Exercise 1

Solution

a)
It is clear that:

$$T_1 \in O(n)$$

$$T_2 \in O(n^2)$$

Given the proper definition of the function class $O(g)$ ("Big O") of a function $g : \mathbb{N} \to \mathbb{R}$ as follows;

$$O(g) := \{f : \mathbb{N} \to \mathbb{R} \mid \exists c \in \mathbb{R}, n_0 \in \mathbb{N}, f(n) \leq cg(n), \forall n > n_0\}$$

b)
We assume $n \neq 0$, for obviously $T_1(0) = T_2(0)$, so T_2 is more efficient if:

$$T_1(n) > T_2(n) \Leftrightarrow 625n > n^2 \Leftrightarrow 625 > n$$

And the opposite is true for $625 < n$, for $n = 625$ the algorithms perform equally.

Exercise 2

Solution

Algorithms $T_1, ..., T_5, T_7, ..., T_9$ are trivial. for T_6, if $n > 1$ then $T_6 \in O(n \log^2(n))$. For T_{10}, notice that:

$$T(2) = 2T(1) + 2 \in O(1),$$

$$T(3) = 2T(2) + 2 = 2[2T(1) + 2] + 2$$

$$\vdots$$

$$T(n) = 2^n[T(1) + 1] + 2 \in O(2^n).$$

Exercise 3

Solution

a)

```cpp
int a = 0, b = 0;
for (i = 0; i < n; i++) {
    a = a + i;
}
for (j = 0; j < m; j++) {
    b = b + j;
}
```

The algorithm operates in $O(n) + O(m)$, the first `for` grows linearly with n and the second one grows with m.

b)

```cpp
float what2(int *arr, int n) {
    int a = 0;
    for (int i = 0; i < n; i++) {
        if(arr[i] > 10) {
            for (int j = 0; j < n; j++) {
                a += n / 2;
            }
        } else {
            printf("ok :(")
        }
    }
}
```

The algorithm has 2 `for's` with n operations each, so it is at least $\in O(n^2)$, it also does check