------------------------------- 34 -----------------------------

NP-Completeness
(Branch-and-bound)

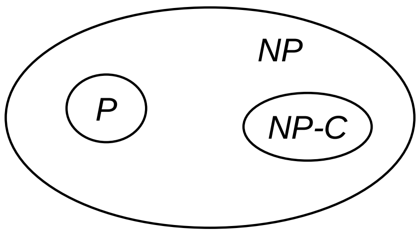
Polynomial time: $O(n^k)$ for some constant k, n is the problem size.

Exponential time: $O(2^n)$, $O(3^{n^3})$, ...

non-deterministic algorithm: an algorithm which
1. guesses an answer, and then
2. verifies the answer.

P: the set of problems that can be solved in $O(n^k)$ time (using a deterministic algorithm).

NP: the set of problems that can be solved in $O(n^k)$ time using a non-deterministic algorithm. (Or, problems whose answers can be verified in $O(n^k)$ time.)



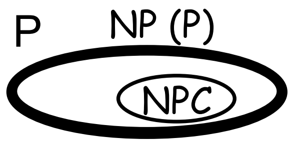
Decision problem: return Yes/No!

Reduction: transform a problem into another.
- * Selection $\Rightarrow$ Sorting
- * Decision version $\Rightarrow$ Optimization version
- * if A $\Rightarrow$ B, then usually B is harder
- * if A $\Rightarrow^p$ B and B $\Rightarrow^p$ C, then A $\Rightarrow^p$ C

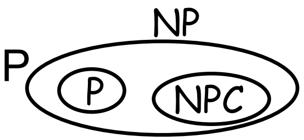
NP-Complete: a problem *A* is in *NP-C* iff (i) *A* is in *NP*, and (ii) all problems in *NP* can be reduced to it in $O(n^k)$ time.
- * all NP-C problems are of the same difficulty
- * all NP$\Rightarrow^p$ A $\cong$ an NP-C$\Rightarrow^p$ A $\cong$ all NP-C$\Rightarrow^p$ A

* If any problem in *NP-C* can be solved in $O(n^k)$ time, then *P*=*NP*. It is believed (not proved) that *P*$\neq$*NP*.

* if NP = P



if NP $\neq$ P

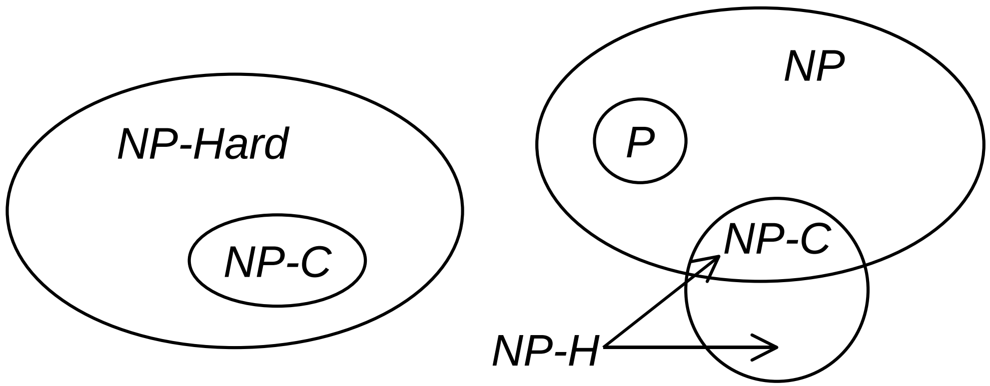


NP-Hard: a problem *A* is in *NP-H* iff

(1) *A* is at least as hard as problems in *NP-C*.

or (2) all problems in *NP* can be reduced to *A* in $O(n^k)$ time.

or (3) a problem in *NP-C* can be reduced to *A* in $O(n^k)$ time.



NP-Complete problems:

* (*Circuit*) *satisfiability problem* (*SAT*):
  $((a \rightarrow b) \vee \neg((\neg a \leftrightarrow c) \vee d)) \wedge \neg b$

* *3-CNF satisfiability problem* (*3SAT*):
  $(a \vee b \vee c) \wedge (a \vee \neg d \vee e) \wedge (b \vee f \vee a)$

* *The subset-sum* (*partition*) *problem*: partition a set of (real) numbers into two subsets of the same sum.

* *The hamiltonian-cycle problem*

* *The traveling-salesperson problem* (*TSP*)

* *The clique problem*

* *The longest simple path problem*

* *The vertex-cover problem*

* *The independent set problem*

* *The graph coloring problem*

* 2SAT $\in$ *P*. (*2CNF* $\in$ *P*)

Question 1: Determine whether the following statements are correct of not.

(1) If a problem is *NP-Complete*, then it can not be solved by any polynomial time algorithm in worst cases.

(2) If a problem is *NP-Complete*, then we have not found any polynomial time algorithm to solve it in worse cases.

(3) If a problem is *NP-Complete*, then it is unlikely that a polynomial time algorithm can be found in the future to solve it in worst cases.

(4) If a problem is *NP-Complete*, then it is unlikely that we can find a polynomial time algorithm to solve it in average cases.

(5) If we can prove that the lower bound of an *NP-Complete* problem is exponential, then we have proved that *NP*≠*P*.

Question 2: Determine whether the following statements are correct of not, and justify your answer.

(a) Any *NP-hard* problem can be solved in polynomial time if there is an algorithm that can solve the satisfiability problem in polynomial time.

(b) Any *NP-Complete* problem can be solved by a polynomial time deterministic algorithm in average if and only if *NP*=*P* is proved.

(c) The clause-monotone satisfiability problem is *NP-Complete*, where a formula is called monotone if each clause of the formula contains either only positive variables or only negative variables.

Question 3: Determine whether the following statements are correct of not.

(1) The NP problems consist of only decision problems.

(2) If a problem is NP-complete, then it can not be solved by any polynomial time algorithm in worst cases.

(3) If we prove that problem *A* can polynomial-time reduce to satisfiability problem, then problem *A* is NP-complete.

(4) If a problem *A* is polynomial time reducible to problem *B* and *B* has a polynomial time algorithm, then problem *A* has a polynomial time algorithm.

(5) If an NP-complete problem can be solved in polynomial time, then NP ≠ P.

(6) The problem of determining whether an integer number is a prime number is an NP-complete problem.

(7) Any NP-hard problem can be solved in polynomial time if there is an algorithm that can solve the satisfiability problem in polynomial time.

(8) The hamiltonian-path problem can be solved in polynomial time on directed acyclic graphs

(9) 3-CNF (the satisfiability problem, in which each clause has exactly three literals) is reducible to 2-CNF problem.

Question 4: Suppose problem P_1 can be reduced to another problem P_2 in $O(n^2)$ time, where n is the input size. Answer the following questions and justify your answer briefly.

(a) If P_1 is *NP-hard*, is P_2 *NP-hard*?

(b) If P_2 is *NP-hard*, is P_1 *NP-hard*?

(c) Suppose P_1 can be solved in $O(f(n))$ time. Is it possible to derive a time lower-bound or a time upper-bound for P_2? If it is possible, what is the time bound?

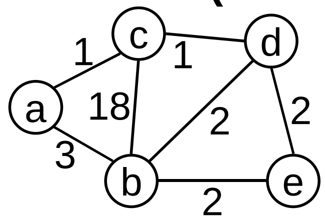
Approximation Algorithm: Let A^* be the optimal solution of an input. An approximation algorithm will produce a solution A such that

$$|A^*-A|/A^* \leq \varepsilon,$$

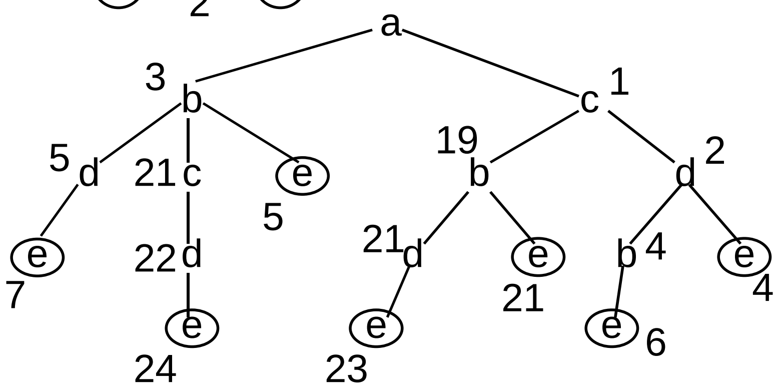
where ε is called the *relative error bound*.

Heuristic Algorithm: may produce a good solution but no guarantee on the error bound.

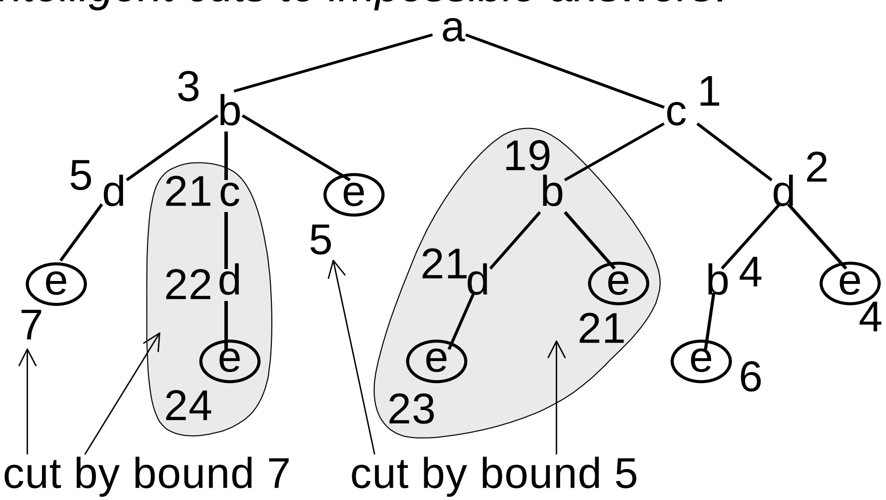
Brute-force (search): Try all possible answers.



: Find a shortest path from a to e.



Branch-and-Bound search: *Brute-force + Intelligent cuts to impossible answers*.



Homework: None.