Solução - Creative Problem 1.3.35 (Random queue; Algs4) e Creative Problem 1.3.36 (Random iterator; Algs4)

Victor Sena Molero (8941317)

April 12, 2016

Contents

1 Introdução **1**

2 Como implementar uma fila aleatória? **1**

3 Como gerar uma permutação aleatória? **2**

4 Como contar permutações? **3**

1 Introdução

Tinham 3 perguntas importantes nesse exercício. Eu vi algumas soluções diferentes e vou comentar as que mais gostei, basicamente a forma como eu faria isso. Nada disso é a única forma de fazer. Mas a ideia era que as soluções fossem equivalentemente boas (com gasto parecido de tempo e memória).
Além disso, se ficar faltando alguma coisa sobre algo que tenha interessado nesta explicação, pode me perguntar. Vou tentar ser breve nas explicações e mais deixar para vocês pistas sobre como encontrar as ferramentas necessárias para resolver isso de uma forma boa do que realmente ensinar tudo num pdf, o que seria impossível.
Eu escrevi isso bem rápido, pode ter erros.

2 Como implementar uma fila aleatória?

Primeiro é importante lembrar que o RandomQueue era uma implementação de fila aleatória. Independente do Bridge e do Histogram, essa classe deveria ser implementada da melhor maneira possível.
Para fazer isso de um jeito bom, é possível usar um vetor de alocação dinâmica, isto é, um vetor que, inicialmente, aloca um espaço fixo de memória e dobra de

tamanho se esse espaço acabar (quando for necessário mais espaço). Alémdisso, é interessante que quando o tamanho vetor alocado for muito grande, ele diminua de tamanho e economize espaço. Aqui, dobrar de tamanho é muito importante, se você aumentar de um por um vai demorar.
Agora, você poderia sempre inserir elementos ao fim do vetor. Quando for necessário remover um elemento aleatório do vetor, basta sortear um índice do vetor aleatóriamente e removê-lo. Perceba que, se fossemos remover, por exemplo, o primeiro elemento deste vetor, teríamos que mover todo elemento do resto do vetor uma posição para trás, porém, para remover o último, basta diminuir o inteiro que representa o tamanho do vetor, ou seja, custa constante. Assim, se sempre sortearmos um elemento qualquer do vetor, trocarmos ele com o ultimo e removermos o ultimo, faremos as remoções em tempo constante.
Me perguntaram se isso enviesa a fila, ou seja, se eu colocar n inteiros distintos numa fila $(a_1, \dots a_n)$ e removê-los um por um, a ordem de saída é uma permutação aleatória uniformemente distribuída destes inteiros? Para isso, devemos assumir que o gerador de números aleatórios do Java é uniformemente distribuído (isso é mentira, ele é próximo disso).
Vamos provar por indução em n.
Se $n = 1$, ao executarmos uma remoção, só existe uma saída possível e uma permutação possível. Todas as permutações têm a mesma probabilidade de serem geradas.
Se $n > 1$, assumindo que uma fila com $k < n$ elementos gera todas as permutações com igual probabilidade.
Agora, seja a_i um elemento qualquer da fila. A chance dele ser o primeiro a ser removido é $1/n$, pois o aleatório do Java gera todo inteiro de 1 a n com igual probabilidade (só que não). Agora, o que nos resta é gerar uma permutação aleatória com os $n - 1$ elementos restantes, pela hipótese de indução, todas as $1/(n-1)!$ permutações restantes com os elementos restantes são equiprováveis se geradas por uma fila aleatória. Note que não importa a ordem na qual os elementos estão dispostos na fila, ou seja, não importa se eu troquei o elemento removido com o ultimo e coloquei o ultimo numa posição qualquer, todas as permutações dos elementos restantes continuam equiprováveis. Então, toda permutação distinta que começa com a_i para todo a_i da tem chance $1/n * 1/(n-1)!$ de ocorrer, logo, já que toda permutação começa com algum a_i, toda permutação distinta tem chance $1/n!$ de acontecer.
Achou essa prova ruim? Tem uma melhor? Achou algo errado? Conteste! Hahaha. Espero ter ajudado.
Tudo certo aqui então? Em resumo, precisamos de um vetor de alocação dinâmica e podemos fazer adições e remoções em tempo constante amortizado.

3 Como gerar uma permutação aleatória?

Para implementar o iterator, você tinha que saber gerar uma permutação aleatória para definir a ordem na qual seu iterator ia percorrer os elementos. Um jeito muito simples de fazer isso era pegar uma RandomQueue e alimentar ela com

a permutação identidade (todos os inteiros de 1 a n). Depois disso, para cada next do vetor, bastava dar um dequeue na fila e retornar aquela posição do vetor. Isso gera todas as permutações de tamanho n com igual probabilidade, assumindo que sua RandomQueue esteja certa. Outra solução simples possível seria inicializar um vetor com a permutação identidade e aplicar um Knuth Shuffle[1] nele. Isso, na verdade, vai basicamente aplicar o mesmo processo que a fila aplica num conjunto de elementos, mas direto no vetor. Depois disso, basta sempre guardar o índice do próximo elemento a ser removido e tudo fica certo.

4 Como contar permutações?

Tem várias soluções pra isso. Você poderia usar uma Árvore de Busca Binária ou uma Hash Table (vocês vão aprender sobre os dois assuntos, se já não aprenderam). Além disso, você pode transformar uma permutação num inteiro entre 0 e um valor k conveniente e só usar um vetor de tamanho $k+1$ como vetor de frequências para contar, em constante, a frequência deles. Isso facilita também a montagem do histograma.
Para transformar permutações de tamanho n em inteiros de 0 até n^n (inclusive) basta olhar para elas como um vetor de inteiros de n posições onde cada elemento é estritamente menor que n (indexar a permutação em 0 é essencial) e calcular $s = \sum_{i=0}^{n-1} a_{i+1} * n^i$ onde a_i é o inteiro na i-ésima posição do vetor, note o quanto isso é parecido com escrever um número na base n. Além disso, é possível transformar uma permutação de tamanho n em um inteiro de 0 a $n! - 1$ olhando para ela como um número no Sistema Numeral Fatorial[2] (obrigado muito pelo aluno que usou esse nome no relatório, eu conhecia mas não sabia o nome, ia ter que explicar sem a ajuda da Wikipedia). A conversão de uma permutação para um inteiro no Sistema Numeral Fatorial é fácilmente feita em n^2 e um pouco mais difícil de fazer em $n \lg n$ (não acho que dê pra fazer melhor que isso) com uma ABB ou uma Fenwick Tree. Já que as permutações para o histograma iam até 6 e isso só era usado no histograma não tinha problema fazer em n^2. Essa ultima solução é bem elegante e é a minha preferida.

[1] https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle
[2] https://en.wikipedia.org/wiki/Factorial_number_system