ponteRevisado

January 25, 2025

1 Xogo do Río

Ás dúas beiras dun río temos 12 fichas que debemos distribuír en 12 casas cada unha numerada do 1 ao 12. Cada xogador tira alternativamente dous dados e suma as puntuacións. A ficha situada na casa con esa puntuación cruzará o río. Gaña o primeiro que quede sen fichas.

[1]:
```
import random

def todas_descomposicions(numero, n):
    if numero <= 0 or n <= 0:
        raise ValueError("O número e n deben ser maiores que 0.")

    def obter_sumandos(restante, k):
        if k == 1:
            return [[restante]] if restante >= 0 else []
        resultado = []
        for i in range(restante + 1):
            for resto in obter_sumandos(restante - i, k - 1):
                resultado.append([i] + resto)
        return resultado

    todas = obter_sumandos(numero, n - 1)

    return todas

# Exemplo de uso
numero = 12
n = 12
resultado = todas_descomposicions(numero, n)
print(f"Todas as descomposicións de {numero} en {n} sumandos con primeira␣
 ↪posición 0 e sumandos en calquera lugar: {resultado}")
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

```
Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

[2]:
```
descomposicions=todas_descomposicions(12,12)
len(descomposicions)
```

[2]:
```
646646
```

[1]:
```
def partida(xog1,xog2):
    ceros=[0]*11

    while xog1!=ceros and xog2!=ceros:
        #tira dous dados xog1
        x=random.randint(1, 6)
        y=random.randint(1, 6)
        if xog1[x+y-2]>0:
            xog1[x+y-2]-=1
        #tira dous dados xog2
        x=random.randint(1, 6)
        y=random.randint(1, 6)
        if xog2[x+y-2]>0:
            xog2[x+y-2]-=1

    if xog1==ceros:
        return 1
    else:
        return 2
```

[2]:
```
numPartidas=100 #100 partidas cada un en 3 horas (~7000 por segundo), 1000␣
↪partidas 30 horas
```

[43]:
```
from tqdm import tqdm
from copy import deepcopy #para pasar as listas por valor
victorias=[]
for i in tqdm(range(len(descomposicions))):
#for i in tqdm(range(100)):
    victorias.append(0)
    for j in range(numPartidas):
        xog1=deepcopy(descomposicions[i])
        xog2=deepcopy(random.choice(descomposicions))
        winner=partida(xog1,xog2)
        if winner==1:
            victorias[i]+=1
```

```
  0%|          | 191/646646 [00:03<3:16:26, 54.85it/s]
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[43], line 10
      8 xog1=deepcopy(descomposicions[i])
      9 xog2=deepcopy(random.choice(descomposicions))
---> 10 winner=partida(xog1,xog2)
     11 if winner==1:
     12     victorias[i]+=1

Cell In[37], line 7, in partida(xog1, xog2)
      4 while xog1!=ceros and xog2!=ceros:
      5     #tira dous dados xog1
      6     x=random.randint(1, 6)
----> 7     y=random.randint(1, 6)
      8     if xog1[x+y-2]>0:
      9         xog1[x+y-2]-=1

File ~/anaconda3/lib/python3.9/random.py:334, in Random.randint(self, a, b)
    330         raise ValueError("empty range for randrange()")
    332     return istart + istep * self._randbelow(n)
--> 334 def randint(self, a, b):
    335     """Return random integer in range [a, b], including both end points
    336     """
    338     return self.randrange(a, b+1)

KeyboardInterrupt:
```

[33]:
```
import pandas as pd
#Clasificación
d = {'indice': list(range(0,len(victorias))), 'xogadas':descomposicions[0:
 ↪len(victorias)],'victorias': victorias}
df = pd.DataFrame(data=d)
df['ratio'] = df['victorias'].apply(lambda x: str(int(x*100/numPartidas))+"%")
df.sort_values(by=['victorias'],ascending=False)
```

[30]:
```
df10000=df.sort_values(by=['victorias'],ascending=False).head(10000)
df10000
```

[32]:
```
#Gardo os resultados en ficheiros para non ter que volver a pasar o proceso de␣
 ↪3 horas
df.sort_values(by=['victorias'],ascending=False).to_csv('/home/manuel/
 ↪Escritorio/Programacion/python/xogoPonte/100partidas.csv',index=False)
df10000.sort_values(by=['victorias'],ascending=False).to_csv('/home/manuel/
 ↪Escritorio/Programacion/python/xogoPonte/10000mellores.csv',index=False)
```

[3]:
```
#Recupero os resultados
import pandas as pd
df10000=pd.read_csv('/home/manuel/Escritorio/Programacion/python/xogoPonte/
 ↪10000mellores.csv')
df3000=df10000.head(3000)
```

[4]:
```
df3000
```

[4]:

	indice	xogadas	victorias	ratio
0	59698	[0, 0, 1, 0, 2, 5, 2, 1, 1, 0, 0]	99	99%
1	173325	[0, 1, 1, 2, 2, 3, 2, 1, 0, 0, 0]	99	99%
2	43001	[0, 0, 0, 3, 2, 4, 1, 1, 1, 0, 0]	97	97%
3	16705	[0, 0, 0, 0, 4, 3, 2, 2, 1, 0, 0]	97	97%
4	145287	[0, 1, 0, 1, 3, 2, 2, 2, 1, 0, 0]	97	97%
…	…	…	…	…
2995	45995	[0, 0, 0, 4, 1, 4, 0, 3, 0, 0, 0]	89	89%
2996	184498	[0, 1, 2, 1, 2, 0, 2, 3, 0, 1, 0]	89	89%
2997	70147	[0, 0, 1, 1, 4, 1, 1, 3, 1, 0, 0]	89	89%
2998	186684	[0, 1, 2, 2, 2, 1, 1, 3, 0, 0, 0]	89	89%
2999	98082	[0, 0, 2, 2, 4, 0, 2, 2, 0, 0, 0]	89	89%

[3000 rows x 4 columns]

[5]:
```
#Todos contra todos os 3000 mellores (collo 3000 porque así o torneo dura 30
min, con 100000 dura 11 horas)
from copy import deepcopy #para pasar as listas por valor
from tqdm import tqdm
import random

victorias=[0]*3000
for i in tqdm(range(3000)):
    for j in range(3000):
        if i!=j:
            xog1=deepcopy(df3000['xogadas'][i]) # teño que crealo como copia␣
 ↪cada vez porque se modifica na partida
            xog1=xog1[1:]
            xog1=xog1[:-1]
            xog1=list(map(int, xog1.split(','))) #Fago isto porque non se␣
 ↪recoñece o tipo lista no df
            xog2=deepcopy(list(df3000['xogadas'])[j])
            xog2=xog2[1:]
            xog2=xog2[:-1]
            xog2=list(map(int, xog2.split(',')))
            winner=partida(xog1,xog2)
            if winner==1:
                victorias[i]+=1
            else:
```

```
                victorias[j]+=1
```

```
100%|      | 3000/3000 [29:14<00:00,  1.71it/s]
```

[6]:
```
#clasificación
numPartidas=2999*2
d = {'indice': list(range(0,len(victorias))), 'xogadas':
 ↪list(df3000['xogadas']),'victorias': victorias}
dfFinal = pd.DataFrame(data=d)
dfFinal['ratio'] = dfFinal['victorias'].apply(lambda x: str(int(x*100/
 ↪numPartidas))+"%")
dfFinal=dfFinal.sort_values(by=['victorias'],ascending=False)
dfFinal
```

[6]:

	indice	xogadas	victorias	ratio
157	157	[0, 0, 1, 1, 2, 3, 2, 2, 1, 0, 0]	3979	66%
162	162	[0, 0, 1, 1, 3, 3, 2, 1, 1, 0, 0]	3973	66%
2180	2180	[0, 0, 1, 2, 2, 3, 2, 1, 1, 0, 0]	3959	66%
381	381	[0, 0, 1, 1, 2, 3, 3, 1, 1, 0, 0]	3947	65%
110	110	[0, 0, 1, 2, 2, 3, 2, 2, 0, 0, 0]	3932	65%
…	…	…	…	…
1961	1961	[0, 0, 2, 5, 2, 1, 0, 1, 0, 1, 0]	1830	30%
1149	1149	[0, 2, 0, 2, 0, 2, 0, 3, 3, 0, 0]	1735	28%
1772	1772	[0, 1, 3, 0, 1, 1, 1, 1, 2, 2, 0]	1671	27%
2812	2812	[0, 0, 2, 0, 1, 8, 0, 1, 0, 0, 0]	1630	27%
2672	2672	[1, 0, 3, 1, 0, 0, 1, 3, 2, 1, 0]	1447	24%

[3000 rows x 4 columns]

[7]:
```
dfFinal.head(40)
```

[7]:

	indice	xogadas	victorias	ratio
157	157	[0, 0, 1, 1, 2, 3, 2, 2, 1, 0, 0]	3979	66%
162	162	[0, 0, 1, 1, 3, 3, 2, 1, 1, 0, 0]	3973	66%
2180	2180	[0, 0, 1, 2, 2, 3, 2, 1, 1, 0, 0]	3959	66%
381	381	[0, 0, 1, 1, 2, 3, 3, 1, 1, 0, 0]	3947	65%
110	110	[0, 0, 1, 2, 2, 3, 2, 2, 0, 0, 0]	3932	65%
1071	1071	[0, 0, 1, 1, 2, 4, 2, 1, 1, 0, 0]	3915	65%
266	266	[0, 0, 0, 2, 2, 4, 2, 1, 1, 0, 0]	3887	64%
948	948	[0, 0, 0, 2, 2, 3, 2, 2, 1, 0, 0]	3873	64%
625	625	[0, 0, 1, 2, 3, 2, 2, 1, 1, 0, 0]	3863	64%
2178	2178	[0, 0, 1, 2, 2, 3, 1, 2, 1, 0, 0]	3858	64%
1481	1481	[0, 0, 0, 1, 2, 3, 3, 2, 1, 0, 0]	3858	64%
2170	2170	[0, 0, 1, 2, 2, 2, 3, 1, 1, 0, 0]	3854	64%
63	63	[0, 0, 0, 1, 3, 3, 2, 2, 1, 0, 0]	3836	63%
57	57	[0, 0, 1, 2, 1, 3, 2, 2, 1, 0, 0]	3832	63%
283	283	[0, 0, 1, 1, 2, 2, 3, 2, 1, 0, 0]	3831	63%

52	52	[0, 0, 0, 2, 2, 3, 3, 1, 1, 0, 0]	3817	63%
1945	1945	[0, 0, 1, 1, 3, 3, 2, 2, 0, 0, 0]	3816	63%
11	11	[0, 0, 1, 2, 3, 3, 2, 1, 0, 0, 0]	3814	63%
780	780	[0, 0, 1, 2, 2, 2, 2, 2, 1, 0, 0]	3807	63%
265	265	[0, 0, 1, 2, 3, 3, 1, 1, 1, 0, 0]	3804	63%
1032	1032	[0, 0, 0, 2, 2, 4, 2, 2, 0, 0, 0]	3801	63%
646	646	[0, 0, 0, 1, 3, 3, 3, 1, 1, 0, 0]	3799	63%
395	395	[0, 0, 0, 1, 2, 4, 2, 2, 1, 0, 0]	3796	63%
8	8	[0, 0, 1, 2, 2, 4, 2, 1, 0, 0, 0]	3795	63%
649	649	[0, 0, 1, 2, 3, 2, 2, 2, 0, 0, 0]	3794	63%
160	160	[0, 0, 0, 2, 3, 3, 2, 1, 1, 0, 0]	3793	63%
471	471	[0, 0, 1, 0, 2, 3, 3, 2, 1, 0, 0]	3792	63%
1771	1771	[0, 0, 1, 1, 3, 3, 1, 2, 1, 0, 0]	3792	63%
21	21	[0, 0, 1, 1, 2, 3, 3, 2, 0, 0, 0]	3786	63%
527	527	[0, 0, 1, 1, 1, 3, 3, 2, 1, 0, 0]	3780	63%
53	53	[0, 0, 0, 2, 2, 3, 3, 2, 0, 0, 0]	3773	62%
802	802	[0, 0, 0, 2, 3, 3, 2, 2, 0, 0, 0]	3760	62%
642	642	[0, 0, 1, 2, 3, 3, 1, 2, 0, 0, 0]	3756	62%
670	670	[0, 0, 1, 2, 2, 4, 2, 0, 1, 0, 0]	3742	62%
291	291	[0, 0, 0, 1, 3, 3, 3, 2, 0, 0, 0]	3740	62%
80	80	[0, 0, 1, 1, 2, 4, 2, 2, 0, 0, 0]	3739	62%
2330	2330	[0, 0, 0, 1, 3, 4, 2, 1, 1, 0, 0]	3739	62%
1039	1039	[0, 0, 1, 2, 3, 2, 1, 2, 1, 0, 0]	3738	62%
2012	2012	[0, 0, 1, 2, 3, 3, 2, 0, 1, 0, 0]	3734	62%
16	16	[0, 0, 0, 2, 2, 2, 3, 2, 1, 0, 0]	3730	62%

[32]:
```
dfFinal.to_csv('/home/manuel/Escritorio/Programacion/python/xogoPonte/final.
↪csv',index=False)
```

[ ]:
```

```