Union-Find Algorithm | Set 1 (Detect Cycle in a an Undirected Graph)

A *disjoint-set data structure* is a data structure that keeps track of a set of elements partitioned into a number of disjoint (non-overlapping) subsets. A *union-find algorithm* is an algorithm that performs two useful operations on such a data structure:

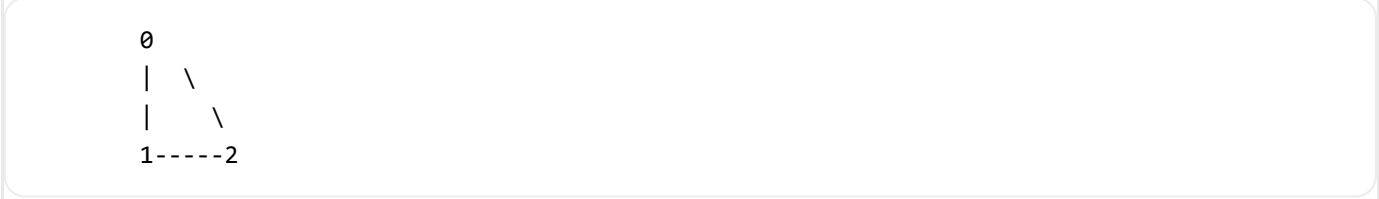
Find: Determine which subset a particular element is in. This can be used for determining if two elements are in the same subset.

Union: Join two subsets into a single subset.

In this post, we will discuss an application of Disjoint Set Data Structure. The application is to check whether a given graph contains a cycle or not.

Union-Find Algorithm can be used to check whether an undirected graph contains cycle or not. Note that we have discussed an algorithm to detect cycle. This is another method based on *Union-Find*. This method assumes that graph doesn't contain any self-loops.
We can keeps track of the subsets in a 1D array, lets call it parent[].

Let us consider the following graph:



For each edge, make subsets using both the vertices of the edge. If both the vertices are in the same subset, a cycle is found.

Initially, all slots of parent array are initialized to -1 (means there is only one item in every subset).

```
0   1   2
-1 -1  -1
```

Now process all edges one by one.

Edge 0-1: Find the subsets in which vertices 0 and 1 are. Since they are in different subsets, we take the union

of them. For taking the union, either make node 0 as parent of node 1 or vice-versa.

```
 0   1   2    <----- 1 is made parent of 0 (1 is now representative of subset {0, 1})
 1  -1  -1
```

Edge 1-2: 1 is in subset 1 and 2 is in subset 2. So, take union.

```
 0   1   2    <----- 2 is made parent of 1 (2 is now representative of subset {0, 1, 2})
 1   2  -1
```

Edge 0-2: 0 is in subset 2 and 2 is also in subset 2. Hence, including this edge forms a cycle.

How subset of 0 is same as 2?

0->1->2 // 1 is parent of 0 and 2 is parent of 1

Based on the above explanation, below are implementations:

C/C++

```
// A union-find algorithm to detect cycle in a graph
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// a structure to represent an edge in graph
struct Edge
{
    int src, dest;
};

// a structure to represent a graph
struct Graph
{
    // V-> Number of vertices, E-> Number of edges
    int V, E;

    // graph is represented as an array of edges
    struct Edge* edge;
};

// Creates a graph with V vertices and E edges
struct Graph* createGraph(int V, int E)
{
    struct Graph* graph =
           (struct Graph*) malloc( sizeof(struct Graph) );
    graph->V = V;
    graph->E = E;

    graph->edge =
        (struct Edge*) malloc( graph->E * sizeof( struct Edge ) );

    return graph;
}

// A utility function to find the subset of an element i
int find(int parent[], int i)
{
```

```
    if (parent[i] == -1)
        return i;
    return find(parent, parent[i]);
}

// A utility function to do union of two subsets
void Union(int parent[], int x, int y)
{
    int xset = find(parent, x);
    int yset = find(parent, y);
    parent[xset] = yset;
}

// The main function to check whether a given graph contains
// cycle or not
int isCycle( struct Graph* graph )
{
    // Allocate memory for creating V subsets
    int *parent = (int*) malloc( graph->V * sizeof(int) );

    // Initialize all subsets as single element sets
    memset(parent, -1, sizeof(int) * graph->V);

    // Iterate through all edges of graph, find subset of both
    // vertices of every edge, if both subsets are same, then
    // there is cycle in graph.
    for(int i = 0; i < graph->E; ++i)
    {
        int x = find(parent, graph->edge[i].src);
        int y = find(parent, graph->edge[i].dest);

        if (x == y)
```



```
{
    /* Let us create following graph
         0
        |  \
        |    \
        1-----2 */
    struct Graph* graph = createGraph(3, 3);

    // add edge 0-1
    graph->edge[0].src = 0;
    graph->edge[0].dest = 1;

    // add edge 1-2
    graph->edge[1].src = 1;
    graph->edge[1].dest = 2;

    // add edge 0-2
    graph->edge[2].src = 0;
    graph->edge[2].dest = 2;

    if (isCycle(graph))
        printf( "Graph contains cycle" );
    else
        printf( "Graph doesn't contain cycle" );
```

```
    return 0;
}
```

Run on IDE

Java

```
// Java Program for union-find algorithm to detect cycle in a graph
import java.util.*;
import java.lang.*;
import java.io.*;

class Graph
{
    int V, E;    // V-> no. of vertices & E->no.of edges
    Edge edge[]; // /collection of all edges

    class Edge
    {
        int src, dest;
    };

    // Creates a graph with V vertices and E edges
    Graph(int v,int e)
    {
        V = v;
        E = e;
        edge = new Edge[E];
        for (int i=0; i<e; ++i)
            edge[i] = new Edge();
    }

    // A utility function to find the subset of an element i
    int find(int parent[], int i)
    {
        if (parent[i] == -1)
            return i;
        return find(parent, parent[i]);
    }

    // A utility function to do union of two subsets
    void Union(int parent[], int x, int y)
    {
        int xset = find(parent, x);
        int yset = find(parent, y);
        parent[xset] = yset;
    }


    // The main function to check whether a given graph
    // contains cycle or not
    int isCycle( Graph graph)
    {
        // Allocate memory for creating V subsets
        int parent[] = new int[graph.V];

        // Initialize all subsets as single element sets
        for(int i=0; i<graph.V; ++i)
            parent[i]=-1;

        // Iterate through all edges of graph, find subset of both
```

```
        // vertices of every edge, if both subsets are same, then
        // there is cycle in graph.
        for (int i = 0; i < graph.E; ++i)
        {
            int x = graph.find(parent, graph.edge[i].src);
            int y = graph.find(parent, graph.edge[i].dest);

            if (x == y)
                return 1;

            graph.Union(parent, x, y);
        }
        return 0;
    }

    // Driver Method
    public static void main (String[] args)
    {
        /* Let us create following graph
         0
        |  \
        |    \
        1-----2 */

        Graph graph = new Graph(3,3);

        // add edge 0-1
        graph.edge[0].src = 0;
        graph.edge[0].dest = 1;

        // add edge 1-2
        graph.edge[1].src = 1;
        graph.edge[1].dest = 2;

        // add edge 0-2
        graph.edge[2].src = 0;
        graph.edge[2].dest = 2;

        if (graph.isCycle(graph)==1)
            System.out.println( "Graph contains cycle" );
        else
            System.out.println( "Graph doesn't contain cycle" );
    }
}
```

Run on IDE

Output:

```
Graph contains cycle
```

Note that the implementation of *union()* and *find()* is naive and takes O(n) time in worst case. These methods can be improved to O(Logn) using *Union by Rank or Height*. We will soon be discussing *Union by Rank* in a separate post.

This article is compiled by Aashish Barnwal and reviewed by GeeksforGeeks team. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

78 Comments Category: Graph Tags: Graph

Related Posts:

- Iterative Depth First Traversal of Graph
- Print all Jumping Numbers smaller than or equal to a given value
- Hopcroft–Karp Algorithm for Maximum Matching | Set 2 (Implementation)
- Hopcroft–Karp Algorithm for Maximum Matching | Set 1 (Introduction)
- Length of shortest chain to reach a target word
- Find same contacts in a list of contacts
- Minimum time required to rot all oranges
- Karger's algorithm for Minimum Cut | Set 2 (Analysis and Applications)

0 Average Difficulty : **0/5.0**
No votes yet.

(Login to Rate)

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

Ayush Shukla • 22 days ago

There is no need to 'find' again in union function because you have already used 'find' in isCycle function.

• Reply • Share ›

shaidan15 • 2 months ago

.

• Reply • Share ›

Neha • 2 months ago

HOW TO DO USING STL??

• Reply • Share ›

Neha ➚ Neha • 2 months ago

sry for caps

• Reply • Share ›

Billionaire • 2 months ago

Very clear explanations!

1 • Reply • Share ›

This comment was deleted.

Geek ➚ Guest • 3 months ago

Graph g(4,3);
g.addEdge(0,1);
g.addEdge(3,2);
g.addEdge(1,2);

Returns wrong output.

0-----1
|
|
3-----2

• Reply • Share ›

Tirthanu Ghosh • 3 months ago

Using the find functions in both main() and find() is redundant

```
int x = find(parent, graph->edge[i].src);
int y = find(parent, graph->edge[i].dest);

int xset = find(parent, x);
```

int yset = find(parent, y);

You should use either of these two, as they evaluate to the same results.

• Reply • Share ›

Abhimanyu Khosla ➚ Tirthanu Ghosh • 2 months ago

No you are wrong.. Try inserting the Edge values in the order 0-1,0-2 and 1-2 and then you will understand the importance of finding xset in UNINON. :)

• Reply • Share ›

Tirthanu Ghosh ➚ Abhimanyu Khosla • 2 months ago

I suggest you do the same..

The program works fine without calculating xset or yset.

• Reply • Share ›

joy asthana ➚ Tirthanu Ghosh • 2 months ago

Yes,,,,,u r right :-)

• Reply • Share ›

emiljano gjiriti • 3 months ago

In the second case, why 2 becomes the parent of 1? 1 has rank=1 while 2 has rank=-1.

• Reply • Share ›

basaisehi • 4 months ago

why are we using such union and find functions which are taking O(n)?

can't we assign set number of i directly to parent[i]?

so that we can have O(1) union and find. like below.

```
int find(int parent[], int i)
{
if (parent[i] == -1)
return i;
return parent[i];
}

void Union(int parent[], int x, int y)
{
int xset = find(parent, x);
int yset = find(parent, y);
if (xset < yset)
parent[x] = parent [y]=xset;
else parent[x] = parent [y]=yset;
```

}

• Reply • Share ›

basaisehi ➚ basaisehi • 4 months ago

leave it..
found the problem...

• Reply • Share ›

Ashish • 4 months ago

There is no use of the Find() calls in the union function. They are redundant.

```
#include<iostream>
#include<stdio.h>
#include<string.h>
using namespace std;;

struct edge
{
int src,dest;
};

int find(int a,int b[])
{
if(b[a]==-1)
return a;
else
return find(b[a],b);
}
```

see more

• Reply • Share ›

D.Dash • 4 months ago

Can any body explain what is the use of
int xset = find(parent, x);
int yset = find(parent, y);

• Reply • Share ›

Varun Mehta ➚ D.Dash • 4 months ago

While doing Union we first find parent of x and then y
then in parent array we set parent of x as y

• Reply • Share ›

Sagar • 5 months ago

can any one explain me the recursive calling in find there is some mistake there is guess.

• Reply • Share ›

praveen kumar ➚ Sagar • 5 months ago

there was nothing wrong with the recursive find since the representative of a subset has a value of -1 ,we initalise all the entries in the array to -1 to form disjoint subsets with single elements and if there are more than a single element in a disjoint subset we are searching for -1 because it is where the parent of the disjoint subset reside and we are going ahead as long as we find the parent(has a array value of -1) of that disjoint subset.else we traverse the way we do in a linked list where the link is the index of the next item in its disjoint subset.parent[i] is equal to the link which is used as the argument in the recursive call.

• Reply • Share ›

Sagar ➚ praveen kumar • 5 months ago

ohh i get it know i wasnt able to figure out how the recursion would end :P Thanks.

• Reply • Share ›

vergil • 5 months ago

no need to determine xset any yset as x and y are themselves indicative of sets in which u and v of (u,v) belong, respectively....

• Reply • Share ›

Nobody • 5 months ago

Can't we use the same approach as we did in case of directed graphs in the last example?(DFS leading to a node already visited)
Thanks

• Reply • Share ›

Anup Rai ➚ Nobody • 5 months ago

Yes that is in the next page.Just the difference from the directed graph is that u don't need a recur array u just need a visited array.
Just take care of the immediate parent.

2 • Reply • Share ›

Nobody ➚ Anup Rai • 5 months ago

Thanks

• Reply • Share ›

IOACC • 5 months ago

A perfect implementation of Union-Find:

https://avdongre.wordpress.com...

IOACC • 5 months ago

Well, cycle detection in undirected Graph is trivial using DFS.

More resource : http://www.cs.nyu.edu/courses/...

• Reply • Share ›

Andy Toh • 7 months ago

My compile-time solution:

http://ideone.com/6MawGe

• Reply • Share ›

Guest ➚ Andy Toh • 5 months ago

Thanks for sharing Andy, could you put some comments as well, please?

• Reply • Share ›

Hans • 7 months ago

English please. We British do not know how to learn foreign languages. It is to hard for us

• Reply • Share ›

Saurabh ➚ Hans • 6 months ago

U should go through the code once which is much easier to comprehend although algorithm is perfectly laid

• Reply • Share ›

Ravi ➚ Hans • 6 months ago

It is "too" hard for us
to be more precise :)

1 • Reply • Share ›

Guest • 7 months ago

For those struggling to understand the implementation and grasp concept of union-find algo. Hope this helps! :)

http://www.users.csbsju.edu/~l...

@GeeksforGeeks There's no need to call find again in union!

1 • Reply • Share ›

arvind kumar • 8 months ago

Can I say Graph is cyclic if the no. of edges are equal to or greater than no. of vertices ? I am assuming that it is undirected and connected graph.

1 • Reply • Share ›

Manoj ➚ arvind kumar • 2 months ago

you are correct but It fails if two edges are given as input with same src and destination

^ | v • Reply • Share ›

devakar verma ➚ arvind kumar • 8 months ago

Yes, I think so...

^ | v • Reply • Share ›

Kakul • 9 months ago

Why so hard explanation? I can't understand clearly. Please be a bit more explanation.

^ | v • Reply • Share ›

Pavlos Polianidis ➚ Kakul • 8 months ago

You start with the initial subsets which consist of just one vertex; i.e. one subset for each vertex. {0}, {1}, {2}

Then you need to iterate through all the edges and perform a union task on the subsets that the two vertices belong to.

for example,

for the edge 0-1 we merge the {0}, {1} subsets; so now we have {0, 1} and {2}.

for the edge 1-2 we merge the {0, 1} and {2}; so now we have one set {0, 1, 2}

finally for the edge 0-2 we cannot apply the union method since both the vertices are in the same subset; And that's where we can spot the cycle

2 ^ | v • Reply • Share ›

Shreyas Dawkhare • 10 months ago

https://www.youtube.com/watch?...

2 ^ | v • Reply • Share ›

Sanghwa Jung • 10 months ago

my java code is that http://javamusician.blogspot.k...

1 ^ | v • Reply • Share ›

piyush jain • a year ago

not good explanation here on geeksforgeeks, too many grammatical and typing mistakes

" We can keeps track of the subsets in a 1D array "

3 ^ | v • Reply • Share ›

John ➚ piyush jain • 8 months ago

Why grammar when semantics?

• Reply • Share ›

piyush jain ➚ piyush jain • a year ago

he used the structure parent array, without explaining what it is and what is the purpose.

• Reply • Share ›

helper • a year ago

though not significant, my java implementation, using the traditional adjacency list representation.

http://ideone.com/KuEwOR

• Reply • Share ›

Inaa Bella • a year ago

I ask please..We can't know the number of cycles in the graph and the vertices of each cycle???

• Reply • Share ›

its_dark ➚ Inaa Bella • a year ago

We can know the number of cycles.
When you find (x==y) in the above code, don't return. Just increment the count of no of cycles and continue.

For vertices of each cycle, you will have to store the immediate parent of each vertex.

Right now, we are updating the parent of each vertex in the array. This is to make the union find algorithm efficient. If you don't change the parent each time and just store the adjacent element, you can print the vertices in the cycle, but the complexity of union find will go up.

OR
you can also store the adjacent element in a separate array, and still maintain the order of union find.

This is basically a space vs time trade off.

3 • Reply • Share ›

Pauli ➚ its_dark • a month ago

Hi I have been
trying to modify the code to find each vertex in the cycle, but I haven't quite managed to do so. Do you have any hints on how to do this, or a code that

can
find all cycles and vertices thereof?

Reply • Share ›

Inaa Bella its_dark • a year ago

Thanks for your response, then the complexity will be exponential ?

Reply • Share ›

Inaa Bella its_dark • a year ago

Thanks for your response, then the complexity will be exponential ??

Reply • Share ›

brahma • a year ago

Simple Implementation
http://ideone.com/UJzI1h

2 • Reply • Share ›

Mragank Yadav • a year ago

For all those who are thinking that calling find() function again in Union() is redundant,

It is not and is absolutely mandatory.
Here understand it with the following example:
Consider this your graph:
0----1--2
..... | /
.....3

now our parent would be:
0 1 2 3
-1 -1 -1 -1

On passing Edge (0-1), parent becomes
0 1 2 3
1 -1 -1 -1

Edge (1-2) gives

see more

2 • Reply • Share ›

Load more comments