學號：N26111871 姓名：廖威任

1. Use p and q and simulating the experiment results until it is a success.

```python
p = 0.2 #成功的機率
first_start_time = time.time()
n_count = [0] * 100 #統計每一次的模擬中需要嘗試幾次才能成功
for i in range(10000000): #做一千萬次模擬
    n = 1 #記錄每一次模擬中嘗試的次數，從第一次開始
    while random.random() > p: #利用隨機亂數模擬結果，若失敗則再試一次
        n += 1
    n_count[n - 1] += 1 #將嘗試次數統計在列表中
first_end_time = time.time()
print("First time：", first_end_time - first_start_time)
```

變數 p 代表的是成功的機率，可調為 0.5、0.8 等等，利用隨機亂數做一千萬次模擬，每當嘗試失敗就重新再做一次，計算出每次模擬中第幾次才會成功，並統計出分布結果。

2. Use traditional Inverse Transform method.

```python
second_start_time = time.time()
interval = [] #紀錄每個區間的頭尾
head = 0 #代表每個區間的起始，初始為0
for i in range(1, 101): #將區間劃分為100份
    pr = (1 - p) ** (i - 1) * p #計算出在第i次成功的機率
    interval.append((head,head + pr)) #求出每個區間的頭跟尾
    head = head + pr

inverse_count = [0] * 100 #統計每一次的模擬中需要嘗試幾次才能成功
for i in range(10000000): #做一千萬次試驗
    u = random.random() #利用隨機亂數模擬結果
    for j in range(100): #將隨機亂數和區間做比對
        if u < interval[j][1]: #若符合條件則代表找到該區間
            inverse_count[j] += 1
            break
second_end_time = time.time()
print("Second time：", second_end_time - second_start_time)
```

此段分為兩個部分，首先利用公式$P\{X=n\}=p(1-p)^{n-1}$算出在第 n 次成功才的機率，進而推導出每個機率區間的頭尾。接下來利用隨機亂數做一千萬次模擬，將結果和區間做比對，找到符合該結果的區間，並統計出分布結果。

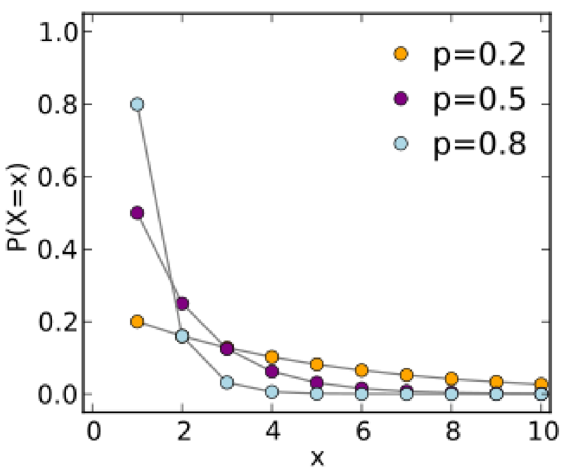
3.Use log

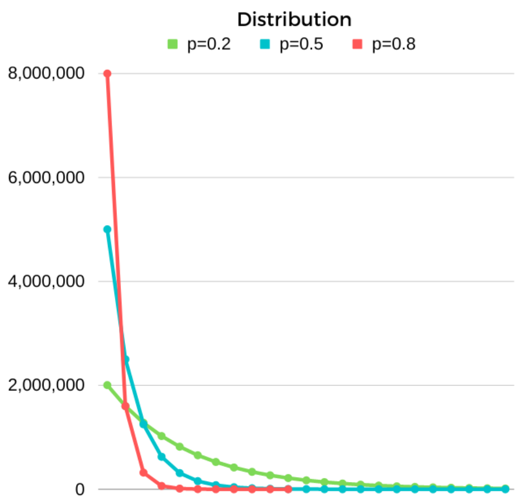
```
third_start_time = time.time()
x_count = [0] * 100 #統計每一次的模擬中需要嘗試幾次才能成功
for i in range(10000000): #做一千萬次模擬
    u = random.random() #利用隨機亂數模擬結果
    x = int(math.log(1 -u) / math.log(1 - p)) + 1 #帶入公式得出在第x次成功
    x_count[x - 1] += 1
third_end_time = time.time()
print("Third time：", third_end_time - third_start_time)
```

利用隨機變數做一千萬次模擬，接結果帶入公式：

X=Int(log(U)/log(q))+1 ≡ X=Int(log(1-U)/log(q))+1 後可以得出在第幾次才成功，並統計出分布結果。

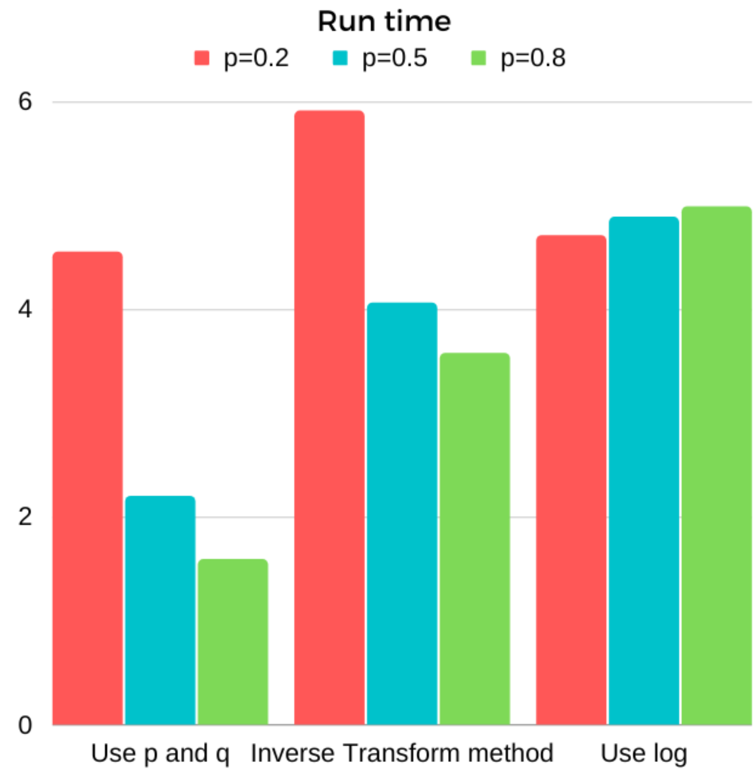
4.Distribution





左圖為幾何分布的機率質量函數(圖取自維基百科)，右圖為本次作業模擬的分布結果，圖表顯示本次模擬和實際函數相似。

5.Run time



以下將 Use p and q 簡稱為法 a、Inverse Transform method 簡稱為法 b、Use log 簡稱為法 c。

此圖表為使用不同的方法來模擬幾何分布的 run time，可以發現在 p=0.2 時，法 a 和法 c 的 run time 相當，且比法 b 快，這是因為法 b 除了要劃分出區間外，在產生出隨機變數後還要在各個區間做比對，時間複雜度為 O(nk)(n 為資料數，k 為區間數)，然而另外兩個方法並不會受到區間數的影響，法 a 只需當

試到成功後即可換下一筆亂數，法 c 只需將亂數帶入公式後即可換下一筆亂數。

從另一個角度來看，當 p 提升後，法 a 和法 b 的 run time 大幅降低，因為在每次的模擬中法 a 能夠有更高的機率在前幾次嘗試就成功，相同的，法 b 也能夠在前幾次的比對就找到符合的區間而不用繼續向後比對，然而法 c 的 run time 卻不受到 p 值的影響,因為每次的模擬都是利用帶入公式來產生結果。