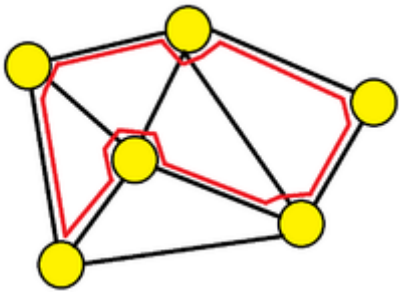
Hamiltonian path

In the mathematical field of graph theory, a **Hamiltonian path** (or **traceable path**) is a path in an undirected or directed graph that visits each vertex exactly once. A **Hamiltonian cycle** (or **Hamiltonian circuit**) is a cycle that visits each vertex exactly once. A Hamiltonian path that starts and ends at adjacent vertices can be completed by adding one more edge to form a Hamiltonian cycle, and removing any edge from a Hamiltonian cycle produces a Hamiltonian path. Determining whether such paths and cycles exist in graphs (the Hamiltonian path problem and Hamiltonian cycle problem) are NP-complete.



A Hamiltonian cycle around a network of six vertices

Hamiltonian paths and cycles are named after William Rowan Hamilton who invented the icosian game, now also known as *Hamilton's puzzle*, which involves finding a Hamiltonian cycle in the edge graph of the dodecahedron. Hamilton solved this problem using the icosian calculus, an algebraic structure based on roots of unity with many similarities to the quaternions (also invented by Hamilton). This solution does not generalize to arbitrary graphs.

Despite being named after Hamilton, Hamiltonian cycles in polyhedra had also been studied a year earlier by Thomas Kirkman, who, in particular, gave an example of a polyhedron without Hamiltonian cycles.[1] Even earlier, Hamiltonian cycles and paths in the knight's graph of the chessboard, the knight's tour, had been studied in the 9th century in Indian mathematics by Rudrata, and around the same time in Islamic mathematics by al-Adli ar-Rumi. In 18th century Europe, knight's tours were published by Abraham de Moivre and Leonhard Euler.[2]

Definitions

A *Hamiltonian path* or *traceable path* is a path that visits each vertex of the graph exactly once. A graph that contains a Hamiltonian path is called a **traceable graph**. A graph is **Hamiltonian-connected** if for every pair of vertices there is a Hamiltonian path between the two vertices.

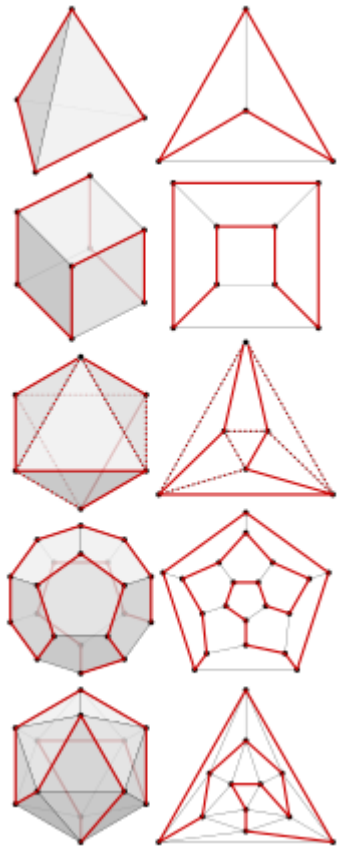
A *Hamiltonian cycle*, *Hamiltonian circuit*, *vertex tour* or *graph cycle* is a cycle that visits each vertex exactly once. A graph that contains a Hamiltonian cycle is called a **Hamiltonian graph**.

Similar notions may be defined for *directed graphs*, where each edge (arc) of a path or cycle can only be traced in a single direction (i.e., the vertices are connected with arrows and the edges traced "tail-to-head").

A Hamiltonian decomposition is an edge decomposition of a graph into Hamiltonian circuits.

A *Hamilton maze* is a type of logic puzzle in which the goal is to find the unique Hamiltonian cycle in a given graph.[3][4]

Examples



Orthographic projections and Schlegel diagrams with Hamiltonian cycles of the vertices of the five platonic solids – only the octahedron has an Eulerian path or cycle, by extending its path with the dotted one.

Examples

- A complete graph with more than two vertices is Hamiltonian

- Every cycle graph is Hamiltonian
- Every tournament has an odd number of Hamiltonian paths (Rédei 1934)
- Every platonic solid, considered as a graph, is Hamiltonian[5]
- The Cayley graph of a finite Coxeter group is Hamiltonian (For more information on Hamiltonian paths in Cayley graphs, see the Lovász conjecture.)
- Cayley graphs on nilpotent groups with cyclic commutator subgroup are Hamiltonian.[6]

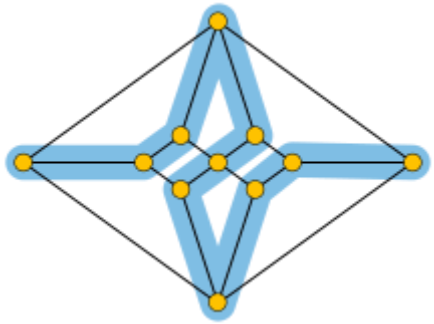
- The flip graph of a convex polygon or equivalently, the rotation graph of binary trees, is Hamiltonian.[7][8]

Properties

Any Hamiltonian cycle can be converted to a Hamiltonian path by removing one of its edges, but a Hamiltonian path can be extended to Hamiltonian cycle only if its endpoints are adjacent.

All Hamiltonian graphs are biconnected, but a biconnected graph need not be Hamiltonian (see, for example, the Petersen graph).[9]

An Eulerian graph G (a connected graph in which every vertex has even degree) necessarily has an Euler tour, a closed walk passing through each edge of G exactly once. This tour corresponds to a Hamiltonian cycle in the line graph $L(G)$, so the line graph of every Eulerian graph is Hamiltonian. Line graphs may have other Hamiltonian cycles that do not correspond to Euler tours, and in particular the line graph $L(G)$ of every Hamiltonian graph G is itself Hamiltonian, regardless of whether the graph G is Eulerian.[10]



The Herschel graph is the smallest possible polyhedral graph that does not have a Hamiltonian cycle. A possible Hamiltonian path is shown.

A tournament (with more than two vertices) is Hamiltonian if and only if it is strongly connected.

The number of different Hamiltonian cycles in a complete undirected graph on n vertices is $(n – 1)!/2$ and in a complete directed graph on n vertices is $(n – 1)!$. These counts assume that cycles that are the same apart from their starting point are not counted separately.

Bondy–Chvátal theorem

The best vertex degree characterization of Hamiltonian graphs was provided in 1972 by the Bondy–Chvátal theorem, which generalizes earlier results by G. A. Dirac (1952) and Øystein Ore. Both Dirac's and Ore's theorems can also be derived from Pósa's theorem (1962). Hamiltonicity has been widely studied with relation to various parameters such as graph density, toughness, forbidden subgraphs and distance among other parameters.[11] Dirac and Ore's theorems basically state that a graph is Hamiltonian if it has *enough edges*.

The Bondy–Chvátal theorem operates on the **closure** cl(G) of a graph G with n vertices, obtained by repeatedly adding a new edge uv connecting a nonadjacent pair of vertices u and v with $\deg(v) + \deg(u) \geq n$ until no more pairs with this property can be found.

Bondy–Chvátal Theorem (1976) — A graph is Hamiltonian if and only if its closure is Hamiltonian.

As complete graphs are Hamiltonian, all graphs whose closure is complete are Hamiltonian, which is the content of the following earlier theorems by Dirac and Ore.

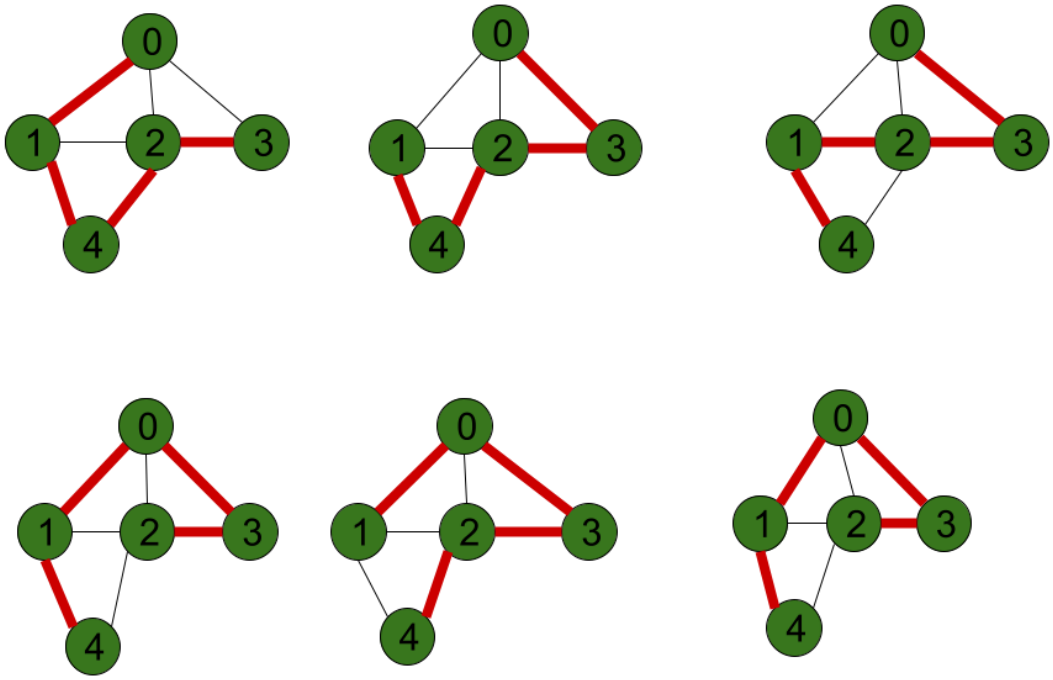
Hamiltonian Path (Using Dynamic Programming)

Given an adjacency matrix **adj[][]** of an undirected graph consisting of **N** vertices, the task is to find whether the graph contains a Hamiltonian Path or not. If found to be true, then print **"Yes"**. Otherwise, print **"No"**.

A Hamiltonian path is defined as the path in a directed or undirected graph which visits each and every vertex of the graph exactly once.

Examples:

Input: adj[][] = {{0, 1, 1, 1, 0}, {1, 0, 1, 0, 1}, {1, 1, 0, 1, 1}, {1, 0, 1, 0, 0}}
Output: Yes
Explanation:
There exists a Hamiltonian Path for the given graph as shown in the image below:



Input: adj[][] = {{0, 1, 0, 0}, {1, 0, 1, 1}, {0, 1, 0, 0}, {0, 1, 0, 0}}
Output: No

Recommended: Please try your approach on ***{IDE}*** first, before moving on to the solution.

Naive Approach: The simplest approach to solve the given problem is to generate all the possible permutations of **N** vertices. For each permutation, check if it is a valid Hamiltonian path by checking if there is an edge between adjacent vertices or not. If found to be true, then print **"Yes"**. Otherwise, print **"No"**.

Time Complexity: *O(N * N!)*
Auxiliary Space: *O(1)*

Efficient Approach: The above approach can be optimized by using Dynamic Programming and Bit Masking which is based on the following observations:

- The idea is such that for every subset **S** of vertices, check whether there is a hamiltonian path in the subset **S** that ends at vertex **v** where **v € S**.
- If **v** has a neighbor **u**, where **u € S – {v}**, therefore, there exists a Hamiltonian path that ends at vertex **u**.
- The problem can be solved by generalizing the subset of vertices and the ending vertex of the Hamiltonian path.

Follow the steps below to solve the problem:

- Initialize a boolean matrix **dp[][]** in dimension **N*2^N** where **dp[j][i]** represents whether there exists a path in the subset or not represented by the mask **i** that visits each and every vertex in **i** once and ends at vertex **j**.
- For the base case, update **dp[i][1 << i] = true**, for **i** in range **[0, N – 1]**
- Iterate over the range **[1, 2^N – 1]** using the variable **i** and perform the following steps:
    - All the vertices with bits set in mask **i,** are included in the subset.
    - Iterate over the range **[1, N]** using the variable **j** that will represent the end vertex of the hamiltonian path of current subset mask **i** and perform the following steps:
        - If the value of **i** and 2^j is true, then iterate over the range **[1, N]** using the variable **k** and if the value of **dp[k][i^2^j]** is **true**, then mark **dp[j][i]** is **true** and break out of the loop.
        - Otherwise, continue to the next iteration.
- Iterate over the range using the variable **i** and if the value of **dp[i][2^N – 1]** is **true**, then there exists a hamiltonian path ending at vertex **i**. Therefore, print **"Yes"**. Otherwise, print **"No"**.

Below is the implementation of the above approach:

```
// C++ program for the above approach

#include <bits/stdc++.h>
using namespace std;
const int N = 5;

// Function to check whether there
// exists a Hamiltonian Path or not
bool Hamiltonian_path(
    vector<vector<int> >& adj, int N)
{
    int dp[N][(1 << N)];
```

```
    // Initialize the table
    memset(dp, 0, sizeof dp);

    // Set all dp[i][(1 << i)] to
    // true
    for (int i = 0; i < N; i++)
        dp[i][(1 << i)] = true;

    // Iterate over each subset
    // of nodes
    for (int i = 0; i < (1 << N); i++) {

        for (int j = 0; j < N; j++) {

            // If the jth nodes is included
            // in the current subset
            if (i & (1 << j)) {

                // Find K, neighbour of j
                // also present in the
                // current subset
                for (int k = 0; k < N; k++) {

                    if (i & (1 << k)
                        && adj[k][j]
                        && j != k
                        && dp[k][i ^ (1 << j)]) {

                        // Update dp[j][i]
                        // to true
                        dp[j][i] = true;
                        break;
                    }
                }
            }
        }
    }

    // Traverse the vertices
    for (int i = 0; i < N; i++) {

        // Hamiltonian Path exists
        if (dp[i][(1 << N) - 1])
            return true;
    }

    // Otherwise, return false
    return false;
}

// Driver Code
int main()
{

    // Input
    vector<vector<int> > adj = { { 0, 1, 1, 1, 0 },
                                 { 1, 0, 1, 0, 1 },
                                 { 1, 1, 0, 1, 1 },
```

```
                                    { 1, 0, 1, 0, 0 } };
    int N = adj.size();

    // Function Call
    if (Hamiltonian_path(adj, N))
        cout << "YES";
    else
        cout << "NO";

    return 0;
}
```

Output:

```
YES
```

Time Complexity: $O(N^2 * 2^N)$
Auxiliary Space: $O(N * 2^N)$