APPENDIX B

Complexity (NP-completeness)

Contents

- Optimization problem
- Decision problem
- Running time
- P: Solvable in polynomial time
- NP: Verifiable in polynomial time
- Reductions
- NP-completeness
- Strongly/weakly NP-complete
- NP-hard

Optimization problems

Def.

An optimization problem is given by:

- Set of instances. Each instance consists of
- a set of feasible solutions and
- a cost function.
- the goal, which is either to minimize or maximize the cost function.

Example: **Shortest path**

- Instance: Graph G=(V,E), s,t∈V, and cost c(e) for each e∈E.
- Solution: Path from s to t.
- Cost: Length (cost) of the path
- Goal: Minimizing cost of the solution.

Optimization problems

Def.

An optimization problem is given by:

- Set of instances. Each instance consists of
- a set of feasible solutions and
- a cost function.
- the goal, which is either to minimize or maximize the cost function.

Example: **Euclidean Traveling Salesman Problem**

- Instance: *n* points in the plane
- Solution: Tour going through all points.
- Cost: Length of the tour.
- Goal: Minimizing length of the tour.

Decision problems

The answer to a decision problem is YES or NO.

Optimization problems can be written as decision problems

Example: **Shortest path**

- Instance: Graph G=(V,E), s,t∈V, and cost c(e) for each e∈E and a number K.
- Solution: Path from s to t.
- Cost: Length (cost) of the path
- Question: Is there a solution of cost at most K?

Three levels of optimization:

1. Optimizing: Find an optimal solution.
2. Evaluating: What is the optimal value?
3. Deciding: Is OPT ≤ K? (≥ K for maximization problem)

Clearly, 1 → 2 → 3. If we can do 1 then we can do 2 and 3.

Usually, the other direction holds as well. That means, if we can solve the decision problem then we can also optimize.

See next page ..

Three levels of optimization:

[3→2] Deciding → Evaluating :

ex. shortest path:
Try, K=1,2,3,… until the answer is YES. (Denote this value by OPT).

[2→1] Evaluating → Optimizing:

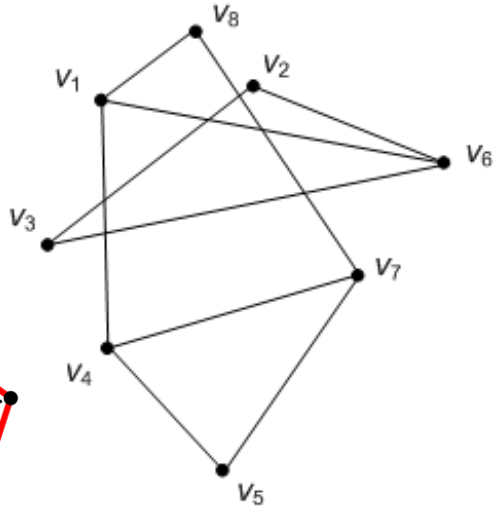
ex. shortest path:

Label edges in arbitrary order: $e_1, e_2 \ldots e_m$.
For i=1…m do
    If Shorestpath(G-e_i) = OPT then remove e_i from G.
The remaining edges will form a shortest s,t path.

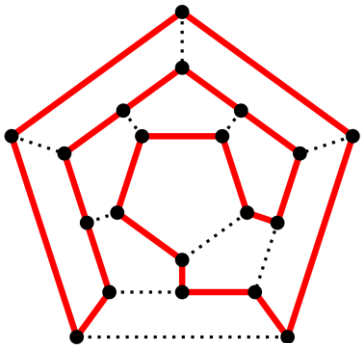
Pure decision problems

Examples

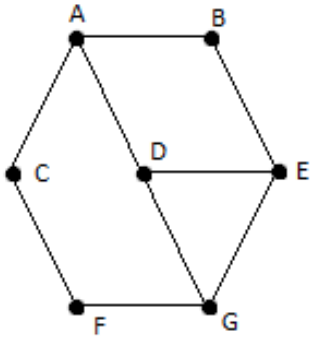
- Is the graph connected?

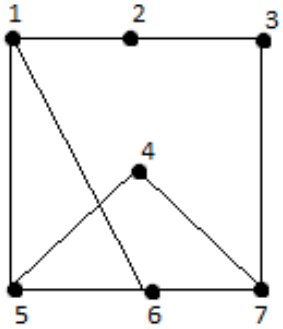


- Is the graph Hamiltonian?



- Are two graphs isomorphic?





- Is 2183300213 a prime number? (yes)

Since any optimization probem has a natural decision version with roughly the same complexity, we may restrict the formal definitions about complexity to decision problems.

Running time

Example:

- Dijkstra' s shortest path algorithm runs in $O(n^2)$ time.
- The TSP problem can solved in $O(n^2 2^n)$ time.

In general, the running time

- is a function of input parameters
- gives a upper bound on the number of `elementary' operations done by the algorithm.

An algorithm for some (decision) problem runs in polyniomal time if for every instance of the problem, the number of steps taken by the algorithm is bounded by some polynomial in the input size.

n	$\log n$	n	$n\log n$	n^2	n^3	2^n
4	2	4	8	16	64	16
8	3	8	24	64	512	256
16	4	16	64	256	4,096	65,536
32	5	32	160	1,024	32,768	4,294,967,296
64	6	64	384	4,094	262,144	$1.84 * 10^{19}$
128	7	128	896	16,384	2,097,152	$3.40 * 10^{38}$
256	8	256	2,048	65,536	16,777,216	$1.15 * 10^{77}$
512	9	512	4,608	262,144	134,217,728	$1.34 * 10^{154}$
1024	10	1,024	10,240	1,048,576	1,073,741,824	$1.79 * 10^{308}$

The Growth Rate of the Six Popular functions

Complexity classes

P: class of all decision problems that can be solved in polynomial time.

- Shortest path
- Graph connectivity
- Max flow
- . . . . .

NP: class of all decision problems that can be verified in polynomial time.

- A decision problem can be verified in polynomial time if for every *yes-instance* of the problem there is a proof (a witness) that can be checked in polynomial time.

Complexity classes

- TSP ∈ NP?
  - Yes! Given an instance and a number K, we can check if OPT≤ K if someone gives us a tour of length K.
- Graph isomorphism ∈ NP?
  - Yes! If we are given the right bijection between the vertices then it is easy to check that the graphs are the same.
- Primality ∈ NP?
  - Yes, but the proof not so easy.

Reductions

A decision problem A is reducable to a decision problem B (notation $A \leq B$) if there is an algorithm such that

- for every instance I of A it produces an instance I' of B
- it runs in polynomial time
- I is yes-instance of A $\leftrightarrow$ I' is yes-instance of B.

Example Hamiltonial Cycle (=A) reduces to TSP (=B)

More generaly, we say A reduces to B if there is an algorithm which solves A and which

- runs in polynomial time
- can use an algorithm for B at unit cost.

NP-completeness

A decision problem A is NP-complete if

- A∈NP
- Every problem in NP can be reduced to A

(more on SAT in Chaper 6)

Theorem (Cook-Levin 1971)
The satisfyability problem (SAT) is NP-complete.

Note that transitivity holds for reductions: If A ≤ B and B ≤ C then A ≤ C.

Corollary: To prove that a probem A is NP-complete it is enough to show that

- A∈NP
- There is some NP-complete problem B that can be reduced to A.

Example:

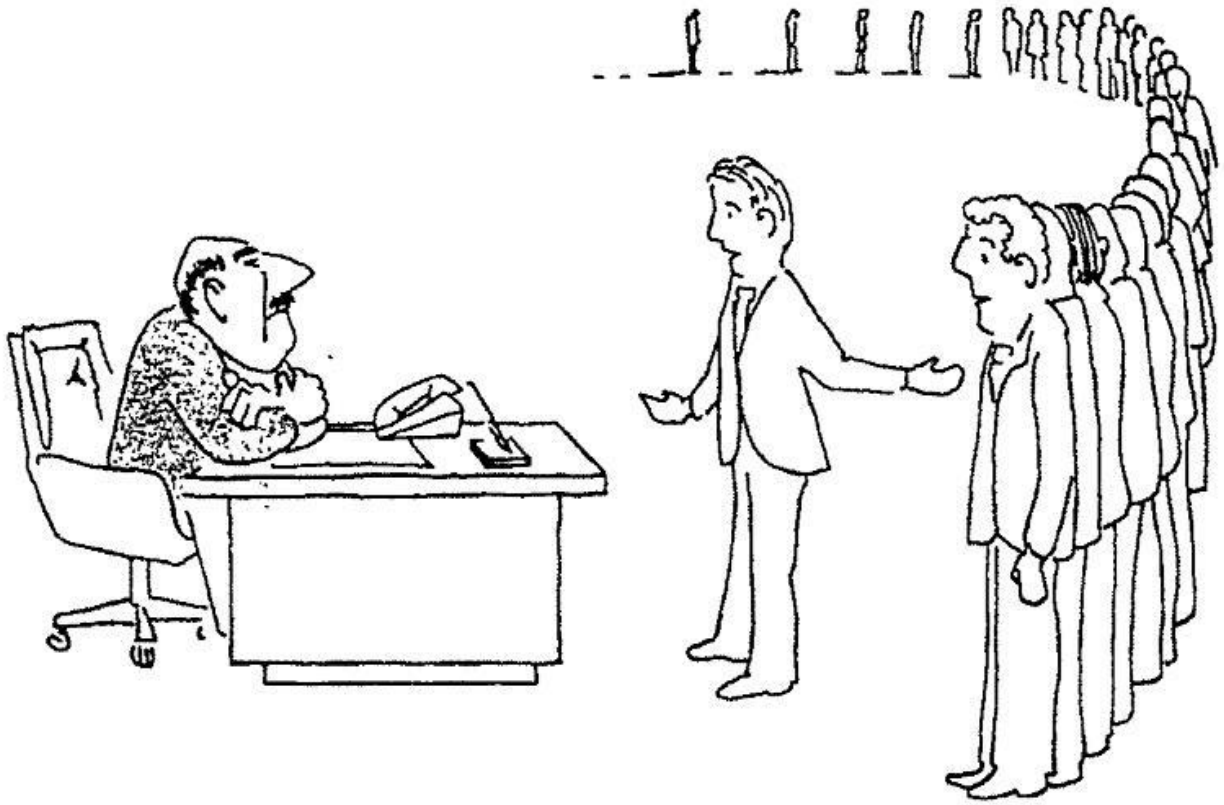
Assume you can find a reduction from SAT to TSP (It can be done). Then, every problem in NP can be reduced to TSP. → TSP is NP-complete.

How to prove that a problem has no effcient algorithm?

“ I could not find an efficient algorithm. Maybe no such algorithm exists.”

NP-completeness gives no proof but at least gives a strong argument.

“ I could not find an efficient algorithm. But neither could these people! “



co-NP

NP was defined for YES-instances, i.e., checking YES is easy.

co-NP: class of all decision problems for which the NO-instances can be verified in polynomial time.

- TSP ∈ co-NP?
  - Strong believe that it is not.
- Graph isomorphism ∈ co-NP?
  - Maybe… maybe not.
- Primality ∈ co-NP?
  - Yes.

strong/weak NP-completeness

A decision problem A is strongly NP-complete if it is NP-complete even when the numbers are poynomially bounded.

Example TSP is strongly NP-complete. (We showed that HC can be reduced to TSP with edge distances 1 and 2.)

Example The Partition problem is NP-complete. However, if we restrict to instance for which all numbers are polynomially bounded then the problem can be solved in polynomial time.

We say that Partition is only weakly NP-complete.

∈ P	NP-complete
Shortest path	Longest path
Matching	3Dimensional Matching
2-coloring	3-coloring
planar 4-coloring	planar 3-coloring
Chinese postman	TSP
LP	ILP
....	....
(short list)	(long list)

NP-complete / NP-hard

Remember:

A decision problem A is NP-complete if:

- $A \in NP$
- Every problem in NP can be reduced to A

A decision problem or optimization problem A is NP-hard if:

- Every problem in NP can be reduced to A.

Note that an optimization problem is NP-hard if its decision version is NP-complete.