Pedram - Mirelmi - 610398176

Same testing just with SJF algorithm

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from MySJF import *
```

base condition

In [ ]:
```
number_of_tests = 500
wished_range = 1 * 10**3
context_switch_time = 0
tester = SimpleSJF(n = 1 * 10**1,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                       in_range=wished_range,
                                                       shape=(n, 2)
                                                   )
                   )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 1000
{
    "response times": {
        "mean of mean": 1297.0886,
        "mean of std": 1178.2562323896445,
        "std of mean": 433.479854030196,
        "std of std": 320.0019327050384
    },
    "waiting times": {
        "mean of mean": 1297.0886,
        "mean of std": 1178.2562323896445,
        "std of mean": 433.479854030196,
        "std of std": 320.0019327050384
    },
    "turnaround times": {
        "mean of mean": 1793.1906,
        "mean of std": 1387.878476675675,
        "std of mean": 524.2042336834376,
        "std of std": 322.7692482880111
    },
    "total number of processes": 5000,
    "maximum burst time": "999"
}
```

double wished_range

In [ ]:
```
number_of_tests = 500
wished_range = 2 * 10**3
context_switch_time = 0
tester = SimpleSJF(n = 1 * 10**2,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                       in_range=wished_range,
                                                       shape=(n, 2)
                                                   )
                   )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 2000
{
    "response times": {
        "mean of mean": 32356.8581,
        "mean of std": 29255.914477212435,
        "std of mean": 2909.1942329330627,
        "std of std": 1911.5670138044811
    },
    "waiting times": {
        "mean of mean": 32356.8581,
        "mean of std": 29255.914477212435,
        "std of mean": 2909.1942329330627,
        "std of std": 1911.5670138044811
    },
    "turnaround times": {
        "mean of mean": 33356.70496,
        "mean of std": 29803.04241817522,
        "std of mean": 2964.0259661574146,
        "std of std": 1909.1197796639995
```

```
    },
    "total number of processes": 50000,
    "maximum burst time": "1999"
}
```

double number of processes

In [ ]:

```
number_of_tests = 500
wished_range = 1 * 10**3
context_switch_time = 0
tester = SimpleSJF(n = 2 * 10**2,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                   )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 1000
{
    "response times": {
        "mean of mean": 32642.99635,
        "mean of std": 29376.491982206728,
        "std of mean": 2144.0642377027857,
        "std of std": 1408.9709676087793
    },
    "waiting times": {
        "mean of mean": 32642.99635,
        "mean of std": 29376.491982206728,
        "std of mean": 2144.0642377027857,
        "std of std": 1408.9709676087793
    },
    "turnaround times": {
        "mean of mean": 33140.70904,
        "mean of std": 29652.8463272136,
        "std of mean": 2164.2579119352613,
        "std of std": 1408.010871642323
    },
    "total number of processes": 100000,
    "maximum burst time": "999"
}
```

Everything seems linear! Let's plot

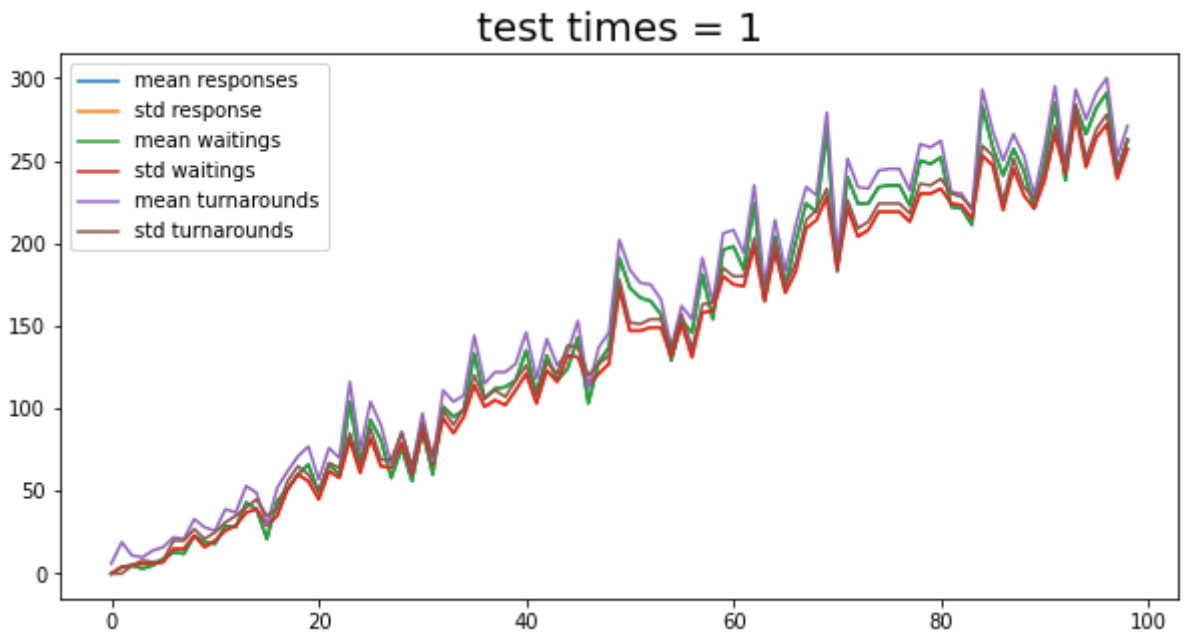
base condition with 1 time test and context switch = 0

In [ ]:

```
wished_range = 10*2
test_times = 1
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleSJF(n=procces_count,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
```

```
plt.legend(loc=2, prop={'size': 10})
plt.title('test times = 1',size=20)
```

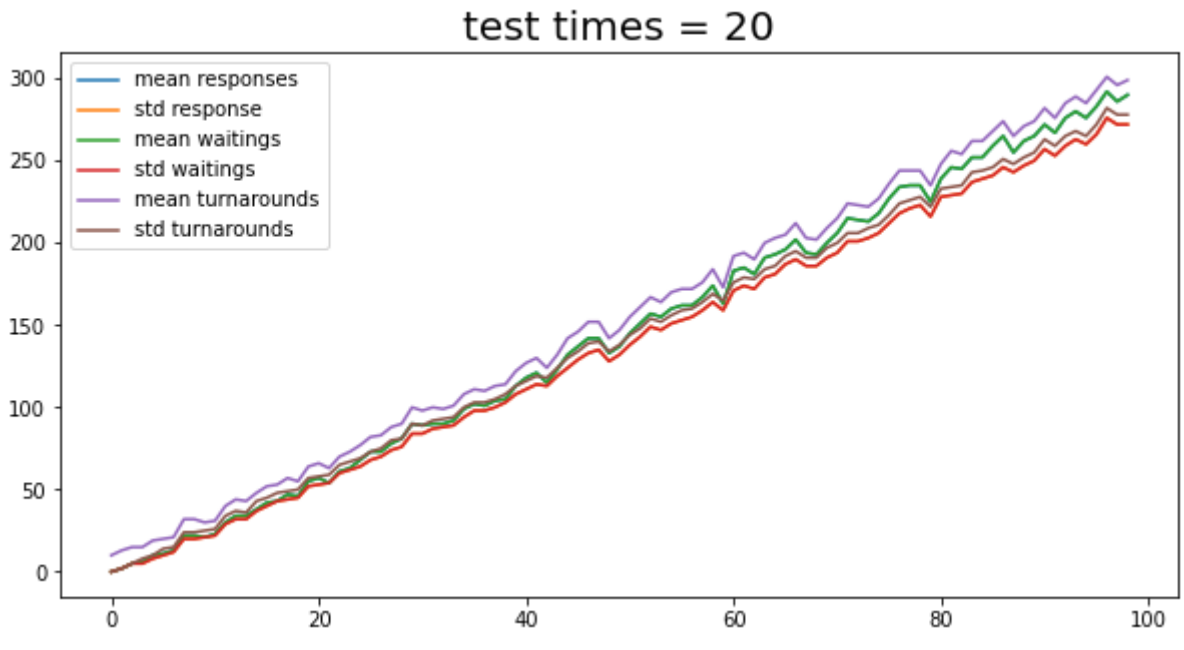
Out[ ]:
```
Text(0.5, 1.0, 'test times = 1')
```



test times = 20

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleSJF(n=procces_count,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                   )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('test times = 20',size=20)
```

Out[ ]:
```
Text(0.5, 1.0, 'test times = 20')
```

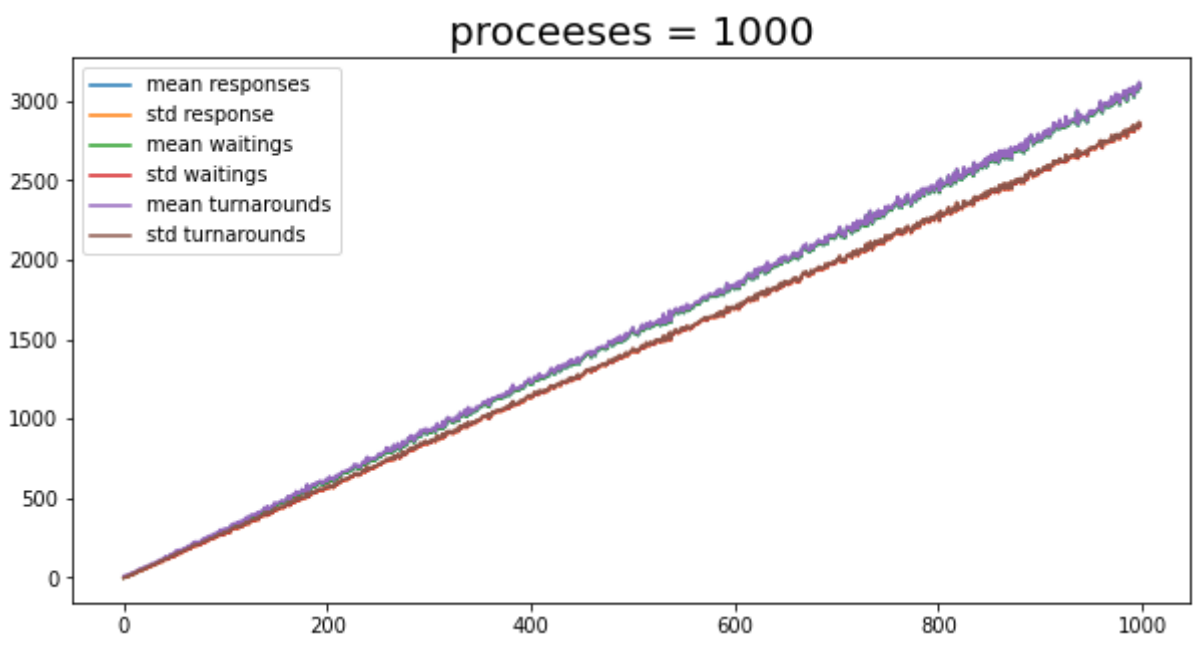


process count = 1 to 1000

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 1000):
    tester = SimpleSJF(n=procces_count,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                           in_range=wished_range,
                                                           shape=(n, 2)
                                                       )
                      )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))
```

In [ ]:
```
a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('proceeses = 1000',size=20)
```

Out[ ]:
```
Text(0.5, 1.0, 'proceeses = 1000')
```

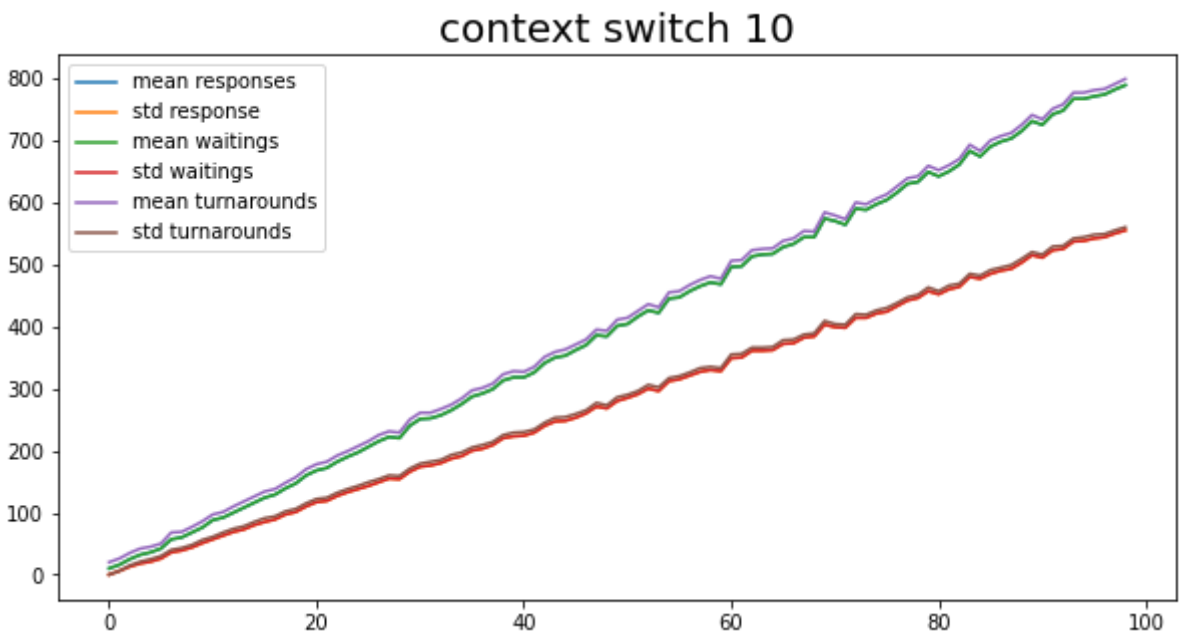


context switch = 10

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 10
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleSJF(n=procces_count,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                           in_range=wished_range,
                                                           shape=(n, 2)
                                                       )
                      )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
```

```
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('context switch 10',size=20)
```

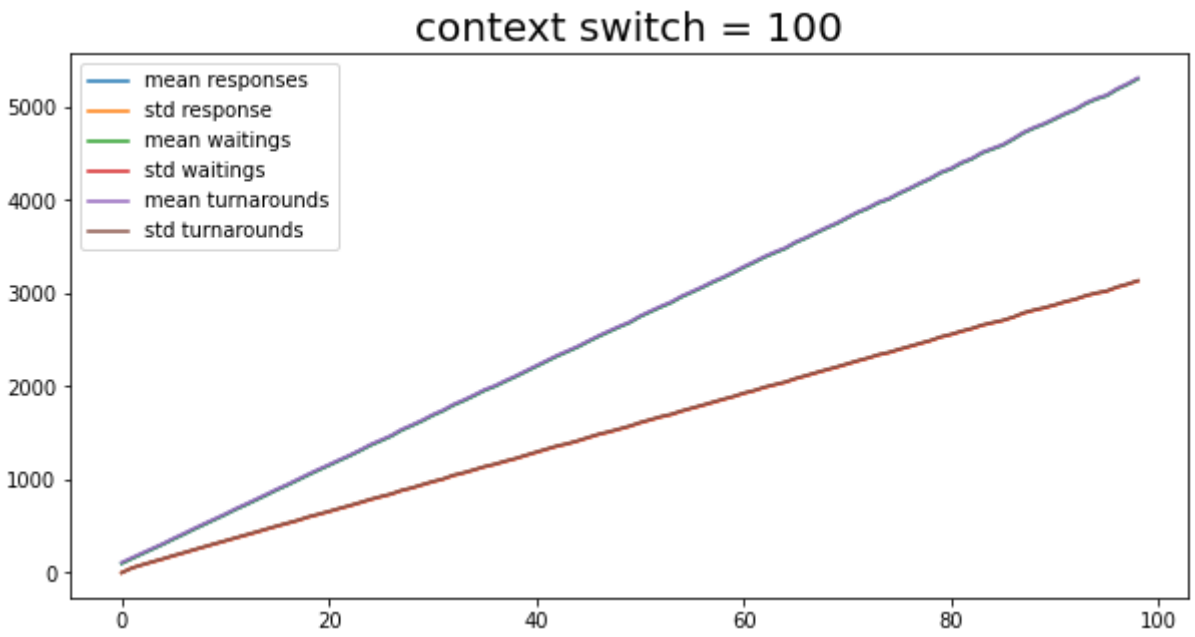
Out[ ]:
```
Text(0.5, 1.0, 'context switch 10')
```



context switch = 100

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 100
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleSJF(n=procces_count,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                       )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('context switch = 100',size=20)
```

Out[ ]:
```
Text(0.5, 1.0, 'context switch = 100')
```



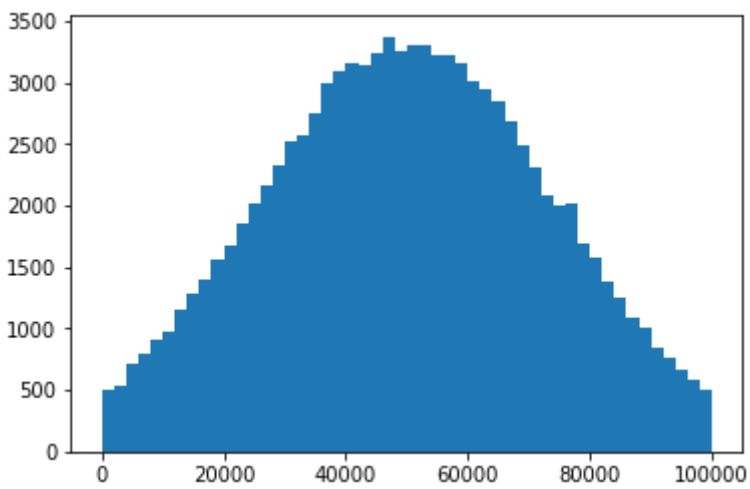
Normal distribution

base condition

In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                        mean=mean,
                                                        std=std,
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                   )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 17790585.756490733,
        "mean of std": 13970514.985083908,
        "std of mean": 391670.8903052225,
        "std of std": 227486.43870330957
    },
    "waiting times": {
        "mean of mean": 17790585.756490733,
        "mean of std": 13970514.985083908,
        "std of mean": 391670.8903052225,
        "std of std": 227486.43870330957
    },
    "turnaround times": {
        "mean of mean": 17840577.933213376,
        "mean of std": 13991936.544361904,
        "std of mean": 392343.5184162763,
        "std of std": 227448.21034551246
    },
    "total number of processes": 477196,
    "maximum burst time": "99998"
}
```



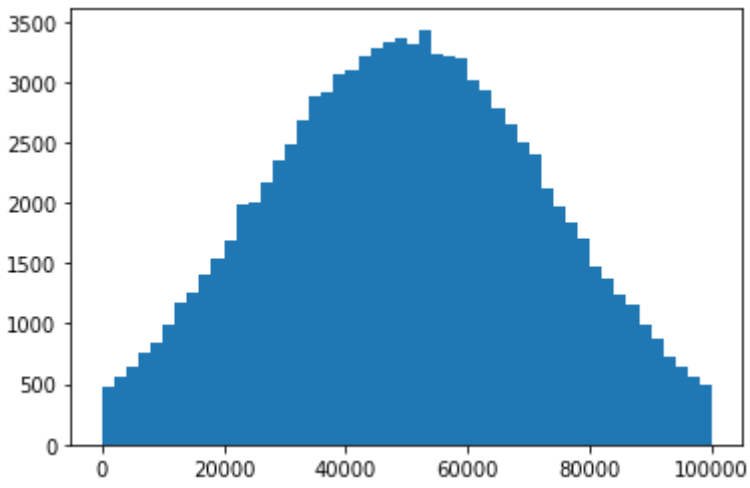
double number of processes

In [ ]:

```
number_of_tests = 500
wished_range = 10**5
```

```
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 2* 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                           mean=mean,
                                                           std=std,
                                                           in_range=wished_range,
                                                           shape=(n, 2)
                                                       )
                   )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 35680915.96210812,
        "mean of std": 27970961.56343126,
        "std of mean": 547800.0561049815,
        "std of std": 324669.7154823607
    },
    "waiting times": {
        "mean of mean": 35680915.96210812,
        "mean of std": 27970961.56343126,
        "std of mean": 547800.0561049815,
        "std of std": 324669.7154823607
    },
    "turnaround times": {
        "mean of mean": 35730920.05148506,
        "mean of std": 27992365.178749677,
        "std of mean": 548283.1542647128,
        "std of std": 324653.6536912798
    },
    "total number of processes": 954578,
    "maximum burst time": "99999"
}
```

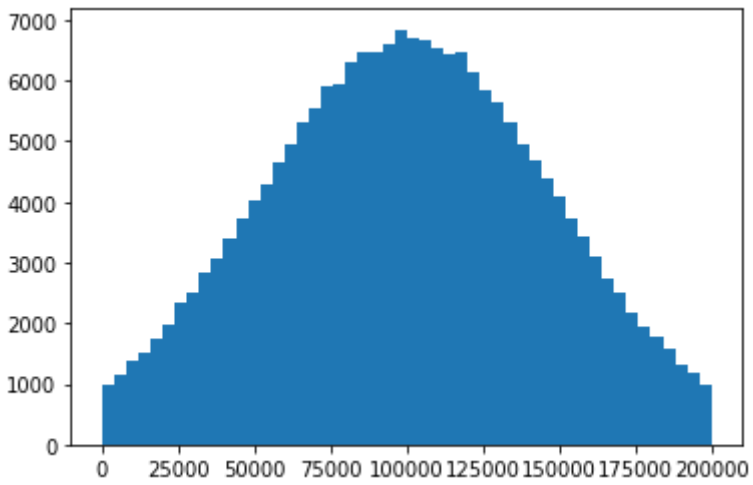


In [ ]:

```
number_of_tests = 500
wished_range = 2 * 10**5
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                           mean=mean,
                                                           std=std,
                                                           in_range=wished_range,
                                                           shape=(n, 2)
                                                       )
                   )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 1.0000000000E+05 and std of 5.0000000000E+04
{
    "response times": {
        "mean of mean": 35624993.581114516,
        "mean of std": 27947673.94327842,
        "std of mean": 716713.9906205523,
        "std of std": 426257.33518732304
    },
    "waiting times": {
        "mean of mean": 35624993.581114516,
        "mean of std": 27947673.94327842,
        "std of mean": 716713.9906205523,
        "std of std": 426257.33518732304
    },
    "turnaround times": {
        "mean of mean": 35725005.522313,
```

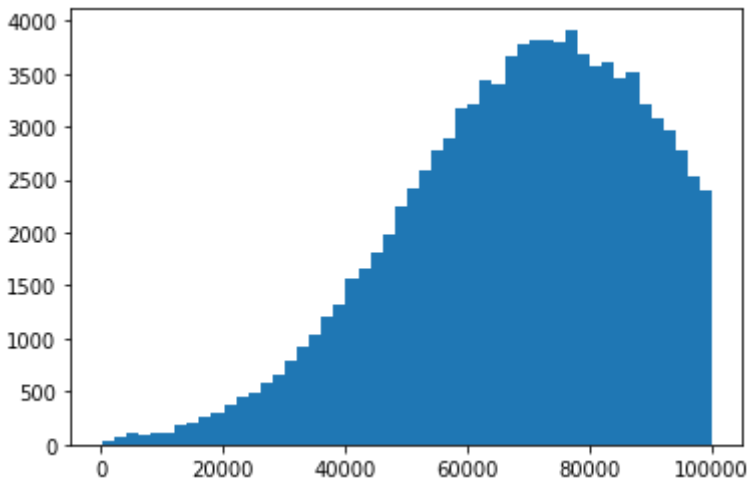
```
        "mean of std": 27990410.56964572,
        "std of mean": 717953.897833931,
        "std of std": 426253.7894977384
    },
    "total number of processes": 477200,
    "maximum burst time": "199995"
}
```



In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (6)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                    mean=mean,
                                                    std=std,
                                                    in_range=wished_range,
                                                    shape=(n, 2)
                                                )
                   )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 7.5000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 23815867.951187868,
        "mean of std": 16858297.68444574,
        "std of mean": 403939.3597874497,
        "std of std": 225991.6545981997
    },
    "waiting times": {
        "mean of mean": 23815867.951187868,
        "mean of std": 16858297.68444574,
        "std of mean": 403939.3597874497,
        "std of std": 225991.6545981997
    },
    "turnaround times": {
        "mean of mean": 23883833.72089653,
        "mean of std": 16876965.48036248,
        "std of mean": 404459.65765529685,
        "std of std": 225869.70953596433
    },
    "total number of processes": 420024,
    "maximum burst time": "99999"
}
```

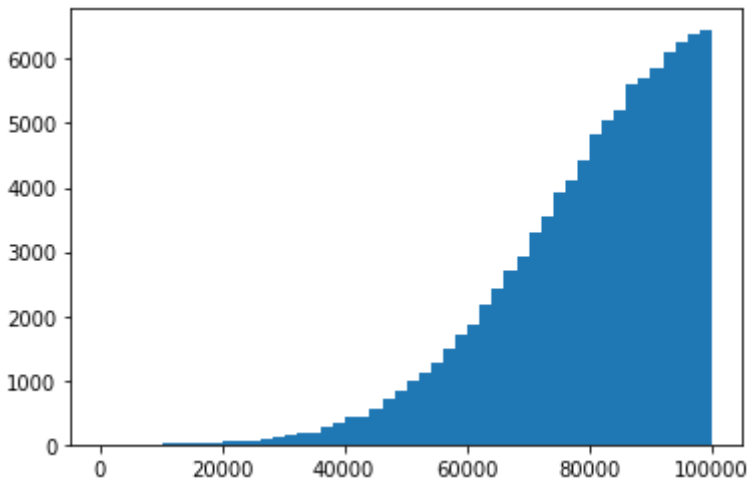


In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (8)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
```

```
                                                                    mean=mean,
                                                                    std=std,
                                                                    in_range=wished_range,
                                                                    shape=(n, 2)
                                                                )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

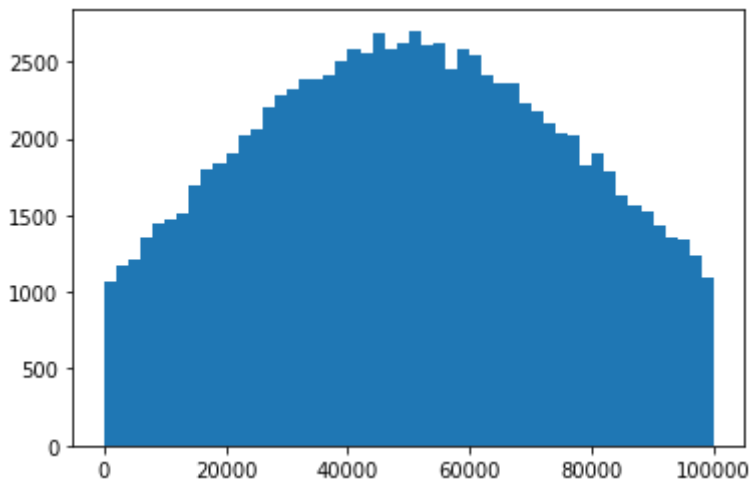
```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 1.0000000000E+05 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 17857700.38205012,
        "mean of std": 11807691.032510506,
        "std of mean": 477043.93962927134,
        "std of std": 293836.41860055475
    },
    "waiting times": {
        "mean of mean": 17857700.38205012,
        "mean of std": 11807691.032510506,
        "std of mean": 477043.93962927134,
        "std of std": 293836.41860055475
    },
    "turnaround times": {
        "mean of mean": 17937722.7051429,
        "mean of std": 11821625.707755042,
        "std of mean": 477395.24385195726,
        "std of std": 293750.79952281725
    },
    "total number of processes": 250128,
    "maximum burst time": "99999"
}
```



In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (4)*(wished_range)/8
std = 3*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleSJF(n = 10**3,
                   context_switch_time=context_switch_time,
                   randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                                    mean=mean,
                                                                    std=std,
                                                                    in_range=wished_range,
                                                                    shape=(n, 2)
                                                                )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 3.7500000000E+04
{
    "response times": {
        "mean of mean": 14382459.958590584,
        "mean of std": 12062309.28321587,
        "std of mean": 416920.0527535749,
        "std of std": 280931.5468485477
    },
    "waiting times": {
        "mean of mean": 14382459.958590584,
        "mean of std": 12062309.28321587,
        "std of mean": 416920.0527535749,
        "std of std": 280931.5468485477
    },
    "turnaround times": {
        "mean of mean": 14432491.185774418,
        "mean of std": 12087137.931328187,
        "std of mean": 417705.1566324006,
        "std of std": 280938.8949013706
    },
    "total number of processes": 408723,
    "maximum burst time": "99999"
}
```



In [ ]:

```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
std = wished_range / 4
for mean in range(1, wished_range):
    tester = SimpleSJF(n=100,
                       context_switch_time=context_switch_time,
                       randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                         mean=mean,
                                                         std=std,
                                                         in_range=wished_range,
                                                         shape=(n, 2)
                                                     )
                   )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('stats',size=20)
plt.xlabel('mean')
```

Out[ ]: Text(0.5, 0, 'mean')