Problem 1：

定义计算的函数:

```python
def generate_events(lambda_, simulation_time):
    time = 0
    events = []
    while time < simulation_time:
        time += np.random.exponential(1 / lambda_)
        events.append(time)
    return np.array(events)
```

将时间差算出:

```python
# 模拟产生事件序列
events_1 = generate_events(lambda_1, simulation_time)
events_2 = generate_events(lambda_2, simulation_time)
print(events_1)
print(events_2)

# 模拟计算事件1的时间差
delta = []
for i in range(len(events_1)-1):
    delta_i = events_1[i+1] - events_1[i]
    delta.append(delta_i)
```

算出 count 并且画图:

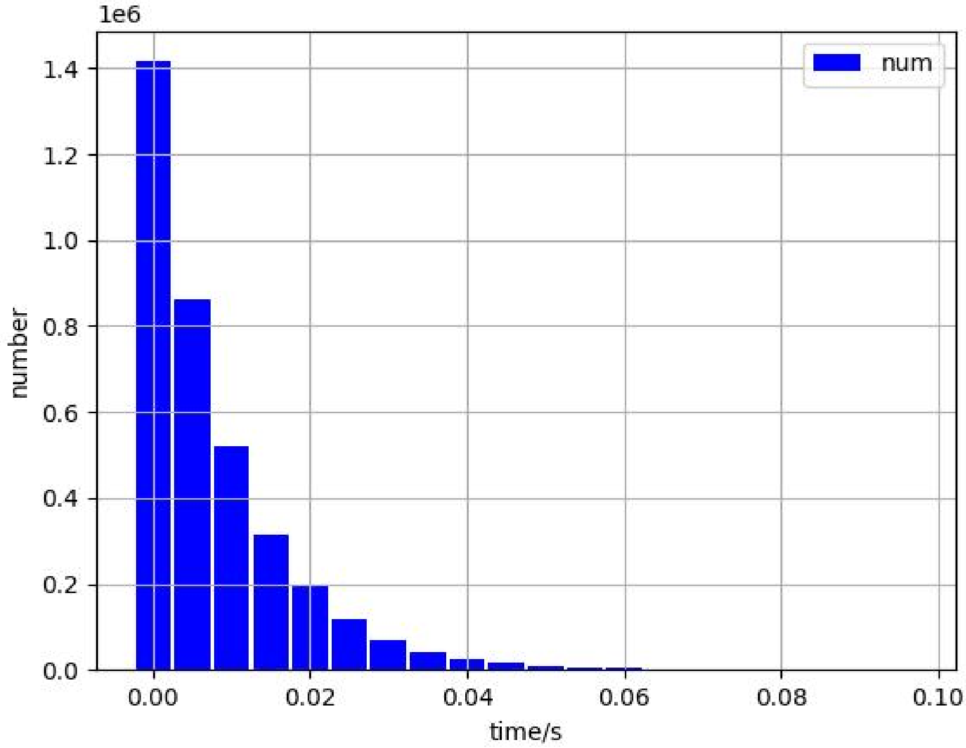
```python
# 以0.005s为时间间隔计算次数
number_1 = []
for i in range(20):
    count = 0
    for j in range(len(delta)):
        if i * 0.005 < delta[j] < (i+1) * 0.005 :
            count += 1
    number_1.append(count)
time_1 = [i * 0.005 for i in range(20)]

# 画出number_1的直方分布图
plt.bar(time_1, number_1 , width = 0.0045, color = 'blue', label = 'num')
plt.xlabel('time/s')
plt.ylabel('number')
plt.grid(True)
plt.legend()
plt.show()
```

接下来得到结果和图像:

上面是输出 event_1,下面是输出 event_2

```
[1.05696944e-02 3.61012657e-02 4.39516854e-02 ... 3.59999853e+04
 3.59999947e+04 3.60000164e+04]
[4.27016747e-03 1.93748559e-02 2.85522806e-02 ... 3.59999942e+04
 3.59999950e+04 3.60000026e+04]
```

最后得到 event_1 的统计图像

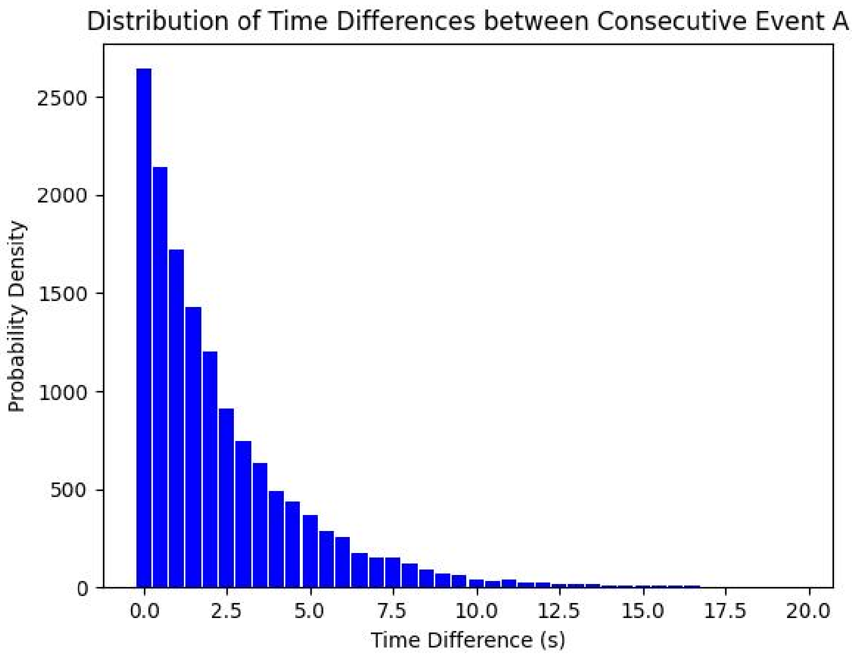


关于事例 A:

我利用了指针来优化时间复杂度，以确保程序可以在比较短的时间内算出。

```python
# 计算事件A的时间差
1 usage
def calculate_event_time_differences(events_1, events_2, tau):
    # 对事件列表进行排序
    events_1_sorted = np.sort(events_1)
    events_2_sorted = np.sort(events_2)

    event_times = []
    i = 0  # 光电管1事件索引
    j = 0  # 光电管2事件索引

    while i < len(events_1_sorted) and j < len(events_2_sorted):
        time_diff = np.abs(events_1_sorted[i] - events_2_sorted[j])
        if time_diff < tau:
            event_times.append(min(events_1_sorted[i], events_2_sorted[j]))

        # 移动指针
        if events_1_sorted[i] < events_2_sorted[j]:
            i += 1
        else:
            j += 1

    return np.array(event_times)
```

最后统计得到图像（统计代码与上面的相似故不发出，我勇的 0.5s 为单位进行统计，详见代码包）：



我认为这个分布满足指数分布。

然后计算出事例次数再算出事例率:

```
Number of events A found: 14344
Estimated rate of events A: 0.39844444444444443
```

模拟得到的事例次数是 14344，事例率是 0.398444，由于理论值为 0.3997，因此理论值和模拟值的误差是 0.3141%。

Problem 2:

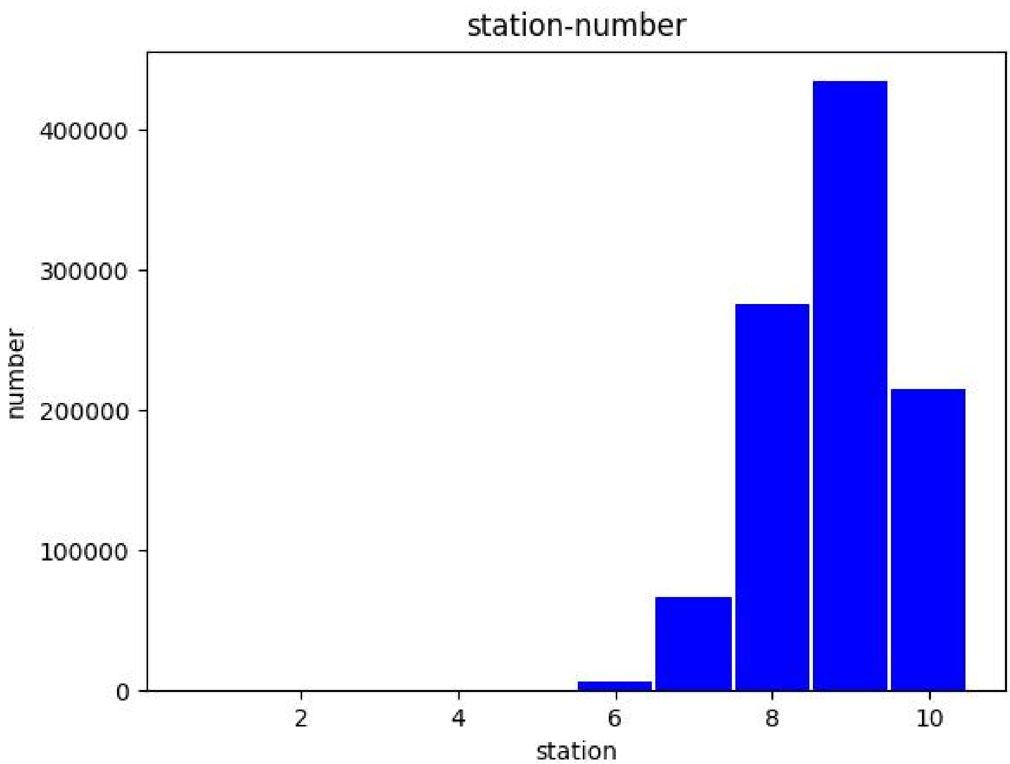
```
for k in range(number):
    c = []
    for i in range(20):
        c_i = random.randint( a: 1, b: 10)
        c.append(c_i)

    count = []
    #计数
    for j in range(len(c)):
        if c[j] in count:
            continue
        else:
            count.append(c[j])

    count1 = len(count)
    countnumber += count1
    station[count1 - 1] += 1
```

利用循环来进行计算，模拟二十次的车站选择。然后计入总数 number，之后将之提取入列表方便作图，之后把图像做出来就行了。

输出结果:

站数的平均值： 8.784915

再输出图像:



Problem 3:

(1) 每次价格的期望：

$$E[X_n] = p\cdot 1.7X_{n-1} + (1-p)\cdot 0.5X_{n-1} = 1.1X_{n-1}$$

每天价格方差：$V[X_n] = p(1.7X_{n-1}-1.1X_{n-1})^2 + (1-p)(1.1X_{n-1}-0.5X_{n-1})^2$

$$\Rightarrow V[X_n] = 0.81X_{n-1}^2$$

故 $n\to\infty$ 时：$\log(X_n)$ 近似服从正态分布

$$E(\log X_n) = \log(X_0) + n\log(1.1)$$

$$V(\log X_n) = n\log(0.81)$$

(2). $n\to\infty$ 时：$E(X_n) = 1.1^n X_0 \to \infty$

(3) n 趋于无穷大时：设涨70%的次数是 n_1

则 $\Omega = C_n^{n_1}$. 当 Ω 取极大值时：

$n_1 = [\frac{n}{2}]$. 则涨与跌的次数一样.

$$\Rightarrow X_n = 0.85^{\frac{n}{2}} X_0 \to 0.$$

故最有可能趋近于0

这是我对于前三问的阐述。

对于第四小问，我先定义了随机涨跌的函数，并且让他运行 n 次，总共进行 simulations 次。

```python
def simulate_stock_price(n, simulations):
    # 初始化价格为 100 元
    prices = np.full(simulations, fill_value: 100.0)

    # 模拟每一天的价格变化
    for _ in range(n):
        # 生成随机涨跌因子
        factors = np.random.choice( a: [1.7, 0.5], size=simulations)
        # 更新价格
        prices *= factors

    # 计算平均价格
    average_price = np.mean(prices)
    return average_price
```

然后再计算理论值，通过运行一亿次得到结果。

```python
def theoretical_average_price(n):
    return 100 * (1.1 ** n)


# 模拟参数
simulations = 100000000
ns = [1, 10, 30, 60]

# 计算模拟值并打印结果
for n in ns:
    simulated_average_price = simulate_stock_price(n, simulations)
    theoretical = theoretical_average_price(n)
    print(f"n={n}: 模拟平均价格={simulated_average_price:.2f}, 理论值={theoretical:.2f}")
```

得到结果：

```
n=1: 模拟平均价格=109.99, 理论值=110.00
n=10: 模拟平均价格=259.47, 理论值=259.37
n=30: 模拟平均价格=1731.97, 理论值=1744.94
n=60: 模拟平均价格=33110.91, 理论值=30448.16
```

对于第五小问：

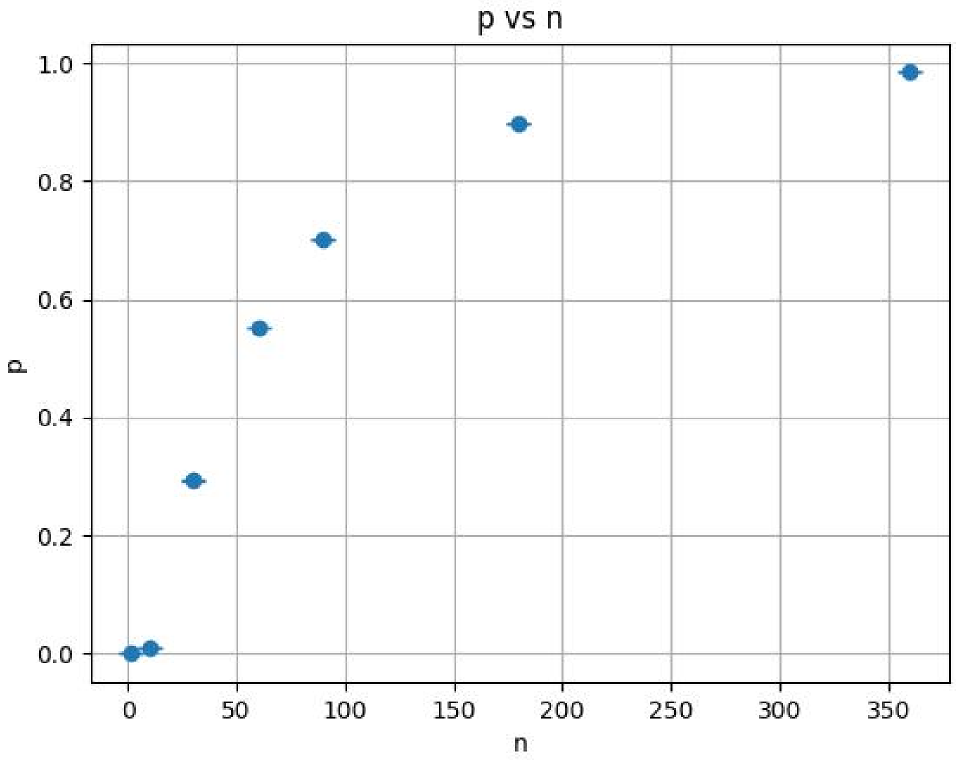
我先在前一问的基础上加上了小于 1 的概率。

```python
def simulate_stock_price(n, simulations):
    # 初始化价格为 100 元
    prices = np.full(simulations, fill_value: 100.0)

    # 模拟每一天的价格变化
    for _ in range(n):
        # 生成随机涨跌因子
        factors = np.random.choice( a: [1.7, 0.5], size=simulations)
        # 更新价格
        prices *= factors

    # 计算小于 1 的价格数量
    count_less_than_1 = np.sum(prices < 1)
    # 计算概率
    probability = count_less_than_1 / simulations
    return probability
```

然后写出误差棒并且再次画出图像:

```python
# 计算概率并绘制图表
probabilities = []
for n in ns:
    probability = simulate_stock_price(n, simulations)
    probabilities.append(probability)

# 计算误差棒
error = np.sqrt(np.array(probabilities) * (1 - np.array(probabilities)) / simulations)
```

随后得到结论:


p vs n
p
n
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350