$ NP

- Input: graph $G = (V, E)$ and integer k
- Certificate: a set $S \subseteq V$ of size k
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G, k)|)$ for some polynomial function p

MIS (Maximum Independent Set) $\in$ NP

- Input: graph $G = (V, E)$ and integer k
- Certificate: a set $S \subseteq V$ of size k
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G, k)|)$ for some polynomial function p
- Certifier B: $B((G, k), S) = 1$ if and only if S is an independent set in G
- Clearly, B runs in polynomial time

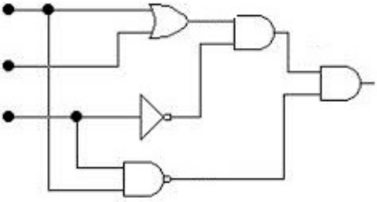
MIS (Maximum Independent Set) $\in$ NP

- Input: graph $G = (V, E)$ and integer k
- Certificate: a set $S \subseteq V$ of size k
- $|\text{encoding}(S)| \leq p(|\text{encoding}(G, k)|)$ for some polynomial function p
- Certifier B: $B((G, k), S) = 1$ if and only if S is an independent set in G
- Clearly, B runs in polynomial time
- $\text{MIS}(G, k) = 1 \quad \iff \quad \exists S,\ B((G, k), S) = 1$

Circuit Satisfiablity (Circuit-Sat) Problem

Input: a circuit with and/or/not gates

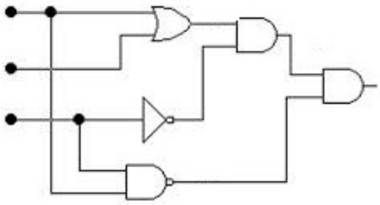
Output: whether there is an assignment such that the output is 1?



Circuit Satisfiablity (Circuit-Sat) Problem

Input: a circuit with and/or/not gates

Output: whether there is an assignment such that the output is 1?



- Is Circuit-Sat $\in$ NP?

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

- Is $\overline{\text{HC}} \in \text{NP}$?

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

- Is $\overline{\text{HC}} \in \text{NP}$?
- Can Alice convince Bob that G is a yes-instance (i.e, G does not contain a HC), if this is true.

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

- Is $\overline{\text{HC}} \in \text{NP}$?
- Can Alice convince Bob that G is a yes-instance (i.e, G does not contain a HC), if this is true.
- Unlikely

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

- Is $\overline{\text{HC}} \in \text{NP}$?
- Can Alice convince Bob that G is a yes-instance (i.e, G does not contain a HC), if this is true.
- Unlikely
- Alice can only convince Bob that G is a no-instance

$\overline{\text{HC}}$

Input: graph $G = (V, E)$

Output: whether G does not contain a Hamiltonian cycle

- Is $\overline{\text{HC}} \in$ NP?
- Can Alice convince Bob that G is a yes-instance (i.e, G does not contain a HC), if this is true.
- Unlikely
- Alice can only convince Bob that G is a no-instance
- $\overline{\text{HC}} \in$ Co-NP

The Complexity Class Co-NP

Def. For a problem X, the problem $\overline{X}$ is the problem such that $\overline{X}(s) = 1$ if and only if $X(s) = 0$.

Def. Co-NP is the set of decision problems X such that $\overline{X} \in$ NP.

Def. A tautology is a boolean formula that always evaluates to 1.

Tautology Problem

Input: a boolean formula

Output: whether the formula is a tautology

- e.g. $(\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3) \vee x_1 \vee (\neg x_2 \wedge x_3)$ is a tautology

Def. A tautology is a boolean formula that always evaluates to 1.

Tautology Problem

Input: a boolean formula

Output: whether the formula is a tautology

- e.g. $(\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3) \vee x_1 \vee (\neg x_2 \wedge x_3)$ is a tautology
- Bob can certify that a formula is not a tautology

Def. A tautology is a boolean formula that always evaluates to 1.

Tautology Problem

Input: a boolean formula

Output: whether the formula is a tautology

- e.g. $(\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3) \vee x_1 \vee (\neg x_2 \wedge x_3)$ is a tautology
- Bob can certify that a formula is not a tautology
- Thus Tautology $\in$ Co-NP

P $\subseteq$ NP