# 1 Code

```
def get_critical_events(A, t):
    n_events = 0
    n = len(A)
    for i in range(n):
        for j in range(i + 1, n):
            if A[i] > t * A[j]:
                n_events += 1

    return n_events
```

## 1.1 Time complexity analysis

First, The initialization of *n_events* and $n$ takes $O(1)$.

```
for i in range(n):
    for j in range(i + 1, n):
```

1. The outer loop runs $n$ times since it need to go overall the element in the array.
2. The inner loop runs $n - i - 1$ times since the constraint $j > i$.

$$
\begin{aligned}
T(n) &= \sum_{i=0}^{n-1} \sum_{j=i+1}^{n-1} 1 \\
&= \sum_{i=0}^{n-1} (n - i - 1) \\
&= \sum_{i=0}^{n-1} n - \sum_{i=0}^{n-1} i - \sum_{i=0}^{n-1} 1 \\
&= n^2 - \frac{n(n-1)}{2} - n \\
&= \frac{n^2 - n}{2}
\end{aligned}
$$

For each iteration, the comparison between $A[i]$ and t*$A[j]$ and increment of counter take $O(1)$. Therefore, the total number of the function is $O(n^2)$.