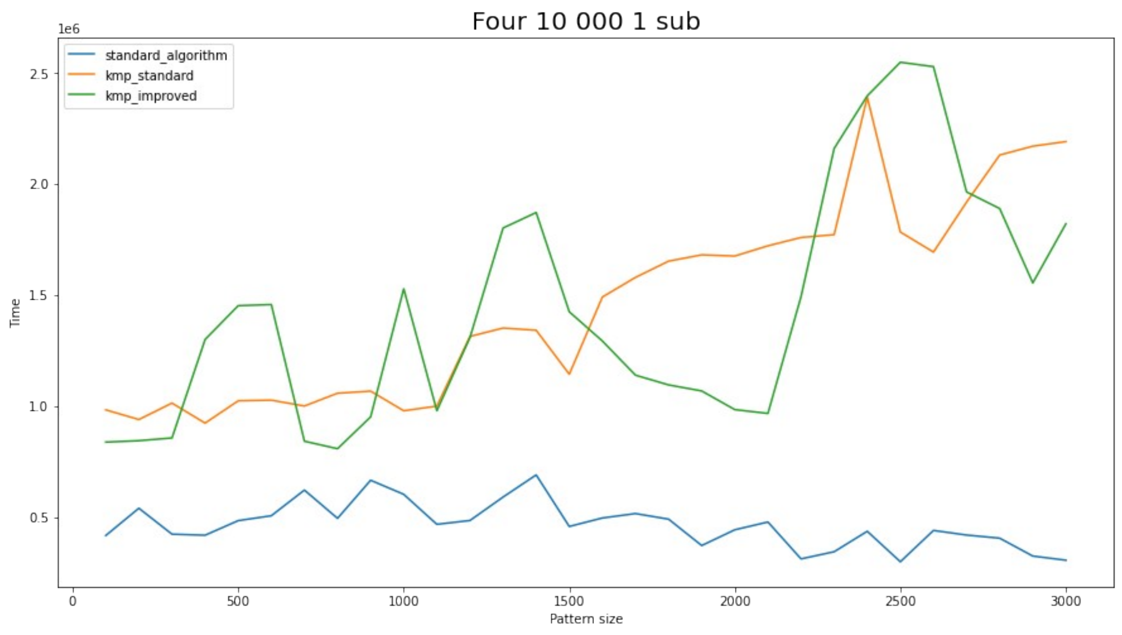
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

Four text 10 000 (1 sub):

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Four 10 000  1.txt") as file:
    for nums in file:
        if len(nums) > 4 and nums[4] == "d":
            current_algorithm = "StandardAlgorithm"
            continue
        elif len(nums) > 4 and nums[4] == "S":
            current_algorithm = "KMP_Standard"
            continue
        elif len(nums) > 4 and nums[4] == "I":
            current_algorithm = "KMP_Improved"
            continue
        if current_algorithm == "StandardAlgorithm":
            k = nums.split(" ")
            standard_algorithm.append(int(k[1]))
        elif current_algorithm == "KMP_Standard":
            k = nums.split(" ")
            kmp_standard.append(int(k[1]))
        elif current_algorithm == "KMP_Improved":
            k = nums.split(" ")
            kmp_improved.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Pattern size")
ax.set_ylabel("Time")
ax.plot(pattern_size, standard_algorithm, label =
'standard_algorithm')
ax.plot(pattern_size, kmp_standard, label = 'kmp_standard')
ax.plot(pattern_size, kmp_improved, label = 'kmp_improved')
ax.set_title("Four 10 000 1 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```

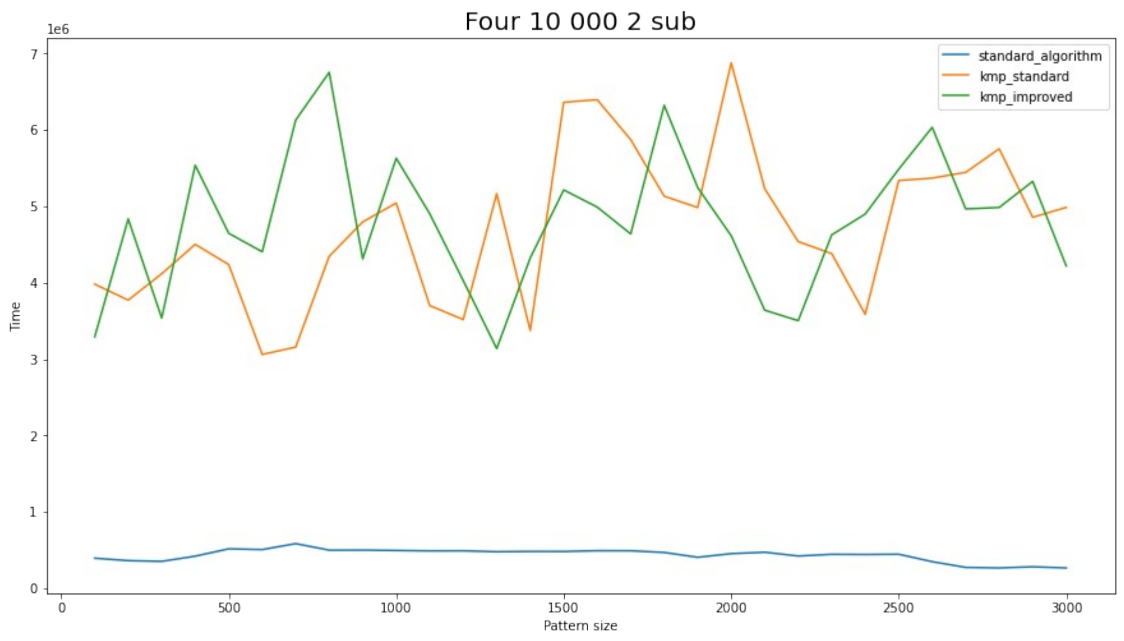


Вывод:

Тут даже несмотрия на то, что только один символ подстановки стандартный алгоритм по времени всё равно лидирует по сравнению с бинарным алфавитом. Это объясняется тем, что тут даже для 1 подстановки нужно перебрать в КМП 4 возможные буквы, раньше нужно было только 2.

Four text 10 000 (2 subs):

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Four 10 000  2.txt") as file:
    for nums in file:
        if len(nums) > 4 and nums[4] == "d":
            current_algorithm = "StandardAlgorithm"
            continue
        elif len(nums) > 4 and nums[4] == "S":
            current_algorithm = "KMP_Standard"
            continue
        elif len(nums) > 4 and nums[4] == "I":
            current_algorithm = "KMP_Improved"
            continue
        if current_algorithm == "StandardAlgorithm":
            k = nums.split(" ")
            standard_algorithm.append(int(k[1]))
        elif current_algorithm == "KMP_Standard":
            k = nums.split(" ")
            kmp_standard.append(int(k[1]))
        elif current_algorithm == "KMP_Improved":
            k = nums.split(" ")
            kmp_improved.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Pattern size")
ax.set_ylabel("Time")
ax.plot(pattern_size, standard_algorithm, label =
'standard_algorithm')
ax.plot(pattern_size, kmp_standard, label = 'kmp_standard')
ax.plot(pattern_size, kmp_improved, label = 'kmp_improved')
ax.set_title("Four 10 000 2 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```



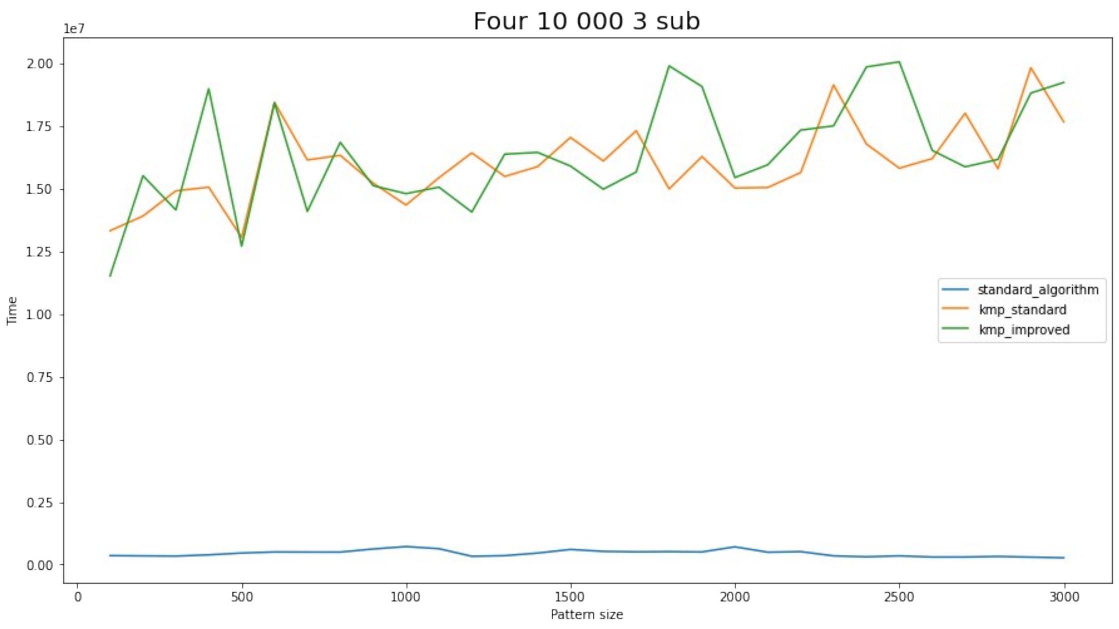
Вывод:

Тут на КМП слишком сильные скачки. По сути из-за большего кол-ва символов в алфавите может происходить больше несовпадений, это всё замедляет. Для бинарного символы могли чаще совпадать, а значит двигались мы быстрее.

Four text 10 000 (3 subs):

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
```

```
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Four 10 000  3.txt") as file:
    for nums in file:
        if len(nums) > 4 and nums[4] == "d":
            current_algorithm = "StandardAlgorithm"
            continue
        elif len(nums) > 4 and nums[4] == "S":
            current_algorithm = "KMP_Standard"
            continue
        elif len(nums) > 4 and nums[4] == "I":
            current_algorithm = "KMP_Improved"
            continue
        if current_algorithm == "StandardAlgorithm":
            k = nums.split(" ")
            standard_algorithm.append(int(k[1]))
        elif current_algorithm == "KMP_Standard":
            k = nums.split(" ")
            kmp_standard.append(int(k[1]))
        elif current_algorithm == "KMP_Improved":
            k = nums.split(" ")
            kmp_improved.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Pattern size")
ax.set_ylabel("Time")
ax.plot(pattern_size, standard_algorithm, label =
'standard_algorithm')
ax.plot(pattern_size, kmp_standard, label = 'kmp_standard')
ax.plot(pattern_size, kmp_improved, label = 'kmp_improved')
ax.set_title("Four 10 000 3 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```

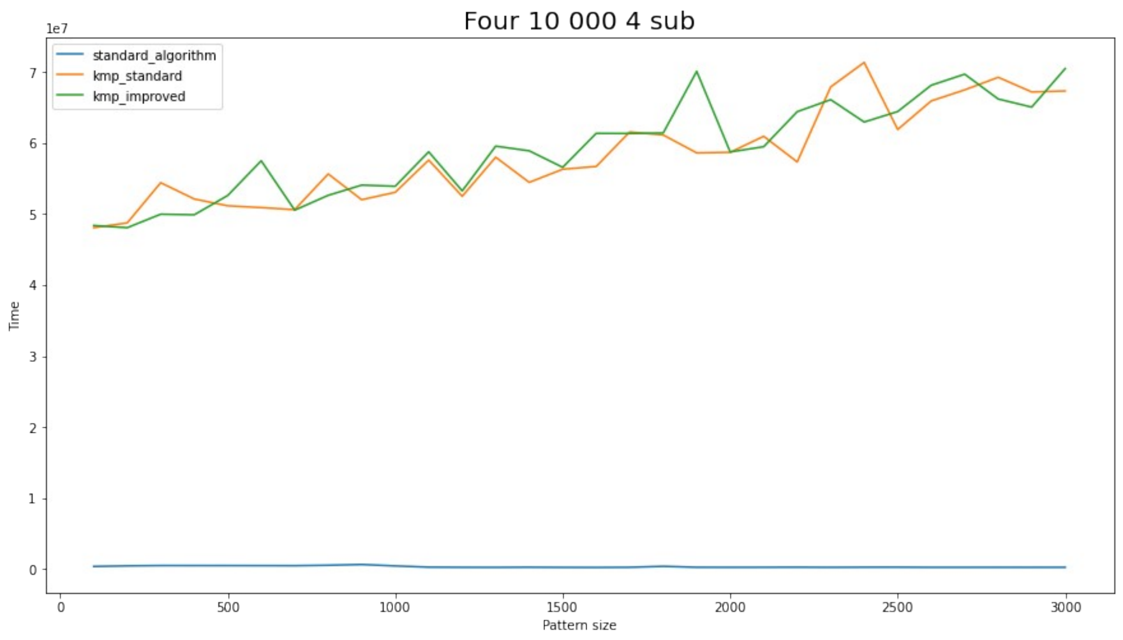


Вывод:

Тут в кмп больше стабильности

Four text 10 000 (4 subs):

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Four 10 000  4.txt") as file:
    for nums in file:
        if len(nums) > 4 and nums[4] == "d":
            current_algorithm = "StandardAlgorithm"
            continue
        elif len(nums) > 4 and nums[4] == "S":
            current_algorithm = "KMP_Standard"
            continue
        elif len(nums) > 4 and nums[4] == "I":
            current_algorithm = "KMP_Improved"
            continue
        if current_algorithm == "StandardAlgorithm":
            k = nums.split(" ")
            standard_algorithm.append(int(k[1]))
        elif current_algorithm == "KMP_Standard":
            k = nums.split(" ")
            kmp_standard.append(int(k[1]))
        elif current_algorithm == "KMP_Improved":
            k = nums.split(" ")
            kmp_improved.append(int(k[1]))
fig, ax = plt.subplots()
fig.set_size_inches(15,8)
ax.set_xlabel("Pattern size")
ax.set_ylabel("Time")
ax.plot(pattern_size, standard_algorithm, label =
'standard_algorithm')
ax.plot(pattern_size, kmp_standard, label = 'kmp_standard')
ax.plot(pattern_size, kmp_improved, label = 'kmp_improved')
ax.set_title("Four 10 000 4 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```



Вывод:

На 4 подстановках время КМП крайне велико, слишком много всего перебирать.