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

NP-Completeness (Branch-and-bound)

34-1x

Polynomial time: $O(n^k)$ for some constant k, n is the problem size. polynomial → easy, can solve it

Exponential time: $O(2^n)$, $O(3^{n^3})$, ... non-polynomial → hard, cannot solve it

(verification algo.)

non-deterministic algorithm: an algorithm which
1. guesses an answer, and then 34-1a
2. verifies the answer. non-deterministic sort ? 34-1x

polynomial (deterministic) 計算

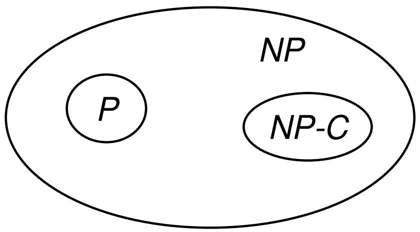
P: the set of problems that can be solved in $O(n^k)$ time (using a deterministic algorithm).

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.) 驗算 34-1y 34-1b

polynomial

non-deterministic

*會計算就一定會驗算



34-1z

"δ(s, v) < 45 ?" "|f*| > 36 ?" 34-2

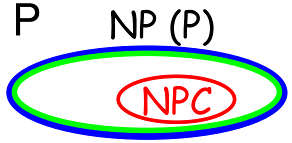
Decision problem: return Yes/No!

Reduction: transform a problem into another. 34-2a
- * Selection ⇨ Sorting
- * Decision version ⇨ Optimization version
- * if A ⇨ B, then usually B is harder (B $\geq_{hard}$ A) 34-2de
- * if A $⇨^p$ B and B $⇨^p$ C, then A $⇨^p$ C 34-2x
  reduction (hardness) is transitive 34-2fg

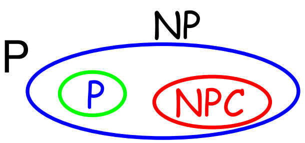
NP-Complete: a problem A is in NP-C iff (i) A is in NP, and (ii) all problems in NP can be reduced to it in $O(n^k)$ time. 34-2h
- * all NP-C problems are of the same difficulty
- * all NP $⇨^p$ A ≅ an NP-C $⇨^p$ A ≅ all NP-C $⇨^p$ A

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

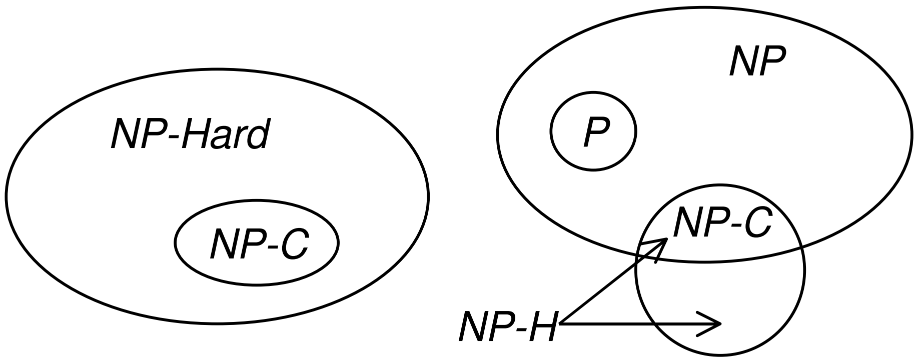
* if NP = P



if NP ≠ 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. all NP $⇨^p$ A
or (3) a problem in NP-C can be reduced to A in $O(n^k)$ time. an (all) NP-C $⇨^p$ A



34-3ab

34-3x

NP-Complete problems:

$\begin{cases} 0\text{: false} \\ 1\text{: true} \end{cases}$

Cook Thm. (Turing award)

* *(Circuit) satisfiability problem* (*SAT*):

  $((a \rightarrow b) \vee \neg((\neg a \leftrightarrow c) \vee d)) \wedge \neg b$

  2^n assignments (Clearly, SAT $\in$ NP)

* *3-CNF satisfiability problem* (*3SAT*):

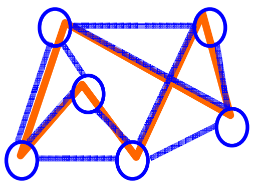
  $(a \vee b \vee c) \wedge (a \vee \neg d \vee e) \wedge (b \vee f \vee a)$

  (conjunctive normal form)

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

  visit each vertex exactly once !

* *The hamiltonian-cycle problem*

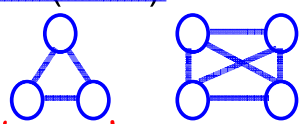


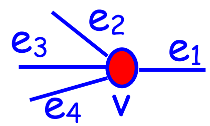
$\Downarrow$

* *The traveling-salesperson problem (TSP)*

* *The clique problem*

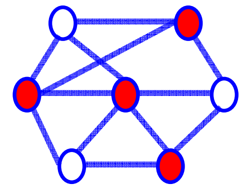
  completely connected subgraph



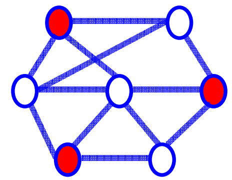


* *The longest "simple" path problem*

* *The vertex-cover problem*



* *The independent set problem*

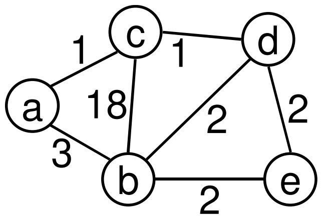


* *The graph coloring problem* min # of colors

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

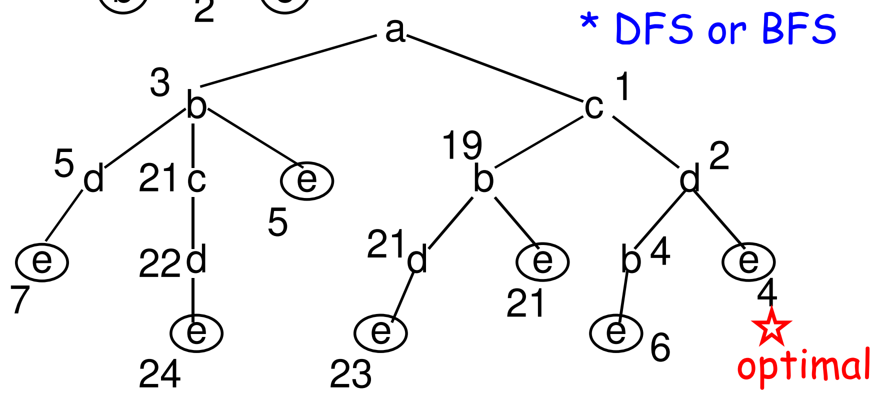
34-4a 34-4x 34-4b

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



: Find a shortest path from a to e.

* DFS or BFS

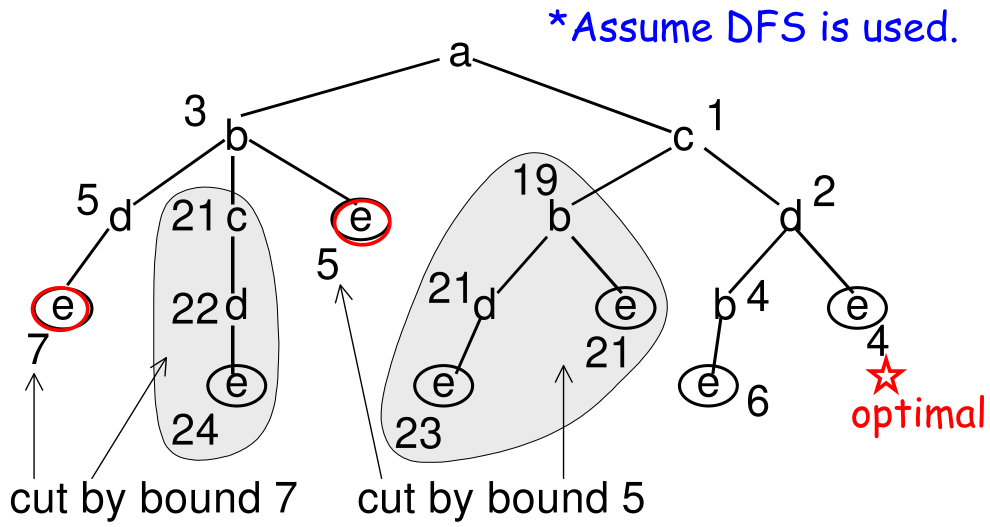


(all simple paths)

34-4z

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

34-5x



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

$|A^*-A|/A^* \le \varepsilon$, e.g. $\varepsilon(n)$ = 0.5 -> error within 50%

where ε is called the *relative error bound*.

啟發式的 (自以為是，一相情願，不負責任)

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

Homework: None.

34-5a

Exercises

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. N

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

(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. y

(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. N?

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

Question 2: Determine whether the following statements are correct of not.

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

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

(3) 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.
y

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

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

* Big CS News in 2001
* Pseudo-Polynomial

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

(7) 3-CNF (the satisfiability problem, in which each clause has exactly three literals) is reducible to 2-CNF problem. poly. reduction???
N?

* SAT $\to_p$ 3CNF ???

Question 3: 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.
N

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

(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.
y

By yourself, using 34-def.

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.

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

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

N (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?