Эмпирический анализ гибридного алгоритма MERGE+INSERTION SORT Данные к заданию

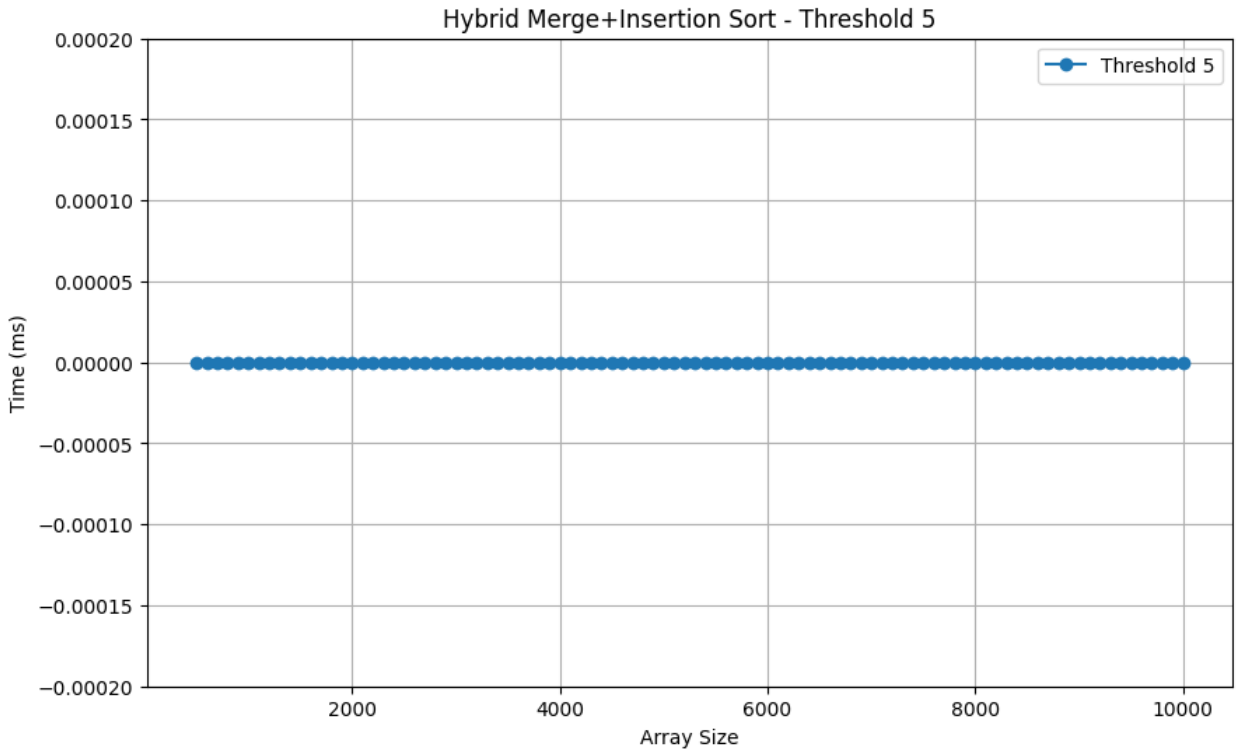
```
import matplotlib.pyplot as plt
sizes = [x for x in range(500, 10001, 100)]

thresholds = [5, 10, 20, 30, 50]
num_thresholds = len(thresholds)

H_random_times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1,
1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1,
2, 1, 2, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1,
1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1,
2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1,
1, 1, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 1,
2, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 1,
1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1]

H_reverse_times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1,
1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,
```

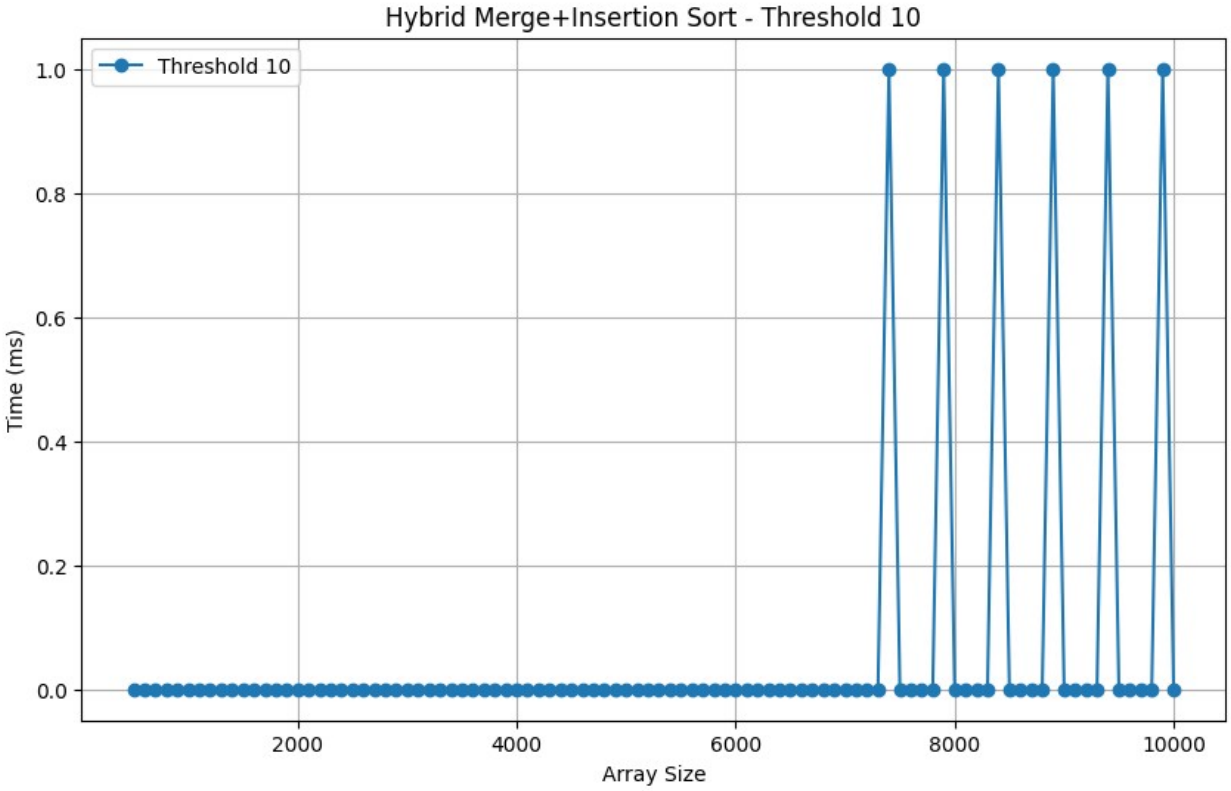
```
0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

H_nearly_sorted_times = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0,
0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0,
1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0]
```

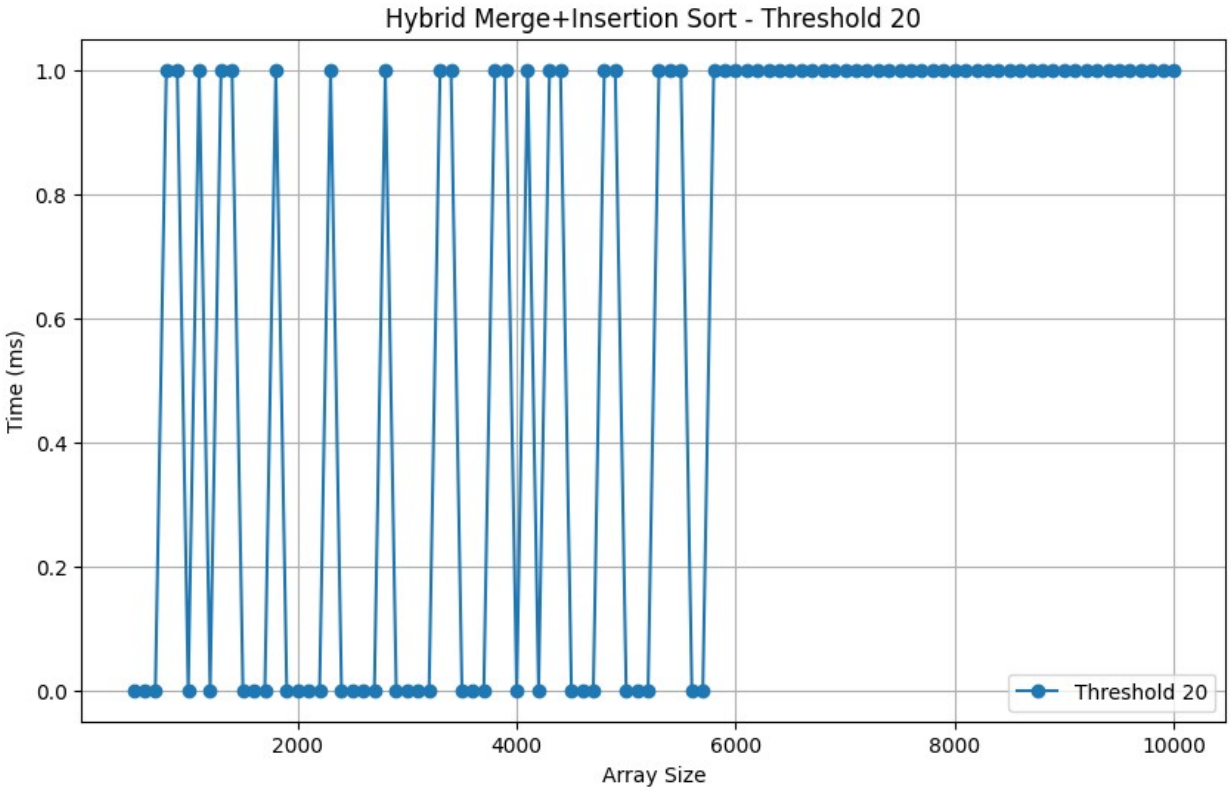
График 1: Построение графика для случайных данных

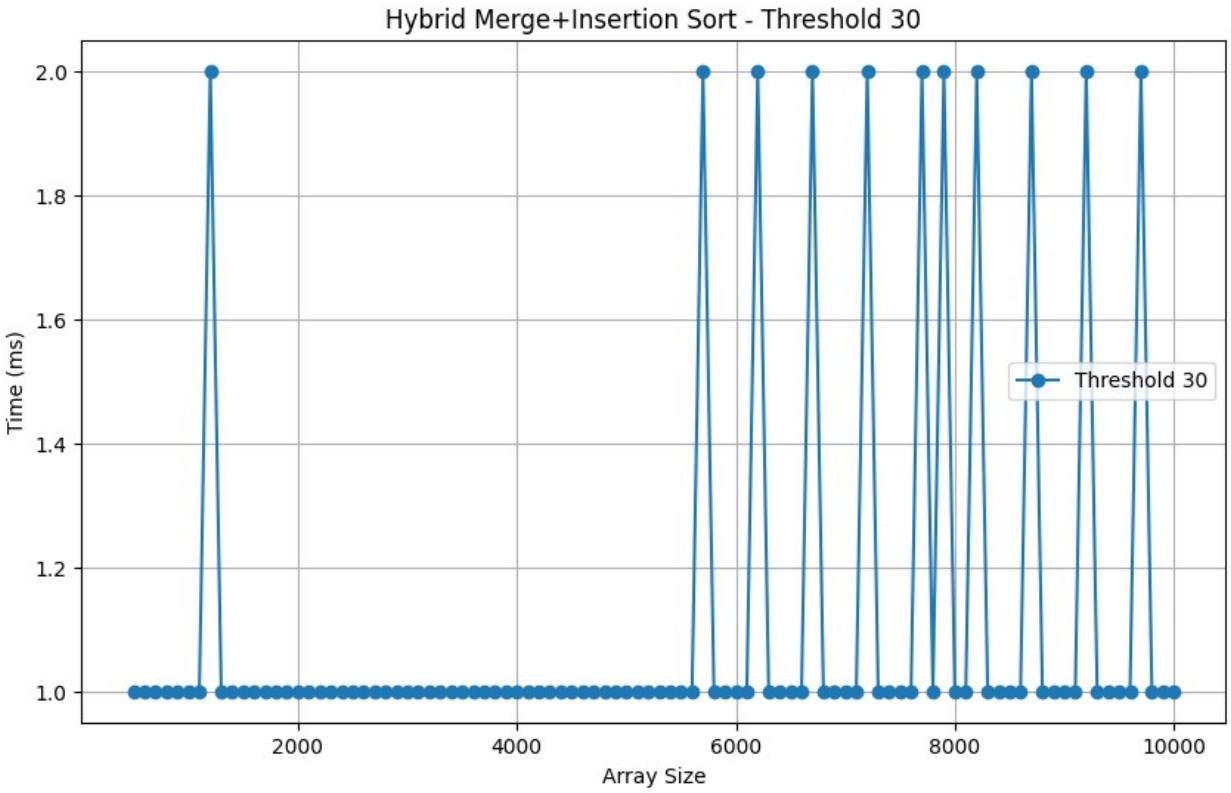
```
# Разделение данных по порогам
values_per_threshold = len(H_random_times) // num_thresholds
hybrid_times_by_threshold = [
    H_random_times[i * values_per_threshold:(i + 1) *
values_per_threshold]
    for i in range(num_thresholds)
]

for i, times in enumerate(hybrid_times_by_threshold):
    plt.figure(figsize=(10, 6))
    if thresholds[i] == 5:
```

```
        plt.ylim(-0.0002, 0.0002)
    plt.plot(sizes, times, marker='o', label=f'Threshold
{thresholds[i]}')
    plt.xlabel('Array Size')
    plt.ylabel('Time (ms)')
    plt.title(f'Hybrid Merge+Insertion Sort - Threshold
{thresholds[i]}')
    plt.legend()
    plt.grid(True)
    plt.show()
```




Hybrid Merge+Insertion Sort - Threshold 10
Threshold 10
Time (ms)
Array Size
0.0
0.2
0.4
0.6
0.8
1.0
2000
4000
6000
8000
10000


Hybrid Merge+Insertion Sort - Threshold 20
Threshold 20
Time (ms)
Array Size
0.0
0.2
0.4
0.6
0.8
1.0
2000
4000
6000
8000
10000



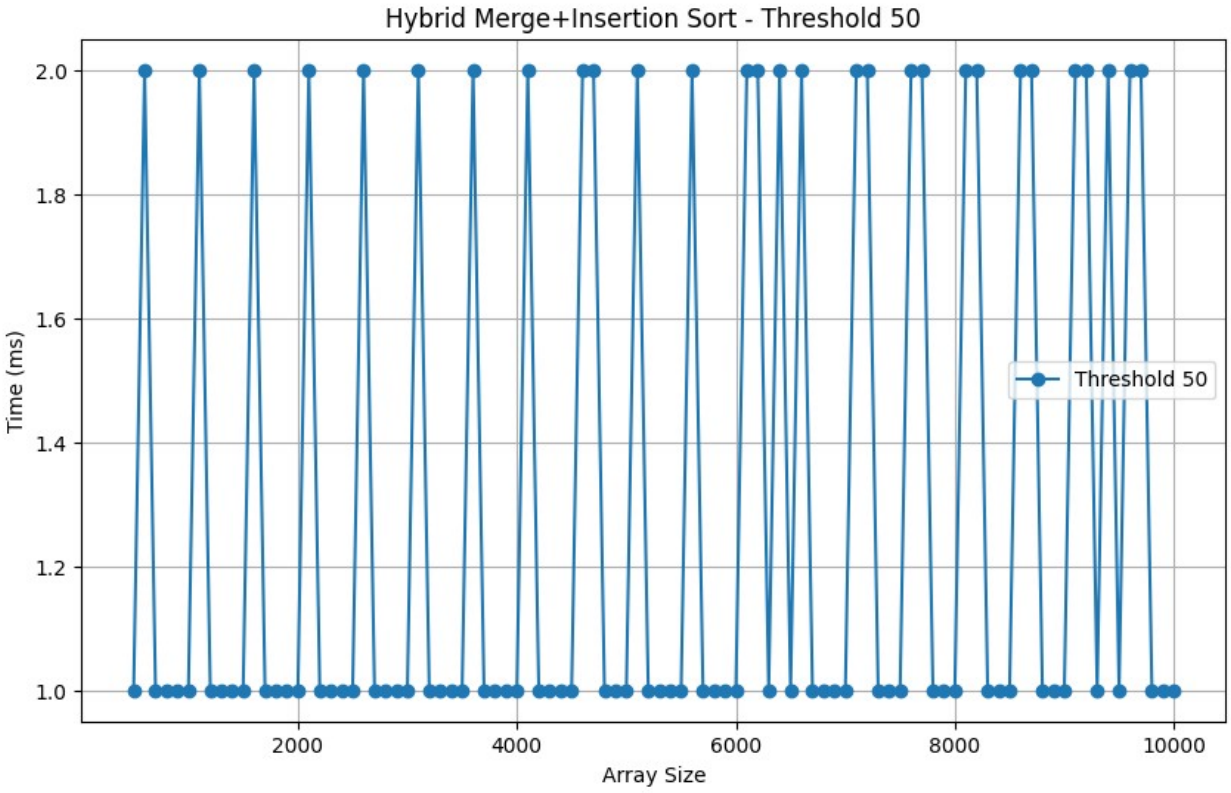
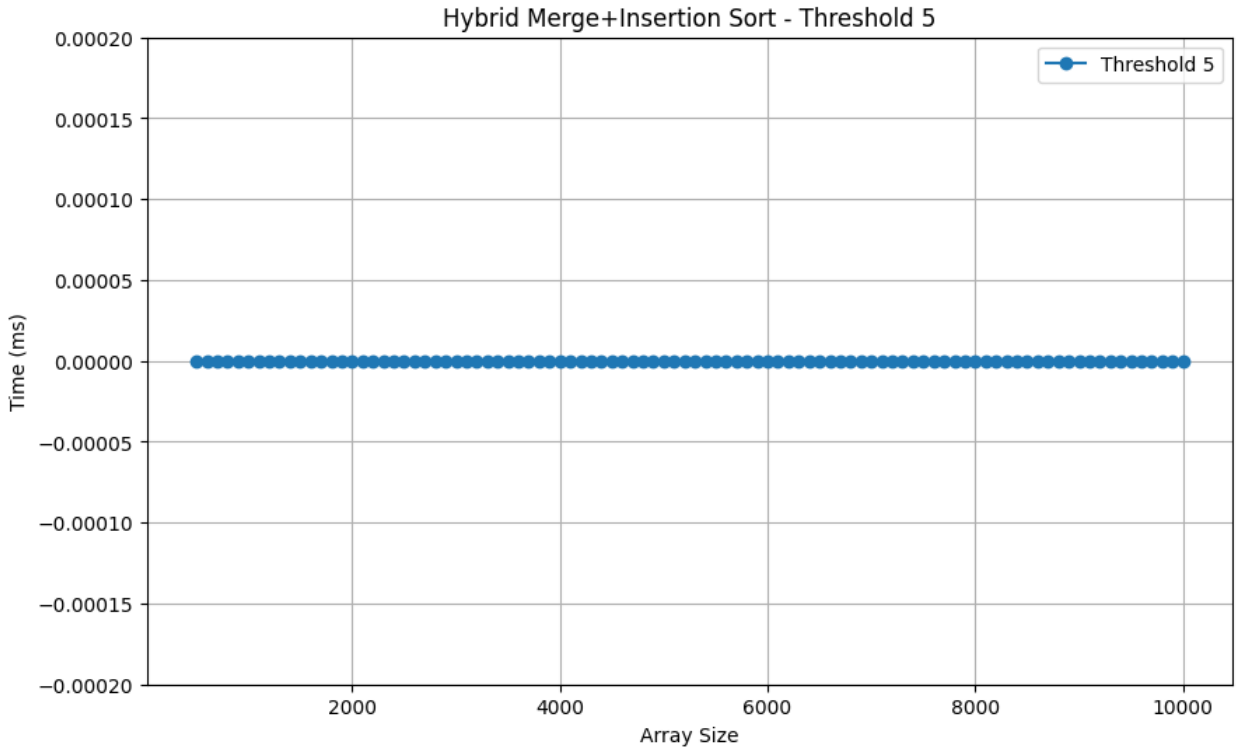
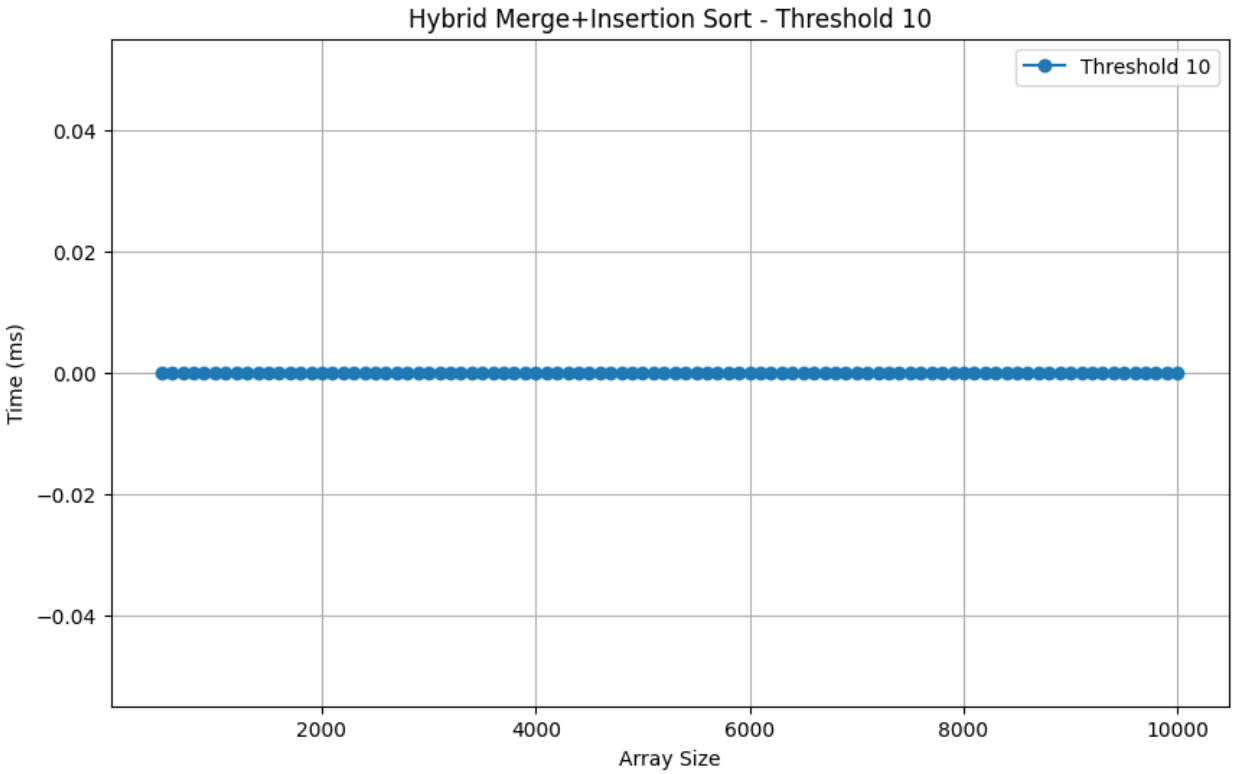
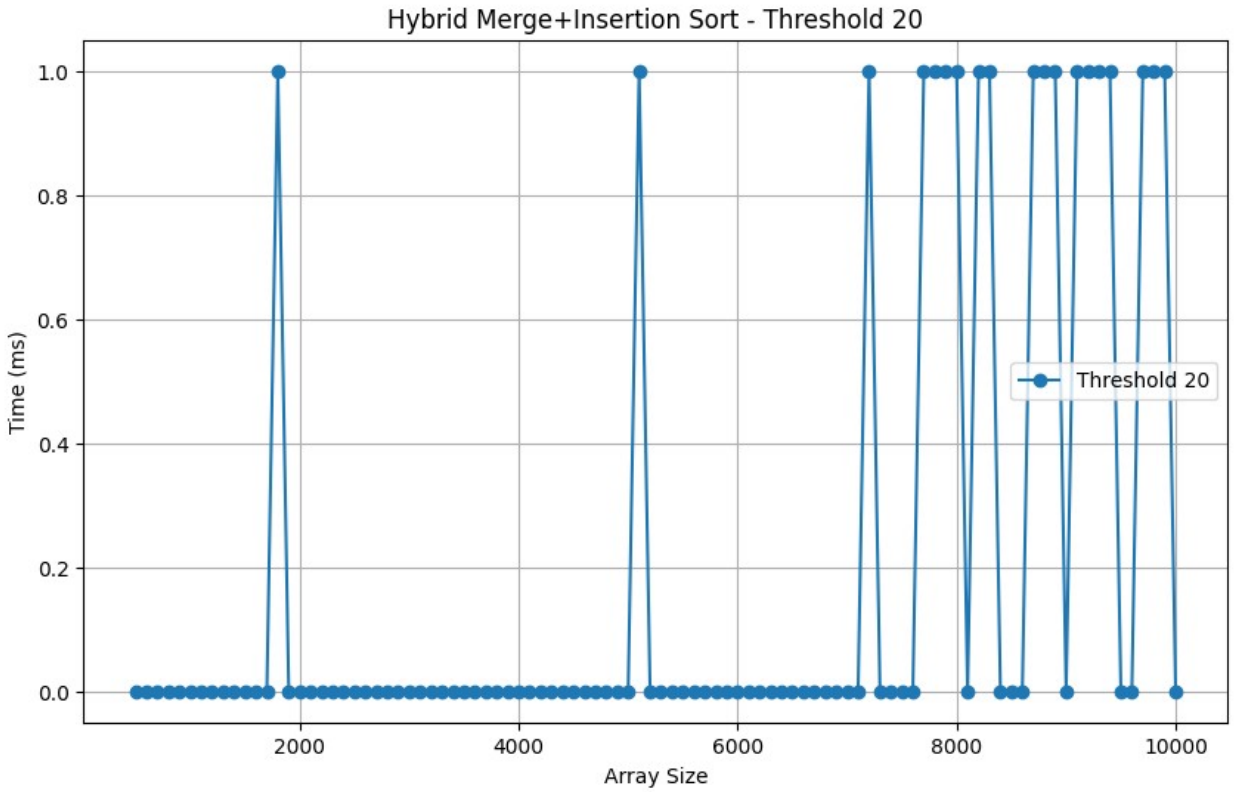


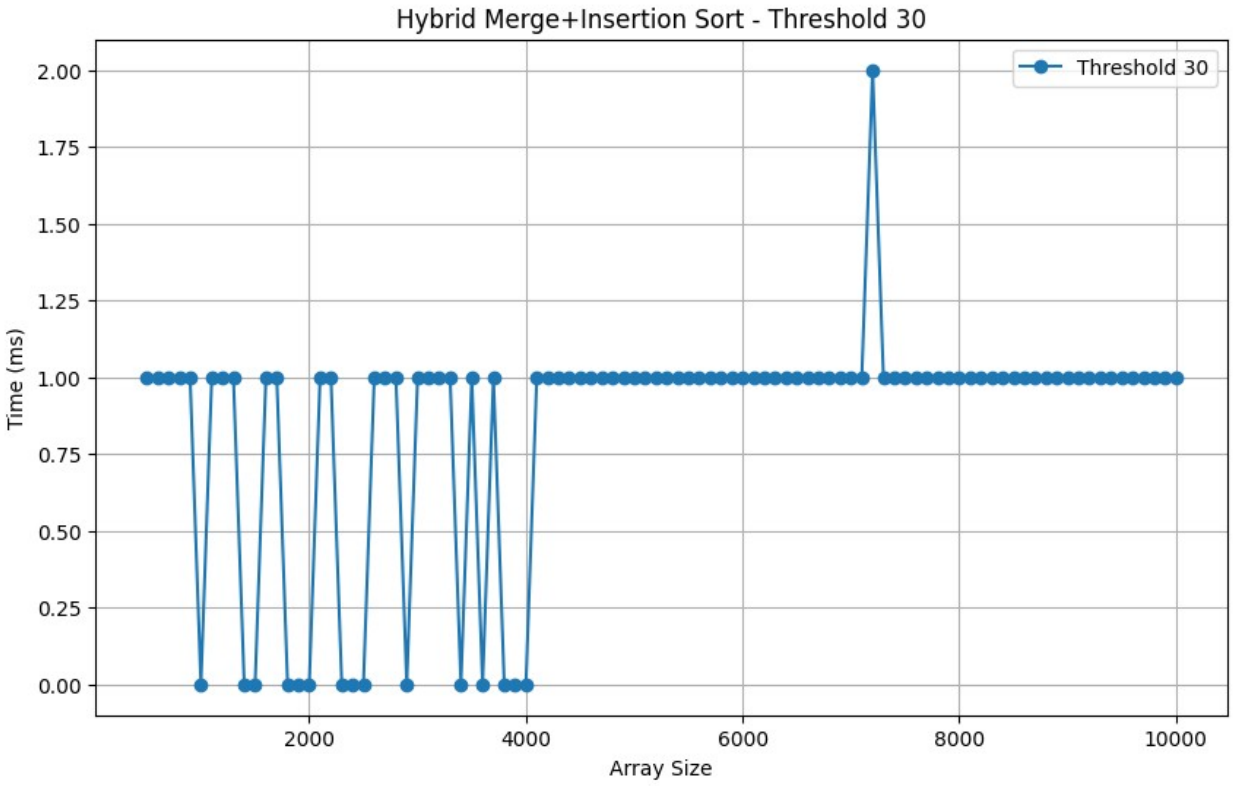
График 2: Построение графика для отсортированных в обратном порядке данных

```python
# Разделение данных по порогам
values_per_threshold = len(H_reverse_times) // num_thresholds
hybrid_times_by_threshold = [
    H_reverse_times[i * values_per_threshold:(i + 1) *
values_per_threshold]
    for i in range(num_thresholds)
]

for i, times in enumerate(hybrid_times_by_threshold):
    plt.figure(figsize=(10, 6))
    if thresholds[i] == 5:
        plt.ylim(-0.0002, 0.0002)
    plt.plot(sizes, times, marker='o', label=f'Threshold
{thresholds[i]}')
    plt.xlabel('Array Size')
    plt.ylabel('Time (ms)')
    plt.title(f'Hybrid Merge+Insertion Sort - Threshold
{thresholds[i]}')
    plt.legend()
    plt.grid(True)
    plt.show()
```




Hybrid Merge+Insertion Sort - Threshold 10
Threshold 10
0.04
0.02
0.00
−0.02
−0.04
Time (ms)
2000
4000
6000
8000
10000
Array Size


Hybrid Merge+Insertion Sort - Threshold 20
Threshold 20
1.0
0.8
0.6
0.4
0.2
0.0
Time (ms)
2000
4000
6000
8000
10000
Array Size



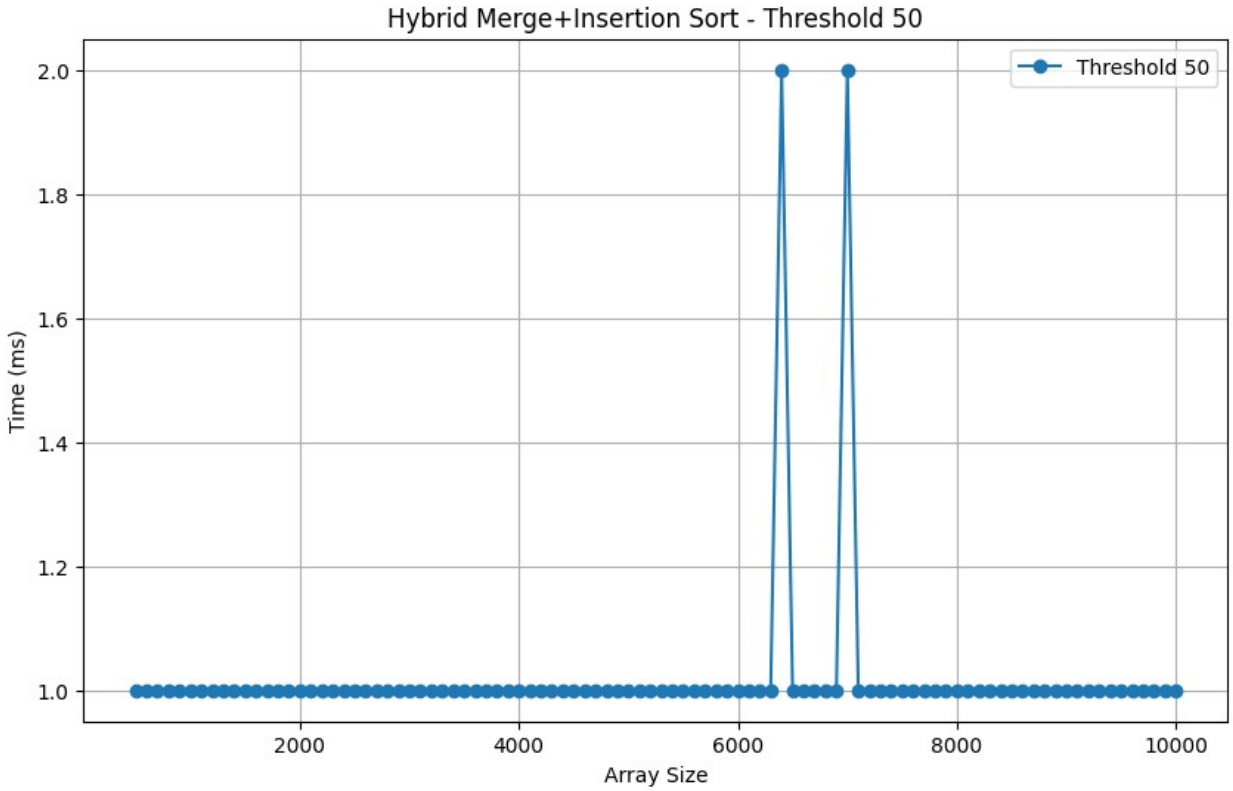
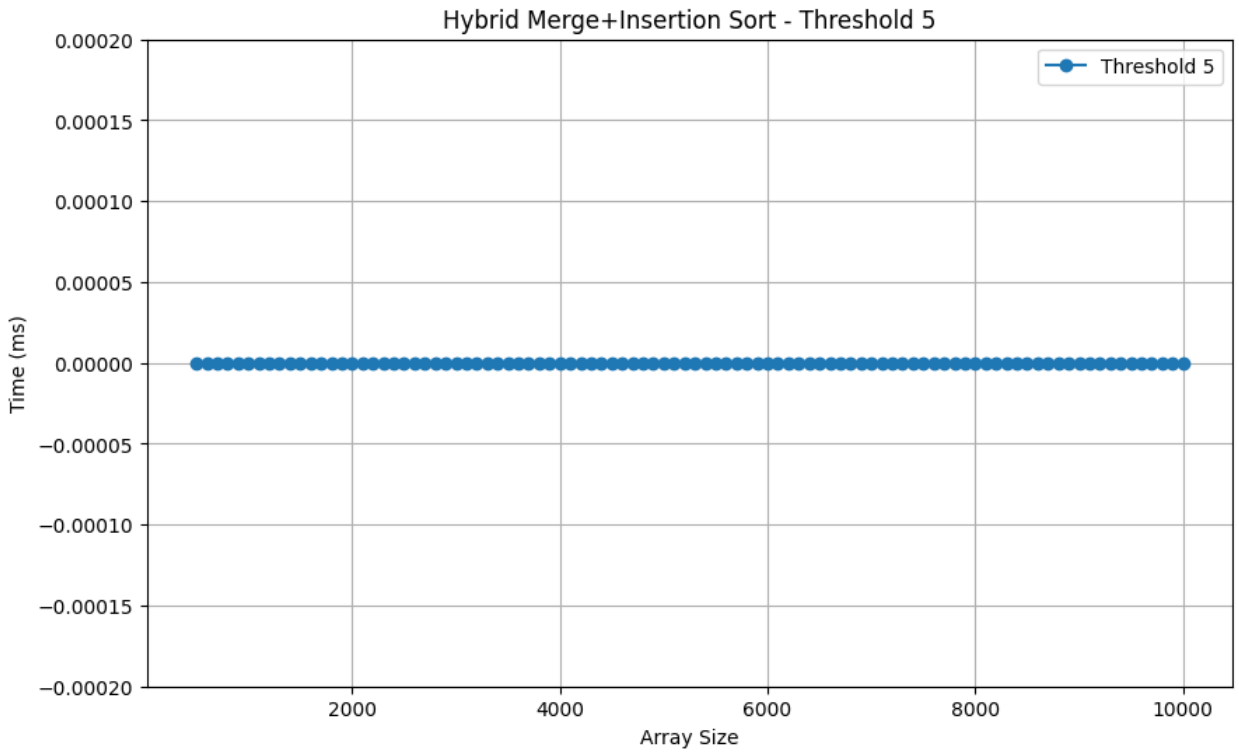
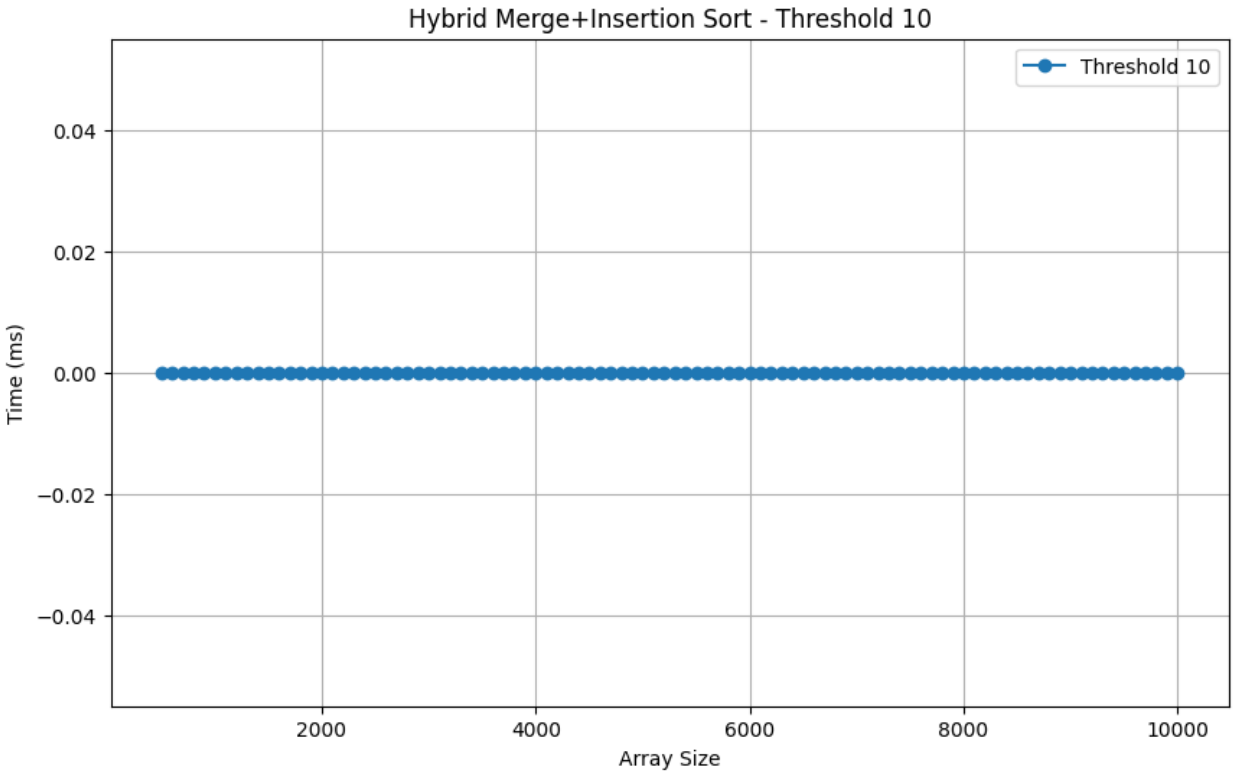
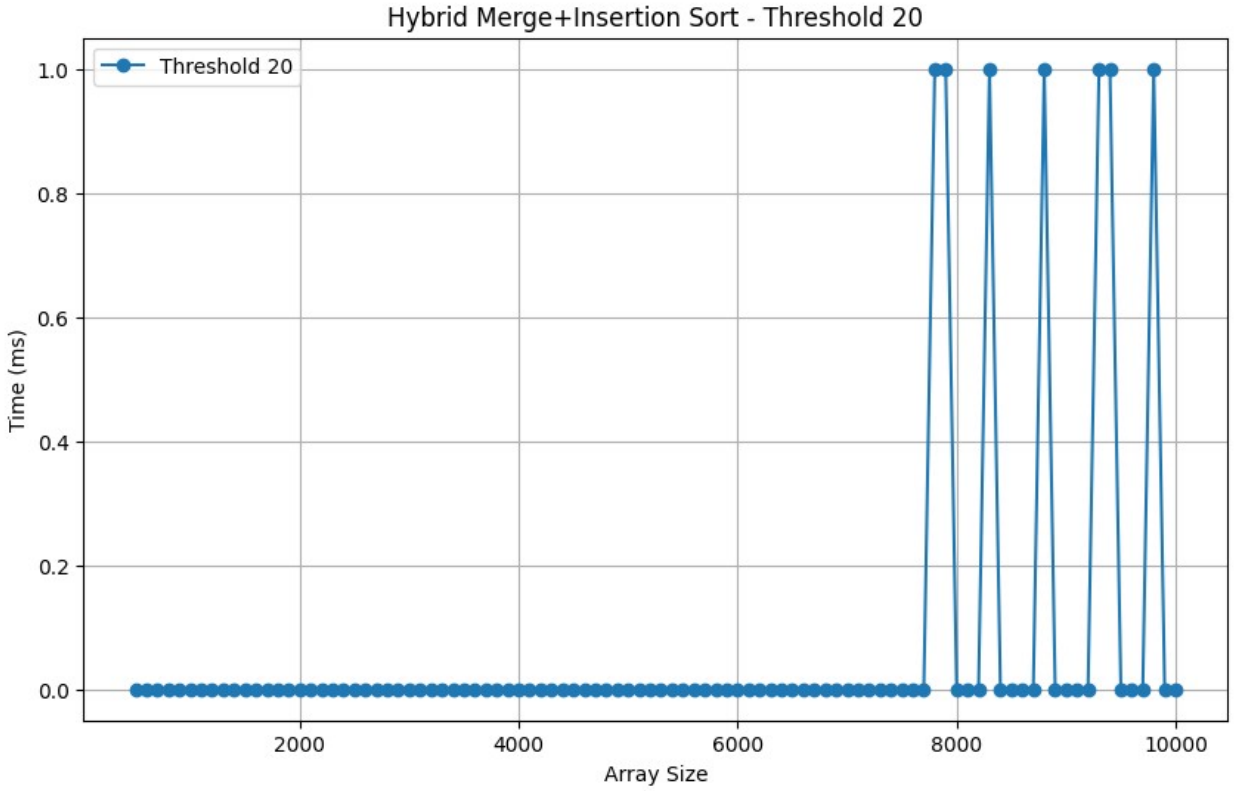


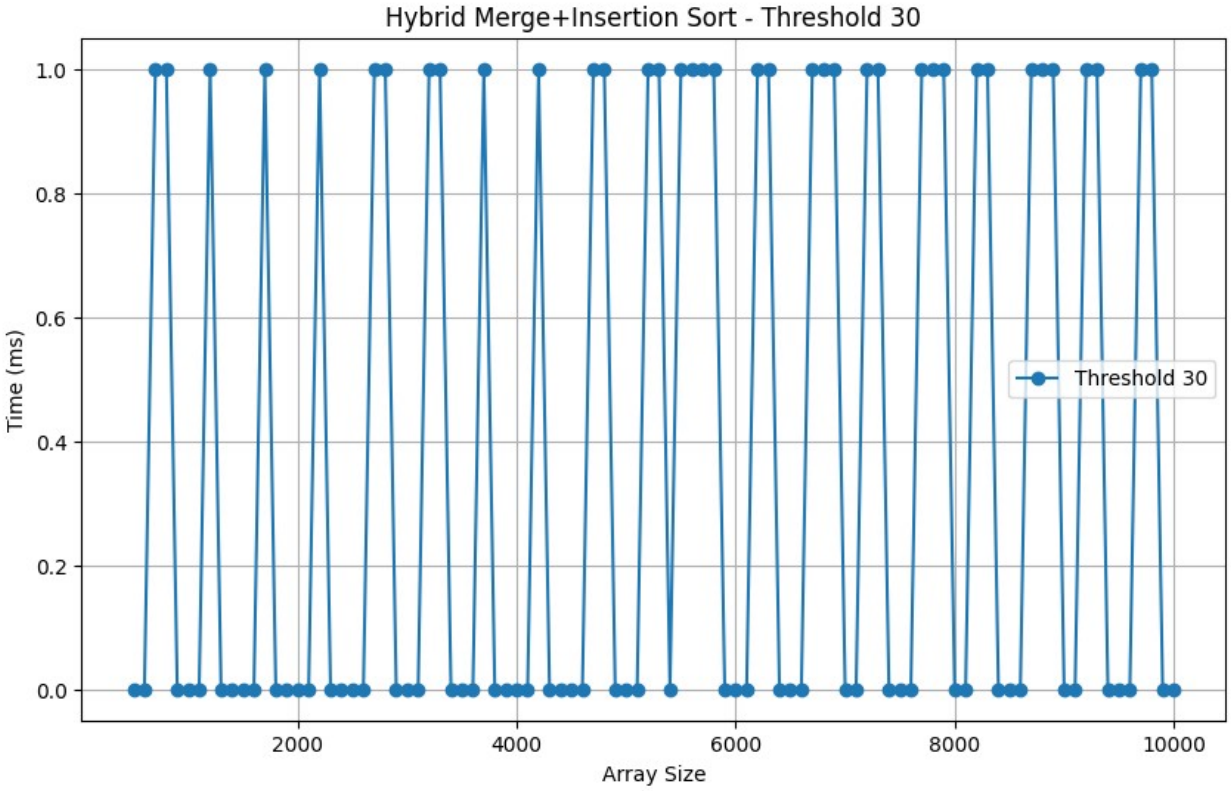
График 3: Построение графика для "почти" отсортированных данных

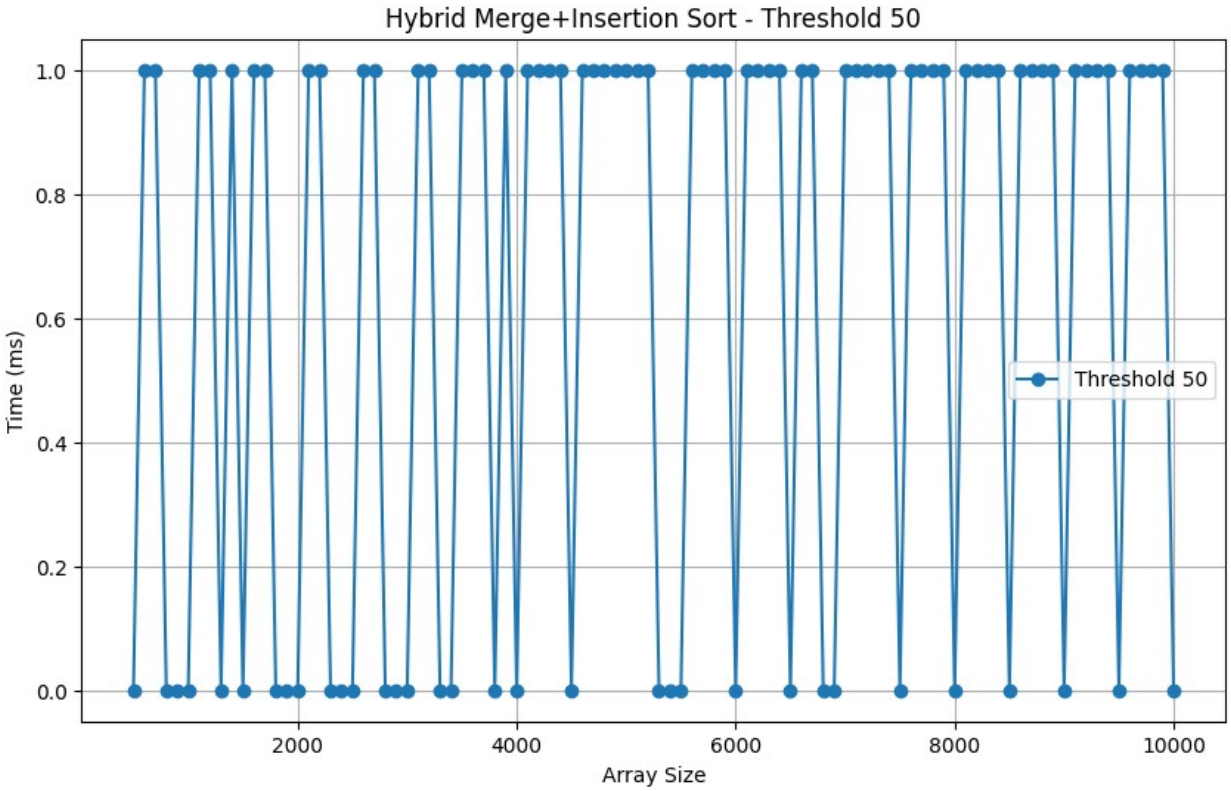
```python
# Разделение данных по порогам
values_per_threshold = len(H_nearly_sorted_times) // num_thresholds
hybrid_times_by_threshold = [
    H_nearly_sorted_times[i * values_per_threshold:(i + 1) *
values_per_threshold]
    for i in range(num_thresholds)
]

for i, times in enumerate(hybrid_times_by_threshold):
    plt.figure(figsize=(10, 6))
    if thresholds[i] == 5:
        plt.ylim(-0.0002, 0.0002)
    plt.plot(sizes, times, marker='o', label=f'Threshold
{thresholds[i]}')
    plt.xlabel('Array Size')
    plt.ylabel('Time (ms)')
    plt.title(f'Hybrid Merge+Insertion Sort - Threshold
{thresholds[i]}')
    plt.legend()
    plt.grid(True)
    plt.show()
```




Hybrid Merge+Insertion Sort - Threshold 10
Threshold 10
0.04
0.02
0.00
−0.02
−0.04
Time (ms)
2000
4000
6000
8000
10000
Array Size


Hybrid Merge+Insertion Sort - Threshold 20
Threshold 20
1.0
0.8
0.6
0.4
0.2
0.0
Time (ms)
2000
4000
6000
8000
10000
Array Size


Hybrid Merge+Insertion Sort - Threshold 30
Time (ms)
Array Size
Threshold 30
0.0
0.2
0.4
0.6
0.8
1.0
2000
4000
6000
8000
10000


Hybrid Merge+Insertion Sort - Threshold 50
Time (ms)
Array Size
Threshold 50
0.0
0.2
0.4
0.6
0.8
1.0
2000
4000
6000
8000
10000