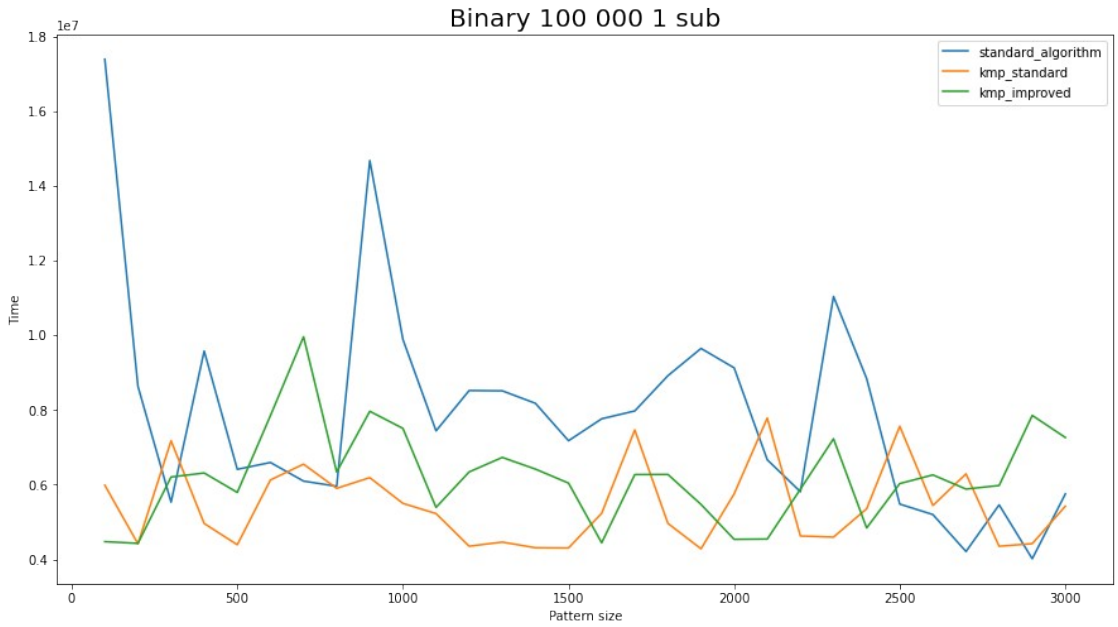
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

Binary text 100 000 (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("Binary 100 000  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("Binary 100 000 1 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```



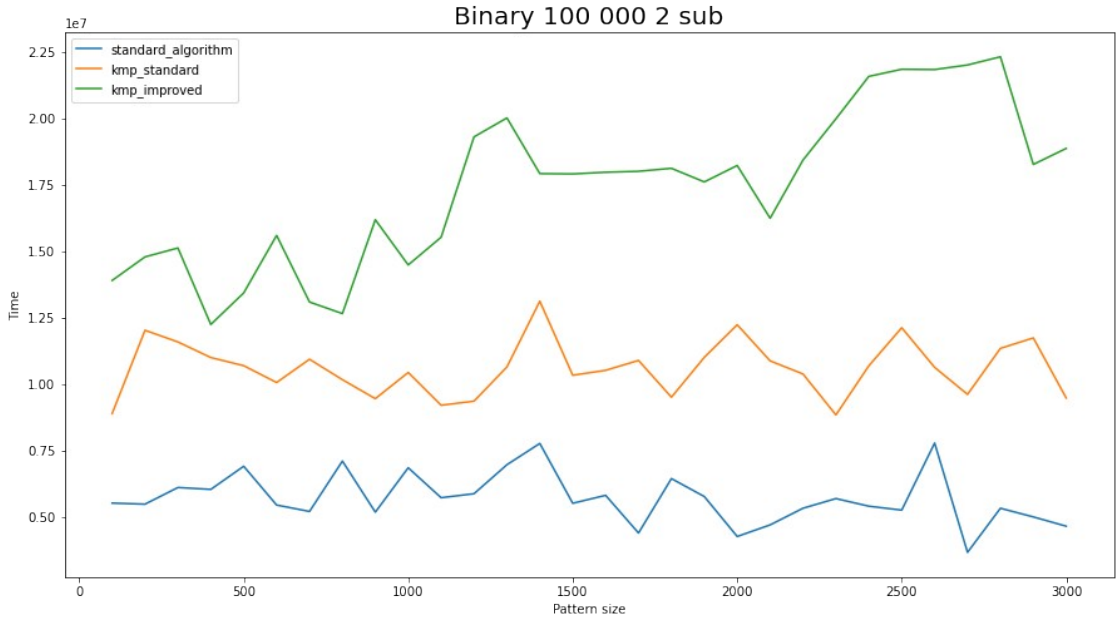
Вывод:

При одном символе подстановки стандартный в среднем работает хуже всего, так как для кмп не приходится перебирать много варинатов.

Binary text 100 000 (2 sub)

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Binary 100 000  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("Binary 100 000 2 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```



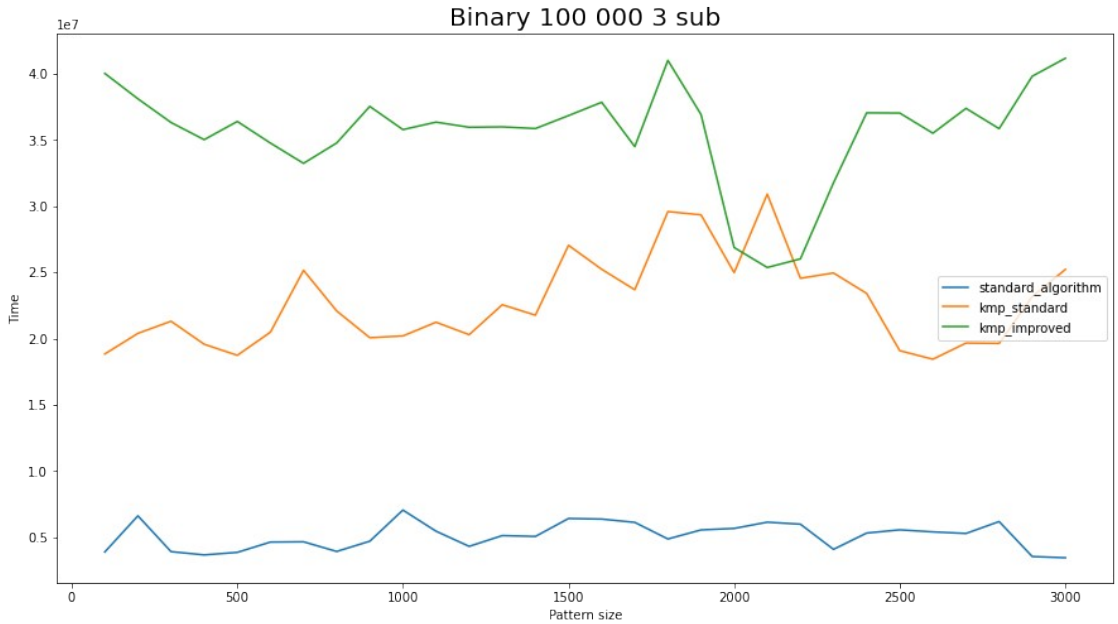
Вывод:

Кажется, самый красивый график из всех. С уточнеинем граней стабильно хуже стандратного КМП. А вот просто стандартный берёт на себя лидерство, так как не приходится устраивать перебор варинтов, как в КМП.

Binary text 100 000 (3 sub)

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Binary 100 000  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("Binary 100 000 3 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```

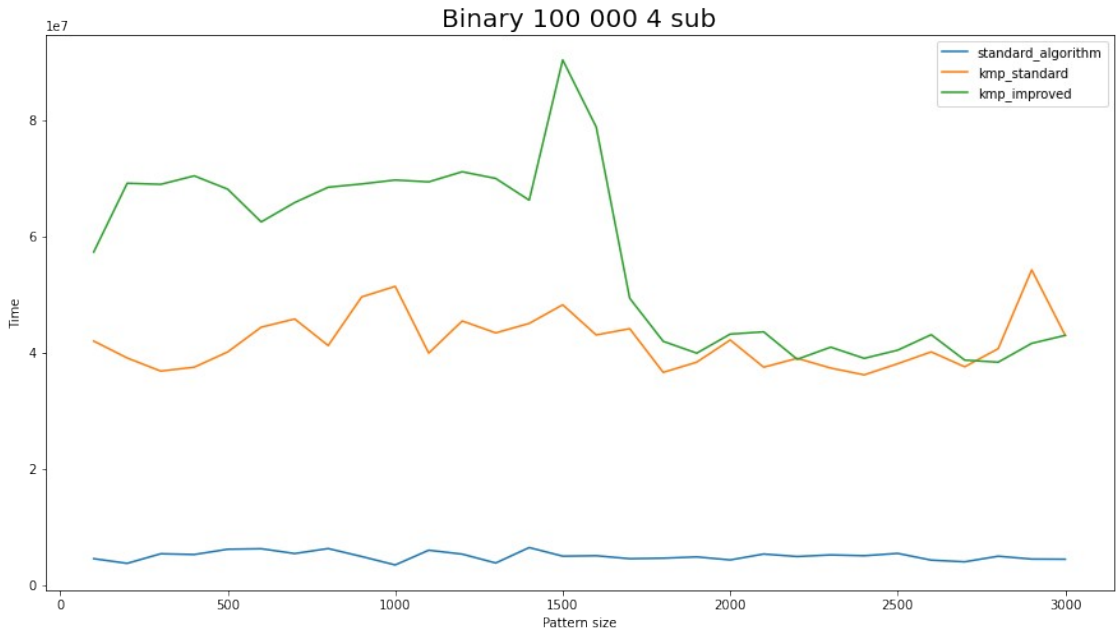


Вывод:

Ситуация похожа на прошлый график, толкько что в один момент времени на 2200 длине стандартный КМП замедялет ход. Ещё интересно, что на меньших данных в графиках КМП больше стабильности

Binary text 100 000 (4 sub)

```
pattern_size = []
for i in range(100, 3001, 100):
    pattern_size.append(i)
standard_algorithm = []
kmp_standard = []
kmp_improved = []
current_algorithm = ""
with open("Binary 100 000  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("Binary 100 000 4 sub", fontsize= 20)
plt.legend(loc='best')
plt.show()
```



Вывод:

Опять же на мальньких данных КМП стабильны