Problem 1

Answer. Assume the two-dimensional TM is $M$. It runs in time $T(n)$. The positions that $M$ could have visited on the 2-dimension tape are contained in the $T(n) \times T(n)$ square. Now we construct another 1-dimension TM $M_1$.

$M_1$ has 2 tapes $t_1, t_2$.

We will map the two-dimensional tape of $M$ to the tape $t_1$ of $M_1$, similar as what we did in class to map the set of rational numbers to the set of natural numbers. Specifically, the $x$th position in $t_1$ corresponds to the position $(p,q)$ on the 2-dimensional tape of $M$. For any position $(p,q)$ on the 2-dimension tape, assume $(p,q)$ is on the diagonal of a square. The number of positions on the diagonal is $d = p + q + 1$, thus we set $x = \sum_{i=1}^{d-1} + p$.

The tape $t_2$ is used to record the current position of the tape head $s$ of $M$, the current position of the tape head $s_1$ for $t_1$ of $M_1$, a number $c$ which indicates how many steps are left from the current position of $s_1$ to its destination $x$. Also $t_2$ is used as a working tape for some intermediate computations.

$M_1$ simulates one step of $M$ as follows. First update the symbol at the current position, and compute the new position $(p,q)$ of head $s$ of $M$ according to the transition function. Then we can compute the corresponding position $x$ on $t_1$. After we get $x$ we compute the distance $c$ between $x$ and the current position of $s_1$. Then $M_1$ starts to move $s_1$ to its destination. Each time $s_1$ moves 1 step, decrease $c$ by 1. $M_1$ keeps moving $s_1$ until $c = 0$.

One move at $(p,q)$ in the 2-dimension tape corresponds to at most $d + 1 = O(T(n))$ steps on the 1-dimension tape $t_1$. Since $M$ runs in time $O(T(n))$, $M_1$ runs in time $O(T(n)^2)$.

Problem 2

Answer. Consider an arbitrary RAM TM $M$ with running time $T(n)$. We give a standard TM $M'$ which can simulate $M$ with running time $O(T(n)^2)$.

$M$ and $M'$ have the following difference. $M'$ does not have the array $A$. But it has an additional tape $A'$ which is used to simulate $A$. The tape $A'$ contains a list of elements $A'[0], A'[1], A'[2], \ldots, A'[l]$. Each $A'[i]$ contains an address and a symbol.
Whenever $M'$ enters the state $q_{\text{access}}$, it does the following. If its address tape contains $\downarrow i \downarrow R$, then $M'$ will scan $A'$ to find the address $\downarrow i \downarrow$. When it finds $\downarrow i \downarrow$ in $A'[i^*]$, it reads the the symbol in $A'[i^*]$, writing it next to $R$ on the address tape. If the address tape contains $\downarrow i \downarrow W\sigma$, then $M'$ also scans the tape $A'$. When it retrieves the address $\downarrow i \downarrow$ in $A'[i^*]$, the symbol in $A'[i^*]$ is rewritten to be $\sigma$. If $M'$ does not retrieve the address $\downarrow i \downarrow$ by reading all the element in $A'$, then $M'$ sets $t = t + 1$ and creates a new element $A'[t]$ which contains the address $\downarrow i \downarrow$ and the symbol $\sigma$.

The time complexity for $R$ and $W$ operations are both $O(T(n))$ because we only need to go through all the elements in $A'$ for once and there are at most $T(n)$ visited cells in $A'$. As $M$ runs in the same way as $M'$ does except for the $R$ and $W$ operations, the time complexity of $M'$ is $O(T(n)^2)$.

Problem 3

Answer. Reduce the function UC to $T$.

Assume we have a TM $R$ that decides $T$. Now we are going to construct another TM $S$ that computes UC.

$S$ works in the following way. Assume the input is $\langle M \rangle$. Then $S$ first constructs another TM $M_1$.

Assume the input for $M_1$ is $x$. $M_1$ is described as follows.

- If $x \neq 001$ then $M_1$ accepts.
- Else, meaning $x = 001$, run $M(\langle M \rangle)$. If $M(\langle M \rangle) = 1$, then $M_1$ accepts; if $M(\langle M \rangle) = 0$, $M_1$ rejects.

Then $S$ uses $R$ to get $R(\langle M_1 \rangle)$. Set $S(\langle M \rangle) = 1 - R(\langle M_1 \rangle)$.

Here is the explanation. If the input $\langle M \rangle$ is s.t. $M(\langle M \rangle) = 1$, $R$ will accept $\langle M_1 \rangle$ as $M_1$ accepts every string. So $S(\langle M \rangle) = 0$. On the other hand, if $M(\langle M \rangle) = 0$ or $M$ does not halt on $\langle M \rangle$, then $M_1$ does not accept $x = 001$. So $\langle M_1 \rangle$ is not in $T$. Thus $R(\langle M_1 \rangle) = 0$ and $S(\langle M \rangle) = 1$.

According to the definition of UC, $S$ computes UC successfully. However we know that UC is uncomputable. This means that $T$ is not decidable.

Problem 4

Answer. (b) The TM $M$ that decides TRIANGLEFREE: Given a graph $G = (V,E)$, $M$ checks every triple of vertices to see whether they are all distinct and connected. If yes then $M$ rejects, otherwise $M$ accepts.
As $M$ will try every triple of vertices, it can find the triangle if $G$ does have one. And it will not find it if $G$ does not have one.

Assume the input length is $n$. The total number of triple of vertices is $\binom{|V|}{3}$ which is a polynomial of $n$. As checking the connectivity between two vertices can be done in polynomial time, the time complexity of $M$ is a polynomial of $n$. So TRIANGLEFREE $\in \mathsf{P}$.

(c) The TM $M$ that decides $\mathsf{BIPARTITE}$: Given a graph $G = (V,E)$, $M$ conducts a depth first search on $G$ to divide $V$ into to sets $A$ and $B$. Every vertex is set to be unvisited at the beginning. At the start, $M$ visits an arbitrary vertex $u$, setting $u$ to be visited, putting it into $A$. During the search, whenever visiting an unvisited vertex $w$ from a visited vertex $v$, $M$ does the following. First, $M$ set $w$ to be visited. Then $w$ is put into $A$ if $v$ is in $B$ and $w$ is put into $B$ otherwise. At last, $M$ checks every visited vertices $x$ adjacent to $w$, if $w$ and $x$ are in the same set, then $M$ rejects and halts. After the search, if $M$ still does not reject, it accepts.

Now we prove the correctness of the algorithm.

If $M$ accepts, then $M$ has divided $V$ into $A$ and $B$. We claim that there is no edge between vertices in $A$. Suppose there is an edge $(u,v)$ in $A$. Assume $u$ is visited before $v$ being visited. When $M$ visits $v$, it will then check every visited vertex adjacent to $v$, finding that $u$ and $v$ are in the same set. So $M$ will reject, contradicting that $M$ accepts. So the claim holds. Similarly there is no edge between vertices in $B$. Thus $G \in \mathsf{BIPARTITE}$.

On the other hand, if $M$ rejects, there exists a circle in $G$ which has an odd number of vertices. This circle cannot be bi-partied. So $G \notin \mathsf{BIPARTITE}$.

As the depth first search has time complexity $O(|V| + |E|)$, the overall running time of $M$ is polynomial of its input length. So $\mathsf{BIPARTITE} \in \mathsf{P}$. \hfill \qed