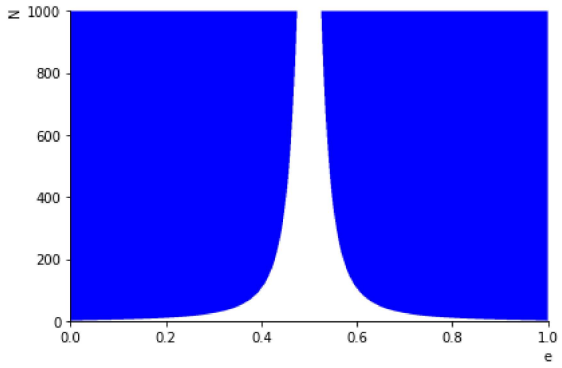
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

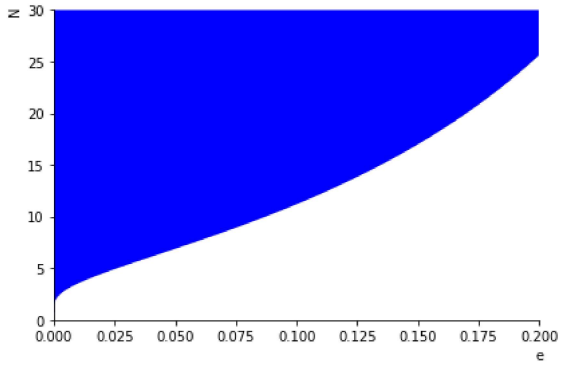
In [ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
from sympy import *
## part c: contout plot (plotted with implicit equations)
e, N = symbols('e N')
plot_implicit(Eq((0.095*0.5**N) / (0.905*(e**(N/2))*((1-e)**(N/2)) + 0.095*0.5**N), 0.99), (e, 0, 1), (N, 0, 150))
plt.show()
```

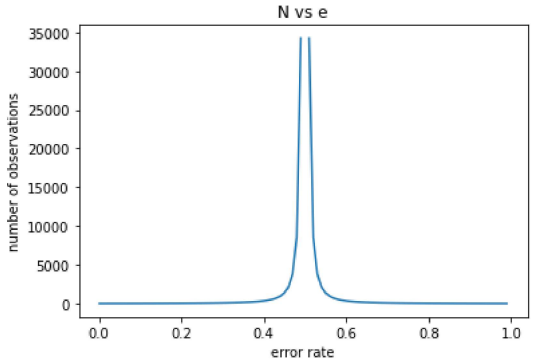


In [ ]:
```
# closer look
e, N = symbols('e N')
plot_implicit(Eq((0.095*0.5**N) / (0.905*(e**(N/2))*((1-e)**(N/2)) + 0.095*0.5**N), 0.99), (e, 0, 0.2), (N, 0, 30))
plt.show()
```

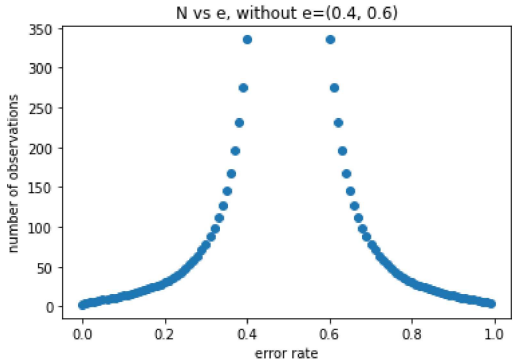


In [47]:
```
## part c: solved analytically

e = [0.001, 0.005] + list(np.arange(0.01, 1, 0.01))
N = [0] * len(e)

aa = 0.95**2
bb = 0.05**2
ab = 1 - aa - bb
alpha = 0.99

for i in range(len(e)):
  x = (0.99 * (aa+bb)) / (ab - alpha * ab)
  numerator = np.log(x)
  denom = np.log(0.5) - 0.5*np.log(e[i]-e[i]**2)
  N[i] = numerator / denom

plt.plot(e, N)
plt.xlabel("error rate")
plt.ylabel("number of observations")
plt.title("N vs e")
plt.show()
```

```
<ipython-input-47-e63e16d82f10>:13: RuntimeWarning: divide by zero encountered in double_scalars
  N[i] = numerator / denom
```



In [50]:
```
e_without_error_4s = list(filter(lambda x: x <= 0.4 or x >= 0.6, e))
N_without_4s = list(filter(lambda x: x < 400, N))
plt.scatter(e_without_error_4s, N_without_4s)
plt.xlabel("error rate")
plt.ylabel("number of observations")
plt.title("N vs e, without e=(0.4, 0.6)")
plt.show()
```



In [51]:
```
import math
e = list(map(lambda x: round(x, 4), e))
integer_N = list(map(lambda x: math.ceil(x) if not math.isinf(x) else x, N))
print(list(zip(e, integer_N)))
```

```
[(0.001, 3), (0.005, 4), (0.01, 5), (0.02, 6), (0.03, 7), (0.04, 8), (0.05, 9), (0.06, 10), (0.07, 11), (0.08, 12), (0.09, 13),
(0.1, 14), (0.11, 15), (0.12, 16), (0.13, 18), (0.14, 19), (0.15, 21), (0.16, 23), (0.17, 24), (0.18, 26), (0.19, 29), (0.2, 3
1), (0.21, 34), (0.22, 37), (0.23, 40), (0.24, 44), (0.25, 48), (0.26, 53), (0.27, 58), (0.28, 64), (0.29, 71), (0.3, 79), (0.3
1, 88), (0.32, 99), (0.33, 112), (0.34, 127), (0.35, 146), (0.36, 168), (0.37, 196), (0.38, 231), (0.39, 277), (0.4, 336), (0.4
1, 416), (0.42, 529), (0.43, 693), (0.44, 945), (0.45, 1363), (0.46, 2134), (0.47, 3799), (0.48, 8555), (0.49, 34240), (0.5, in
f), (0.51, 34240), (0.52, 8555), (0.53, 3799), (0.54, 2134), (0.55, 1363), (0.56, 945), (0.57, 693), (0.58, 529), (0.59, 416),
(0.6, 336), (0.61, 277), (0.62, 231), (0.63, 196), (0.64, 168), (0.65, 146), (0.66, 127), (0.67, 112), (0.68, 99), (0.69, 88),
(0.7, 79), (0.71, 71), (0.72, 64), (0.73, 58), (0.74, 53), (0.75, 48), (0.76, 44), (0.77, 40), (0.78, 37), (0.79, 34), (0.8, 3
1), (0.81, 29), (0.82, 26), (0.83, 24), (0.84, 23), (0.85, 21), (0.86, 19), (0.87, 18), (0.88, 16), (0.89, 15), (0.9, 14), (0.9
1, 13), (0.92, 12), (0.93, 11), (0.94, 10), (0.95, 9), (0.96, 8), (0.97, 7), (0.98, 6), (0.99, 5)]
```

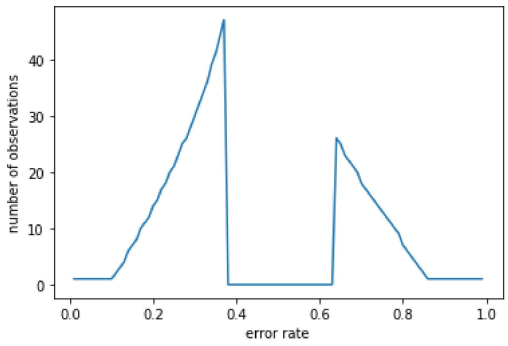
In [ ]:
```
## part d
aa = 0.95**2
bb = 0.05**2
ab = 1 - aa - bb

e = np.arange(0.01, 1, 0.01)
N = np.arange(1, 100000)
min_N = [0] * len(e)


def f(e, n):
  numerator = 0.5**(100+n) * ab
  denom = aa * e**(30+n) * (1-e)**70 + bb * e**70 * (1-e)**(30+n) + ab * 0.5**(n+100)

  return numerator / denom >= 0.99

for i in range(len(e)):
  for n in N:
    if f(e[i], n):
      min_N[i] = n
      break
```

```
<ipython-input-20-fa7eb9e49ee2>:15: RuntimeWarning: invalid value encountered in double_scalars
  return numerator / denom >= 0.99
```

In [ ]:
```
e = list(map(lambda x: round(x, 2), e))
print(list(zip(e, min_N)))
```

```
[(0.01, 1), (0.02, 1), (0.03, 1), (0.04, 1), (0.05, 1), (0.06, 1), (0.07, 1), (0.08, 1), (0.09, 1), (0.1, 1), (0.11, 2), (0.12, 3), (0.13, 4), (0.14, 6), (0.15, 7), (0.16, 8), (0.17, 10), (0.18, 11), (0.19, 12), (0.2, 14), (0.21, 15), (0.22, 17), (0.23, 18), (0.24, 20), (0.25, 21), (0.26, 23), (0.27, 25), (0.28, 26), (0.29, 28), (0.3, 30), (0.31, 32), (0.32, 34), (0.33, 36), (0.34, 39), (0.35, 41), (0.36, 44), (0.37, 47), (0.38, 0), (0.39, 0), (0.4, 0), (0.41, 0), (0.42, 0), (0.43, 0), (0.44, 0), (0.45, 0), (0.46, 0), (0.47, 0), (0.48, 0), (0.49, 0), (0.5, 0), (0.51, 0), (0.52, 0), (0.53, 0), (0.54, 0), (0.55, 0), (0.56, 0), (0.57, 0), (0.58, 0), (0.59, 0), (0.6, 0), (0.61, 0), (0.62, 0), (0.63, 0), (0.64, 26), (0.65, 25), (0.66, 23), (0.67, 22), (0.68, 21), (0.69, 20), (0.7, 18), (0.71, 17), (0.72, 16), (0.73, 15), (0.74, 14), (0.75, 13), (0.76, 12), (0.77, 11), (0.78, 10), (0.79, 9), (0.8, 7), (0.81, 6), (0.82, 5), (0.83, 4), (0.84, 3), (0.85, 2), (0.86, 1), (0.87, 1), (0.88, 1), (0.89, 1), (0.9, 1), (0.91, 1), (0.92, 1), (0.93, 1), (0.94, 1), (0.95, 1), (0.96, 1), (0.97, 1), (0.98, 1), (0.99, 1)]
```

In [ ]:
```
plt.plot(e, min_N)
plt.xlabel("error rate")
plt.ylabel("number of observations")
plt.show()
```



In [ ]: