GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Backtracking | Set 6 (Hamiltonian Cycle)

Hamiltonian Path in an undirected graph is a path that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian Path such that there is an edge (in graph) from the last vertex to the first vertex of the Hamiltonian Path. Determine whether a given graph contains Hamiltonian Cycle or not. If it contains, then print the path. Following are the input and output of the required function.

Input:
A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is adjacency matrix representation of the graph. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0.

Output:

An array path[V] that should contain the Hamiltonian Path. path[i] should represent the ith vertex in the Hamiltonian Path. The code should also return false if there is no Hamiltonian Cycle in the graph.

For example, a Hamiltonian Cycle in the following graph is {0, 1, 2, 4, 3, 0}. There are more Hamiltonian Cycles in the graph like {0, 3, 4, 2, 1, 0}

```
(0)--(1)--(2)
 |   / \   |
 |  /   \  |
 | /     \ |
(3)-------(4)
```
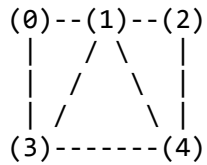
And the following graph doesn't contain any Hamiltonian Cycle.

```
(0)--(1)--(2)
 |   / \   |
 |  /   \  |
 | /     \ |
(3)       (4)
```
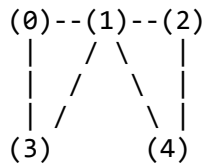
Naive Algorithm

Generate all possible configurations of vertices and print a configuration that satisfies the given constraints. There will be n! (n factorial) configurations.

```
while there are untried conflagrations
{
    generate the next configuration
    if ( there are edges between two consecutive vertices of this
       configuration and there is an edge from the last vertex to
       the first ).
    {
       print this configuration;
       break;
    }
}
```

Backtracking Algorithm

Create an empty path array and add vertex 0 to it. Add other vertices, starting from the vertex 1. Before adding a vertex, check for whether it is adjacent to the previously added vertex and not already added. If we find such a vertex, we add the vertex as part of the solution. If we do not find a vertex then we return false.

Implementation of Backtracking solution

Following is C/C++ implementation of the Backtracking solution.

```
// Program to print Hamiltonian cycle
#include<stdio.h>

// Number of vertices in the graph
#define V 5

void printSolution(int path[]);

/* A utility function to check if the vertex v can be added at index 'pos'
   in the Hamiltonian Cycle constructed so far (stored in 'path[]') */
bool isSafe(int v, bool graph[V][V], int path[], int pos)
{
```

```
    /* Check if this vertex is an adjacent vertex of the previously
       added vertex. */
    if (graph [ path[pos-1] ][ v ] == 0)
        return false;

    /* Check if the vertex has already been included.
      This step can be optimized by creating an array of size V */
    for (int i = 0; i < pos; i++)
        if (path[i] == v)
            return false;

    return true;
}

/* A recursive utility function to solve hamiltonian cycle problem */
bool hamCycleUtil(bool graph[V][V], int path[], int pos)
{
    /* base case: If all vertices are included in Hamiltonian Cycle */
    if (pos == V)
    {
        // And if there is an edge from the last included vertex to the
        // first vertex
        if ( graph[ path[pos-1] ][ path[0] ] == 1 )
           return true;
        else
          return false;
    }

    // Try different vertices as a next candidate in Hamiltonian Cycle.
    // We don't try for 0 as we included 0 as starting point in in hamCycle()
    for (int v = 1; v < V; v++)
    {
        /* Check if this vertex can be added to Hamiltonian Cycle */
        if (isSafe(v, graph, path, pos))
        {
            path[pos] = v;

            /* recur to construct rest of the path */
            if (hamCycleUtil (graph, path, pos+1) == true)
                 return true;

            /* If adding vertex v doesn't lead to a solution,
               then remove it */
            path[pos] = -1;
        }
    }

    /* If no vertex can be added to Hamiltonian Cycle constructed so far,
       then return false */
    return false;
}

/* This function solves the Hamiltonian Cycle problem using Backtracking.
   It mainly uses hamCycleUtil() to solve the problem. It returns false
   if there is no Hamiltonian Cycle possible, otherwise return true and
   prints the path. Please note that there may be more than one solutions,
   this function prints one of the feasible solutions. */
bool hamCycle(bool graph[V][V])
{
    int *path = new int[V];
    for (int i = 0; i < V; i++)
```

```c
        path[i] = -1;

    /* Let us put vertex 0 as the first vertex in the path. If there is
       a Hamiltonian Cycle, then the path can be started from any point
       of the cycle as the graph is undirected */
    path[0] = 0;
    if ( hamCycleUtil(graph, path, 1) == false )
    {
        printf("\nSolution does not exist");
        return false;
    }

    printSolution(path);
    return true;
}

/* A utility function to print solution */
void printSolution(int path[])
{
    printf ("Solution Exists:"
            " Following is one Hamiltonian Cycle \n");
    for (int i = 0; i < V; i++)
        printf(" %d ", path[i]);

    // Let us print the first vertex again to show the complete cycle
    printf(" %d ", path[0]);
    printf("\n");
}

// driver program to test above function
int main()
{
    /* Let us create the following graph
       (0)--(1)--(2)
        |   / \   |
        |  /   \  |
        | /     \ |
       (3)-------(4)    */
    bool graph1[V][V] = {{0, 1, 0, 1, 0},
                      {1, 0, 1, 1, 1},
                      {0, 1, 0, 0, 1},
                      {1, 1, 0, 0, 1},
                      {0, 1, 1, 1, 0},
                     };

    // Print the solution
    hamCycle(graph1);

    /* Let us create the following graph
       (0)--(1)--(2)
        |   / \   |
        |  /   \  |
        | /     \ |
       (3)       (4)    */
    bool graph2[V][V] = {{0, 1, 0, 1, 0},
                      {1, 0, 1, 1, 1},
                      {0, 1, 0, 0, 1},
                      {1, 1, 0, 0, 0},
                      {0, 1, 1, 0, 0},
                     };
```

```
    // Print the solution
    hamCycle(graph2);

    return 0;
}
```

Output:

```
Solution Exists: Following is one Hamiltonian Cycle
 0  1  2  4  3  0

Solution does not exist
```

Note that the above code always prints cycle starting from 0. Starting point should not matter as cycle can be started from any point. If you want to change the starting point, you should make two changes to above code.
Change "path[0] = 0;" to "path[0] = s;" where s is your new starting point. Also change loop "for (int v = 1; v < V; v++)" in hamCycleUtil() to "for (int v = 0; v < V; v++)".

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

Related Topics:

- [Assign directions to edges so that the directed graph remains acyclic](#)
- [K Centers Problem | Set 1 (Greedy Approximate Algorithm)](#)
- [Find the minimum cost to reach destination using a train](#)
- [Applications of Breadth First Traversal](#)
- [Optimal read list for given number of days](#)
- [Print all paths from a given source to a destination](#)
- [Minimize Cash Flow among a given set of friends who have borrowed money from each other](#)
- [Boggle (Find all possible words in a board of characters)](#)

Tags: [Backtracking](#), [Graph](#)

[📷]   **Tweet**   **g+1** 0

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

35 Comments   **GeeksforGeeks**   **1** Login ▾

♥ **Recommend** 2   ↗ Share   Sort by Newest ▾

[👤] | Join the discussion… |

[👤]   **Itachi Uchiha** · a month ago
         http://ideone.com/uCETqY

         A better and bit efficient version of the above implementation!. For those who didn't understand

the above code check this one. Its commented well. :)

⌃ | ⌄ · Reply · Share ›

Santhosh · 3 months ago

Same thing in java

package com.santhosh.test;

import java.util.ArrayList;

public class HamiltonCycle {

public static ArrayList<graphnode> visitedNodes = new ArrayList<graphnode>();

public static boolean isHamiltonCycle(GraphPath path, GraphNode A) {

System.out.println("Checking for Path From:" + A);

if (visitedNodes.size() > path.node.size()) {

if (visitedNodes.get(0).equals(A)) {

System.out.println("yes Hamilton Graph");

return true;

see more

⌃ | ⌄ · Reply · Share ›

Dheeraj Sachan · 4 months ago
https://github.com/dheeraj9198...
⌃ | ⌄ · Reply · Share ›

Meena Chaudhary · 6 months ago
I solved it using DFS and time complexity is $O(V^2)$ Below is the code. Hope it helps.

public boolean check() {

Stack<node> nodeStack = new Stack<>();

nodeStack.add(root);

root.setIsOnStack();

while (!nodeStack.isEmpty()) { //O(V), where V is total number of vertices

Node currentNode = nodeStack.peek(); //1

System.out.println(currentNode.getLabel());

```
for (Entry<node, boolean=""> entry : currentNode.getNeighbourList().entrySet()) { // V

    Node currentNeighbourer = entry.getKey();

    if (!currentNeighbourer.isOnStack()) { // 1
```

see more

∧ | ∨ · Reply · Share ›

Harsh Kara · 7 months ago

@geeksforgeeks
IS THIS TRUE ?
if the maximun length of cycle is equal to the number of vertices then we can say that hamilton cycle is present otherwise it's NOT
so we can easily inplement this by dfs
please correct me if wrong :)

∧ | ∨ · Reply · Share ›

Arpit → Harsh Kara · 4 months ago
also the cycle should contain all the vertices

EXACTLY ONCE except the starting vertex considering it to be a cycle.

∧ | ∨ · Reply · Share ›

arjomanD · a year ago
i think my code is much simpler

http://paste.ubuntu.com/743219...

∧ | ∨ · Reply · Share ›

arjomanD · a year ago
I had a question . (or maybe im wrong cuz i havn't still read the code) does this program find Hamiltonian path ?

1 ∧ | ∨ · Reply · Share ›

hxgxs1 · a year ago
time complexity should be O(N!)..there id a for loop in a recursive call so...
T(N)=N*(T(N-1) + O(1)) or
T(N)=N*(N-1)*(N-2)...=O(N!)

∧ | ∨ · Reply · Share ›

AlienOnEarth · a year ago
What would be the time complexity for this problem? Wikipedia says it can not be solved in Polynomial time as its a NP-Complete problem

∧ | ∨ · Reply · Share ›

iyer → AlienOnEarth · 7 months ago

i have the same question. did you get the answer?

∧ | ∨ · Reply · Share ›

Guest · a year ago

bool hamCycle(bool graph[V][V],bool*seen,int vertex)

```
{
int i;
for(i=0;i<v;i++) {="" if(graph[vertex][i]="1" &&="" seen[i]="false)" {="" seen[i]="true;"
stack[++top]="i;" if(hamcycle(graph,seen,i))="" return="" true;="" else="" top--;="" }="" }=""
return="" false;="" }="">
```

2 ∧ | ∨ · Reply · Share ›

Guest · a year ago

```
# include <stdio.h>
#include <stdbool.h>
// Number of vertices in the graph
#define V 5

void printSolution(int path[]);

int stack[V];
int top;
bool hamCycle(bool graph[V][V],bool*seen,int vertex)

{
int i;
for(i=0;i<v;i++) {="" if(graph[vertex][i]="1" &&="" seen[i]="false)" {="" seen[i]="true;"
stack[++top]="i;" if(hamcycle(graph,seen,i))="" return="" true;="" else="" top--;="" }="" }=""
return="" false;="" }="" driver="" program="" to="" test="" above="" function="" int="" main()=""
{="" *="" let="" us="" create="" the="" following="" graph="" (0)--(1)--(2)="" |="" \="" |="" |="" \=""
|="" |="" \="" |="" (3)-------(4)="" *="" bool="" graph1[v][v]="{{0," 1,="" 0,="" 1,="" 0},="" {1,=""
0,="" 1,="" 1,="" 1},="" {0,="" 1,="" 0,="" 0,="" 1},="" {1,="" 1,="" 0,="" 0,="" 1},="" {0,="" 1,=""
1,="" 1,="" 0},="" };="" print="" the="" solution="" bool="" seen[v]="{false};" seen[0]="true;"
stack[0]="0;" hamcycle(graph1,seen,0)="" if(top="V)" printf("there="" exists=""
hamcycle\n");="" int="" i;="" for(i="0;i&lt;V;i++)" printf("%d="" --="> ",stack[i] );
return 0;
}
```

∧ | ∨ · Reply · Share ›

da3m0n · a year ago

Time Complexity please..

∧ | ∨ • Reply • Share ›

Guest · 2 years ago

Algorithm for finding whether a graph is Hamiltonian or not:
Do DFS &maintain count of the adjacent unvisited vertices
If (there is no adjacent unvisited vertex){
if the count is equal to the num of vertices
return true;
else
return false;
}

8 ∧ | ∨ • Reply • Share ›

> **Siya** → Guest · 4 days ago
>
> There is a check whether all the vertices's are present but here is no condition to check whether every vertex is visited exactly once. I am not convinced with your solution.
>
> ∧ | ∨ • Reply • Share ›

> **Sriharsha g.r.v** → Guest · a year ago
>
> hi can u pls elaborate and explain with example..
>
> 1 ∧ | ∨ • Reply • Share ›

Born Actor · 2 years ago

```C++
[sourcecode language="C++"]
/* #include <iostream>
#include<string>
#include<sstream>
#include<iomanip>
# include <stdio.h>
# include <math.h>
#include <vector>
#include <stdlib.h>
using namespace std;
int a[50][50];
int n;

std::vector < pair <int, int > >edges;
int visited[50];
int cycle(int node);
void print();
int main()
```

see more

∧ | ∨ • Reply • Share ›

AMIT · 2 years ago

I think a minor improvement is possible...using a matrix soln to indicate which indices has already been visited,we can find isSafe in O(1)time.

```
 /* Paste your code here (You may delete these lines if not writing code) */
// Program to print Hamiltonian cycle
#include<stdio.h>

// Number of vertices in the graph
#define V 5
void print(int soln[V])
{
    int i,j;
    printf("solution exists\n");
    for(i=0;i<V;i++,printf("\n"))
    //for(j=0;j<V;j++)
    printf("%d  ",soln[i]);
    printf("\n\n");
}
```

see more

∧ | ∨ · Reply · Share ›

begfairouz → AMIT · 2 years ago

Hi,

In the beggining i have to thank you for your program, it works well.
But i tried to use it for a graph[V][V] with V=120 but the program doesn't return result (neither positive or negative). Did you try it for big matrices?

∧ | ∨ · Reply · Share ›

neham · 2 years ago

just a quick question - Hamiltonian Cycle belongs to NP-Complete problem how can we solve it using backtracking technique?

∧ | ∨ · Reply · Share ›

GeeksforGeeks · 2 years ago

Ankit: The main task of program is to find out whether a given graph contains Hamiltonian Cycle or not. And if there is a cycle, then print the cycle. The above program always does that. The starting point of cycle doesn&#039t matter, a cycle is a cycle and can be printed in any way.

If you don&#039t like 0 as a starting point of cycle, we have added a note below to change the starting point :)

starting point 1)

Also, we have removed self loops in the matrix representation of graph. Keep it up!

︿ | ﹀ · Reply · Share ›

Ankit Paharia · 2 years ago

The given input graph[][] has a bug.... it won&#039t work for all the input vertices such as 1,2,3,4... Why input graph is considered having self loops like graph[0][0]=1, graph[1][1]=1 etc. It should be -
bool graph[V][V] = {{0, 1, 0, 1, 0},

{1, 0, 1, 1, 1},

{0, 1, 0, 0, 1},

{1, 1, 0, 0, 1},

{0, 1, 1, 1, 0},

};.

If the loop in bool hamCycleUtil() is changed as -
for (int v = 0; v < V; v++){}.
it will work for all the vertices having hamiltonian cycle.... please check if I am correct or not....

︿ | ﹀ · Reply · Share ›

NNavneet · 2 years ago

why in the input graph[i][i]=1 , if we start with vertex 1 or any other vertex than 0 as starting vertex then code does not return correct path , it actually return that there is no path existing.

you correct this by making graph[i][i]=0 for all i's, and stating the for from v=0 in the hamCycleUtil function
Here is the code :

```
  #include<stdio.h>
 #include<iostream>
 // Number of vertices in the graph
 #define V 5
 using namespace std;


 void printSolution(int path[])
 {
        for(int i=0;i<V;i++)
        {
                cout<<path[i]<<" ";
```

see more

∧ | ∨ · Reply · Share ›

GeeksforGeeks → NNavneet · 2 years ago

The main task of program is to find out whether a given graph contains Hamiltonian Cycle or not. And if there is a cycle, then print the cycle. The above program always does that. The starting point of cycle doesn't matter, a cycle is a cycle and can be printed in any way.

If you don't like 0 as a starting point of cycle, we have added a note below the code to change the starting point :)

Also, we have removed self loops in the matrix representation of graph. Keep it up!

∧ | ∨ · Reply · Share ›

Anshul Agrawal · 2 years ago

grt work buddy..really helped me alot..

∧ | ∨ · Reply · Share ›

AAZ · 2 years ago

What would be the runtime complexity of the program ?

∧ | ∨ · Reply · Share ›

spark9 · 3 years ago

IsSafe function complexity can be improved by using better data structures. We can use an array[V] to mark whether a vertex is visited or not.

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

∧ | ∨ · Reply · Share ›

atul · 3 years ago

code will not work for all inputs

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

∧ | ∨ · Reply · Share ›

kartik → atul · 3 years ago

Please provide more details of your comment. Why do think that the code will not work? Do you have some sample graph for which it didn't work?

∧ | ∨ · Reply · Share ›

atul · 3 years ago

why assumption has been made that node 0 is connected to node 1.

if given graph is the one given below..then code will fail right??

```
(0)--(4)--(3)
 |   / \   |
 |  /   \  |
 | /     \ |
(2)-------(1)
```

[sourcecode language=""]
/* Paste your code here (You may delete these lines if not writing code) */

∧ | ∨ · Reply · Share ›

atul → atul · 3 years ago

ignore above comment

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

∧ | ∨ · Reply · Share ›

kavish · 3 years ago

good post....a more generalised version of knight's tour problem...

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

∧ | ∨ · Reply · Share ›

Venki · 3 years ago

How can we extend the solution to start at an arbitrary vertex? One way is to keep track of "pos" and wrap back to zero when it crosses array bound, and include in the solution validity whether all the vertices included in the position vector.

Is there any better way?

∧ | ∨ · Reply · Share ›

kartik → Venki · 3 years ago

One simple way is to generate the path array as it is being generated in the above code. Then generate rotations of the path array. Every rotation of the path array will be a Hamiltonian Cycle.

∧ | ∨ · Reply · Share ›

✉ Subscribe   Ⓓ Add Disqus to your site   ▷ Privacy   **DISQUS**

Google™ Custom Search

-
-
-
  - [Interview Experiences](#)
  - [Advanced Data Structures](#)
  - [Dynamic Programming](#)
  - [Greedy Algorithms](#)
  - [Backtracking](#)
  - [Pattern Searching](#)
  - [Divide & Conquer](#)
  - [Mathematical Algorithms](#)
  - [Recursion](#)
  - [Geometric Algorithms](#)
-

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](#)
- Follow @GeeksforGeeks

- # Recent Comments

  - ## Ashish Aggarwal

    Try Data Structures and Algorithms Made Easy -...

    Algorithms · 17 minutes ago

  - ## Vlad

    Thanks. Very interesting lectures.

    Expected Number of Trials until Success · 1 hour ago

  - ## cfh

    My implementation which prints the index of the...

    Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

  - ## Gaurav pruthi

    forgot to see that part ;)

    Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

  - ## saeid aslami

    thanks

    Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 1 hour ago

  - ## Cracker

    Implementation:...

    Implement Stack using Queues · 2 hours ago

  -

@geeksforgeeks, Some rights reserved   Contact Us!
Powered by WordPress & MooTools, customized by geeksforgeeks team