Seminar on automata, fall 1996

Quantum computation

Part 2.

IV. Notations and basic concepts.

Here $|\Sigma|$ will denote the cardinality of a set Σ. Notation Σ^* will stand for the set of all finite sequences of the members of Σ. The elements of Σ^* are called the *words* over alphabet Σ. If $x = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_n}$ is a word, then $|x|$ means the length of x which, in this example, is n. The length of the *empty word* ε is defined to be 0. A set $L \subseteq \Sigma^*$ is called a *formal language* over Σ.

A classical *Turing machine* consists of a finite alphabet, states, (potentially infinite) tape and of *transition rules*. More precisely,

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, F).$$

Here Q is a finite set of the states, Σ is the finite alphabet that is expected to contain a special blank symbol $*$, $\delta : Q \times \Sigma \to Q \times \Sigma \times \{L, 0, R\}$ is the transition function, q_0 is the *initial state* and F is the set of the *final states*. Sets Q and Σ are assumed to be disjoint. Here we want the transition function δ to be completely defined.

The *tape* $\mathcal{T}$ is the set of all mappings $T : \mathbb{Z} \to \Sigma \cup Q$ such that $T(i) \neq *$ only for finitely many $i \in \mathbb{Z}$ and that $T(k) \in Q$ for exactly one $k \in \mathbb{Z}$. Symbol $T_t(k)$ is called the contents of the k:th cell at time t. If $T(k) \in Q$, then $T(k+1)$ is called the *currently scanned symbol*.

Intuitively the tape represents a countable sequence of memory cells that continues infinitely in both directions. The unique symbol from Q on the tape works as the state marker and the read-write head position marker at the same time. The precise definition of the tape becomes convinient since each cell has an index to refer with.

The *configuration* of a Turing machine is a mapping in $\mathcal{T}$. It is clear that there are only contably many configurations. A configuration T is a *final* configuration if $T(k) \in F$ for some k.

The *computation* of a classical Turing machine is a countable sequence of configurations. Each computation is completely determined by the first member T_0 which is called *initial configuration*. The initial configuration depends on the *input word*. Let $x = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_n}$ be an input word. Then it is required that $T_0(0) = q_0$, $T_0(j) = \sigma_{i_j}$ when $j \in \{1, \ldots, n\}$ and $T(k) = *$ for all other values of k. So the letters of the input word are initially stored in cells 1, 2, ..., n and the state of the machine is q_0 in the beginning.

When $i \geq 1$, member T_i is obtained from T_{i-1} by the transition rules. We say that configuration T_{i-1} *yields* T_i *in one step* and denote $T_{i-1} \vdash T_i$. Relations $\vdash^k$ "yields in k steps" and $\vdash^*$ "yields" are defined in the obvious way. In this formalism

the computation of a Turing machine does not stop, but reaching a final state is thought to be a sign of complete computation and the result can be read on the tape. We say that the Turing machine *halts* if a final configuration is reached. Otherwise the machine *diverges*.

Assume now temporarily that there are two final states: YES and NO, and that the machine $\mathcal{M}$ always halts. The machine defines then a formal language over Σ, namely

$$\mathrm{L}(\mathcal{M}) = \{x \mid \text{ machine } \mathcal{M} \text{ halts in YES-state with input } x\}.$$

We say that $\mathrm{L}(\mathcal{M})$ is the *language decided by* $\mathcal{M}$. If a formal language is of form $L = L(\mathcal{M})$ for some Turing machine $\mathcal{M}$, it means that there is an *algorithm* which decides whether a given word x belongs to that language or not. The languages that are not of form $L(\mathcal{M})$ are called *undecidable*.

If the machine always either halts in YES-state or diverges, a formal language is again defined:

$$\mathrm{L}(\mathcal{M}) = \{x \mid \text{ machine } \mathcal{M} \text{ halts with input } x\}.$$

Language $\mathrm{L}(\mathcal{M})$ is called the language *accpeted* by machine $\mathcal{M}$.

If a language L is decided by a Turing machine $\mathcal{M}$ then there clearly exists a Turing machine $\mathcal{M}'$ that accepts L. The family of formal languages that are accepted by Turing machines is called *recursively enumerable* languages and denoted by **RE**. Languages accepted by Turing machines are called *recursive* languages. The set of recursive languages is denoted by **R**. A cardinality argument shows that the major part of formal languages are not even recursively enumerable. Turing machines can also be considered as function calculators: input x yields some output word or the machine diverges. Thus a Turing machine specifies a partially defined function $\Sigma^* \to \Sigma^*$. A function that is defined by always halting Turing machine is called *recursive function*.

A *nondeterministic Turing machine*, NTM, is defined in the same way as the classical one, but instead of transition function there is a *transition relation*

$$\delta \subseteq Q \times \Sigma \times Q \times \Sigma \times \{L, 0, R\}$$

for which we assume that for each pair $(q, \sigma) \in Q \times \Sigma$ there is at least one triplet $(q_1, \sigma_1, d) \in Q \times \Sigma \times \{L, 0, R\}$ such that $(q, \sigma, q_1, \sigma_1, d) \in \delta$.

The *computation tree* or shortly the *computation* of a NTM is a labelled tree with the initial configuration as a root entry and the descendant of each vertex defined by the transition relation in the obvious way. The computation of a NTM *halts* if there is a path from the root entry to a vertex labelled with a final configuration. A nondeterministic Turing machine accepts an input word, if there is at least one accepting path in the computation tree, otherwise NTM rejects the input word.

A *probabilistic Turing machine* is also defined as the classical one, but the transition function is replaced with the transition probability function

$$\delta : Q \times \Sigma \times Q \times \Sigma \times \{L, 0, R\} \to [0, 1]$$

that satisfies

$$\sum_{\substack{(q,\sigma,d)\\ \in Q\times\Sigma\times\{L,0,R\}}} \delta(q_1,\sigma_1,q,\sigma,d)=1$$

for any choise of q_1 and σ_1. If the current state is q_1 and symbol σ_1 is being scanned, the value $\delta(q_1,\sigma_1,q,\sigma,d)$ gives the probability to enter state q, overwrite σ_1 with σ and to move to direction d.

The *computation tree* of a probabilistic Turing machine is a tree having vertices labelled with configuration and edges labelled with probabilities. The structure of the tree is determined as an obvious analogue to the nondeterministic case. The *level* of a vertex is the number of edges in the path from the root to the vertex. The *probability of a vertex* is the product of the probabilities on the path from the root to the vertex. A probabilistic Turing machine *halts*, if there is a vertex having non-zero probability labelled with a final configuration.

V. Quantum Turing machine.

Let $\mathcal{C}$ be the linear span of all configurations over the field of complex numbers. Clearly $\mathcal{C}$ is an infinite-dimensional complex vector space with basis $\mathcal{T}$. Each element in $\mathcal{C}$ can be represented as a finite sum

$$\mathbf{c}=\alpha_1 T_1+\alpha_2 T_2+\ldots+\alpha_n T_n,$$

where $T_i\in\mathcal{T}$. We will define an inner product in $\mathcal{C}$ by

$$\langle T_i \mid T_j\rangle=\begin{cases}1, & \text{if } i=j\\ 0 & \text{otherwise,}\end{cases}$$

and extending this in the only possible way. The vector space $\mathcal{C}$ is called the *configuration space* and the unit-length elements in $\mathcal{C}$ are called *superpositions*.

A *Quantum Turing machine* is defined as the earlier ones, but here

$$\delta: Q\times\Sigma\times Q\times\Sigma\times\{L,0,R\}\to\mathbb{C}$$

is the *transition amplitude function*. It is required that δ satisfies the following conditions:

1) For all $(q_1,\sigma_1)\in Q\times\Sigma$ the sum of squared absolute values of the amplitudes leaving the current configuration equals to one:

$$\sum_{\substack{(q,\sigma,d)\\ \in Q\times\Sigma\times\{L,0,R\}}} |\delta(q_1,\sigma_1,q,\sigma,d)|^2=1.$$

2) For all different pairs $(q_1,\sigma_1)\neq(q_2,\sigma_2)\in Q\times\Sigma$ the corresponding sequences of the amplitudes are orthogonal:

$$\sum_{\substack{(q,\sigma,d)\\ \in Q\times\Sigma\times\{L,0,R\}}} \delta(q_1,\sigma_1,q,\sigma,d)\overline{\delta}(q_2,\sigma_2,q,\sigma,d)=0.$$

3) Fixed any $(q_1, \sigma_1, \sigma_1')$, $(q_2, \sigma_2, \sigma_2') \in Q \times \Sigma \times \Sigma$ and $d_1 \neq d_2 \in \{L, 0, R\}$, then

$$\sum_{q \in Q} \delta(q_1, \sigma_1, q, \sigma_1', d_1)\overline{\delta}(q_2, \sigma_2, q, \sigma_2', d_2) = 0,$$

so the sequences of the amplitudes of reaching a state q from different directions must also be orthogonal.

The *computation tree* of a quantum Turing machine is a tree having vertices labelled with configurations and edges labelled with transition amplitudes. The root is labelled with the initial configuration. The *amplitude* of a vertex is the product of the amplitudes on the path from the root to the vertex. The entries of the vertices at level k always determine a superposition

$$\alpha_1 T_{i_1} + \alpha_2 T_{i_1} + \ldots + \alpha_n T_{i_n}, \tag{5-1}$$

Where T_{i_j} are the entries at level k and α_j are the amplitudes of the corresponding vertices. By the requirement 1) (5-1) is of unit length. Thus the computation tree always induces a sequence of superpositions where the first member is the initial configuration and $\mathbf{c}_i$ is determined by $\mathbf{c}_{i-1}$ and by the transition amplitude function when $i \geq 1$.

Let T_1, T_2, T_3, ... be an enumeration of all configurations. We define mapping $U : \mathcal{C} \to \mathcal{C}$ by

$$U(T) = \sum_{i=1}^{k} \alpha_i T_{j_i},$$

where T_{j_i} are configurations that can be obtained from T in one step with amplitude α_i. Mapping U is then extended into a linear mapping in the only possible way and U is called the *time evolution* of the quantum Turing machine.

Lemma V.1. *Let U^* be the adjoint mapping of U. Then $U^*U = I$, so U is injective.*

The outline of the proof. To find U^* we write

$$U(T_i) = \sum_{l=1}^{\infty} \alpha_{li} T_l.$$

Recall that α_{li} is the amplitude of reaching T_l from T_i in one step and that the sum above is actually finite. One checks that

$$U^*(T_i) = \sum_{l=1}^{\alpha} \overline{\alpha}_{il} T_l$$

really is the adjoint mapping of U. Here also the sum is finite. It is easy to check that

$$U^*(U(T_i)) = \sum_{l=1}^{\infty} \Big(\sum_{k=1}^{\infty} \alpha_{ki} \overline{\alpha}_{kl} \Big) T_l.$$

It follows from the requirement 1) that

$$\sum_{k=1}^{\infty} |\alpha_{ki}|^2 = 1,$$

and it remains to check that

$$\sum_{k=1}^{\infty} \alpha_{ki}\overline{\alpha}_{kl} = 0$$

whenever $l \neq i$. This follows from conditions 2) and 3). □

Theorem V.2. *Mapping U is unitary.*

Proof. It remains to show that U is surjective, since then U has an inverse mapping, which, by the previous lemma, has to be U^*. It also suffices to show that each basis vector T_i is in the image of U. Suppose, for the contradiction that $U(\mathbf{c}) \neq T_N$ for any $\mathbf{c}$. Then all the configurations locally looking like T_N also are out of the range.

Let $n \geq 4$ and establish $K = n\,|Q|\,|\Sigma|^n$ configurations $T_{i_1}, \ldots, T_{i_K}$ having cells other than $\{0, 1, \ldots, n\}$ blank and cell n not scanned. These configurations generate a K-dimensional subspace $V \subset \mathcal{C}$. Also $\dim U(V) = K$, because U is injective.

On the other hand, in one step from *any* T_{i_k} one can go into another T_{i_l} or exit the chosen n cells (there are at most $2\,|Q|\,|\Sigma|^n$ configurations to exit into), so we can go into at most $K + 2\,|Q|\,|\Sigma|^n$ configurations. But at least $(n-2)\,|\Sigma|^{n-3}$ of T_{i_k} locally look like T_N, so they cannot be reached from anywhere. Consequently, all images of $T_{i_1}, \ldots, T_{i_K}$ can be represented as

$$U(T_{i_k}) = \sum_{l \in \mathcal{J}} \alpha_l T_{j_l}$$

where $|J| \leq K + |Q|\,|\Sigma|^n - (n-2)\,|\Sigma|^{n-3}$. Therefore

$$K = \dim U(V) \leq K + |Q|\,|\Sigma|^n - (n-2)\,|\Sigma|^{n-3},$$

which is equivalent to $n \leq |Q|\,|\Sigma|^3 + 2$. But n can be chosen arbitrarily large, which rises a contradiction. □

Corollary. *The computation of a quantum Turing machine is reversible.*

The unitarity of the time evolution mapping of a quantum Turing machine is in some sense paradoxical: That the future computation is determined when the initial configuration is known is easy to handle, but here also the *past computation is known*, so we can in principle determine the superposition before the inital one! However, the determinism is violated by the observation.

VI. A finite-dimensional model

We fix two natural numbers $M \leq N$ and assume that a quantum Turing machine never scanns cells other than $\{-N+1, \ldots, N\}$ in forward of backward computation if the length of the input is at most M.

Let $\mathcal{S}$ be the set of those superpositions that will occur when the length of the input is at most M, more presicely, let

$$\mathcal{I} = \{\mathbf{c} \in \mathcal{T} \mid \mathbf{c} \text{ is a configuration of form } \ldots * * * q_0 x_1 \ldots x_m * * * \ldots \text{ with } q_0 \text{ in cell 0 and } m \leq M\},$$

and

$$\mathcal{S} = \bigcup_{\mathbf{c} \in \mathcal{I}} \{U^k(\mathbf{c}) \mid k \in \mathbb{Z}\}.$$

Clearly $|\mathcal{S}| \leq 2N\,|Q|\,|\Sigma|^{2N}$ by the assumption. Next we consider the finite-dimensional subspace $\mathcal{C}'$ generated by $\mathcal{S}$. the following statement is obvious:

Lemma VI.1. *The restriction of U on $\mathcal{C}'$ is a unitary mapping* $\mathrm{dom}(U) \to \mathcal{C}'$.

Let $d = |Q| + |\Sigma|$. Next we will consider a d-dimensional Hilbert space $\mathcal{H}$ that has orthonormal basis

$$B = \{|\,q_0\rangle, |\,q_1\rangle, \ldots, |\,q_f\rangle, |\,\sigma_1\rangle, |\,\sigma_2\rangle, \ldots, |\,\sigma_n\rangle\},$$

where $Q = \{q_1, \ldots, q_f\}$ and $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$. Let $\hat{\mathcal{H}}$ be a $2N+1$-fold tensor product

$$\hat{\mathcal{H}} = \bigotimes_{i=-N}^{N} \mathcal{H}$$

and $e : \mathcal{C}' \to \hat{\mathcal{H}}$ be the embedding defined in the obvious way. Then a unitary mapping $\hat{U}$ whose domain is the image of $\mathcal{S}$ in $\mathcal{C}'$ can be defined by $\hat{U}(e(x)) = e(U(x))$ and extended into a unitary mapping $\tilde{U} : \hat{\mathcal{H}} \to \hat{\mathcal{H}}$. Space $\hat{\mathcal{H}} = \hat{\mathcal{H}}_N$ is called a *finite model* of a quantum Turing machine.

Let

$$\mathbf{c} = \sum_{i=1}^{m} \alpha_i T_{j_i} \in \mathcal{C}' \tag{5-2}$$

be a superposition and $\mathbf{c}' = e(\mathbf{c})$. If $\mathbf{c}'$ can be represented as a tensor product of $2N+1$ vectors in $\mathcal{H}$, we say that the superposition is *decomposable*. Otherwise the superposition is *entangled*.

The *observation of the superposition* (5-2) yields configuration T_{j_i} with probability $|\alpha_i|^2$. The superposition after the observation is T_{j_i}, so all other branches of the computation tree are destroyed.

The observation described above corresponds to a measurement of an observable of form $\lambda_1 P[\varphi_1] + \ldots \lambda_m P[\varphi_m]$, where $\lambda_1 < \lambda_2 < \ldots \lambda_m$ and for each φ_i are the basis vectors (configurations) of $\hat{\mathcal{H}}$. Measured value λ_i indicates that the basis vector φ_i was observed.

Example (degenerate observables): Let $\{\varphi_1, \varphi_2\}$ be an orthonormal basis of $\mathcal{H}_2$ and $A = 1 \cdot P[\varphi_1] + 2 \cdot P[\varphi_2]$ an observable. Recall that $P[\varphi]$ is the projection onto the

one-dimensional subspace spanned by a uniti-length vector φ. If $\psi = c_1\varphi_1 + c_2\varphi_2$ is a state vector of system to be observed, then value 1 will be obtained with probability $|c_1|^2$ and value 2 with probability $|c_2|^2$. The post-observation state vectors are φ_1 and φ_2 respectively. Consider then the compound system $\mathcal{H}_2 \otimes \mathcal{H}_2$ with orthonormal basis

$$\{\varphi_1 \otimes \varphi_1, \varphi_1 \otimes \varphi_2, \varphi_2 \otimes \varphi_1, \varphi_2 \otimes \varphi_2, \}$$

and an observable

$$B = A \otimes I = 1 \cdot P[\varphi_1 \otimes \varphi_1] + 1 \cdot P[\varphi_1 \otimes \varphi_2] + 2 \cdot P[\varphi_2 \otimes \varphi_1] + 2 \cdot P[\varphi_2 \otimes \varphi_2].$$

Observable B is now *degenerate*, i.e. it has multiple eigenvalues. The spectral projections are now given by

$$E^B(\{1\}) = P[\varphi_1 \otimes \varphi_1] + P[\varphi_1 \otimes \varphi_2]$$

and

$$E^B(\{2\}) = P[\varphi_2 \otimes \varphi_1] + P[\varphi_2 \otimes \varphi_2].$$

In a state determined by a unit-lenght vector

$$\psi = c_1\varphi_1 \otimes \varphi_1 + c_2\varphi_1 \otimes \varphi_2 + c_3\varphi_2 \otimes \varphi_1 + c_4\varphi_2 \otimes \varphi_2$$

we have

$$E^B_{P[\psi]}(\{1\}) = |c_1|^2 + |c_2|^2$$

and

$$E^B_{P[\psi]}(\{2\}) = |c_3|^2 + |c_4|^2 .$$

The post-measurement state vector will be

$$\frac{1}{\sqrt{|c_1|^2 + |c_2|^2}}\big(c_1\varphi_1 \otimes \varphi_1 + c_2\varphi_1 \otimes \varphi_2\big)$$

or

$$\frac{1}{\sqrt{|c_3|^2 + |c_4|^2}}\big(c_3\varphi_2 \otimes \varphi_1 + c_4\varphi_2 \otimes \varphi_2\big)$$

depending on which value was observed. So, essentially observable B corresponds to an observation on the first component, but the second component remains in superposition.

For the *general observation* of cells indexed with set $I \subseteq \{-N, \dots, N\}$ of superposition (5-2) we define an equivalence relation $\sim_I \subseteq \mathcal{T} \times \mathcal{T}$ by

$$T_i \sim_I T_j \iff T_i(k) = T_j(k) \text{ for all } k \in I.$$

and divide configurations in (5-2) into equivalence classes. The *probability* of an equivalence class $[T_{j_k}]$ is defined by

$$P([T_{j_k}]) = \sum_{T_{i_j} \in [T_{j_k}]} |\alpha_j|^2 .$$

The *observation of cells I* yields equivalence class $[T_{j_k}]$ with probability $P([T_{j_k}])$. The post-observation superposition is

$$\frac{1}{\sqrt{\sum_{T_j \in [T_{j_k}]} |a_j|^2}} \sum_{T_j \in [T_{j_k}]} \alpha_j T_j$$

provided the class $[T_{j_k}]$ was observed.

VII. The reversibility of the computation.

A quantum Turing machine is always reversible, but the classical ones are not, in general. In order to conclude that everything we can do with a TM can also be done with a quantum Turing machine, we have to show that the computation can be forced to be reversibe. The device introduced by Bennet [PART 1] is modified here.

A *two-tape Turing machine* consists of two tapes $\mathcal{T}^{(1)}$, $\mathcal{T}^{(2)}$ and of a sixtuple

$$(Q, \Sigma, \Gamma, \delta, q_0, F),$$

where Q is the set of states, Σ and Γ are the alphabets of the first and the second tape respectively. It is required that the unique symbol from Q on both tapes equal at each step. The function

$$\delta : Q \times \Sigma \times \Gamma \to Q \times \Sigma \times \Gamma \times \{L, 0, R\}^2$$

is again called the transition function. The action of the two-tape Turing machine is defined in the obvious way. The *configuration* of a two-tape Turing machine is now and ordered pair $(T^{(1)}, T^{(2)}) \in \mathcal{T}^{(1)} \times \mathcal{T}^{(2)}$ such that if $T^{(1)}(k_1) \in Q$ and $T^{(2)}(k_2) \in Q$, then $T^{(1)}(k_1) = T^{(2)}(k_2)$. Again there is only countably many configurations.

Definition VII.1. A Turing machine $\mathcal{M}$ is (logically) *reversible*, if each configuration uniquely determines the previous one.

Theorem VII.1. *For each one-tape Turing machine there exists a reversible two-tape Turing machine that simulates the original one on the first tape.*

The outline of the proof. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a one-tape machine. We will describe a two-tape machine

$$M' = (Q', \Sigma, (Q \times \Sigma \times \{L, 0, R\}) \cup \{*\}, \delta', q_0, F),$$

where $Q' = Q \cup Q_w \cup Q_r$, $Q_w = \{q_w \mid q \in Q\}$ and $Q_r = \{q_r \mid q \in Q\}$. Sets Q, Q_w and Q_r are assumed to be pairwise disjoint. Transition rules are given in five groups: For each pair $(q, \sigma) \in Q \times \Sigma$ we define

1° $\delta'(q, \sigma, \mathbf{x}) = (q_w, \sigma, \mathbf{x}, 0, R)$ for any $\mathbf{x} \in Q \times \Sigma \times \{L, 0, R\} \cup \{*\}$.

2° $\delta'(q_w, \sigma, *) = (q_1, \sigma_1, (q, \sigma, d_1), d_1, 0)$, where q_1, σ_1 and d_1 are determined by $\delta(q, \sigma) = (q_1, \sigma_1, d_1)$.

3° $\delta'(q_w, \sigma, (p, \eta, d)) = (q_r, \sigma, (p, \eta, d), 0, L)$ for each $(p, \eta, d) \in Q \times \Sigma \times \{L, 0, R\}$.

4°

$$\delta'(q_r, \sigma, (p, \eta, d)) = \begin{cases} (p, \eta, *, 0, L) & \text{if } \delta(p, n) = (q, \sigma, d) \\ (q, \sigma, (p, \eta, d), d, 0) & \text{otherwise} \end{cases}$$

5° $\delta'(q_r, \sigma, *) = (q, \sigma, *, 0, 0)$.

It is a tedious but straightforward task to check that the machine M' is reversible: If the machine is in a state $q \in Q$, there are two possibilities for the previous state, case 2° (Q_w) or cases 4° and 5° (Q_r). If $\mathbf{x} = *$, then we have 5°, and the previous configuration is easy to determine (uniquely). If $\mathbf{x} \neq *$, we have to distinguish between 2° and 4°. But this is easy: Let $\mathbf{x} = (p, \eta, d)$ and σ' the next symbol on the first tape to direction $-d$ from the head. Then chech whether $\delta(p, \eta) = (q, \sigma', d)$. If so, the current configuration was obtained via 2°, otherwise via 4°. The previous configuration in again unique.

In state $q_w \in Q_w$ the only possibility is 1°, and the previous configuration is clearly unique.

In a state $q_r \in Q_r$ we must consider 3° and 4°. To separate these we look at the symbol on the right hand side of the head on the second tape. If the symbol was $*$, 4° has occured, otherwise 3°. In the case 3° the previous configuration is easy to tell, so consider 4°. To reconstruct the previous configuration we find out $\delta(q, \sigma) = (q', \sigma', d)$, move the head of the second tape to the right and print (q, σ, d) there. Then we print σ' on the first tape and set the machine in the state q'_r.

The simulation of the original machine is easy: The input word is written on the first tape and the second tape is assumed to be empty. With this assumption case 3° never occurs and the machine works in time $2f(n)$ if the original one worked in time $f(n)$. □