به نام خدا

تمرین اول کامپیوتری

آمار و احتمال مهندسی

استاد ابوالقاسمی

فاطمه جلیلی

به شماره دانشجویی:810199398

زمان تحویل : 1400/8/14

فهرست

سوال اول – بازی برد و باخت :

- **حل تئوری قسمت الف :**

می دانیم در کل 36 حالت برای پرتاب دو تاس وجود دارد ، متغیر تصادفی S را به عنوان مجموع دو عدد رو شده در نظر می گیریم ، S می تواند هر یک از مقادیر 2 تا 12 را اختیار کند تابع جرم احتمال S را مطابق جدول زیر بدست می آوریم :

	مقادیر رو شده که مجموع آن ها برابر S می شود	احتمال
S=2	(1,1)	1/36
S=3	(1,2)-(2,1)	2/36
S=4	(1,3)-(3,1)-(2,2)	3/36
S=5	(1,4)-(2,3)-(3,2)-(4,1)	4/36
S=6	(1,5)-(2,4)-(3,3)-(4,2)-(5,1)	5/36
S=7	(1,6)-(2,5)-(3,4)-(4,3)-(5,2)-(6,1)	6/36
S=8	(2,6)-(3,5)-(4,4)-(5,3)-(6,2)	5/36
S=9	(3,6)-(4,5)-(5,4)-(6,3)	4/36
S=10	(4,6)-(5,5)-(6,4)	3/36
S=11	(5,6)-(6,5)	2/36
S=12	(6,6)	1/36

اکنون احتمال حالت های ممکن برد ، باخت یا ادامه دادن در مرحله اول را مطابق زیر بدست می آوریم:

result	S	Probability (first round)
win	7,11	$\frac{6}{36}+\frac{2}{36}=\frac{8}{36}$
lose	2,3,12	$\frac{1}{36}+\frac{2}{36}+\frac{1}{36}=\frac{4}{36}$
continue	4,5,6,8,9,10	$\frac{3}{36}+\frac{4}{36}+\frac{5}{36}+\frac{5}{36}+\frac{4}{36}+\frac{3}{36}=\frac{24}{36}$

*در ادامه گزارش کار حالت سوم یعنی continue را حالت تساوی می نامیم.

پس احتمال برد ، باخت یا تساوی بازیکن در مرحله ی اول بدین ترتیب مشخص شد اما در صورتی که بازی ادامه داشته باشد بسته به S بدست آمده در مرحله ی اول که می تواند از بین اعداد 4،5،6،8،9،10 باشد ادامه روند را به 6 حالت تقسیم می کنیم و مطابق قانون احتمال کل احتمال برد را بدست می آوریم .

- جدا از S در مرحله اول که آن را S_1 می نامیم روند محاسبه احتمال برد در هر یک از 6 حالت گفته شده مطابق زیر و یکسان خواهد بود :

مطابق آزمایش برنولی با فرض اینکه $P(S=k)=p$ باشد که در آن $k\epsilon\ (4,5,6,8,9,10)$ داریم :

برای اینکه بازیکن در مرحله ی nام ببرد لازم است تا در هر یک از n-1 مرحله ی قبلی به جز مرحله ی اول که تساوی کرده نه برده باشد و نه باخته باشد .

احتمال $S_1 = k$ مطابق فرض برابر p است

احتمال نبردن و نباختن در هر مرحله برابر احتمال مساوی نشدن مجموع اعداد رو شده با S_1 و 7 نیاوردن است پس برابر $1-p-\frac{6}{36}$ خواهد بود .

احتمال $S=k$ شدن در مرحله ی nام برای برد هم مطابق فرض p است .

بازیکن باید در مرحله 1 مساوی کند و در $n-2$ مرحله بعدی نبرد و نبازد پس احتمال برد در مرحله ی n ام به شرط اینکه در مرحله ی اول $S=k$ باشد طبق توضیحات و طبق اصل ضرب برابر است با:

$$p \times p \times \left(1-p-\frac{6}{36}\right)^{n-2}$$

پس برای محاسبه احتمال کل در هر یک از 6 حالت گفته شده لازم است تا مقدار بالا را به ازای $n=2$ تا $n=\infty$ جمع کنیم که به صورت حد مجموع دنباله ی هندسی بدست خواهد آمد

$$p^2 + p^2\left(1 - p - \frac{1}{6}\right) + p^2\left(1 - p - \frac{1}{6}\right)^2 + \cdots$$
$$= p^2(1 + \left(1 - p - \frac{1}{6}\right) + \left(1 - p - \frac{1}{6}\right)^2 + \cdots)$$
$$p^2\left(\frac{1}{1 - \left(1 - p - \frac{1}{6}\right)}\right) = \frac{p^2}{p + \frac{1}{6}}$$

طبق توضیحات بالا اکنون همانطور که گفته شد در هر یک از 6 حالت احتمال برد را محاسبه می کنیم و طبق قانون احتمال کل آن هارا باهم جمع می کنیم تا احتمال برد به شرط اینکه در مرحله اول حالت تساوی رخ داده باشد بدست بیاید

طبق اطلاعات جدول 1 و 2 داریم :

k	Probability of winning where $S_1 = k$
4	$\left(\frac{3}{36}\right)^2 \times \frac{9}{36}$
5	$\left(\frac{4}{36}\right)^2 \times \frac{36}{10}$
6	$\left(\frac{5}{36}\right)^2 \times \frac{36}{11}$
8	$\left(\frac{5}{36}\right)^2 \times \frac{36}{11}$
9	$\left(\frac{4}{36}\right)^2 \times \frac{36}{10}$
10	$\left(\frac{3}{36}\right)^2 \times \frac{9}{36}$

پس احتمال کل بردن به شرط تساوی در مرحله اول مطابق مجموع زیر طبق قانون احتمال کل بدست می آید:

$$2\left(\frac{9}{36^2}\times\frac{36}{9}+\frac{16}{36^2}\times\frac{36}{10}+\frac{25}{36^2}\times\frac{36}{11}\right)=\frac{1}{18}\times\frac{268}{55}=\frac{134}{495}$$

پس احتمال کل برد شامل برد در مرحله ی اول به علاوه ی احتمال برد در مراحل بعدی به شرط تساوی در مرحله اول که طبق مجموع بالا محاسبه شد ، برابر خواهد بود با :

$$P(winning)=\frac{134}{495}+\frac{8}{36}=\frac{244}{495}$$

- **قسمت ب و ج :**

معرفی توابع مورد نیاز:

برای انجام این دو قسمت 4 تابع تعریف می کنیم

-rollDice() : در روند بازی بازیکن دو تاس را پرتاب مب کند که اعداد رندوم خواهند آمد و طبق آن پیش خواهد رفت ، برای تبدیل این قسمت از بازی به کد این تابع را تعریف می کنیم که با هر بار صدا زدن آن دو عدد رندوم بین 1 تا 6 تولید می کند و مجموع آن ها را به عنوان خروجی به ما می دهد.

-game() : بازی مورد نظر قوانینی دارد که در صورت پروژه توضیح داده شده و در نتیجه این قوانین نتیجه هر دست طبق سه حالت برد ، باخت ، یا ادامه دادن تعیین می شود

برای شبیه سازی این قسمت و نوشتن کد آن این تابع را تعریف می کنیم که به کمک خروجی تابع rollDice ادامه روند را به سه قسمت تقسیم می کند و در هر قسمت نتیجه بازی را خروجی می دهد .

اگر 7 و 11 رو شوند(خروجی تابع rolldice) این تابع win را خروجی می دهد
اگر 2،3 و12 رو شوند (خروجی تابع rolldice) این تابع lose را خروجی می دهد
اگر هر عدد دیگری رو شود تحت یک حلقه while تا زمانی که نتیجه به صورت win یا lose تعیین نشود حلقه ادامه پیدا می کند .
به این صورت که مجموع دو عدد رو شده در مرحله ی اول را در متغیر جدیدی ذخیره می کند و در هر مرحله که دوباره تاس ها ریخته می شوند (توسط تابع rollDice) چک می کند آیا خروجی rollDice (مجموع اعداد رو شده) برابر متغیری که تعریف شد خواهد بود یا نه اگر باشد نتیجه را به صورت win تعیین می کند و لذا کد درمرحله بعدی وارد حلقه while نمی شود.
همچنین چک می کند که کند آیا خروجی rollDice (مجموع اعداد رو شده) برابر 7 است یا نه اگر مجموع 7 شود نتیجه را به صورت loseتعیین می کند و لذا کد درمرحله بعدی وارد حلقه while نمی شود. در نهایت نتیجه را خروجی می دهد.

-()simulate : برای شبیه سازی نیاز داریم که بازی تعداد مشخصی بار تکرار شود بنابراین این تابع به کمک تابع قبلی بازی را به تعداد دفعات ورودی خود تکرار می کند و تعداد برد و باخت ها را شمارش می کند و در نهایت احتمال برد را به صورت حاصل تقسیم تعداد برد ها بر ورودی تابع (تعداد دفعات شبیه سازی) خروجی می دهد.
برای شمارش تعداد برد ها و باخت ها دو متغیر جدید برای تعداد برد و باخت تعریف می کنیم و تحت یک حلقه for که به تعداد ورودی تابع تکرار می شود هر بار نتیجه تابع game را چک می کنیم و اگر نتیجه win بود یکی به تعداد متغیر برد ها و اگر نتیجه lose بود یکی به تعداد متغیر باخت ها اضافه می کنیم.

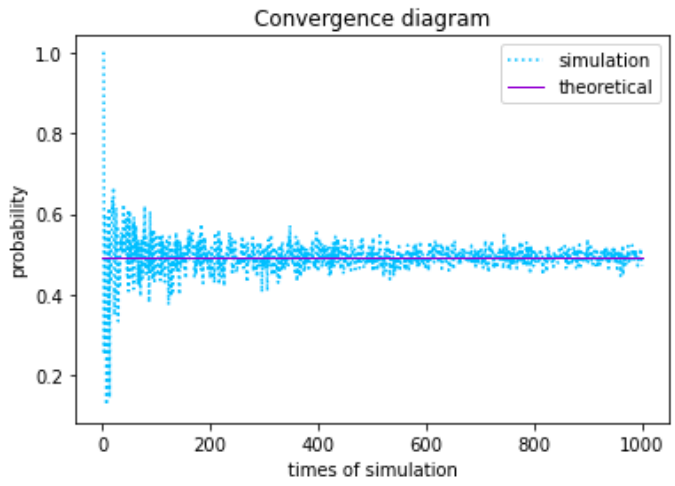
-()relativeerror: در قسمت ب و ج از ما خواسته شده که مقدار خطای نسبی در هر بار شبیه سازی بدست آورده شود لذا تابعی تعریف می کنیم مقدار بدست آمده در هر شبیه سازی را به عنوان ورودی به آن می دهیم و تابع طبق فرمول خطای نسبی با در اختیار داشتن مقدار تئوری که در بخش ا-الف بدست آمد آن را محاسبه می کند و خطای نسبی را خروجی می دهد.

محاسبه ی احتمال برد و خطای نسبی:

با استفاده از توابعی که معرفی کردیم مقدار احتمال برد با 1000 ، 10000 ، 100000 بار شبیه سازی را بدست می آوریم و خطای نسبی را در هر یک از حالات محاسبه می کنیم.
برای این کار طبق تعریف توابع 1000 ، 10000 ، 100000 را به عنوان ورودی به تابع simulate می دهیم و این تابع احتمال برد را خروجی می دهد سپس این سه خروجی را به تابع relativeerror می دهیم تا خطای نسبی هر یک را محاسبه کند و همگی را روی صفحه چاپ می کنیم.

- **قسمت د:**

برای رسم نمودار همگرایی دو لیست تعریف می کنیم و نمودار آن دو را بر حسب هم به وسیله plot از کتابخانه matplotlib رسم می کنیم



لیست اولیه شامل اعداد از 1 تا 1000 است .
لیست دوم شامل خروجی تابع simulate معرفی شده به ازای اعداد درون لیست اول است.
نام نمودار و تیتر کنار محور افق و عمود هم مطابق خواسته سوال تعیین می کنیم

برای مقایسه بهتر مقدار تئوری و نمودار همگرایی که به ازای افزایش تعداد بار های شبیه سازی به مقدار تئوری نزدیک تر می شود ، لیست ثابت دیگری نیز تعریف شده که به ازای همه اعداد درون لیست اول مقدار ثابت تئوری که قبلا بدست آمده را اختیار می کند.
رنگ ، استایل خطوط ، ضخامت خطوط و... را هم با استفاده از ویژگی های تابع plot مشخص می کنیم.

سوال دوم – احتمال پیدا کردن همزاد :

توضیح روند کلی حل مسئله:

از آن جایی که روند کلی کد برای قسمت الف و ب یکسان است ابتدا روند کلی که طبق آن کد نوشت شده را توضیح می دهیم سپس هر بخش را به طور جداگانه توصیف می کنیم.

در این سوال نیست از شبیه سازی استفاده شده است ، برای بدست آمدن احتمال مطلوب در هر سه قسمت الف ، ب ، ج از 10000 شبیه سازی استفاده شده است.

تعداد دفعات شبیه سازی در متغیری به نام timesOfSIM ذخیره شده لذا می توانید آن را تغییر دهید.

بایستی مسئله 10000 بار طبق فرض تکرار شود و تعداد حالات مطلوب شمرده شود بنابراین از یک حلقه for کمک می گیریم که 10000 بار تکرار می شود.
در دنیای واقعی روز تولد افراد عدد رندمی بین 1 تا 365 است لذا با استفاده از radient در کتابخانه random به تعداد افراد عدد تصادفی تولید می کنیم و آن ها را در یک لیست دخیره می کنیم

- برای قسمت دوم سوال که تولد ها بین 1تا n است اعداد تصافی بین 1 تا n تولید می شوند.

اکنون نیاز داریم تا تعداد تکرار های روز تولد افراد را محاسبه کنیم و ببینیم آیا مطلوب (دقیقا 2 نفر یکسان) هست یا خیر ، برای این کار با استفاده از counter در کتابخانه collection تعداد تکرار های هر یک از اعداد داخل لیست را بدست می آوریم .
خروجی لیستی خواهد بود که جلوی هر عدد لیست اولیه تعداد تکرار های آن نوشته شده است ، اما ما تنها تعداد تکرار ها را نیاز داریم پس با استفاده از value اعداد درون لیست خروجی counter را در لیست جدیدی به نام bdayRep_sequence ذخیره می کنیم .
مطلوب سوال زمانی است که همه اعداد داخل لیست اولیه دقیقا یک بار تکرار داشته باشند به جز یک عدد ، برای چک کردن این حالت نیاز داریم که اعداد درون لیست bdayRep_sequence را مرتب کنیم ، لیست مرتب شده را در لیست جدیدی به نام sortedBdaysRep ذخیره می کنیم .
گفته شد باید همه اعداد دقیقا 1 بار تکرار شوند به جز یکی پس در لیست مرتب شده باید اعداد همگی 1 باشند به جز آخرین عدد که باید 2 باشد ، پس به وسیله if چک می کنیم آیا دو عدد آخر لیست مرتب شده (1,2) هستند یا نه
در هر بار طی کردن حلقه for در صورتی که نتیجه مطلوب باشد (if آخر که توضیح داده شد درست باشد) یکی به متغیری که از پیش تعریف کردیم و حالات مطلوب را شمارش می کند به نام only2sameBday اضافه می شود
در آخر برای بدست آوردن احتمال مقدار only2sameBday را تقسیم بر تعداد دفعات شبیه سازی می کنیم .

- **الف:**

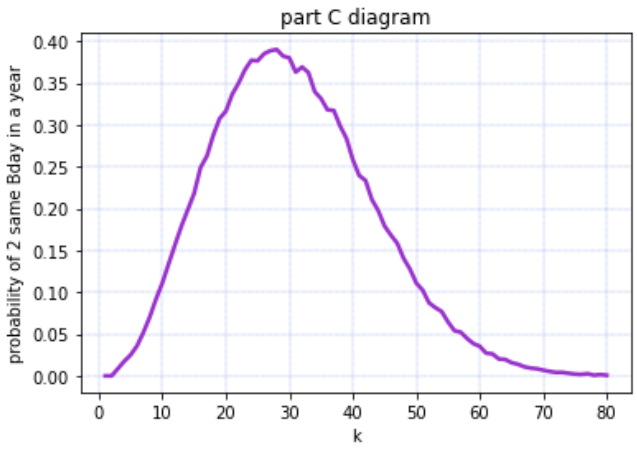
برای قسمت الف سوال مطابق روند توضیحی تعداد افراد را 23 نفر در نظر گرفته لذا 23 عدد رندوم بین 1 تا 365 تولید می کنیم و دقیقا مثل توضیحات قبل در آخر احتمالی که به ازای 10000 بار شبیه یازی بدست آمده را چاپ می کنیم

- **ب:**

برای قسمت ب که خواسته شده تابعی بنویسیم ، تابعی تعریف می کنیم که تعداد افراد و تعداد روز ها را به عنوان ورودی بگیرد ، به تعداد افراد عدد تصادفی از 1 تا تعداد روز ها تولید می کنیم و در آخر تابع پس از طی روند توضیح داده شده احتمال بدست آمده در 10000 بار شبیه سازی را خروجی می دهد.

- **ج:**

برای رسم نمودار دو لیست تعریف می کنیم و نمودار آن دو را بر حسب هم به وسیله plot از کتابخانه matplotlib رسم می کنیم



لیست اول شامل اعداد 1 تا 80 است

لیست دوم شامل خروجی تابع معرفی شده در قسمت ب سوال به ازای ورودی اول 365 و ورودی دوم اعداد درون لیست اول است

رنگ ، استایل خطوط ، ضخامت خطوط ، و grid و... را هم با استفاده از ویژگی های تابع plot مشخص می کنیم.

- به دلیل 80 بار شبیه سازی با تکرار 10000 بار کمی زمان می برد تا نمودار رسم شود (حدود 6 دقیقه)