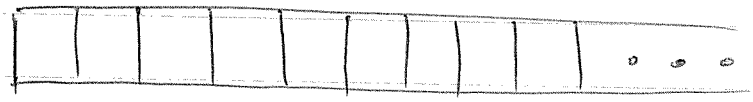
Date / /

GT 6 2017,10,25

A Turing machine (TM) is a mathematical model of computation that defines an abstract machine which manipulates symbols on a strip of tape according to a table of rules. Despite the model's simplicity, given any computer algorithm, a Turing machine can be constructed that is capable of simulating that algorithm's logic.



(May be infinite)

Definition (Decision Problem)

A problem is called a decision problem if its answer is "Yes" or "No".

(•) In the study of computational complexity, an optimization problem is usually formulated as a decision problem.

(•••) Prototype GT problem (decision version): Given n items and two integers d and k ($d>0$ and $k<n$), determine whether $M(d,n) \geq k$ or not.

Deterministic TM

The set of rules prescribes at most one action to be performed for any given situation. (Next step is determined uniquely.)

Non-deterministic TM

The set of rules that prescribes more than one action for a given situation. (Next step has many choices!)

(P: def) A decision problem belongs to P if it can be computed by a polynomial time deterministic TM.

(NP: def) . . . non-deterministic TM.

Time : The number of moves of the TM.

Space : The number of (different) cells on the tape which have been visited by the head during the computation.

PSPACE A decision problem belongs to PSPACE if it can be computed by a polynomial-space deterministic or non-deterministic TM.

$$P \subseteq NP \subseteq PSPACE$$

(***) Whether $P \subsetneq NP$ or $P = NP$ is unknown so far. But, in my opinion proving $P \neq NP$ is of higher probability. ($> \frac{1}{2}$)

(*) Many researchers have claimed the above fact that $P \neq NP$, but so far none of them are "belivable". (In a word, all of them are good explanations but not "proofs".)

Definition (Reducible)

A decision problem A is poly.-time many-one reducible of a decision problem B, denoted by $A \leq_m^P B$, if there is a poly-time computable mapping f from instances of A to instances of B such that A has the "Yes"-answer on x if and only if $f(x)$ has the Yes-answer on $f(x)$.

(•) Many examples are in Graph Theory.

Definition (Completeness)

For a complexity class C, a decision problem is C-complete if (a) A is in C, and

(b) For any decision problem $B \in C$, $B \leq_m^P A$.

(•) A and B are equivalent in C. (In other words.)

==

(••) If B is NP-complete and $B \leq_m^P A$, then A is NP-complete.

(••) A decision problem $\hat{A}$ is co-NP-complete if the complement of $\hat{A}$ is NP-complete.

(否定命題)

Definition (Complement)

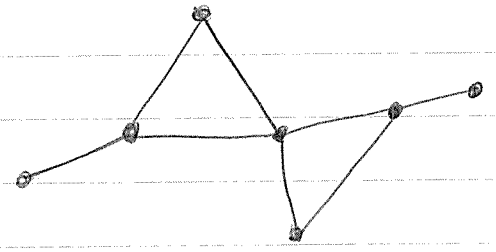
A decision problem B is the complement of another decision problem A if they always obtain different answers on the same input.

The most "popular" NP-complete problem

Definition (Vertex cover) A subset $S \subseteq V(G)$ of a graph G is a vertex cover of G if $\forall e \in E(G)$, $e \cap S \neq \phi$.

The decision version of vertex-cover (VC)

Given a graph $G=(V,E)$ and an integer $k \le |V(G)|$, determine whether there is a set $V' \subseteq V(G)$ of size k such that V' is a vertex cover of G.



Is there a vertex cover of size 3 ? (or 2)

e.g. VC is equivalent to find an independent set of size k. (Can you see it ?)

Exercises Give other examples which are equivalent to VC.

Combinatorial Search problem

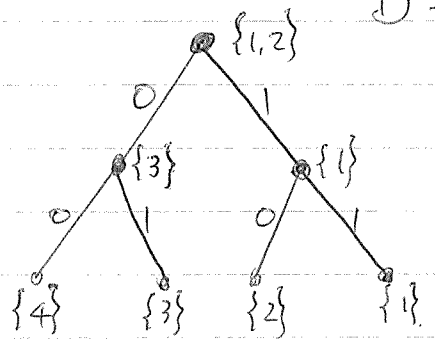
Given a domain D and an integer k, determine whether there is a decision tree of "height" $\le k$ of which each path uniquely determines an object in D.

(*) Notice that in a decision tree of search problems, each internal node corresponds to a Yes-No query and the left and the right sons are the two answers of the query respectively.

(*) No queries on leaves!

e.g. $D = \{1,2,3,4\}$, $k = 2$

(Yes!)



(◦) Each path (from the root to the leave) uniquely determine an object in D (which is positive).

Fact CSP $\in$ PSPACE

Proof. The algorithm of CSP needs $O(k)$ space to store one path of the decision tree if D and k are given.

(◦) You may search for more infos about PSPACE.