به نام خدا

آز پروژه درس سیستم عامل فاز دوم

اعضا

حسین علی حسینی

مهراد میلانلو

پرنیان رضویان پور

علیرضا نوروزی

بهار ۱۴۰۲

آزمایش ۱: پیاده سازی کردن شبیه ساز زمان بندی

۱)

```
0: Arrival of Task 12 (ready queue length = 1)
0: Run Task 12 for duration 2 (ready queue length = 0)
1: Arrival of Task 13 (ready queue length = 1)
2: Arrival of Task 14 (ready queue length = 2)
2: IO wait for Task 12 for duration 1
2: Run Task 14 for duration 1 (ready queue length = 1)
3: Arrival of Task 15 (ready queue length = 2)
3: Wakeup of Task 12 (ready queue length = 3)
3: IO wait for Task 14 for duration 2
3: Run Task 12 for duration 2 (ready queue length = 2)
5: Wakeup of Task 14 (ready queue length = 3)
5: Run Task 14 for duration 1 (ready queue length = 2)
6: Run Task 15 for duration 2 (ready queue length = 1)
8: Run Task 15 for duration 1 (ready queue length = 1)
9: Run Task 13 for duration 2 (ready queue length = 0)
11: Run Task 13 for duration 2 (ready queue length = 0)
13: Run Task 13 for duration 2 (ready queue length = 0)
15: Run Task 13 for duration 1 (ready queue length = 0)
16: Stop
```

۲)

```
0: Arrival of Task 12 (ready queue length = 1)
0: Run Task 12 for duration 2 (ready queue length = 0)
1: Arrival of Task 13 (ready queue length = 1)
2: Arrival of Task 14 (ready queue length = 2)
2: IO wait for Task 12 for duration 1
2: Run Task 13 for duration 2 (ready queue length = 1)
3: Arrival of Task 15 (ready queue length = 2)
3: Wakeup of Task 12 (ready queue length = 3)
4: Run Task 14 for duration 1 (ready queue length = 3)
5: IO wait for Task 14 for duration 2
5: Run Task 15 for duration 2 (ready queue length = 2)
7: Wakeup of Task 14 (ready queue length = 3)
7: Run Task 12 for duration 2 (ready queue length = 3)
9: Run Task 14 for duration 1 (ready queue length = 2)
10: Run Task 13 for duration 4 (ready queue length = 1)
14: Run Task 15 for duration 1 (ready queue length = 1)
15: Run Task 13 for duration 1 (ready queue length = 0)
16: Stop
```

آزمایش ۲: به سوییهره وری ۱۰۰ درصد!

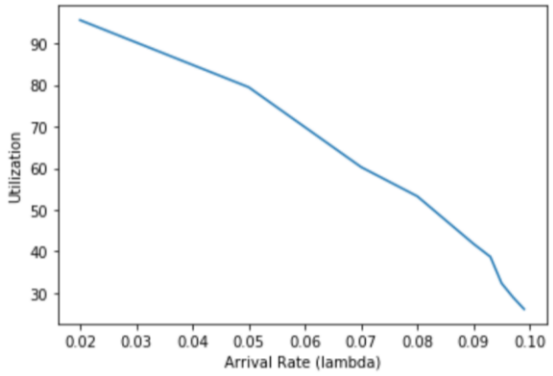
۱) میدانیم امیدریاضی توزیع نمایی با پارامتر λ برابر است با $\frac{1}{\lambda}$ پس برای آن که میانگین فاصله دو تسک متوالی برابر M شود باید $\lambda = \frac{1}{M}$ باشد.

۲) برای اینکه بهروری پروسسور مان برابر ۵۰ درصد شود، باید اکسپکتد زمان فاصله بین هر دو تسک متوالی برابر $2M$ شود که پروسسور ما نیمه اولش را کار کند و نیمه دوم را منتظر بماند پس $\lambda = \frac{1}{2M}$

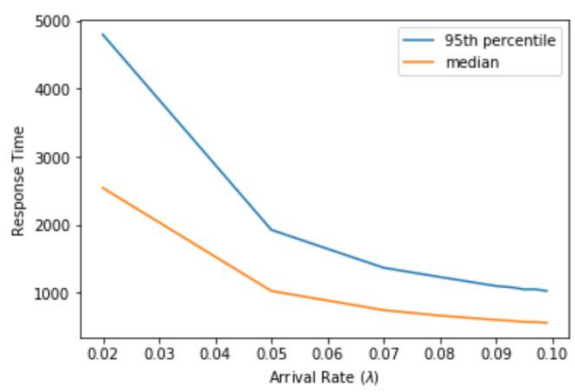
۳) همانطور که در تکه کد زیر دیده میشود لامبدا ای که داده میشود به نوعی مقدار اکسپکتد توزیع نمایی ماست.

```
def make_exp_arrivals(arrival_rate, service_time, n):
    """Make n tasks of exponential distributed arrival intervals and lengths"""
    arrivals = np.cumsum(np.random.exponential(1 / arrival_rate, n))
    lengths = [service_time for _ in range(n)]
    return [Task(arr, run, indefinite, nowait) for arr, run in zip(arrivals, lengths)]
```

حال نمودار را تحلیل میکنیم برای اکسپکتد های کم یعنی فاصله بین تسک ها کم باشد پروسسور ما بی وقفه در حال انجام کار است اما اگر امیدریاضی فاصل بین تسک ها از M بیشتر شود معطلی پروسسور بیشتر شده و بهروری پایین خواهد آمد. مثلا برای $M = 10$



۴) برای این بخش هر چه فاصله بین تسک ها کمتر باشد یعنی پارامتر کوچک تر صف طولانی تری برای اجرا تشکیل شده و زمان پاسخ دهی بسیار زیاد میشود.



۵) از آنجایی که طول تسک ها تفاوتی ندارند از هر زمان بند دیگه ای استفاده کنیم نتیجه بهتری نخواهیم گرفت همانطور که اگر از *RR* استفاده کنیم نتیجه بدتری هم حاصل میشود.

۶) اگر بهروری را بخواهیم به ۱۰۰ برسانیم یعنی پروسسور ما بی وقفه باید کار کند پس یعنی زمان رسیدن تسک ها باید کم باشد یعنی تسک ها خیلی زودتر از آن که نوبت به اجرای آنها برسد آمده اند و این یعنی هر تسک مدت زمان زیادی معطل میماند. پس تاخیر زیادی در پاسخگویی خواهیم داشت.

آزمایش ۳: رعایت عدالت بین فعالیت ها

۱) زیرا تسک T_i تا انجام نشود خبری از تسک T_{i+1} نیست پس یعنی در هر لحظه ۲ تسک برای انجام دادن وجود دارد پس در صف بیشتر از ۲ تسک نداریم.

۲) با توجه به شبیه سازی که در مثال زیر انجام داده ایم میبینیم که احتمال آنکه تسک اول *T* از تسک دوم *S* بیشتر طول بکشد برابر ۵۰ درصد است.

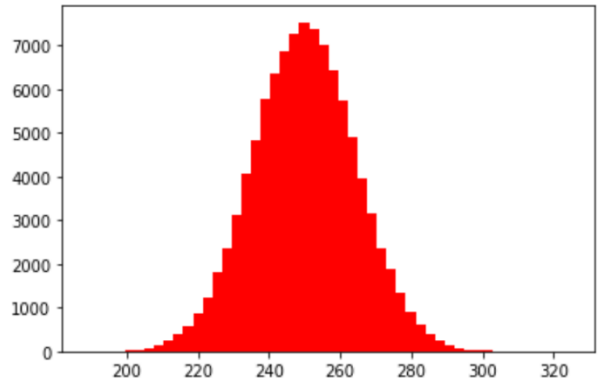
```
lmbda = 5
N = 1e6
success = 0
for _ in range(int(N)):
    S_task = np.random.uniform(0, lmbda)
    T_task = np.random.uniform(0, lmbda)
    if S_task > T_task:
        success += 1
print(success / N)
```

۳) طبق قضیه حد مرکزی جمع تعداد زیادی رویداد *i.i.d* از توزیع نرمال پیروی میکند و مقدار اکسپکتد و واریانس آن برابر مقادیر زیر است:

$$\mu = \Sigma E[S_i]$$
$$var = \Sigma Var(S_i)$$

و با تکه کد زیر هیستوگرام را ساخته ایم که ملاحظه کنید چقدر شبیه توزیع نرمال است.

```
n = 100
sums = [sum(np.random.uniform(0, lmbda, n)) for _ in range(N)]
plt.hist(sums, color="red", bins=50)
plt.show()
```



۴) این جواب از روی تقسیم ۲ متغیر تصادفی نرمال قابل محاسبه است.

Derivation [edit source]

A way of deriving the ratio distribution of $Z = X/Y$ from the joint distribution of the two other random variables X, Y, with joint pdf $p_{X,Y}(x,y)$, is by integration of the following form[3]

$$p_Z(z) = \int_{-\infty}^{+\infty} |y|\, p_{X,Y}(zy, y)\, dy.$$

If the two variables are independent then $p_{XY}(x,y) = p_X(x)p_Y(y)$ and this becomes

$$p_Z(z) = \int_{-\infty}^{+\infty} |y|\, p_X(zy)p_Y(y)\, dy.$$

This may not be straightforward. By way of example take the classical problem of the ratio of two standard Gaussian samples. The joint pdf is

$$p_{X,Y}(x,y) = \frac{1}{2\pi} \exp\left(-\frac{x^2}{2}\right) \exp\left(-\frac{y^2}{2}\right)$$

Defining $Z = X/Y$ we have

$$p_Z(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} |y| \exp\left(-\frac{(zy)^2}{2}\right) \exp\left(-\frac{y^2}{2}\right) dy$$
$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} |y| \exp\left(-\frac{y^2\left(z^2+1\right)}{2}\right) dy$$

Using the known definite integral $\int_0^\infty x \exp\left(-cx^2\right) dx = \frac{1}{2c}$ we get

$$p_Z(z) = \frac{1}{\pi(z^2+1)}$$

which is the Cauchy distribution, or Student's t distribution with $n = 1$

۵) در زیر شبیه سازی را مشاهده میکنید.

```
# Feel free to change this while developing if it takes too long to run
TRIALS = 10000
N = 1.1

unfair_count = 0
fair_count = 0

for _ in range(TRIALS):
    elapsed_0 = sum(np.random.uniform(0, lmbda, n))
    elapsed_1 = sum(np.random.uniform(0, lmbda, n))
    if elapsed_0 / elapsed_1 > N or elapsed_1 / elapsed_0 > N:
        unfair_count += 1
    else:
        fair_count += 1

print("Fraction of runs that were unfair:", unfair_count / (fair_count + unfair_count))
```

```
Fraction of runs that were unfair: 0.2392
```

این عدد بسیار به n و شباهت تسک ها به توزیع نرمال بستگی دارد ولی برای این اعداد در ۲۳ درصد مواقع اختلاف آنها از ۱۰ درصد هم بیشتر است برای رد ادعای بهنام باید از تست های آماری دیگر استفاده کرد.

پایان