第一题

Given $\{X_1, X_2, \ldots, X_n\}$ be the random sample of X with the mean μ and the variance σ^2 ,proof $\frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \rightarrow^d N(0,1)$

Given $M_X(t) = e^{\frac{1}{2}t^2}$,for $X \sim N(0,1)$

Thus $M_Y(t) = e^{\frac{1}{2}t^2}$,for $X \sim N(0,1) \Leftrightarrow Y \rightarrow^d N(0,1)$,for $Y = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}}$ whose MGF is exist for all t

$$
\begin{aligned}
M_Y(t) = E(e^{tY}) &= E(e^{t\frac{\bar{X}-\mu}{\sigma/\sqrt{n}}}) \\
&= E(e^{t\frac{\sum X_i-\mu}{\sigma/\sqrt{n}}}) \\
&= E(e^{\frac{t}{\sqrt{n}}\sum\frac{X_i-\mu}{\sigma}}) \\
&= E(e^{\frac{t}{\sqrt{n}}\frac{X_1-\mu}{\sigma}})\ldots E(e^{\frac{t}{\sqrt{n}}\frac{X_n-\mu}{\sigma}}) \\
&= E(e^{\frac{t}{\sqrt{n}}\frac{X_n-\mu}{\sigma}})^n \\
&= M_{\frac{X_n-\mu}{\sigma}}(\frac{t}{\sqrt{n}})^n
\end{aligned}
$$

By the Taylor's Theorem:

$$
\begin{aligned}
M_{\frac{X_n-\mu}{\sigma}}(\frac{t}{\sqrt{n}}) &= 1 + \frac{t}{\sqrt{n}}E(\frac{X_n-\mu}{\sigma}) + \frac{(\frac{t}{\sqrt{n}})^2}{2}E(\frac{X_n-\mu}{\sigma})^2 + (\frac{t}{\sqrt{n}})^2 * h(\frac{t}{\sqrt{n}}) \\
&= 1 + \frac{(\frac{t}{\sqrt{n}})^2}{2} + (\frac{t}{\sqrt{n}})^2 * h(\frac{t}{\sqrt{n}})
\end{aligned}
$$

Thus,we have

$$
M_Y(t) = M_{\frac{X_n-\mu}{\sigma}}(\frac{t}{\sqrt{n}})^n = (1 + \frac{(\frac{t}{\sqrt{n}})^2}{2} + (\frac{t}{\sqrt{n}})^2 * h(\frac{t}{\sqrt{n}}))^n
$$

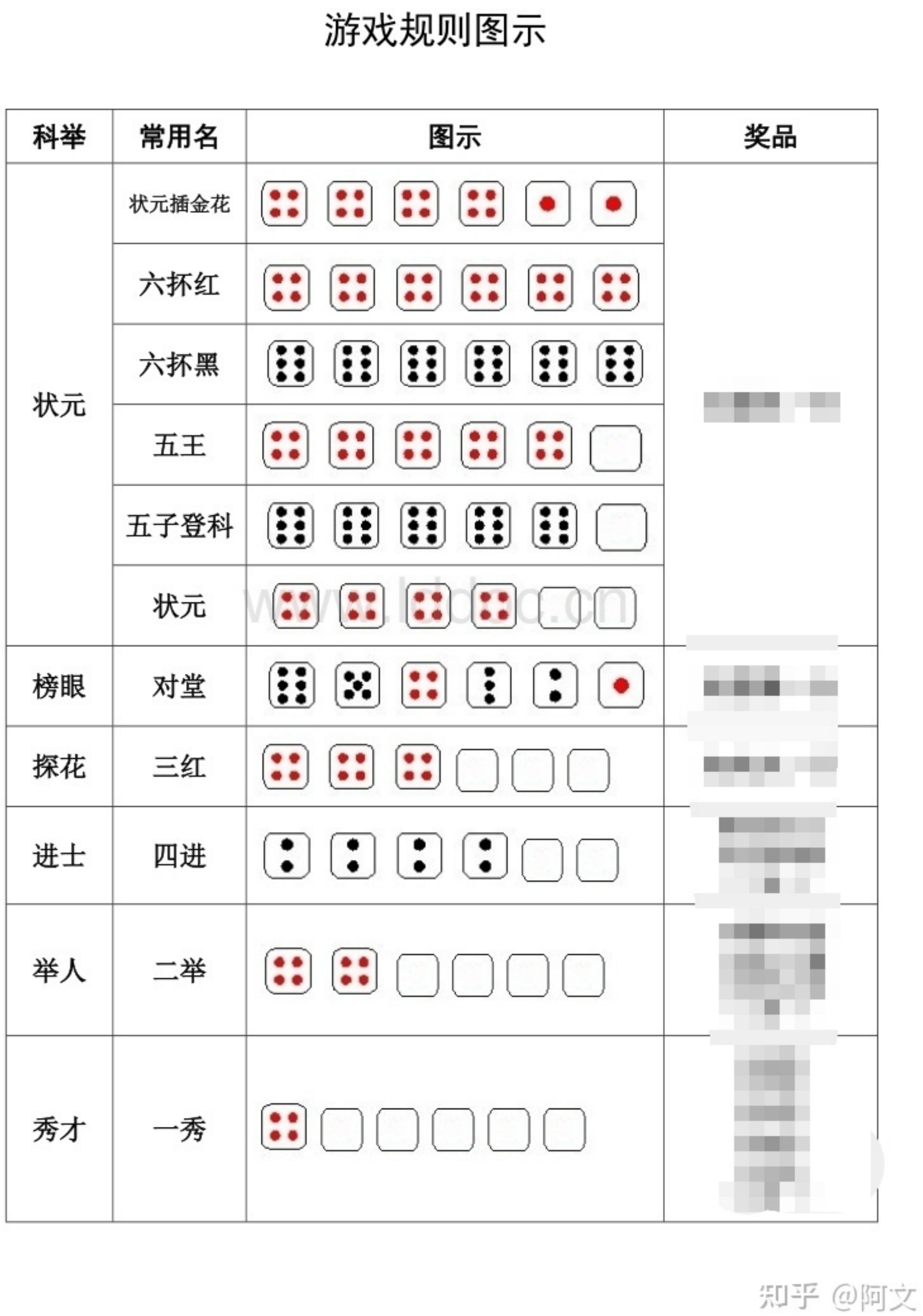
$$
\lim_{n\to\infty} M_Y(t) = \lim_{n\to\infty}(1 + \frac{(\frac{t}{\sqrt{n}})^2}{2} + (\frac{t}{\sqrt{n}})^2 * h(\frac{t}{\sqrt{n}}))^n = e^{\frac{t^2}{2}}
$$

Consequently,we know that $Y = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \sim N(0,1)$

- For a general case that MGF is not necessarily exist, refer to [A Moment Generating Function Proof of the Lindeberg–Lévy Central Limit Theorem]

第二题

Generate the probability result of the game 博饼



游戏规则图示

科举	常用名	图示	奖品
状元	状元插金花	4 4 4 4 1 1	
	六抔红	4 4 4 4 4 4	
	六抔黑	6 6 6 6 6 6	
	五王	4 4 4 4 4	
	五子登科	6 6 6 6 6	
	状元	4 4 4 4	
榜眼	对堂	6 5 4 3 2 1	
探花	三红	4 4 4	
进士	四进	2 2 2 2	
举人	二举	4 4	
秀才	一秀	4	

In [ ]:

```python
import math
def split(sample,num_dies):
    runs = int(math.modf(len(sample)/num_dies)[-1])
    sample = sample[:runs*num_dies].reshape([runs,num_dies])
    return sample
```

In [ ]:

```python
import matplotlib.pyplot as plt
from scipy.stats import rv_discrete
from numpy import array
from numpy import where,sum
def is_jinhua(set):
    #four = where(set==4,1,0)
    #two = where(set==2,1,0)
    return sum(where(set==4,1,0))==4 & sum(where(set==2,1,0))==2

def is_liuhong(set):
    return sum(where(set==4,1,0))==6

def is_liuhei(set):
    return sum(where(set==6,1,0))==6

def is_wuwang(set):
    return sum(where(set==4,1,0))==5

def is_wuzi(set):
    return sum(where(set==6,1,0))==5

def is_zhuangyuan(set):
    return sum(where(set==6,1,0))==4

def is_duitang(set):
    indicator = (array([sum(where(set==i,1,0))==1 for i in range(1,7)]))
    return sum(indicator)==len(indicator)

def is_sanhong(set):
    return sum(where(set==4,1,0))==3

def is_sijin(set):
    return sum(where(set==2,1,0))==4

def is_erju(set):
    return sum(where(set==4,1,0))==2

def is_yixiu(set):
    return sum(where(set==4,1,0))==1
```

In [ ]:

```
def result(sample):
    sample = split(sample,6)
    jinhua = [is_jinhua(i) for i in sample]
    liuhong = [is_liuhong(i) for i in sample]
    liuhei = [is_liuhei(i) for i in sample]
    wuwang = [is_wuwang(i) for i in sample]
    wuzi = [is_wuzi(i) for i in sample]
    zhuangyuan = [is_zhuangyuan(i) for i in sample]
    duitang = [is_duitang(i) for i in sample]
    sanhong = [is_sanhong(i) for i in sample]
    sijin = [is_sijin(i) for i in sample]
    erju = [is_erju(i) for i in sample]
    yixiu = [is_yixiu(i) for i in sample]
    X = sum(array([jinhua,liuhong,liuhei,wuwang,wuzi,zhuangyuan,duitang,sanhong,sijin,e
    return X
```

In [ ]:

```
def generate_sample(n):
    prob_value = (array([1,2,3,4,5,6]),array([1/6,1/6,1/6,1/6,1/6,1/6]))
    num_dies = 6
    die = rv_discrete(a=1,b=6,values=prob_value)
    rst = die.rvs(size=n)
    return rst
sample = generate_sample(1000000)
fig, ax = plt.subplots(figsize=(12, 8))
names = ['jinhua','liuhong','liuhei','wuwang','wuzi','zhuangyuan','duitang','sanhong','
count = result(sample)/len(sample)
plt.stem(names,count)
plt.xlabel('awarded combinations')
plt.ylabel('frequecy of combinations')
plt.title('Bobing Stimulation')
plt.show()
```