**What is a Randomized Algorithm?**

An algorithm that uses random numbers to decide what to do next anywhere in its logic is called a Randomized Algorithm. For example, in Randomized Quick Sort, we use a random number to pick the next pivot (or we randomly shuffle the array). And in [Karger’s algorithm](https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/), we randomly pick an edge.

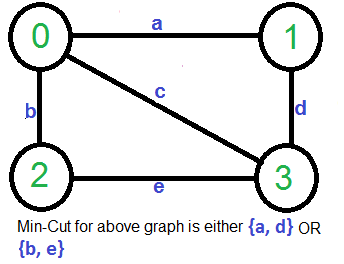
## ****How to analyse Randomized Algorithms?****

Some randomized algorithms have deterministic time complexity. For example, [this](https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/) implementation of Karger’s algorithm has time complexity is O(E). Such algorithms are called [Monte Carlo Algorithms](https://www.geeksforgeeks.org/randomized-algorithms-set-2-classification-and-applications/) and are easier to analyse for worst case.   
On the other hand, time complexity of other randomized algorithms (other than Las Vegas) is dependent on value of random variable. Such Randomized algorithms are called[Las Vegas Algorithms](https://www.geeksforgeeks.org/randomized-algorithms-set-2-classification-and-applications/). These algorithms are typically analysed for expected worst case. To compute expected time taken in worst case, all possible values of the used random variable needs to be considered in worst case and time taken by every possible value needs to be evaluated. Average of all evaluated times is the expected worst case time complexity. Below facts are generally helpful in analysis os such algorithms.

# Karger’s algorithm for Minimum Cut

Given an undirected and unweighted graph, find the smallest cut (smallest number of edges that disconnects the graph into two components).   
The input graph may have parallel edges.

For example consider the following example, the smallest cut has 2 edges.



A Simple Solution use [Max-Flow based](https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/) [s-t cut algorithm](https://www.geeksforgeeks.org/minimum-cut-in-a-directed-graph/) to find minimum cut. Consider every pair of vertices as source ‘s’ and sink ‘t’, and call minimum s-t cut algorithm to find the s-t cut. Return minimum of all s-t cuts. Best possible time complexity of this algorithm is O(V5) for a graph. [How? there are total possible V2 pairs and s-t cut algorithm for one pair takes O(V\*E) time and E = O(V2)].

Below is simple Karger’s Algorithm for this purpose. Below Karger’s algorithm can be implemented in O(E) = O(V2) time.

1) Initialize contracted graph CG as copy of original graph

2) While there are more than 2 vertices.

a) Pick a random edge (u, v) in the contracted graph.

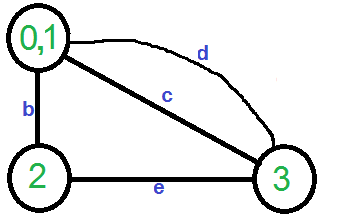
b) Merge (or contract) u and v into a single vertex (update

the contracted graph).

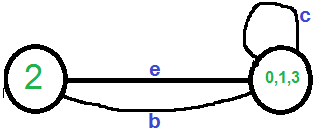
c) Remove self-loops

3) Return cut represented by two vertices.

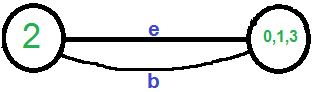
Let us understand above algorithm through the example given.  
Let the first randomly picked vertex be ‘**a**‘ which connects vertices 0 and 1. We remove this edge and contract the graph (combine vertices 0 and 1). We get the following graph.



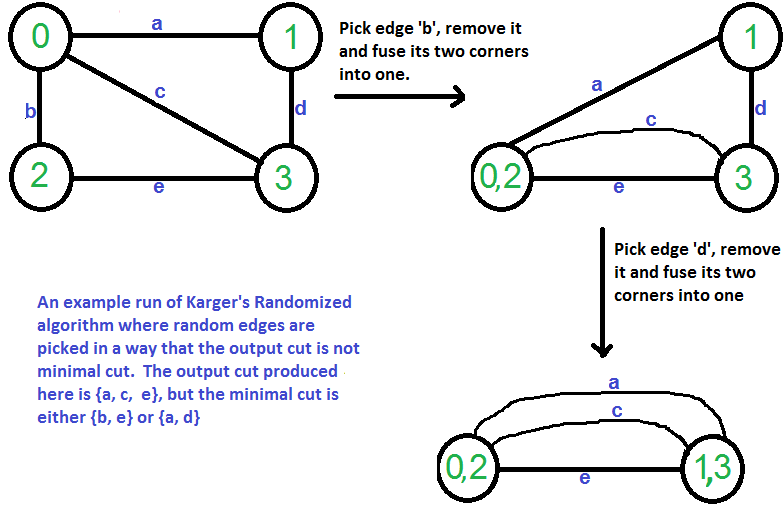
Let the next randomly picked edge be ‘d’. We remove this edge and combine vertices (0,1) and 3.



We need to remove self-loops in the graph. So we remove edge ‘c’



Now graph has two vertices, so we stop. The number of edges in the resultant graph is the cut produced by Karger’s algorithm.  
***Karger’s algorithm is a***[***Monte Carlo algorithm***](http://en.wikipedia.org/wiki/Monte_Carlo_algorithm)***and cut produced by it may not be minimum.***For example, the following diagram shows that a different order of picking random edges produces a min-cut of size 3.



# Randomized Algorithms (Classification and Applications)

## ****Classification****

Randomized algorithms are classified in two categories.

**Las Vegas:** These algorithms always produce correct or optimum result. Time complexity of these algorithms is based on a random value and time complexity is evaluated as expected value. For example, Randomized QuickSort always sorts an input array and [expected worst case time complexity of QuickSort is O(nLogn)](https://www.geeksforgeeks.org/randomized-algorithms-set-1-introduction-and-analysis/).

**Monte Carlo:** Produce correct or optimum result with some probability. These algorithms have deterministic running time and it is generally easier to find out worst case time complexity. For example [this implementation of Karger’s Algorithm](https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/) produces minimum cut with probability greater than or equal to 1/n2 (n is number of vertices) and has worst case time complexity as O(E). Another example is [Fermet Method for Primality Testing](https://www.geeksforgeeks.org/primality-test-set-2-fermet-method/).

**Example to Understand Classification:**

Consider a binary array where exactly half elements are 0

and half are 1. The task is to find index of any 1.

A Las Vegas algorithm for this task is to keep picking a random element until we find a 1. A Monte Carlo algorithm for the same is to keep picking a random element until we either find 1 or we have tried maximum allowed times say k.  
The Las Vegas algorithm always finds an index of 1, but time complexity is determined as expect value. The [expected number of trials before success](https://www.geeksforgeeks.org/expected-number-of-trials-before-success/) is 2, therefore expected time complexity is O(1).  
The Monte Carlo Algorithm finds a 1 with probability [1 – (1/2)k]. Time complexity of Monte Carlo is O(k) which is deterministic

## ****Applications and Scope:****

* Consider a tool that basically does sorting. Let the tool be used by many users and there are few users who always use tool for already sorted array. If the tool uses simple (not randomized) QuickSort, then those few users are always going to face worst case situation. On the other hand if the tool uses Randomized QuickSort, then there is no user that always gets worst case. Everybody gets expected O(n Log n) time.
* Randomized algorithms have huge applications in Cryptography.
* [Load Balancing](https://www.geeksforgeeks.org/load-balancing-on-servers-random-algorithm/).
* Number-Theoretic Applications: [Primality Testing](https://en.wikipedia.org/wiki/Solovay%E2%80%93Strassen_primality_test)
* Data Structures: Hashing, Sorting, Searching, [Order Statistics](https://www.geeksforgeeks.org/kth-smallestlargest-element-unsorted-array-set-2-expected-linear-time/) and Computational Geometry.
* Algebraic identities: Polynomial and [matrix identity verification](https://en.wikipedia.org/wiki/Randomized_algorithm#Verifying_matrix_multiplication). Interactive proof systems.
* Mathematical programming: Faster algorithms for linear programming, Rounding linear program solutions to integer program solutions
* Graph algorithms: Minimum spanning trees, shortest paths, [minimum cuts](https://www.geeksforgeeks.org/kargers-algorithm-for-minimum-cut-set-1-introduction-and-implementation/).
* Counting and enumeration: Matrix permanent Counting combinatorial structures.
* Parallel and distributed computing: Deadlock avoidance distributed consensus.
* Probabilistic existence proofs: Show that a combinatorial object arises with non-zero probability among objects drawn from a suitable probability space.
* Derandomization: First devise a randomized algorithm then argue that it can be derandomized to yield a deterministic algorithm.