Backtracking

By: Ashok Basnet

Lecture Outline

- Introduction to Backtracking method of problem solving
- The 8-queen problem
- Sum of Sub-set problem
- Graph coloring problem
- Hamilton Cycle

Introduction to Backtracking method of problem solving

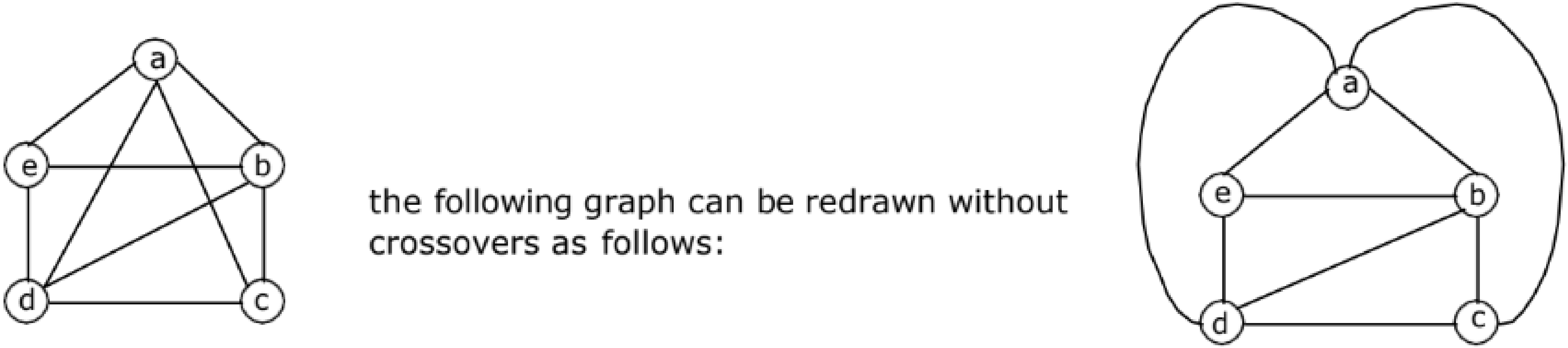
- Backtracking is used to solve problem in which a sequence of objects is chosen from a specified set so that the sequence satisfies some criterion.
- Backtracking is a modified **depth first search** of a tree.
- Backtracking algorithms determine problem solutions by systematically searching the solution space for the given problem instance.
- This search is facilitated by using a tree organization for the solution space.
- Backtracking is the procedure where by, after determining that a node can lead to nothing but dead end, we go back (backtrack) to the nodes parent and proceed with the search on the next child.

Terminology

- **Problem state** is each node in the depth first search tree.
- **Solution states** are the problem states ‘S’ for which the path from the root node to ‘S’ defines a tuple in the solution space.
- **Answer states** are those solution states for which the path from root node to s defines a tuple that is a member of the set of solutions.
- **State space** is the set of paths from root node to other nodes. State space tree is the tree organization of the solution space.
- **Live node** is a node that has been generated but whose children have not yet been generated.
- **E-node** is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.
- **Dead node** is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.

Planar Graphs

- When drawing a graph on a piece of a paper, we often find it convenient to permit edges to intersect at points other than at vertices of the graph.
- These points of interactions are called crossovers.
- A graph G is said to be planar if it can be drawn on a plane without any crossovers; otherwise G is said to be non-planar i.e., A graph is said to be planar.



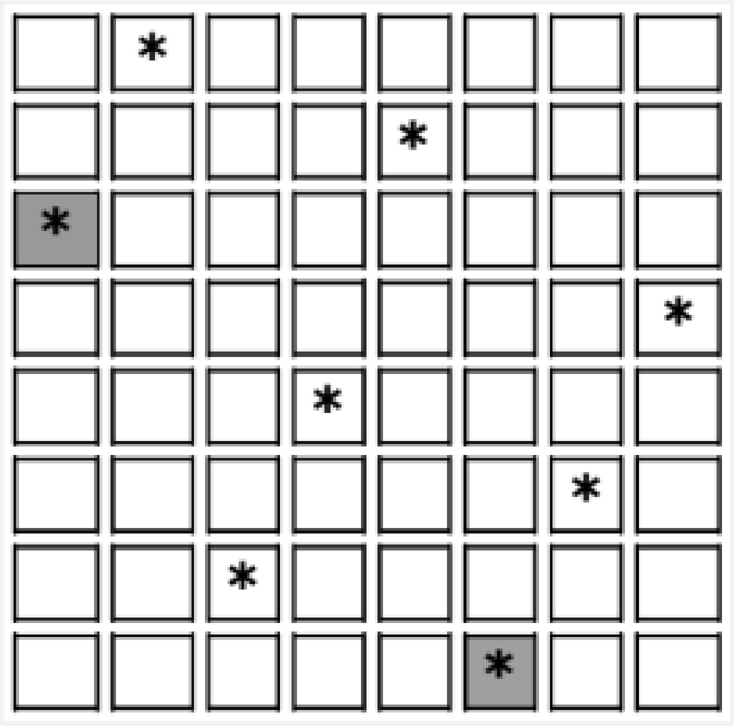
N-Queens Problem

- Let us consider, N = 8. Then 8-Queens Problem is to place eight queens on an 8 x 8 chessboard so that no two “attack”, that is, no two of them are on the same row, column, or diagonal.
- All solutions to the 8-queens problem can be represented as 8-tuples (x1, ... , x8), where x^i is the column of the i^{th} row where the i^{th} queen is placed.
- The explicit constraints using this formulation are $S_i = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $1 < i < 8$. Therefore the solution space consists of 8^8 8-tuples.
- The implicit constraints for this problem are that no two xi’s can be the same (i.e., all queens must be on different columns) and no two queens can be on the same diagonal.
- This realization reduces the size of the solution space from 8^8 tuples to 8! Tuples.

N-Queens Problem

- The promising function must check whether two queens are in the same column or diagonal:
- Suppose two queens are placed at positions (i, j) and (k, l) Then:
- **Column Conflicts:** Two queens conflict if their x_i values are identical.
- **Diag 45 conflict:** Two queens i and j are on the same 45^0 diagonal if: $i – j = k – l$. This implies, $j – l = i – k$.
- **Diag 135 conflict:** $i + j = k + l$. This implies, $j – l = k – i$.
- Therefore, two queens lie on the same diagonal if and only if: $|j – l| = |i – k|$
- Where, j be the column of object in row i for the i^{th} queen and l be the column of object in row ‘k’ for the kth queen.

To check the diagonal clashes, let us take the following tile configuration:



In this example, we have:

i	1	2	3	4	5	6	7	8
x_i	2	5	1	8	4	7	3	6

Let us consider for the 3rd row and 8th row case whether the queens on are conflicting or not. In this case $(i, j) = (3, 1)$ and $(k, l) = (8, 6)$. Therefore:

$$|j - l| = |i - k| \Rightarrow |1 - 6| = |3 - 8|$$

$$\Rightarrow 5 = 5$$

In the above example we have, $|j - l| = |i - k|$, so the two queens are attacking. This is not a solution.

Suppose we start with the feasible sequence 7, 5, 3, 1.

						*	
				*			
		*					
*							

Step 1:

Add to the sequence the next number in the sequence 1, 2, . . . , 8 not yet used.

Suppose we start with the feasible sequence 7, 5, 3, 1.

						*	
				*			
		*					
*							

Step 1:

Add to the sequence the next number in the sequence 1, 2, . . . , 8 not yet used.

Step 2:

If this new sequence is feasible and has length 8 then STOP with a solution. If the new sequence is feasible and has length less then 8, repeat Step 1.

Step 3:

If the sequence is not feasible, then *backtrack* through the sequence until we find the *most recent* place at which we can exchange a value. Go back to Step 1.

1	2	3	4	5	6	7	8	Remarks
7	5	3	1					
7	5	3	1*	2*				$\lvert j - l \rvert = \lvert 1 - 2 \rvert = 1$ $\lvert i - k \rvert = \lvert 4 - 5 \rvert = 1$
7	5	3	1	4				
7*	5	3	1	4	2*			$\lvert j - l \rvert = \lvert 7 - 2 \rvert = 5$ $\lvert i - k \rvert = \lvert 1 - 6 \rvert = 5$
7	5	3*	1	4	6*			$\lvert j - l \rvert = \lvert 3 - 6 \rvert = 3$ $\lvert i - k \rvert = \lvert 3 - 6 \rvert = 3$
7	5	3	1	4	8			
7	5	3	1	4*	8	2*		$\lvert j - l \rvert = \lvert 4 - 2 \rvert = 2$ $\lvert i - k \rvert = \lvert 5 - 7 \rvert = 2$
7	5	3	1	4*	8	6*		$\lvert j - l \rvert = \lvert 4 - 6 \rvert = 2$ $\lvert i - k \rvert = \lvert 5 - 7 \rvert = 2$
7	5	3	1	4	8			*Backtrack*
7	5	3	1	4				*Backtrack*

7	5	3	1	6				
7*	5	3	1	6	2*			$\lvert j - l \rvert = \lvert 1 - 2 \rvert = 1$ $\lvert i - k \rvert = \lvert 7 - 6 \rvert = 1$
7	5	3	1	6	4			
7	5	3	1	6	4	2		
7	5	3*	1	6	4	2	8*	$\lvert j - l \rvert = \lvert 3 - 8 \rvert = 5$ $\lvert i - k \rvert = \lvert 3 - 8 \rvert = 5$
7	5	3	1	6	4	2		*Backtrack*
7	5	3	1	6	4			*Backtrack*
7	5	3	1	6	8			
7	5	3	1	6	8	2		
7	5	3	1	6	8	2	4	**SOLUTION**

Sum of Sub-set problem

- Given a set of non-negative integers and a value **sum**, the task is to check if there is a subset of the given set whose sum is equal to the given **sum**.
- **Examples:**
- ***Input:*** *set[] = {3, 34, 4, 12, 5, 2}, sum = 9*
  Output: True
  Explanation: There is a subset (4, 5) with sum 9.
- ***Input:*** *set[] = {3, 34, 4, 12, 5, 2}, sum = 30*
  Output: False
  Explanation: There is no subset that add up to 30.

Sum of Sub-set problem

- Given positive numbers w_i, $1 \le i \le n$, and m, this problem requires finding all subsets of w_i whose sums are 'm'.
- All solutions are k-tuples, $1 \le k \le n$.
- For example, n = 4, w = (11, 13, 24, 7) and m = 31, the desired subsets are (11, 13, 7) and (24, 7).
- Draw the tree for the given problem to find the solution.

Graph coloring problem

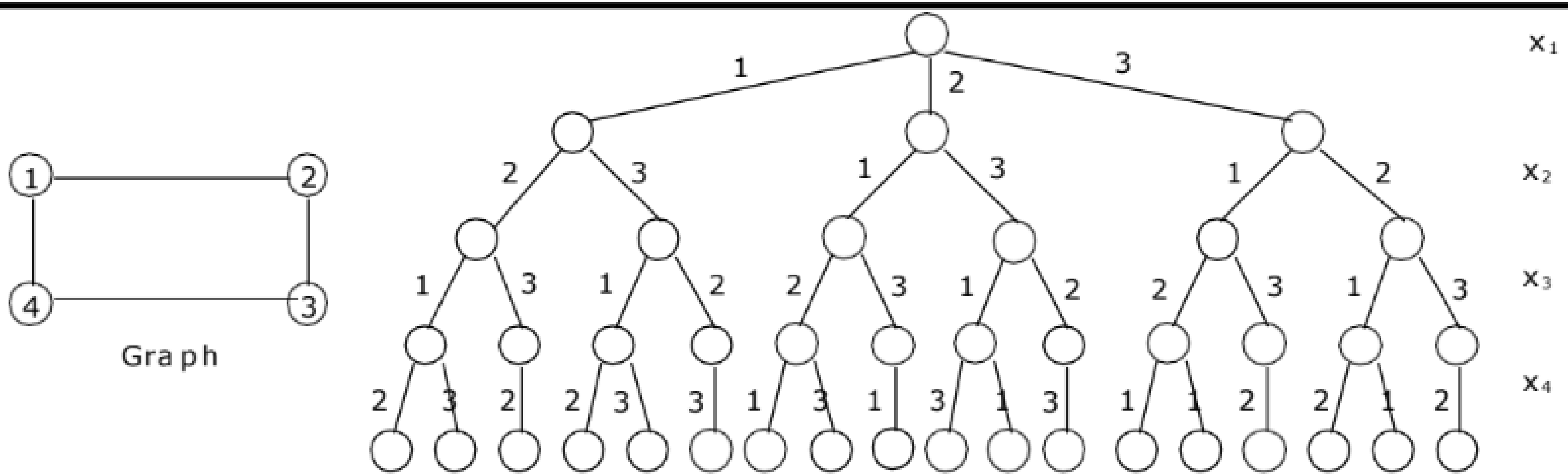
- Let G be a graph and m be a given positive integer.
- We want to discover whether the nodes of G can be colored in such a way that no two adjacent nodes have the same color, yet only m colors are used.
- This is termed the m-colorabiltiy decision problem.
- The m-colorability optimization problem asks for the smallest integer m for which the graph G can be colored.
- Given any map, if the regions are to be colored in such a way that no two adjacent regions have the same color, only four colors are needed.
- For many years it was known that five colors were sufficient to color any map, but no map that required more than four colors had ever been found.
- After several hundred years, this problem was solved by a group of mathematicians with the help of a computer. They showed that in fact four colors are sufficient for planar graphs.

Graph coloring problem

- The function m-coloring will begin by first assigning the graph to its adjacency matrix, setting the array x [] to zero.
- The colors are represented by the integers 1, 2, . . . , m and the solutions are given by the n-tuple (x1, x2, . . ., xn), where xi is the color of node i.

Example:

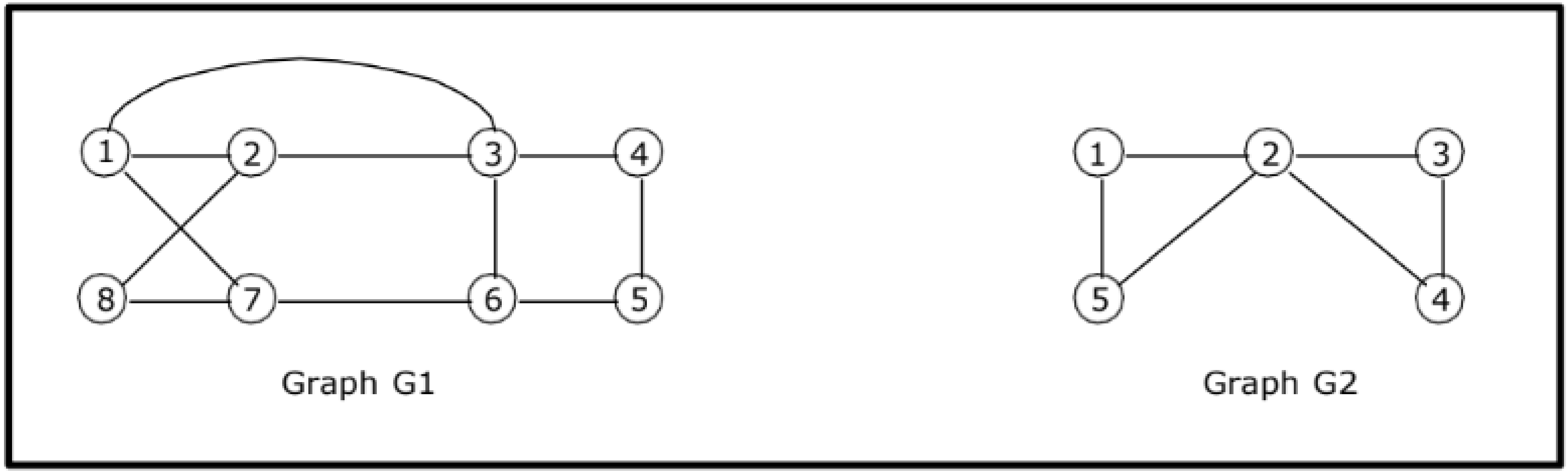
Color the graph given below with minimum number of colors by backtracking using state space tree.



A 4-node graph and all possible 3-colorings

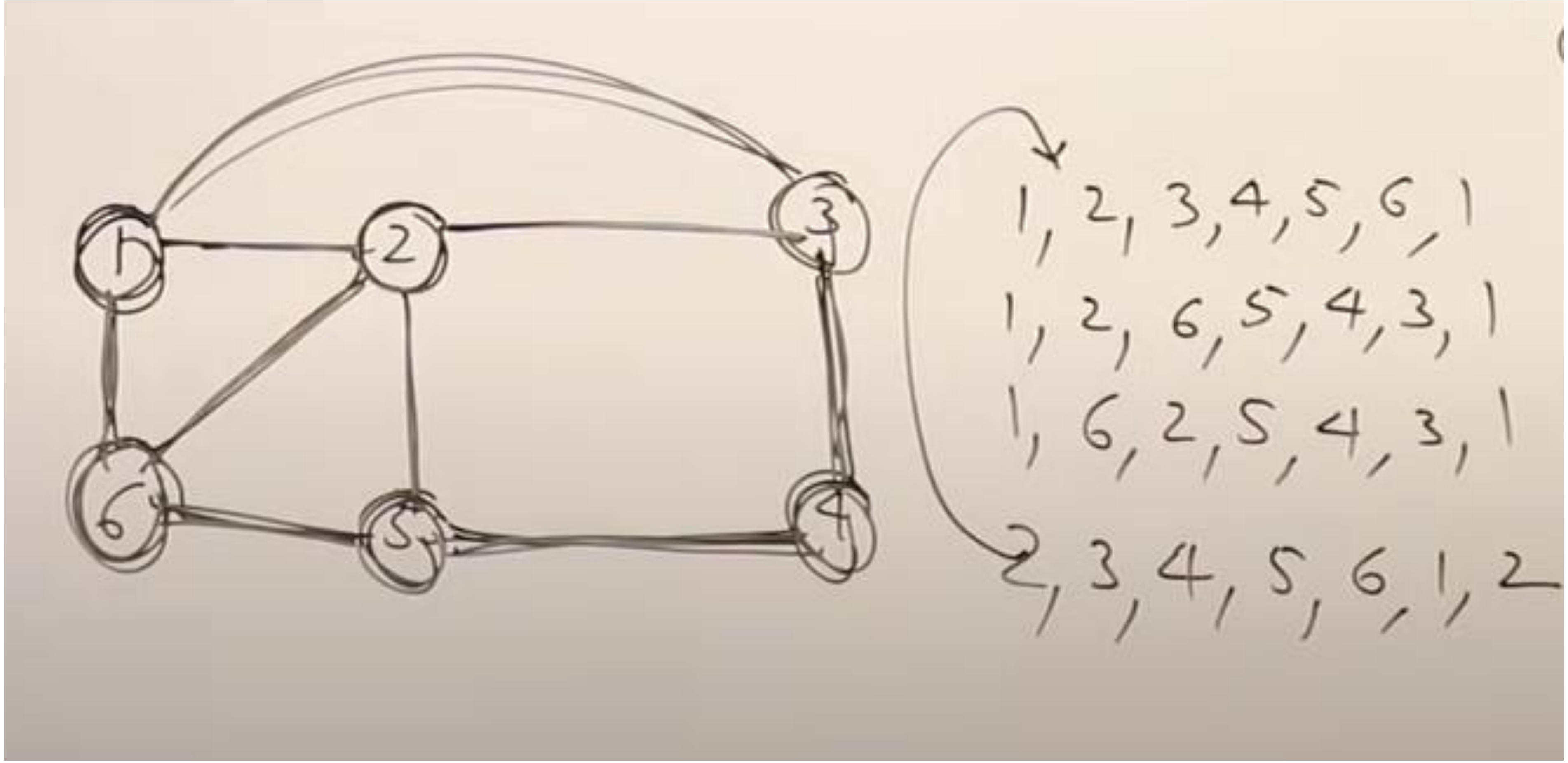
Hamilton Cycle

- Let G = (V, E) be a connected graph with n vertices. A Hamiltonian cycle (suggested by William Hamilton) is a round-trip path along n edges of G that visits every vertex once and returns to its starting position.
- The graph G1 contains the Hamiltonian cycle 1, 2, 8, 7, 6, 5, 4, 3, 1. The graph G2 contains no Hamiltonian cycle.

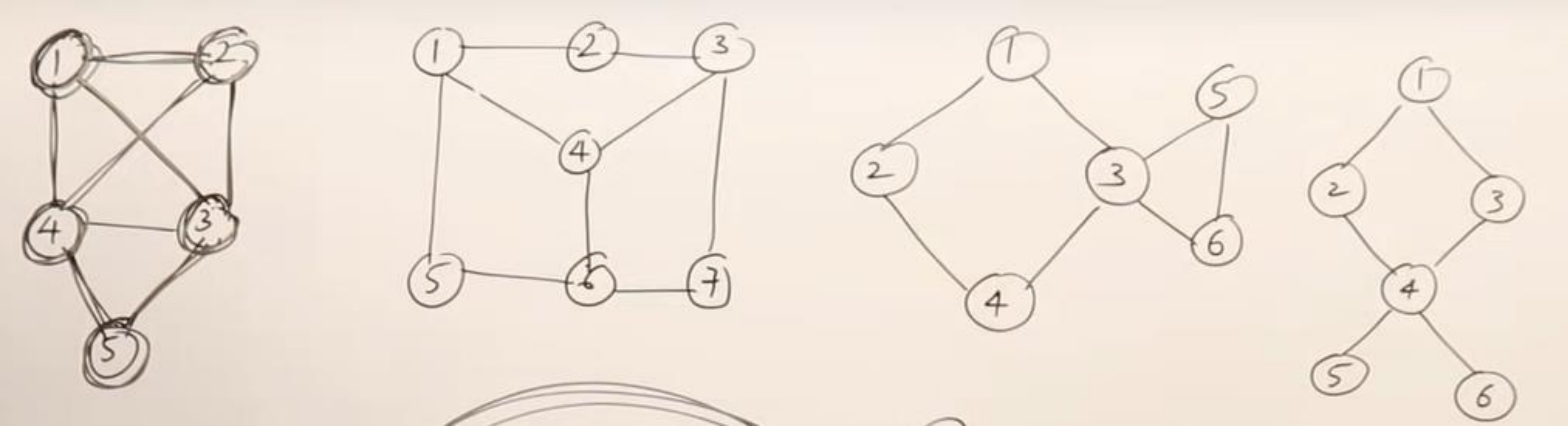


Two graphs to illustrate Hamiltonian cycle

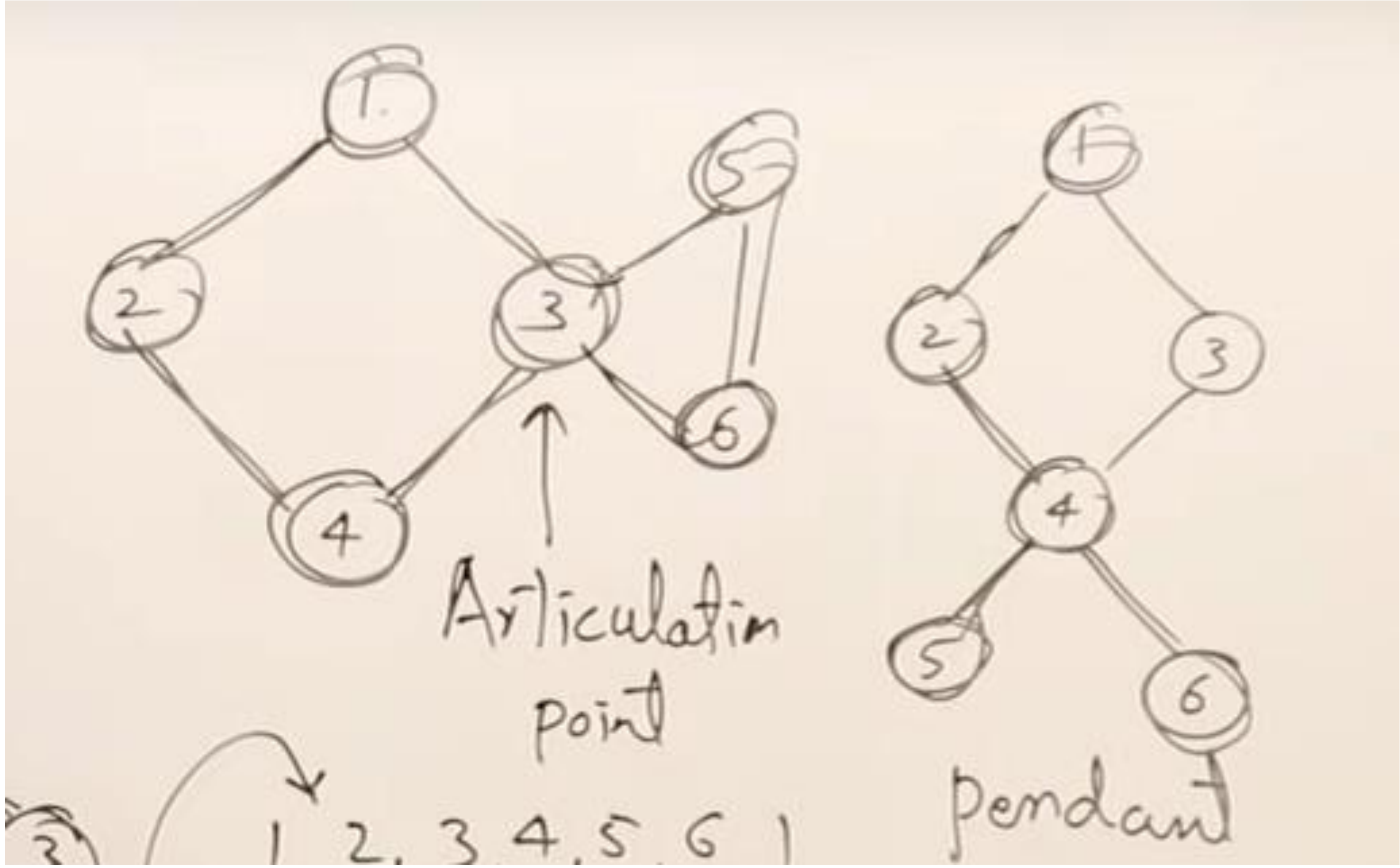
Hamilton Cycle



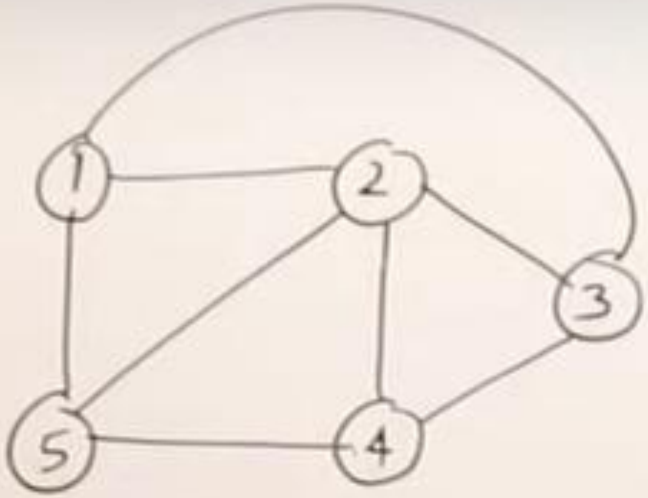
Hamilton Cycle



Hamilton Cycle



Hamilton Cycle



$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

x					
	1	2	3	4	5