CSC196: Great Ideas in Computing

Tutorial 3

5th October 2022

Birthday paradox

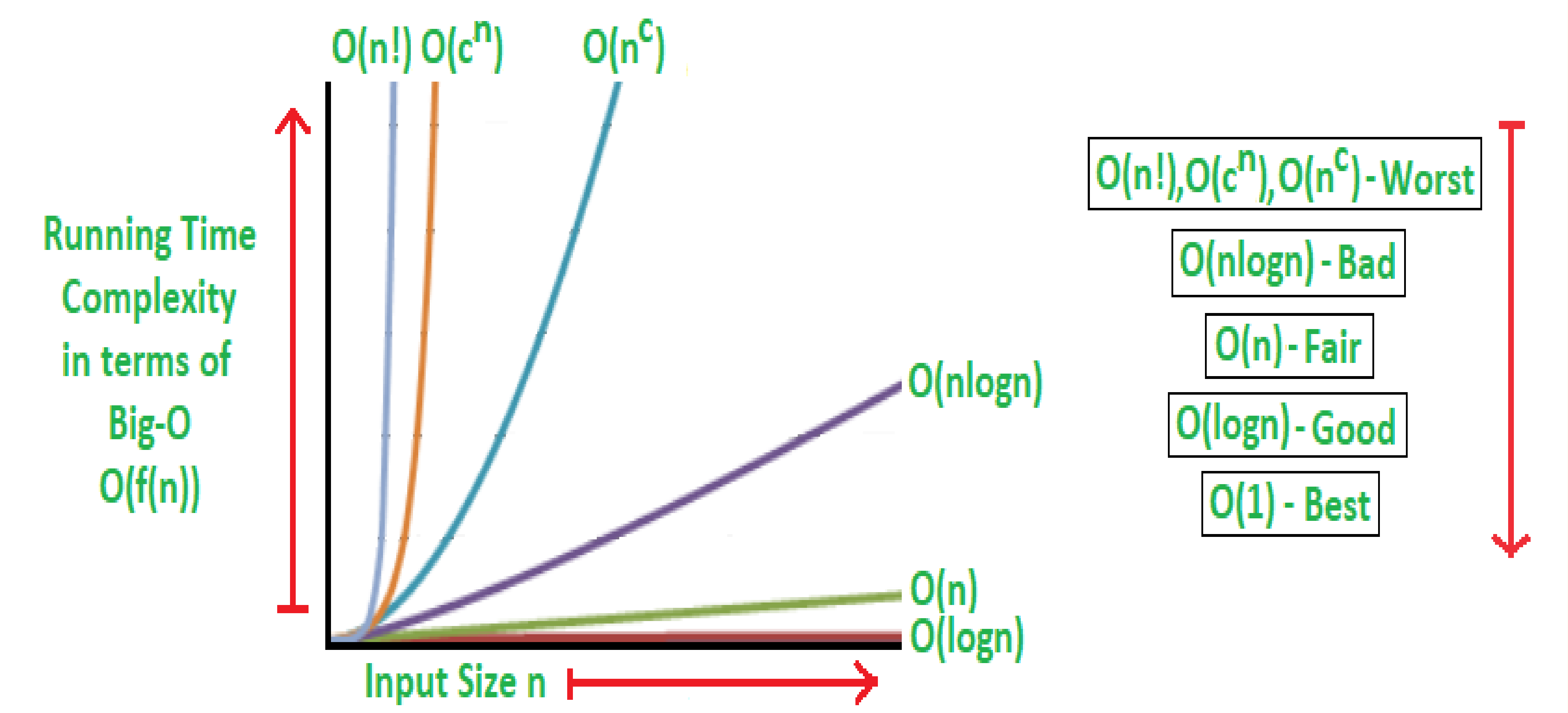
- Write your DOBs on a piece of paper

Q2. Asymptotic analysis of algorithms

- Big O (Upper bound)
  - $f(n)$ is $O(g(n))$
    - a.k.a $f(n)$ is Big oh of $g(n)$
    - a.k.a $f(n)$ is order of $g(n)$
    - $f(n) <= c * g(n)$ for all n > n_0 for an arbitrary n_0 and c

- Examples
  - $O(1)$: Constant time
  - $O(logN)$: Logarithmic time
  - $O(N)$: Linear time
  - $O(N^2)$: Quadratic time
  - $O(N^2 + logN) = O(N^2) + O(logN)$

- Other bounds
  - Big Omega (lower bound)
  - Big Theta (tight bound)



- Reference: https://www.geeksforgeeks.org/analysis-algorithms-big-o-analysis/

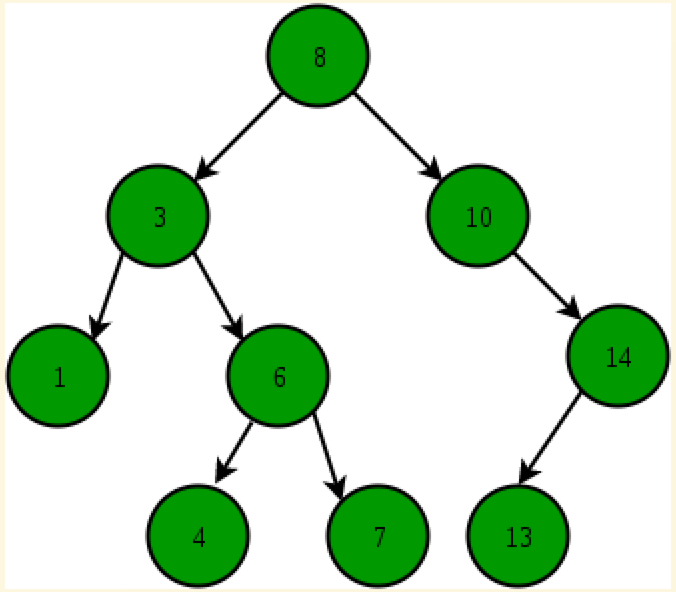
n	Constant $O(1)$	Logarithmic $O(\log n)$	Linear $O(n)$	Linear Logarithmic $O(n \log n)$	Quadractic $O(n^2)$	Cubic $O(n^3)$
1	1	1	1	1	1	1
2	1	1	2	2	4	8
4	1	2	4	8	16	64
8	1	3	8	24	64	512
16	1	4	16	64	256	4,096
1,024	1	10	1,024	10,240	1,048,576	1,073,741,824

- Reference: https://dzone.com/articles/learning-big-o-notation-with-on-complexity

Q2. Binary Search Tree

- Binary Tree
  - There is a root node which has in degree = 0 (no incoming edges)
  - There are leaf nodes which has out degree = 0 (no outgoing edges)
  - Each node has at most two outgoing pointers and at most one incoming pointer
  - Number of levels = O(log(number_of_nodes))

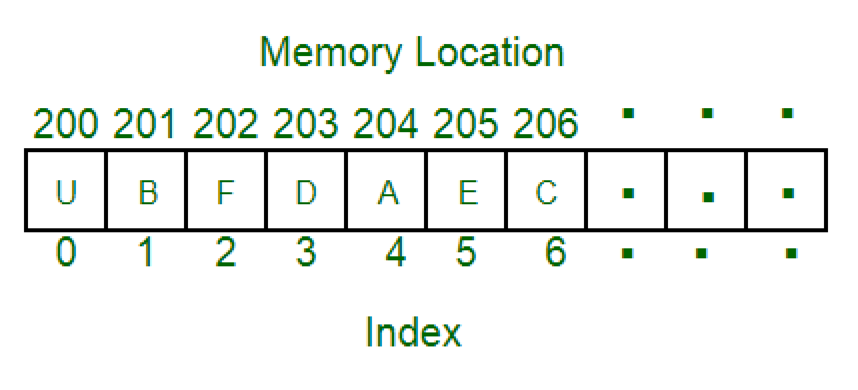
- Binary Search Tree (BST)
  - All nodes in the left subtree have lesser value than root node
  - All nodes in the right subtree have greater value than root node
  - Left and right subtrees are BST as well
- Used to implement ordered dictionary (keys are sorted)



- Reference: https://www.geeksforgeeks.org/binary-search-tree-data-structure/

Q2. Sorted Array

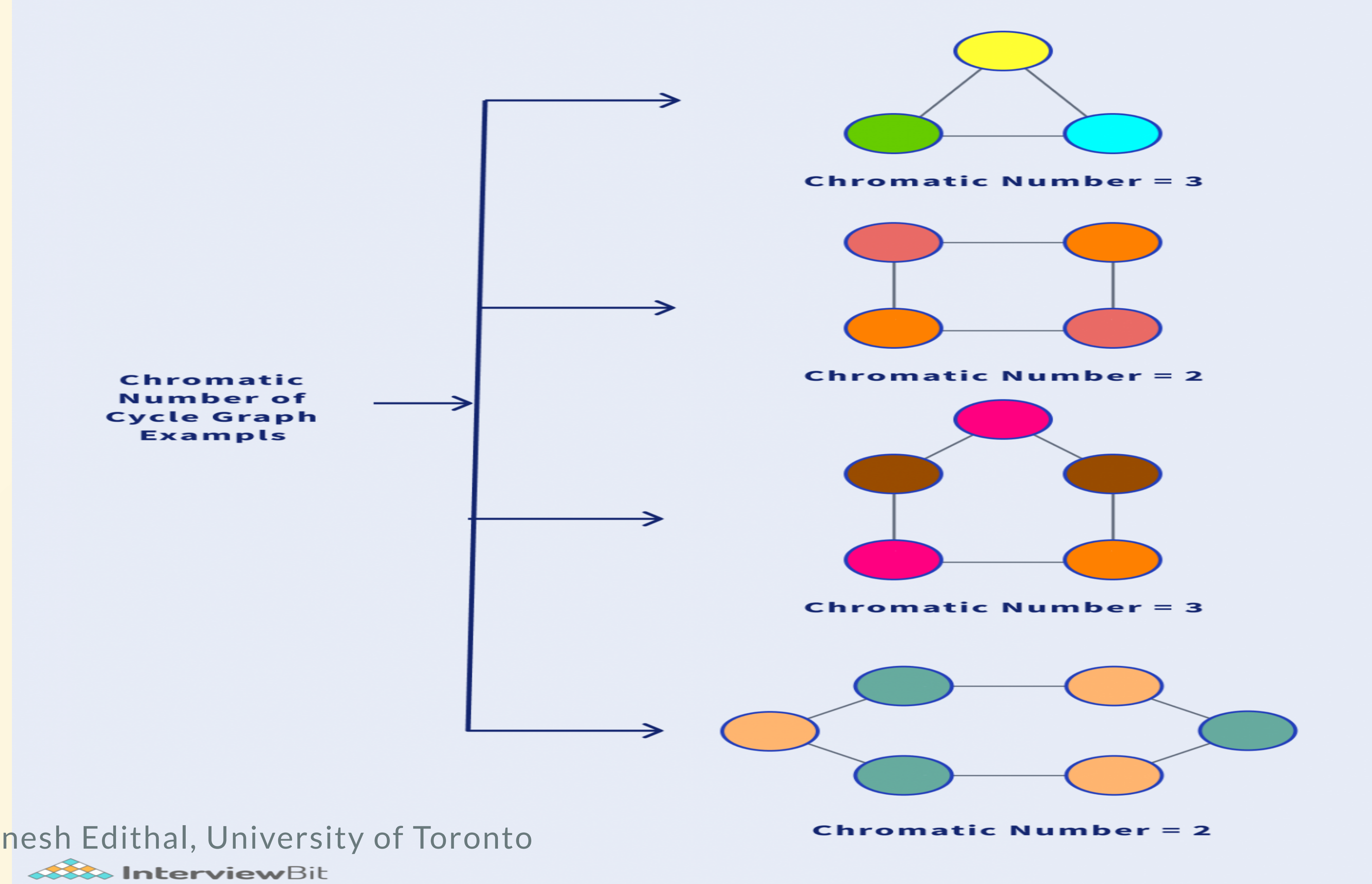
- Array can be implemented using linked list
  - Doubly linked list for traversal from either end
- Searching (reading) is faster in sorted array, however writing is slower
  - Reading: O(log(n))
  - Writing: O(log(n))
- Unsorted array has faster writes but slower reads
  - Reading: O(n)
  - Writing: O(1)



- Reference: https://www.geeksforgeeks.org/array-data-structure/

Q3. Graph coloring problem

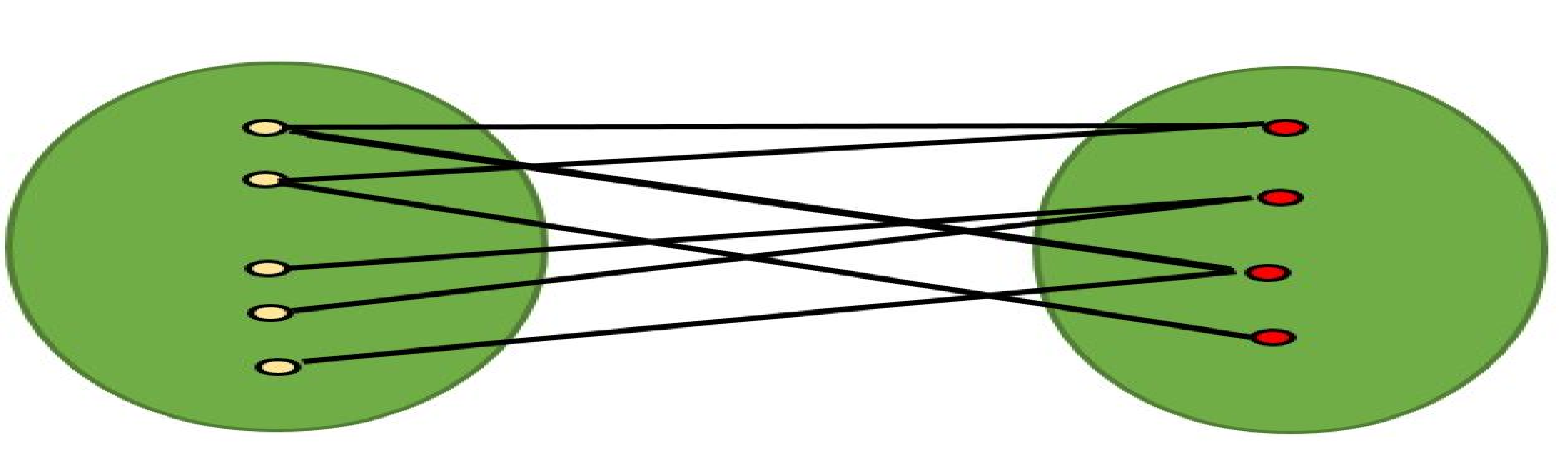
- Vertex coloring is the most common graph coloring problem
- Color the vertices such that no two adjacent vertices are colored the same
  - Graph is k colorable if you can do this with k colors
  - Chromatic number: Least possible value of k
  - Use Breadth first search (BFS) for two coloring
- Reference: https://www.geeksforgeeks.org/graph-coloring-applications/


Chromatic Number of Cycle Graph Exampls
Chromatic Number = 3
Chromatic Number = 2
Chromatic Number = 3
Chromatic Number = 2
InterviewBit

- Finding chromatic number ($N > 3$) is NP complete
- Reference: https://www.interviewbit.com/blog/graph-coloring-problem/

Q3. Bipartite graph

- Vertices can be divided into two disjoint sets
  - All edges are in between the sets



- Reference: https://www.geeksforgeeks.org/bipartite-graph/

Assignment 1

- Deadline 7th October (Friday 8AM)

Birthday Paradox

- Pigeon hole principle: Same birthday guranteed for n > 365 people
- Probability of same birthday is not $n/365$ but
  - P = 1 - $\frac{365*364*...*(365-(n-1))}{365^n} = 1 - \frac{P_n^{365}}{365^n}$
  - For $n = 20, P = 41.1\%$
  - For $n = 30, P = 70.6\%$
- Reference: https://en.wikipedia.org/wiki/Birthday_problem#Calculating_the_probability

Example of computable transformation

- Equivalence of Vertex cover problem and Independent set problem
  - The solution to one problem can be computed using solution to the other problem
- Vertex Cover problem
  - Identify a set of vertices such that all edges are connected to these vertices
- Independent Set problem
  - Identify a set of vertices such that there are no edges between them
- Removing the vertices of a given vertex cover from the graph leaves us with independent set
  - And vice versa