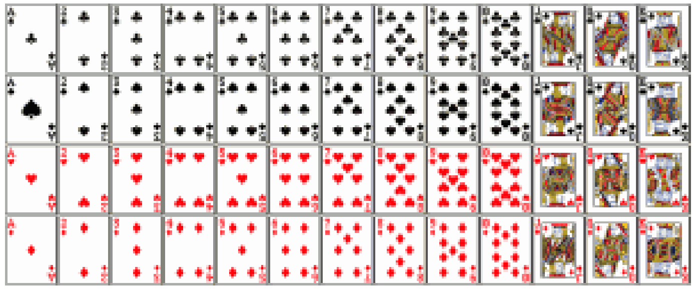
Shuffle a given array

Given an array, write a program to generate a random permutation of array elements. This question is also asked as “shuffle a deck of cards” or “randomize a given array”.



Let the given array be *arr[]*. A simple solution is to create an auxiliary array *temp[]* which is initially a copy of *arr[]*. Randomly select an element from *temp[]*, copy the randomly selected element to *arr[0]* and remove the selected element from *temp[]*. Repeat the same process n times and keep copying elements to *arr[1], arr[2], … .* The time complexity of this solution will be O(n^2).

Fisher–Yates shuffle Algorithm works in O(n) time complexity. The assumption here is, we are given a function rand() that generates random number in O(1) time.
The idea is to start from the last element, swap it with a randomly selected element from the whole array (including last). Now consider the array from 0 to n-2 (size reduced by 1), and repeat the process till we hit the first element.

Following is the detailed algorithm

```
To shuffle an array a of n elements (indices 0..n-1):
  for i from n - 1 downto 1 do
```

Google™ Custom Search

GeeksforGeeks

53,525 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion
Geometric Algorithms

```
        j = random integer with 0 <= j <= i
        exchange a[j] and a[i]
```

Following is C++ implementation of this algorithm.

```
// C Program to shuffle a given array

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

// A utility function to swap to integers
void swap (int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

// A utility function to print an array
void printArray (int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// A function to generate a random permutation of arr[]
void randomize ( int arr[], int n )
{
    // Use a different seed value so that we don't get same
    // result each time we run this program
    srand ( time(NULL) );

    // Start from the last element and swap one by one. We don't
    // need to run for the first element that's why i > 0
    for (int i = n-1; i > 0; i--)
    {
        // Pick a random index from 0 to i
        int j = rand() % (i+1);

        // Swap arr[i] with the element at random index
        swap(&arr[i], &arr[j]);
    }
}
```

Popular Posts

All permutations of a given string
Memory Layout of C Programs
Understanding “extern” keyword in C
Median of two sorted arrays
Tree traversal without recursion and without stack!
Structure Member Alignment, Padding and Data Packing
Intersection point of two Linked Lists
Lowest Common Ancestor in a BST.
Check if a binary tree is BST or not
Sorted Linked List to Balanced BST

```
// Driver program to test above function.
int main()
{
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = sizeof(arr)/ sizeof(arr[0]);
    randomize (arr, n);
    printArray(arr, n);

    return 0;
}
```

Output:

```
7 8 4 6 3 1 2 5
```

The above function assumes that rand() generates a random number.

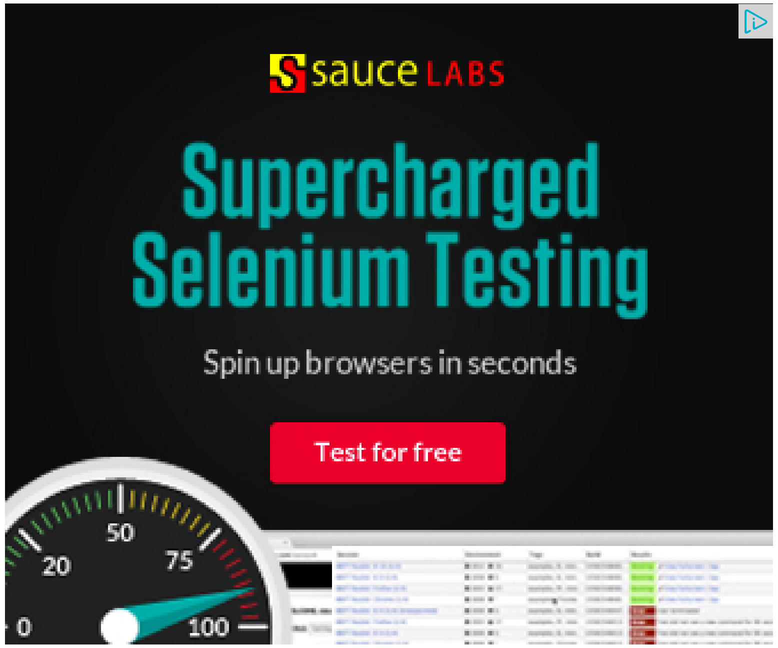
Time Complexity: O(n), assuming that the function rand() takes O(1) time.

How does this work?
The probability that ith element (including the last one) goes to last position is 1/n, because we randomly pick an element in first iteration.

The probability that ith element goes to second last position can be proved to be 1/n by dividing it in two cases.
Case 1: i = n-1 (index of last element):
The probability of last element going to second last position is = (probability that last element doesn't stay at its original position) x (probability that the index picked in previous step is picked again so that the last element is swapped)
So the probability = ((n-1)/n) x (1/(n-1)) = 1/n
Case 2: 0 < i < n-1 (index of non-last):
The probability of ith element going to second position = (probability that ith element is not picked in previous iteration) x (probability that ith element is picked in this iteration)
So the probability = ((n-1)/n) x (1/(n-1)) = 1/n

We can easily generalize above proof for any other position.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



705 Subscribe

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 15 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 55 minutes ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 58 minutes ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 1 hour ago

AdChoices

▶ C++ Code

▶ Java Array

▶ Java Source Code

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

9 Tweet 2 0

Writing code in comment? Please use **ideone.com** and share the link here.

20 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Manni • 2 months ago

Manni's probabilistic approach:

Select 2 random number between 0-n, swap the two elements at the the inde
Repeat the step n/2 times atleast. Contact Manpreet Singh, NIT Durgapur for r

https://www.facebook.com/manpr...

1

Ankur Jain • 3 months ago

Q1 why mod we take mod rand() % (i+1);and not like rand() % (n) ?
Q2 and why didn't take the element at zeroth index ?

alien ➦ Ankur Jain • a month ago

A1: because once you have shuffled nth element, dont replace it again
A2: with what will you replace 0th element with?

Ankur Jain ➦ alien • a month ago

Q1 comment : in the above solution you are swapping with its l
position is only replace with 0,1,2,3,4 and why cant 6 th ,7th , th

Q2 comment : as 0th element is only replace when rand fun gi
for i 0 to n
j=rand()%n;
shuffle the whole array in equal proportion

Sumit Gera • 11 months ago

How did we arrive at the probability for I = n - 1 case?

K.kaushik • 11 months ago

A simple java implementation of the above program with O(n) time complexity

```
public class Shuffle {

        public static int randomize(int min, int max)
        {
                        return (int) (Math.random() * (max-min+1) + m:
        }

        public static void shuffleCards(int cards[],  int n)
        {
                        int i;

                        for(i = 0; i < n; i++)
                        {
                                int j = randomize(i, n-1-i);
```

see more

Poorvank Bhatia • a year ago

why don&#039t we need to run it for the first element?

Poorvank Bhatia • a year ago

why don&#039t we need to run it for the first element?

Praveen · 2 years ago

Also a better explanation is here - http://bost.ocks.org/mike/shuf...

Mahesh · 2 years ago

Can you explain how the first case is order n^2 ?

Kartik ➦ Mahesh · 2 years ago

In first method, we need to remove the selected element from temp. R
c(m-1) time where m is the number of elements in temp[] and c is a cc
2 + ... 1)c = O(n^2)

Ishant Gaurav ➦ Kartik · a year ago

```
class Main
{
public static void main (String[] args) throws java.lang.Exceptio
{
int a[]={0,1,2,3,4,5,6,7,8,9};
int randarray[]=new int[10];
int flag[] = new int[10];
for(int i=0;i<10;i++)
{
randarray[i]=a[i];
}
SecureRandom rand = new SecureRandom();

int i=0;
while(i<=9)
```

{
int num = rand.nextInt(10);
if(i<=9 && flag[num]==0)

see more

Ishant Gaurav Ishant Gaurav • a year ago

I have wrritten acc to first method suggested by geeks f
complexity is o(n*2). Can u plz suggest if m wrong som

raj Ishant Gaurav • a year ago

In First method..
instead of removing selected element and then moving
to left by 1...we can just simply swap it with the last eler
As a result first algorithm will also be 0(n) time..

GeeksforGeeks • 2 years ago

@V: Thanks for the suggestion. We have added srand(time(NULL)) to the orig

@Apeirogon: Thanks for the inputs. We have added a comment before 'for' lo
last only as it makes the program more readable and matches with the standa

V • 2 years ago

You need to call srand(time(NULL)) before your call to randomize() otherwise
on each program execution.

```
/* Paste your code here (You may delete these lines if not writing co
```

Aish • 2 years ago

Hi,

It appears like the same output is generated each time when I try executing the

Output:
3 6 4 7 1 5 8 2

Is there any way by which we can get differnet set of array output each time?

```
/* Paste your code here (You may delete these lines if not writing c
```

adarsh Aish • 2 years ago

```
you must have forgot to write srand() function
```

Apeirogon • 2 years ago

It is a C implementation and not C++ implementation as stated in the post.

The loop iteration in randomize function need not be in reverse order, it is bett

When the code leaves out one iteration, write a comment about the same.

```
void randomize (int *a, int n) {
  for (int i = 0; i < n - 1; i++) { // off-by-one intentional, we do
  swap  (&a[i], &a[i + rand()%(n - i)];
}
```

Rajeev · 2 years ago

Nice

Subscribe   Add Disqus to your site