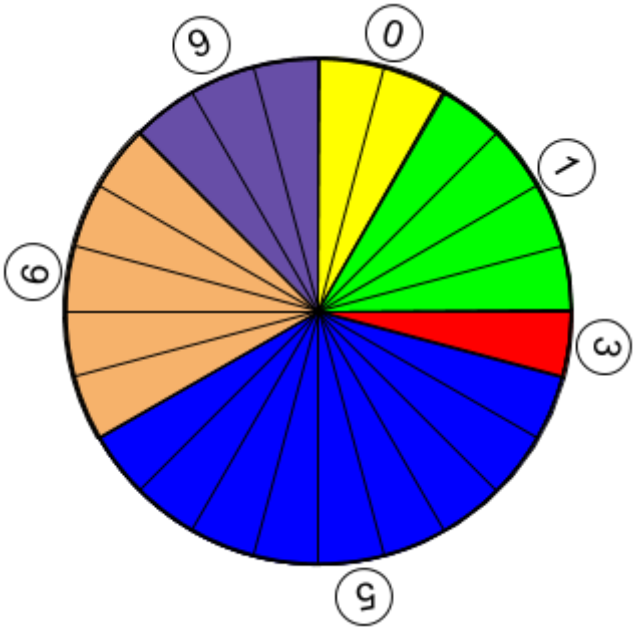
Primeiro Exercício-Programa: Roleta Maluca

BCC 2018 - MAC0110 - Entrega: até 15/04/2018 23:55 pelo PACA

Introdução

Nesse EP nós vamos implementar em Python3 um jogo de roleta com um *twist*: nessa roleta a quantidade de números disponível é configurável e os números sorteados possuem probabilidades diferentes de aparecer; alguns podem até nem ser sorteáveis!



As próximas seções explicam como implementar esse tipo de sorteio enviesado e como deve funcionar o jogo. O enunciado traz também algumas funções de implementação obrigatória, que lhe servirão de metodologia para estruturar e testar seu código. Leia atentamente o enunciado, poste dúvidas no PACA, lembrando sempre de que trechos de código não devem ser compartilhados, e que apenas os recursos da linguagem vistos em aula podem ser empregados na solução (e mesmo alguns desses estão explicitamente interditados no enunciado).

Distribuições de probabilidade enviesadas

A roleta maluca é definida por dois parâmetros principais: a quantidade *N* de valores que podem aparecer (por convenção os valores sorteados estão sempre entre *0* e *N-1*), e a distribuição de probabilidade desses valores. Observe a estrutura radial da roleta na figura acima: nossa roleta maluca terá uma estrutura exatamente igual! Para cada valor *n* passível de ser sorteado será associado um peso aleatório p_n (um inteiro entre 0 e 9), indicando a quantidade de raios alocado para esse valor. No exemplo da figura, *N=10* e os pesos correspondentes são $p_0=2$, $p_1=4$, $p_2=0$, $p_3=1$, $p_4=0$, $p_5=9$, $p_6=5$, $p_7=0$, $p_8=0$ e $p_9=3$. Todos esses pesos devem ser armazenados em uma única variável inteira *dist*, que conterá o resultado da expressão

$$dist = p_0+p_1*10+p_2*10^2+...+p_{N-1}*10^{N-1}$$

Na seção seguinte alguns detalhes da implementação dessa expressão serão discutidos; por enquanto, basta notar que deverão ser gerados N valores aleatórios $p_0, p_1,..., p_{N-1}$, cada um deles entre 0 e 9, que formam a representação decimal do número inteiro *dist*.

Uma vez estabelecidos os valores de *N* e *dist*, resta saber como gerar valores aleatórios entre *0* e *N-1* com a mesma distribuição de probabilidade daquela roleta em estrutura radial com os pesos fixados. Note que se

$$S = \sum_{n=0}^{N-1} p_n$$

então as probabilidades associadas a cada valor n devem ser iguais a p_n/S, e a probabilidade de sorteio de um valor qualquer menor ou igual a *n* deve ser $\frac{\sum_{j=0}^{n} p_n}{S}$. Isso sugere o seguinte algoritmo para gerar um valor aleatório conforme essa distribuição:

```
sorteia(N,dist):
    calcule S de acordo com a fórmula acima
    sorteie um valor aleatório s inteiro entre 0 e S-1
    devolva o menor inteiro n (entre 0 e N-1) tal que s < p0+p1+...+pn
```

Cada um desses passos envolve alguma combinação de mecanismos vistos em aula (contadores, aritmética com dígitos, somas parciais, etc). A próxima seção traz uma visão geral do jogo e discussões específicas sobre as implementações necessárias.

Especificação

Seu programa em Python deve se comportar exatamente como o executável disponível no PACA. O jogo começa com a configuração da roleta maluca: o usuário informa o valor de *N* (entre 2 e 100) e o computador gera a distribuição aleatória *dist*. Usaremos a função `randint` da biblioteca `random` para gerar valores inteiros aleatórios; para isso acrescente ao início do seu código a linha

```
from random import randint
```

A partir desse ponto, cada chamada `randint(a,b)` devolverá um valor inteiro uniformemente sorteado no intervalo [a,b] (incluindo os dois extremos). Você deverá então usar essa função para construir a distribuição enviesada através de uma função com o seguinte esqueleto:

```
# Função obrigatória
def distribuicao(N):
    """Devolve um inteiro dist onde cada dígito n=0,...N-1 representa
       um peso relativo em uma distribuição de probabilidade,
       verificando que dist esteja entre 0 e 10**N-1.
    """
    # calcula, calcula, calcula...
    assert dist in range(10**N)
    return dist
```

A linha `assert dist in range(10**N)` faz com que a execução seja interrompida se o valor calculado para *dist* não estiver entre *0* e 10^N-1 (ou seja, se algo saiu errado no meio do cálculo). A fim de exercitarmos nosso aprendizado, o uso do operador ** (exponenciação) está **interditado** no EP1, ou seja, você terá que construir a distribuição sem usar explicitamente qualquer expressão da forma `x**y`. As únicas duas ocorrências de ** no seu código serão essas aí de cima (uma dentro do comentário, outra no `assert`). Dica: compute as potências $10^0, 10^1, \ldots, 10^{N-1}$ iterativamente, a fim de construir cada um dos termos da soma que define *dist*.

Em seguida o jogo começa! Cada rodada terá sempre a mesma estrutura:

- computador sorteia quem joga primeiro (50% | 50% para humano | computador)
- jogador A escolhe valor inteiro p entre 0 e N-1
- jogador B escolhe valor inteiro q entre 0 e N-1, com q ≠ p
- computador sorteia, anuncia ganhador e atualiza pontuações
- computador pergunta se o jogo deve continuar

Para realizar a jogada, escreveremos uma função com o seguinte esqueleto:

```
# Função obrigatória
def jogada(N,jogador,lanceanterior):
    """Processa e devolve uma jogada do jogador ("humano" ou "computador"),
       que deve estar entre 0 e N-1 e não pode ser igual a lanceanterior.
    """
    # processa, processa, processa…
    assert lance in range(N) and lance != lanceanterior
    return lance
```

A mesma função será usada para os dois jogadores, conforme a ordem estabelecida pelo sorteio, sendo que se `jogador=="humano"` o valor deve ser solicitado do usuário, mas se `jogador=="computador"` deve ser gerado aleatoriamente (sorteio uniforme entre 0 e N-1). Nos dois casos as condições *lance ∈ {0,...,N-1}* e *lance ≠ lanceanterior* devem ser verificadas, mesmo que para isso o programa tenha que repetir a pergunta (ou o sorteio). Para o primeiro jogador de cada jogada a função pode ser chamada com lanceanterior=-1 (para não restringir as escolhas desse jogador).

O sorteio deve ser realizado obrigatoriamente por uma função

```
# Função obrigatória
def sorteia(N,dist):
    """Devolve o resultado de um sorteio enviesado dentre os
       inteiros de 0 a N-1 de acordo com a distribuição dist.
    """
    # computa, computa, computa...
    assert sorteio in range(N)
    return sorteio
```

que realiza a sequência de cálculos indicada na seção anterior. Para somar os dígitos decimais de *dist*, lembre-se que as expressões `dist % 10` (resto da divisão) e `dist // 10` (quociente inteiro da divisão) produzem respectivamente o último dígito e o inteiro formado pelos dígitos que sobram, o que permite ao computador "descascar os dígitos" de um inteiro qualquer (por exemplo, se n=987 então n%10==7 e n//10==98, e fazendo n=n//10 teríamos, no próximo passo, n%10 == 8 e n//10 == 9).

Ganhará a rodada o jogador que mais se aproximar do valor sorteado, considerando a distância "circular" dos inteiros modulo N

```
dist(a,b) = min ( (a-b)%N , (b-a)%N )
```

e desempatando em favor do jogador humano. A pontuação dos dois jogadores começa em 0 e a soma dos pontos será sempre 0. Um acerto do valor exato sorteado resulta numa transferência de crédito de 100 pontos do jogador perdedor para o ganhador da rodada; um acerto aproximado resulta numa transferência de 10

pontos. A continuidade do jogo depende do jogador humano, que responderá ao final de cada rodada à pergunta "Deseja continuar jogando (S/N): "

Está disponível um executável no PACA para complementar o enunciado e tirar dúvidas sobre o funcionamento do jogo em situações específicas. Baixe e teste, e poste quaisquer dúvidas que surgirem no Fórum.

Função Bônus (+1.0 na nota do EP)

Se você quiser enriquecer a interface do jogo, poderá implementar um *easter egg* na forma de uma função

```
# função opcional, com bônus de +1.0 na nota
def desmontaroleta(N,dist):
    """Explicita a estrutura da roleta maluca, enviesada pela
       distribuição dist: para cada inteiro entre 0 e N-1 mostra
       o peso correspondente (entre 0 e 9), a probabilidade
       teórica associada, e a frequência relativa deste dígito
       em uma repetição de 1000 sorteios com essa distribuição.
    """
```

que será acionada no caso do usuário responder à frase "Deseja continuar jogando (S/N): " com a expressão mágica “Abra o jogo!”. Nesse caso seu programa deve produzir um diagnóstico do enviesamento da roleta conforme a descrição da função (experimente fazer o mesmo no executável para observar o formato da saída).

Last but not least…

Lembre-se sempre de:

- Ler as Instruções para Entrega de EPs no PACA
- Não postar códigos no Fórum Geral, nem os compartilhar com colegas
- Não deixar o EP para a última hora: as probabilidades da Internet cair, do HD pifar e vários outros fenômenos extremamente comuns aumenta vertiginosamente quando estamos próximos de deadlines (corolário do Teorema de Murphy).
- Divertir-se programando! (Vocês ainda tem mais de 3,5 anos de BCC pela frente, né? ;-)