MAC0323 Algoritmos e Estruturas de Dados II

Eduardo Pinheiro(8936798)

22-03-2016

1.3.36 Random Iterator

Write a iterator for RandomQueue$\langle Item \rangle$ from the previous exercise that returns the items in random order.

Considerações:

Para compilar e executar o problema:

$ $javac - algs4\ RandomQueue.java$
$ $java - algs4\ RandomQueue$

Para complilar e rodar a TesteVisual:

$ $javac - algs4\ TesteVisual.java$
$ $java - algs4\ TesteVisual\ N\ T$
*Com N sendo os números inseridos na fila e T a quantidade de teste.

Implementação:

RandomQueue implementa a seguinte API:

1.3.35 *Random queue.* A *random queue* stores a collection of items and supports the following API:

public class RandomQueue<Item>	
RandomQueue()	*create an empty random queue*
boolean isEmpty()	*is the queue empty?*
void enqueue(Item item)	*add an item*
Item dequeue()	*remove and return a random item (sample without replacement)*
Item sample()	*return a random item, but do not remove (sample with replacement)*

API for a generic random queue

Write a class RandomQueue that implements this API. *Hint*: Use an array representation (with resizing). To remove an item, swap one at a random position (indexed 0 through N-1) with the one at the last position (index N-1). Then delete and return the last object, as in ResizingArrayStack. Write a client that deals bridge hands (13 cards each) using RandomQueue<Card>.

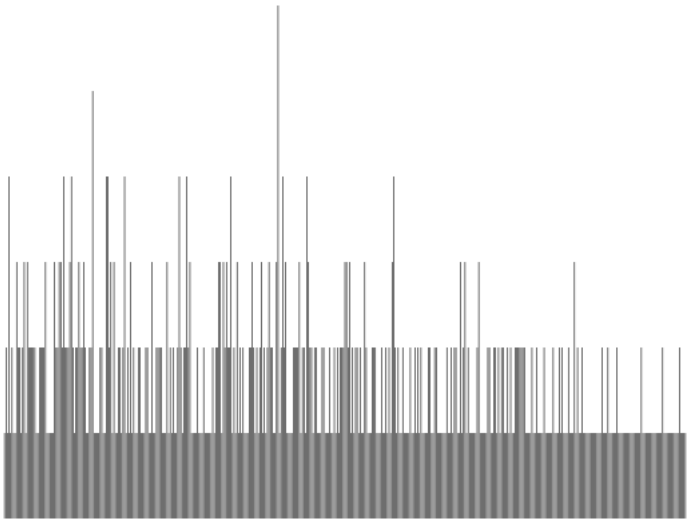
O programa possui também o tipo Permutation, que é uma lista-ligada que guarda as permutações únicas e suas frequências para construir o histograma.

Permutation tem a seguinte API:

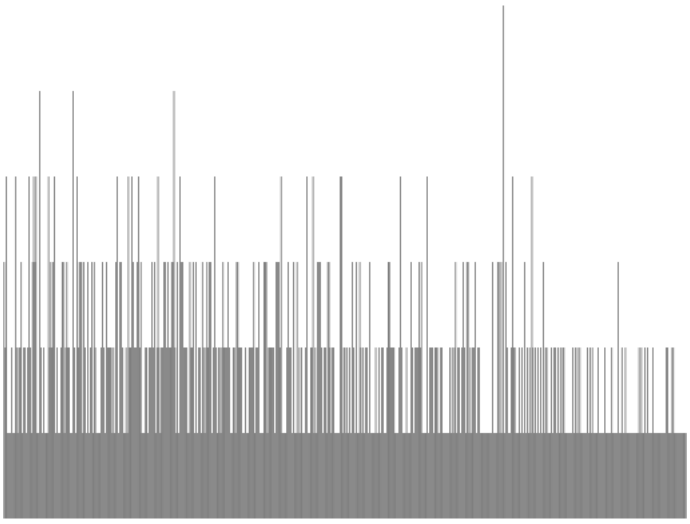
- $public\ Permutation()$
- $public\ void\ append(Integer[]\ item)$
- $public\ void\ print()$
- $public\ Integer[]\ frequencies()$

Testes:

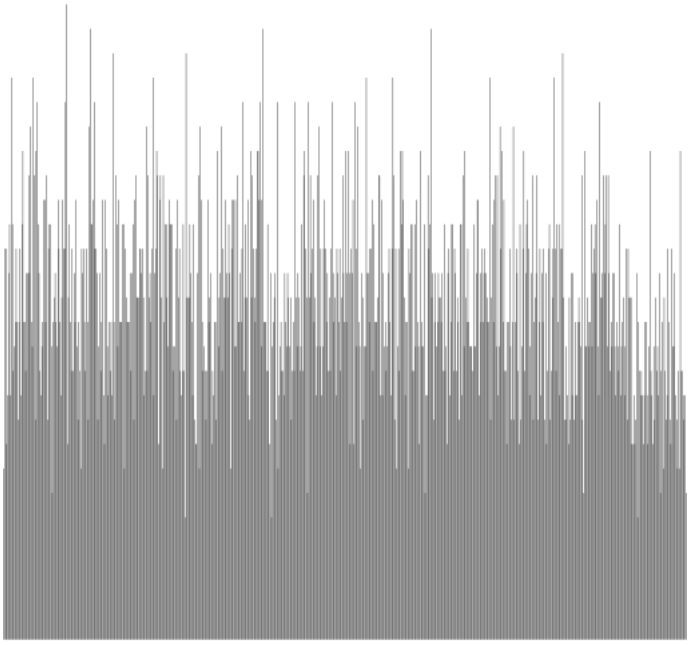
- $ $java - algs4\ TesteVisual\ 6\ 720$



- $ $java - algs4\ TesteVisual\ 6\ 1000$



- $ $java - algs4\ TesteVisual\ 6\ 10000$



- $ $java - algs4\ TesteVisual\ 6\ 100000$

- $ $java - algs4\ TesteVisual\ 6\ 1000000$

Temos que conforme o número T aumenta a quantidade de vezes que cada permutação ocorre vai convergindo para o mesmo número o que se espera pois cada permutação tem a mesma chance de ocorrer, podemos ver que a partir de 4000 testes, aparecem todas as permutações possíveis com N = 6.

Teste estatístico:

Para verificar a correção do iterator, podemos verificar de acordo com uma amostra se cada permutação tem probablidade $\frac{1}{K!}$, para isso podemos fazer um intervalo de confiança, neste estará contido o real valor de p.
Pelo Teorema do limite central, para um N grande na amostra, podemos usar a distribuição normal.

Para calcular o erro temos:

$$\varepsilon = z(\sqrt{\frac{\hat{p}(1-\hat{p})}{N}})$$

Com z sendo o valor da tabela normal para o nível de confiança desejado.
$\hat{p}$ a proporção encontrada na amostra.
N a amostra total.

Para $K = 4$, $N = 100000$, $\hat{p} = \frac{4139}{100000}$, encontrado na amostra gerada pelo teste do programa, e com nível de confiança de 95%($z = 1.96$).

Temos:

$$\varepsilon = 1,96(\sqrt{\frac{\frac{4139}{100000}(1-\frac{4139}{100000})}{100000}})$$

$\varepsilon = 0.0012345$

Com isso temos um intervalo de confiança(IC):

$IC = (\hat{p} - \varepsilon, \hat{p} + \varepsilon)$

$IC = (\frac{4139}{100000} - 0.0012345, \frac{4139}{100000} + 0.0012345)$

$IC = (0,04015, 0,04262)$

Temos que o valor esperado é $p = \frac{1}{24} = 0,04167$, portanto $p \in IC$, concluíndo que com nível de confiança de 95% a probabilidade de uma permutação ocorrer é a esperada de $\frac{1}{K!}$