In [1]:
```python
import pandas as pd
import numpy as np

numNames = open("../../db/names.csv", "r").read().split("\n").__len__()
def get_opt_k(m: int) -> int:
    return np.round(np.log(2)*m/numNames, 0).__int__()

def get_false_positive_rate(m: int, k: int) -> float:
    return (1 - np.exp(-(k*numNames)/m))**k


for M in [100_000, 250_000, 500_000, 750_000, 1_000_000, 2_000_000]:
    k = get_opt_k(M)
    fp_rate = get_false_positive_rate(M, k)
    print(f"M ~= {np.round(M/numNames,0)} * N")
    print(f"Probabilidad falso positivo  {M=:9,}  {k=:2}:  {fp_rate*100:7.4f}%\n")
```

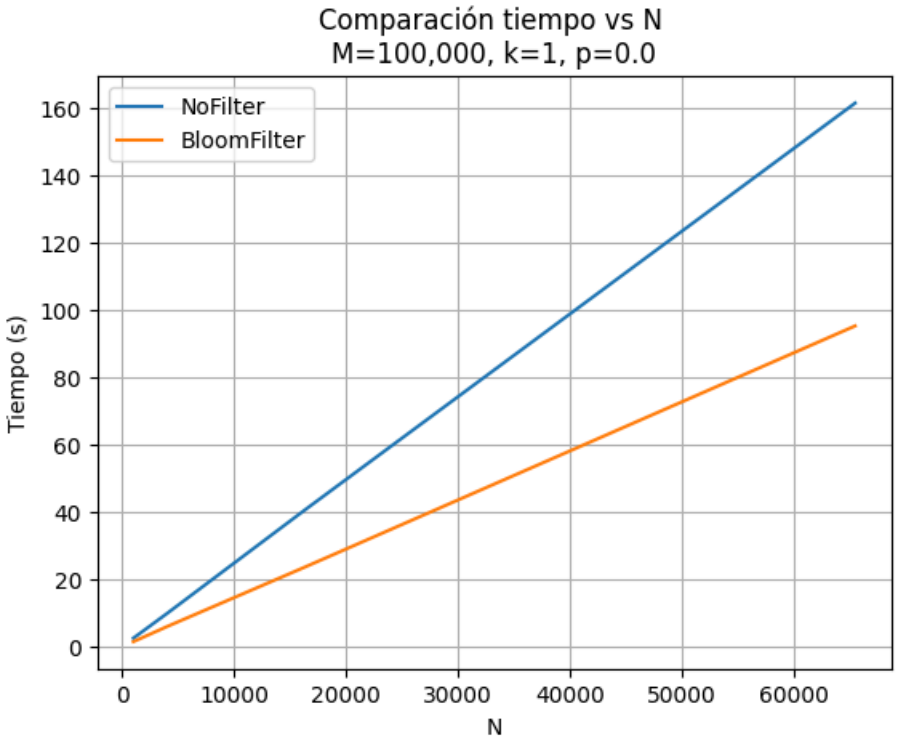
```
M ~= 1.0 * N
Probabilidad falso positivo  M=  100,000  k= 1:  60.8938%

M ~= 3.0 * N
Probabilidad falso positivo  M=  250,000  k= 2:  27.8951%

M ~= 5.0 * N
Probabilidad falso positivo  M=  500,000  k= 4:   7.7814%

M ~= 8.0 * N
Probabilidad falso positivo  M=  750,000  k= 6:   2.1706%

M ~= 11.0 * N
Probabilidad falso positivo  M=1,000,000  k= 7:   0.6019%

M ~= 21.0 * N
Probabilidad falso positivo  M=2,000,000  k=15:   0.0036%
```

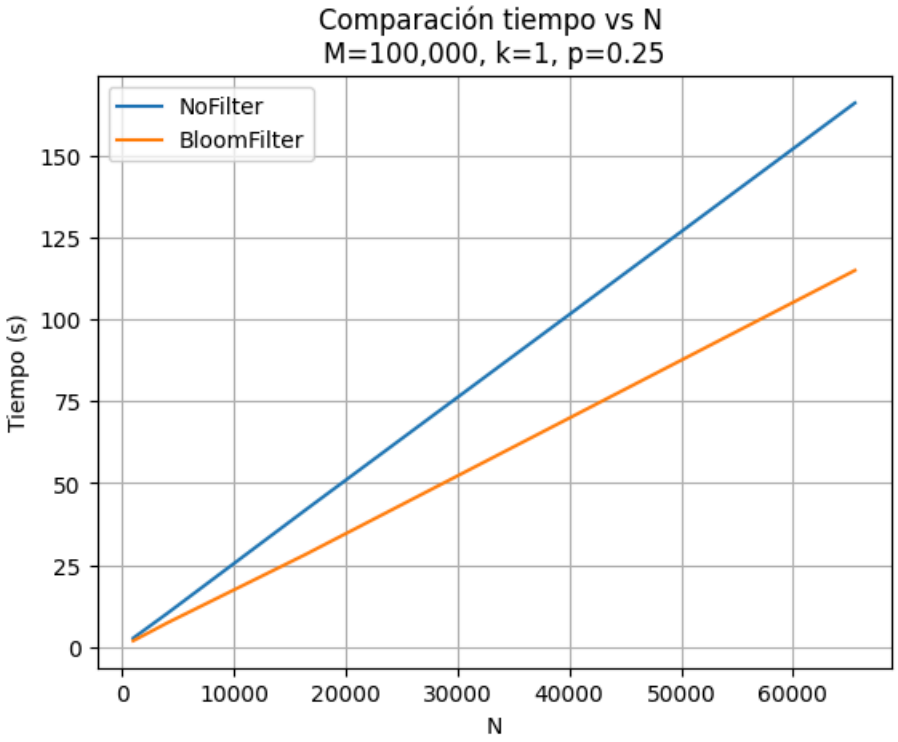
In [2]:
```python
import pandas as pd

def get_fp_rate_df(df: pd.DataFrame, M: int, k: int) -> float:
    df = df[df["M"]  == M]
    df = df[df["k"]  == k]
    df = df[df["DB"] == "BloomFilter"]

    fp = 0      # FP
    total = 0   # FP + TN
    for found, found_in_filter, N, p in zip(df["found"], df["foundInFilter"], df["N"], df["p"]):
        fp += (found_in_filter - found) / N   # Aporte de cada test
        total += (1-p)

    fp_rate = fp / total
    return fp_rate * 100

df = pd.read_csv("../results.csv")


for M in [100_000, 250_000, 500_000, 750_000, 1_000_000, 2_000_000]:
    k = get_opt_k(M)
    print(f"Proporción de falsos positivos  {M=:9,}  {k=:2}:  {get_fp_rate_df(df, M, k):7.4f}%")
```

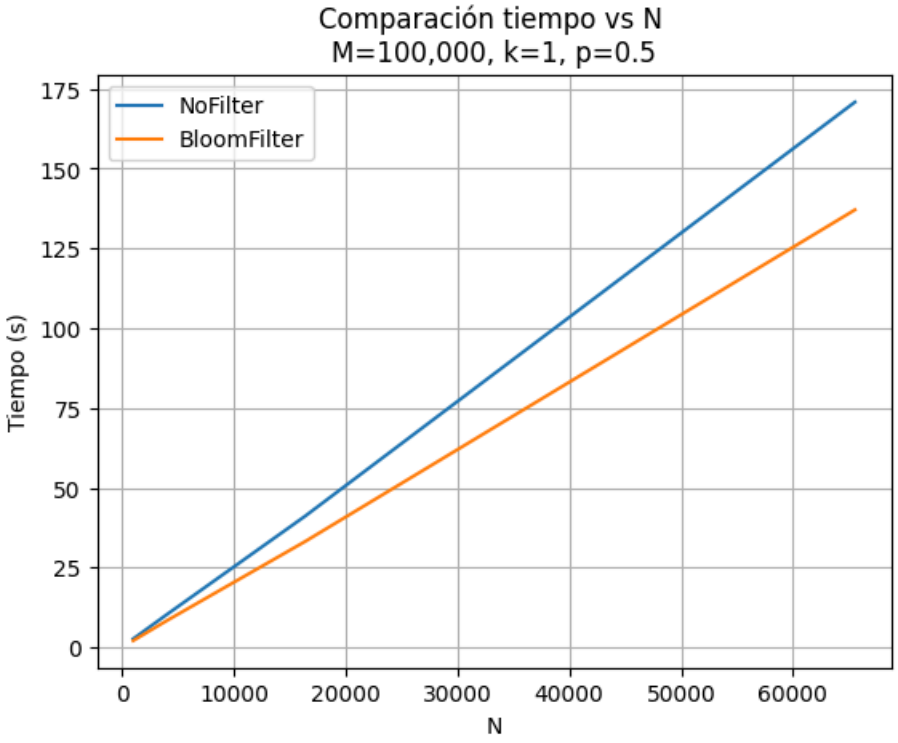
```
Proporción de falsos positivos  M=  100,000  k= 1:  60.5887%
Proporción de falsos positivos  M=  250,000  k= 2:  28.0411%
Proporción de falsos positivos  M=  500,000  k= 4:   7.8459%
Proporción de falsos positivos  M=  750,000  k= 6:   2.2893%
Proporción de falsos positivos  M=1,000,000  k= 7:   0.8389%
Proporción de falsos positivos  M=2,000,000  k=15:   0.0005%
```

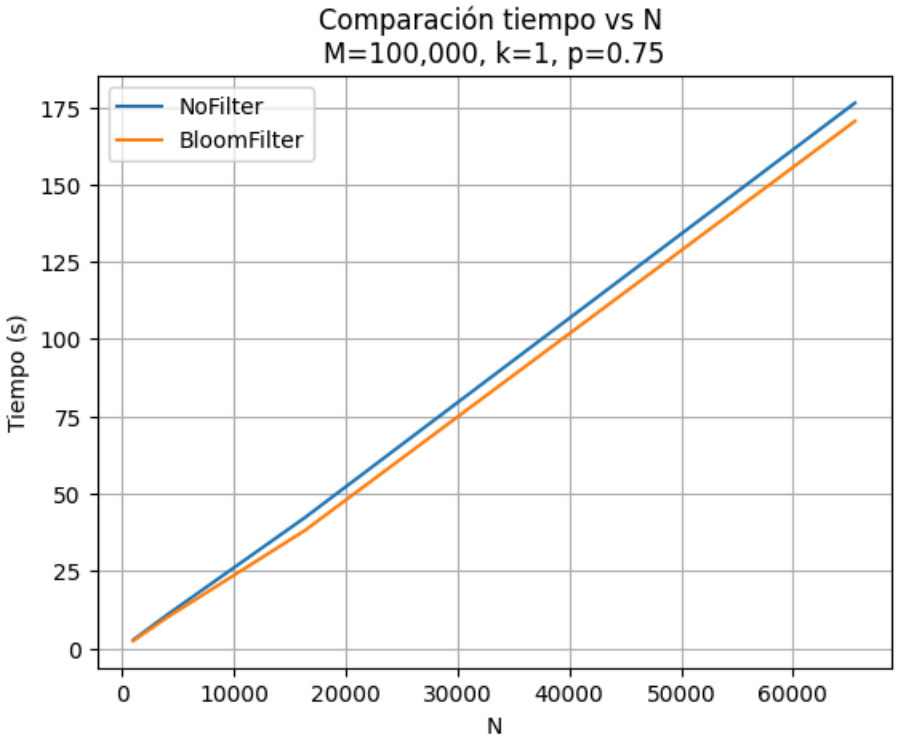
In [3]:
```python
import matplotlib.pyplot as plt

def cmp_N_graph(M: int, k: int, p: float) -> None:
```

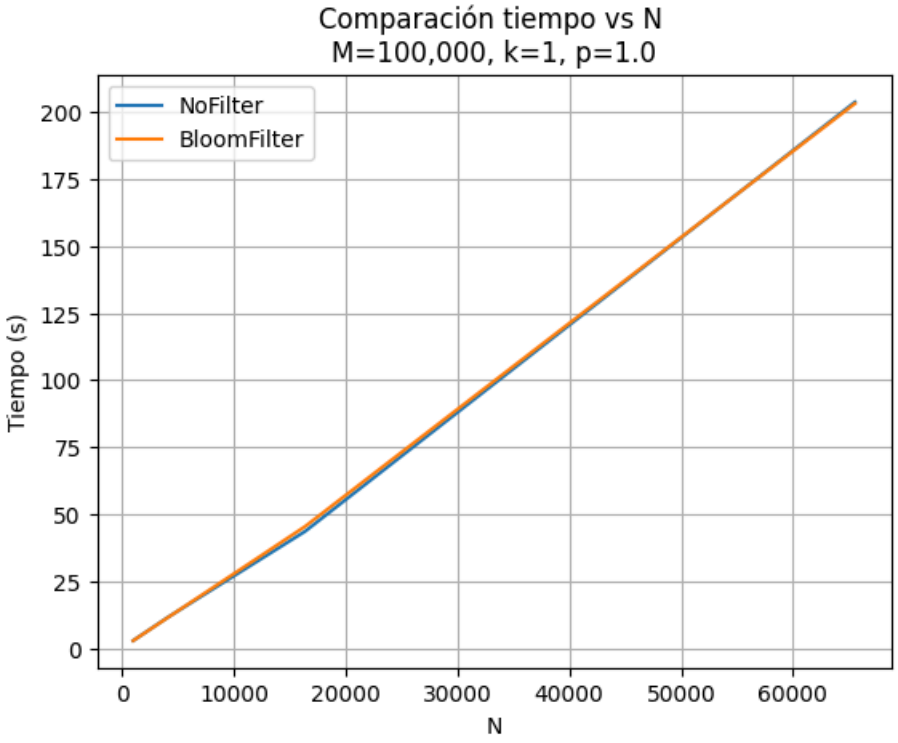
```
    df = pd.read_csv("../results.csv")
    df = df[df["M"] == M]
    df = df[df["k"] == k]
    df = df[df["p"] == p]

    fig, ax = plt.subplots()
    ax.set_title(f"Comparación tiempo vs N \n{M=:,}, {k=}, {p=}")
    ax.set_xlabel("N")
    ax.set_ylabel("Tiempo (s)")

    for db in df["DB"].unique():
        db_df = df[df["DB"] == db]
        ax.plot(db_df["N"], db_df["Time"], label=db)

    ax.legend()
    ax.grid()
    plt.show()


cmp_N_graph(100_000, 1, 0.0)
cmp_N_graph(100_000, 1, 0.25)
cmp_N_graph(100_000, 1, 0.5)
cmp_N_graph(100_000, 1, 0.75)
cmp_N_graph(100_000, 1, 1.0)
```




Comparación tiempo vs N
M=100,000, k=1, p=0.25
NoFilter
BloomFilter
Tiempo (s)
N
0
25
50
75
100
125
150
0
10000
20000
30000
40000
50000
60000

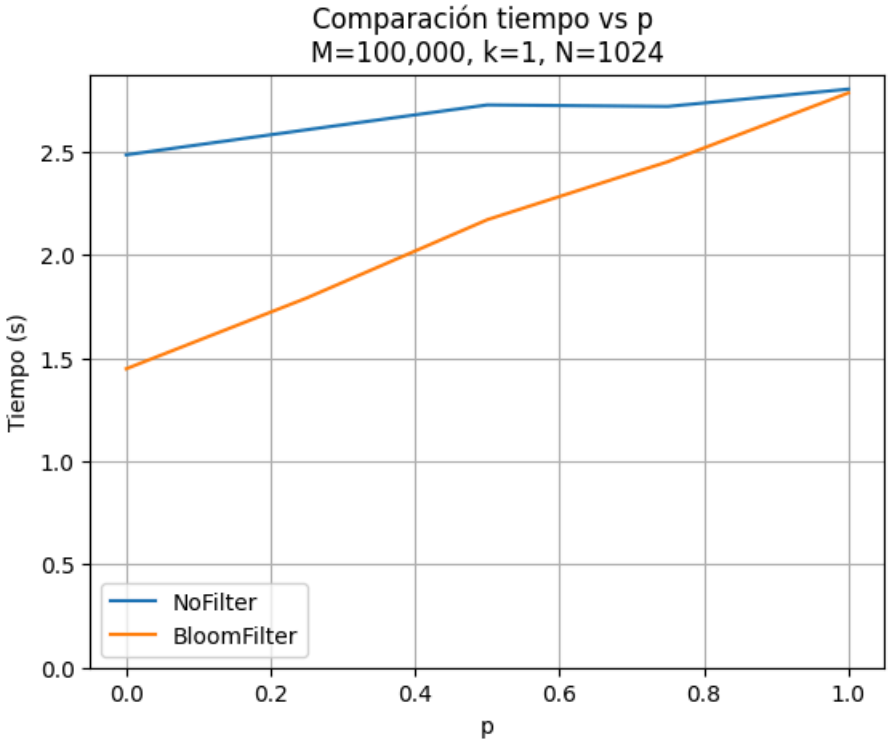

Comparación tiempo vs N
M=100,000, k=1, p=0.5
NoFilter
BloomFilter
Tiempo (s)
N
0
25
50
75
100
125
150
175
0
10000
20000
30000
40000
50000
60000

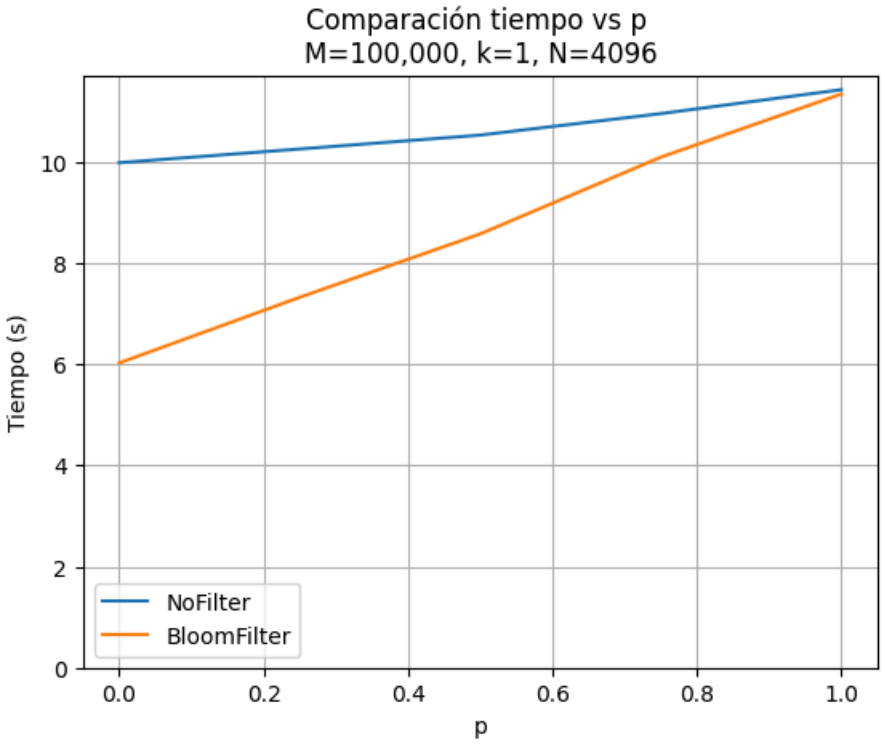


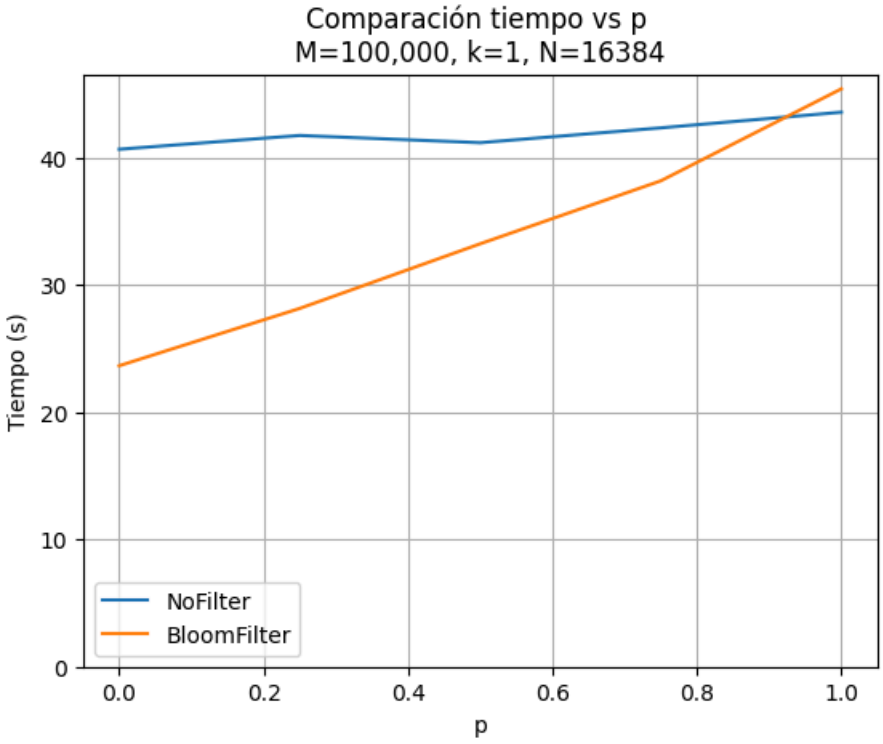


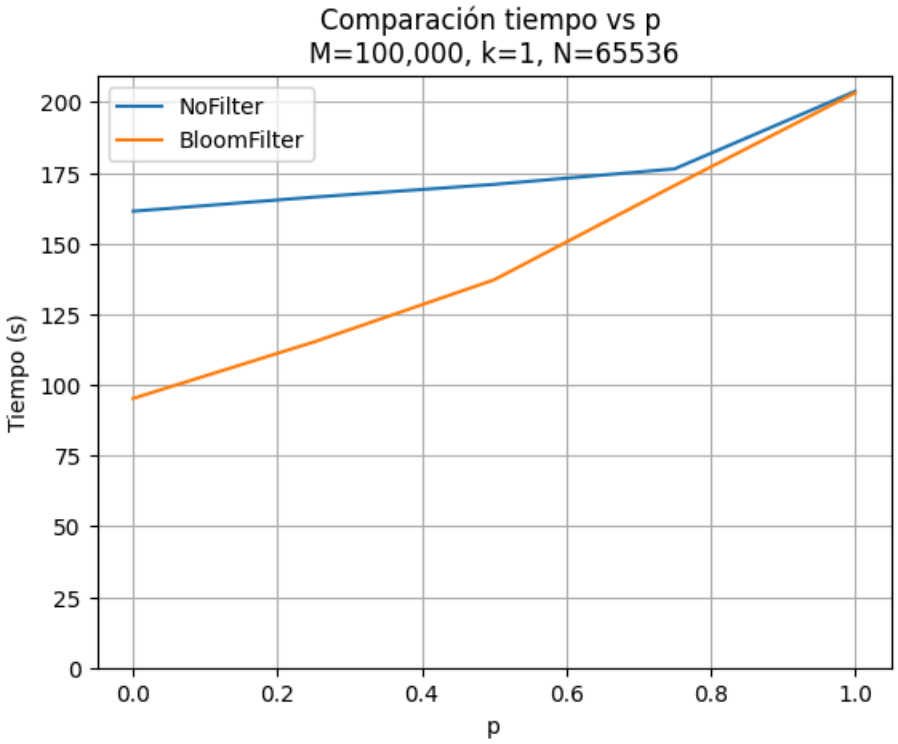
In [4]:
```python
import pandas as pd
import matplotlib.pyplot as plt

def cmp_p_graph(M: int, k: int, N: float) -> None:
    df = pd.read_csv("../results.csv")
    df = df[df["M"] == M]
    df = df[df["k"] == k]
    df = df[df["N"] == N]

    fig, ax = plt.subplots()
    ax.set_title(f"Comparación tiempo vs p \n{M=:,}, {k=}, {N=}")
    ax.set_xlabel("p")
    ax.set_ylabel("Tiempo (s)")
```

```
    for db in df["DB"].unique():
        db_df = df[df["DB"] == db]
        ax.plot(db_df["p"], db_df["Time"], label=db)

    ax.set_ylim(0)
    ax.legend()
    ax.grid()
    plt.show()



cmp_p_graph(100_000, 1, 2**10)
cmp_p_graph(100_000, 1, 2**12)
cmp_p_graph(100_000, 1, 2**14)
cmp_p_graph(100_000, 1, 2**16)
```

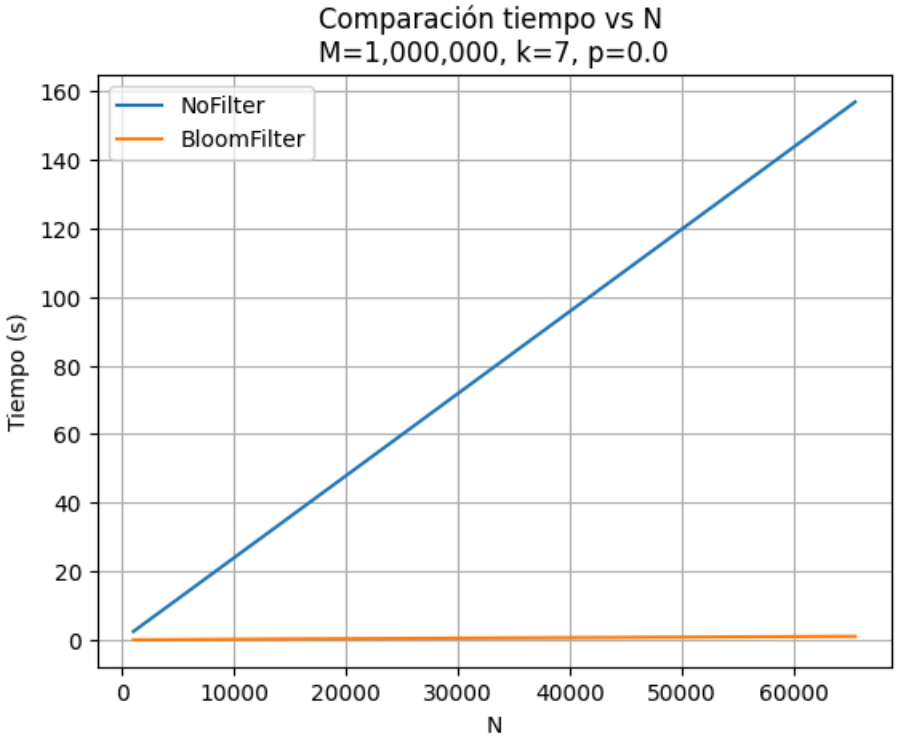


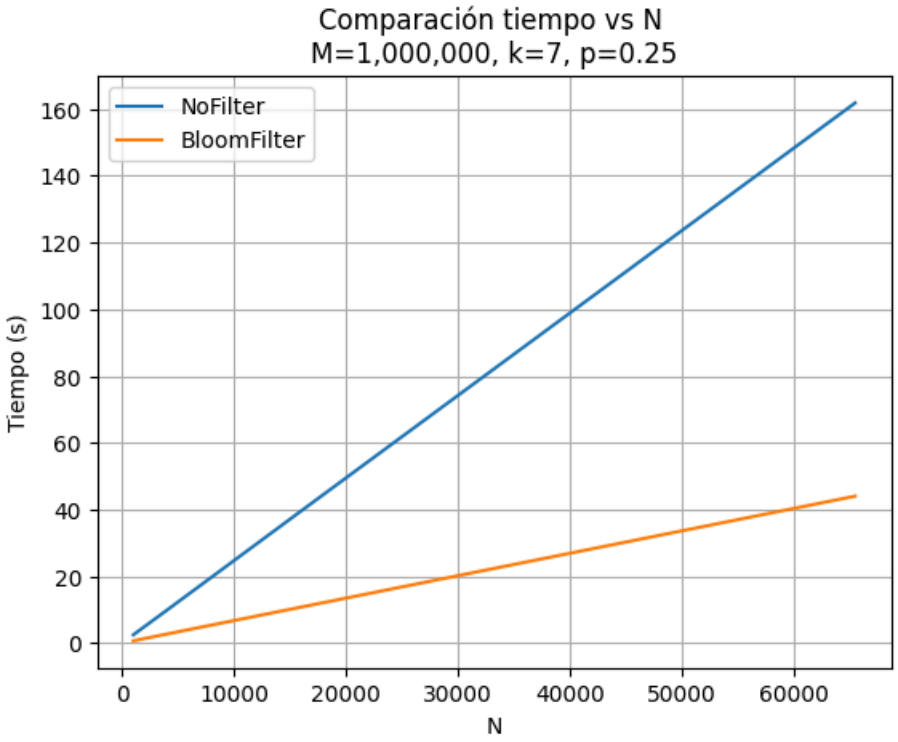

Comparación tiempo vs p
M=100,000, k=1, N=4096
Tiempo (s)
p
NoFilter
BloomFilter
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

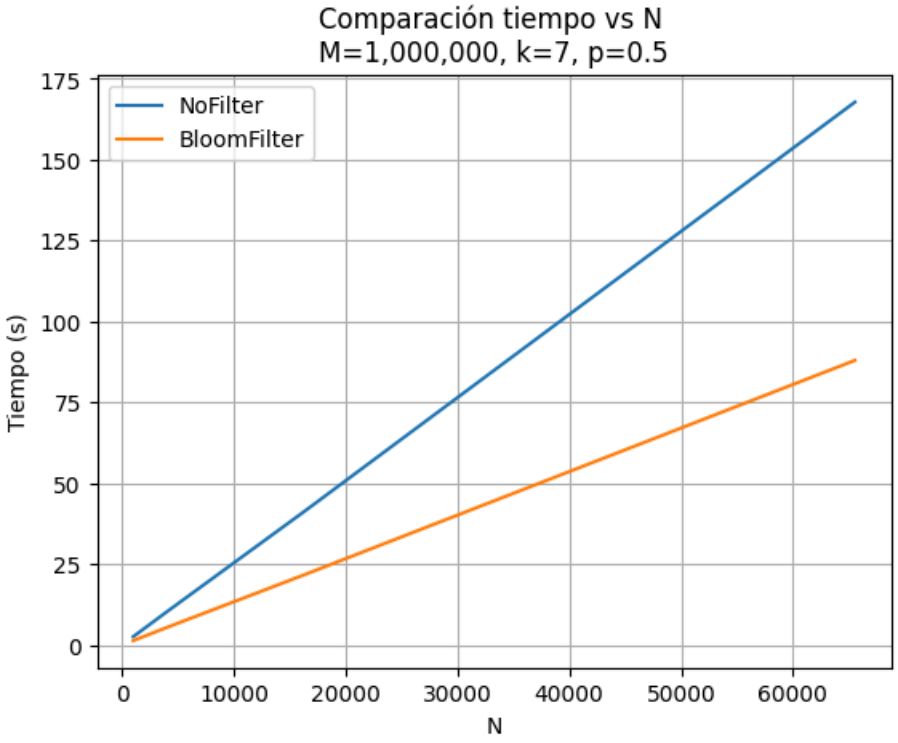

Comparación tiempo vs p
M=100,000, k=1, N=16384
Tiempo (s)
p
NoFilter
BloomFilter
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0

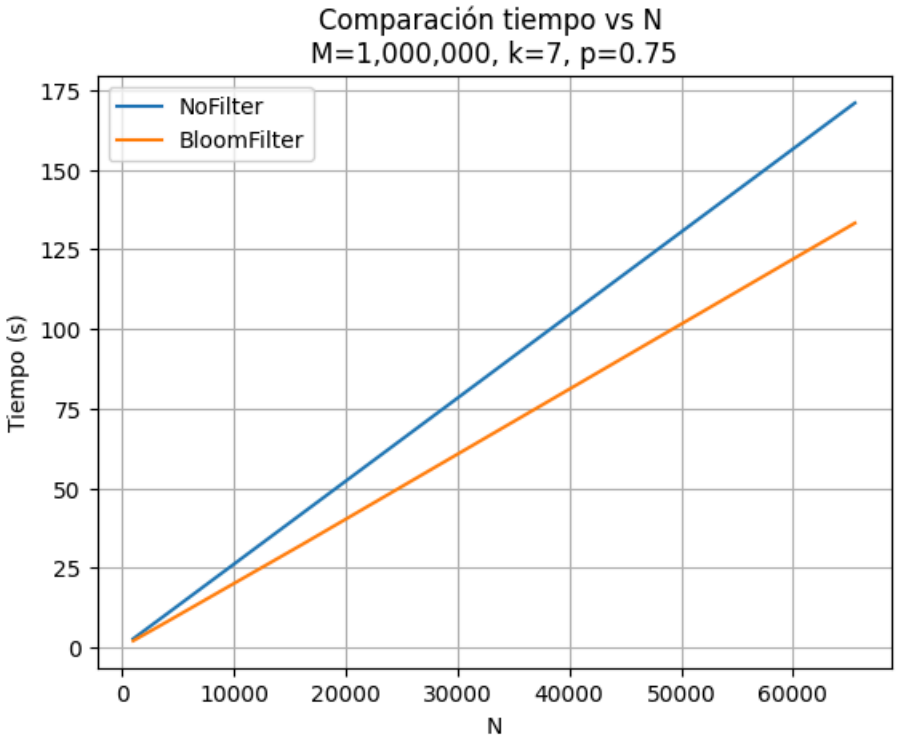


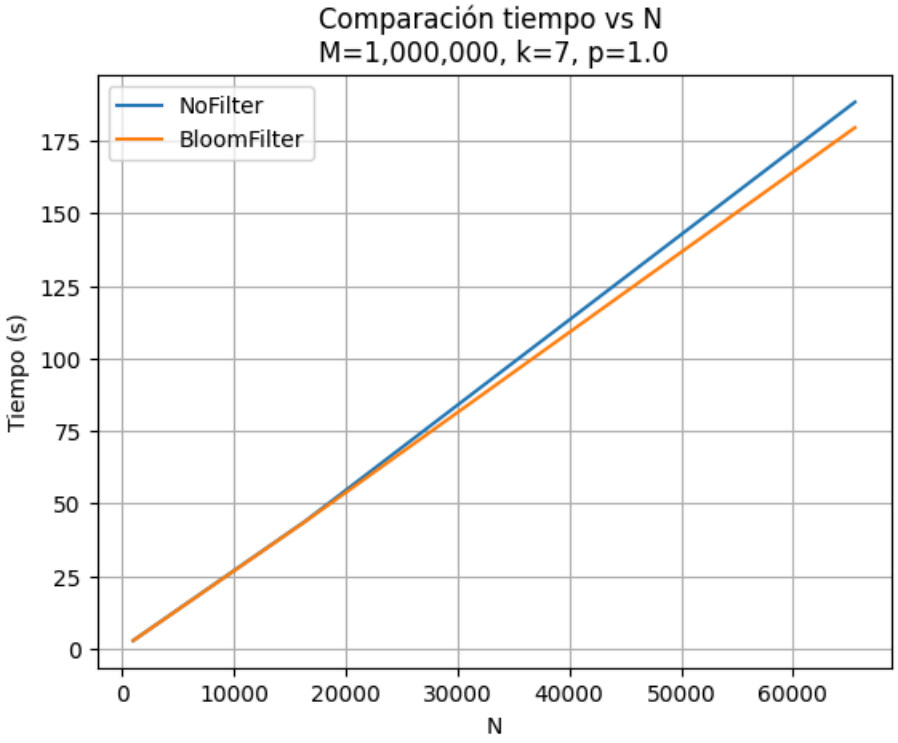
In [5]:
```
cmp_N_graph(1_000_000, 7, 0.0)
cmp_N_graph(1_000_000, 7, 0.25)
cmp_N_graph(1_000_000, 7, 0.5)
cmp_N_graph(1_000_000, 7, 0.75)
cmp_N_graph(1_000_000, 7, 1.0)
```



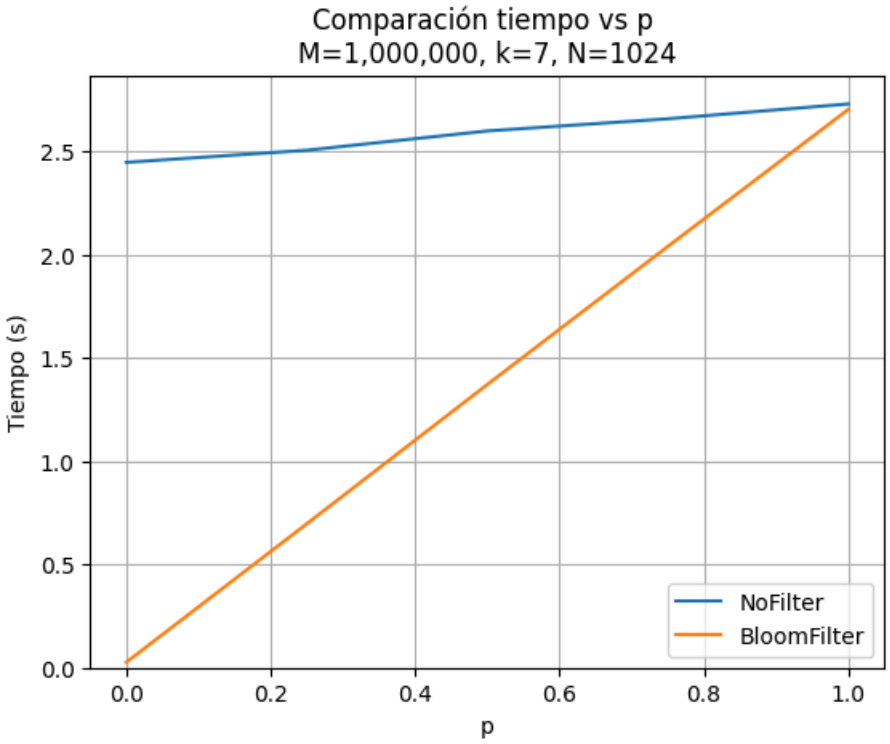

Comparación tiempo vs N
M=1,000,000, k=7, p=0.25
NoFilter
BloomFilter
Tiempo (s)
N
0
20
40
60
80
100
120
140
160
0
10000
20000
30000
40000
50000
60000

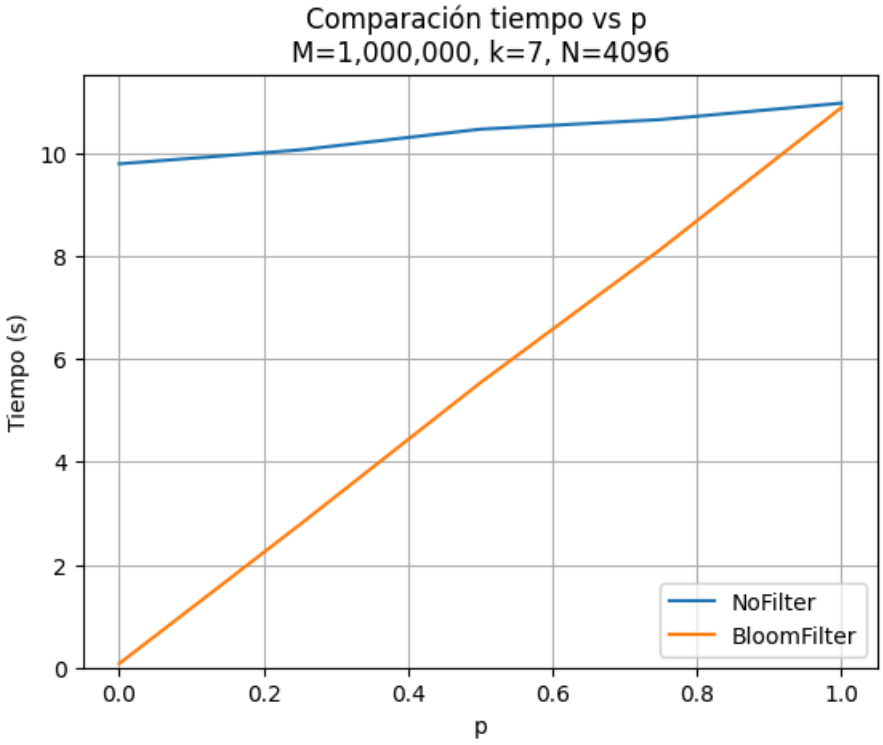

Comparación tiempo vs N
M=1,000,000, k=7, p=0.5
NoFilter
BloomFilter
Tiempo (s)
N
0
25
50
75
100
125
150
175
0
10000
20000
30000
40000
50000
60000

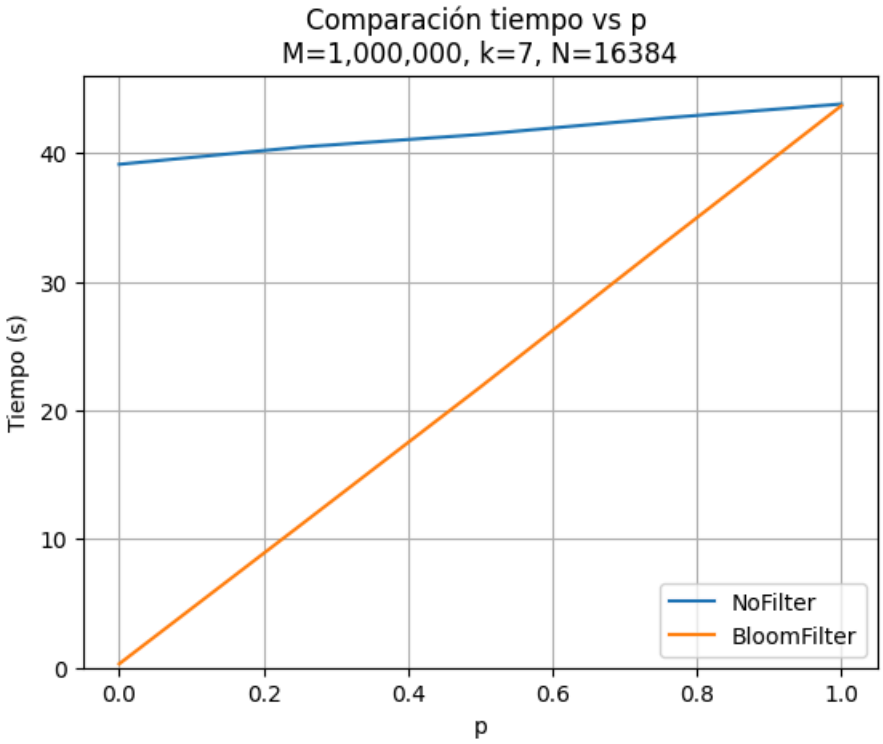


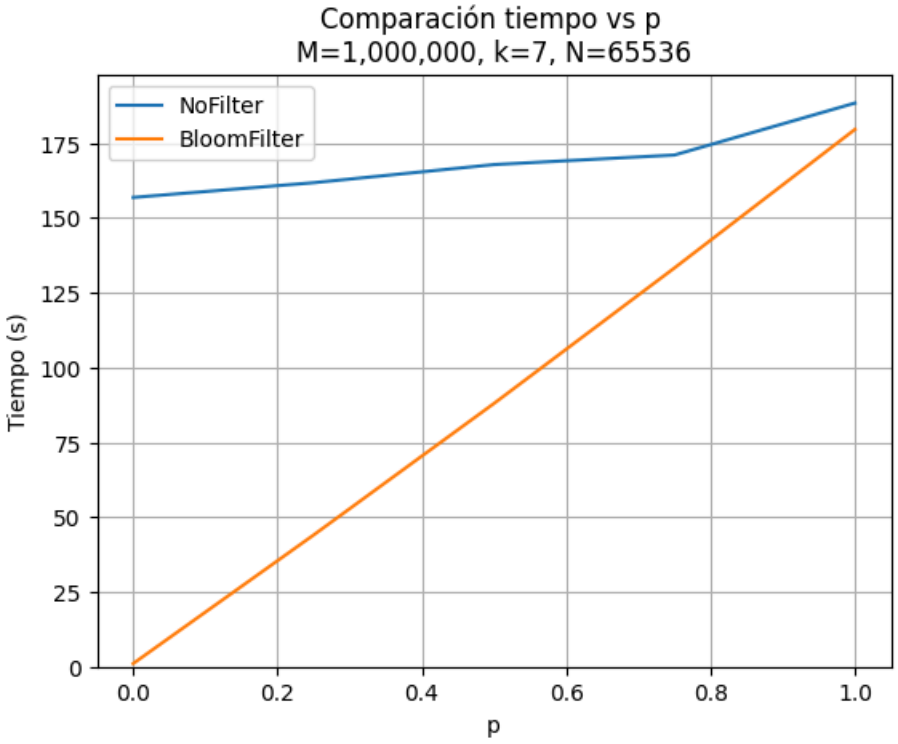


In [6]:
```
cmp_p_graph(1_000_000, 7, 2**10)
cmp_p_graph(1_000_000, 7, 2**12)
cmp_p_graph(1_000_000, 7, 2**14)
cmp_p_graph(1_000_000, 7, 2**16)
```


Comparación tiempo vs p
M=1,000,000, k=7, N=1024
Tiempo (s)
p
NoFilter
BloomFilter
0.0
0.5
1.0
1.5
2.0
2.5
0.0
0.2
0.4
0.6
0.8
1.0


Comparación tiempo vs p
M=1,000,000, k=7, N=4096
Tiempo (s)
p
NoFilter
BloomFilter
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0


Comparación tiempo vs p
M=1,000,000, k=7, N=16384
Tiempo (s)
p
NoFilter
BloomFilter
0
10
20
30
40
0.0
0.2
0.4
0.6
0.8
1.0


Comparación tiempo vs p
M=1,000,000, k=7, N=65536
Tiempo (s)
p
NoFilter
BloomFilter
0
25
50
75
100
125
150
175
0.0
0.2
0.4
0.6
0.8
1.0