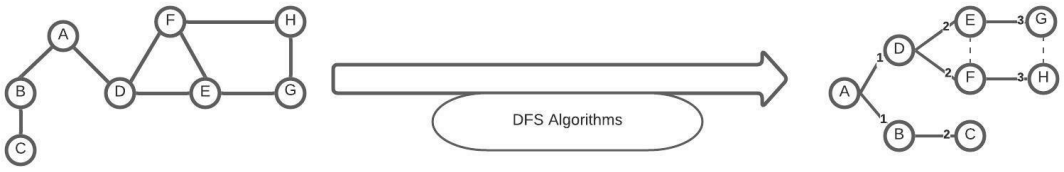
BREADTH FIRST SEARCH

Represent figures in spanning tree figures using python



>> Queue declaration

>> Visited DICT: visited DICT built as Frequency array because I will need it in testing

>> Discovered DICT: Test if the value which needed to added in queue has previous added or not. Duplicated value in queue not necessary.

>> Represent the figure as DICT

>> Build initialization: 1-All visited DICT value = False

2-All Queue status DICT Value = False

3-Put initial vertex

>> Looping {

- Test vertex if visited or not

if not

- make it visited

- test adjacent value of this vertex. Visited or not

if not

- test adjacent value of this vertex. Stated in queue or not.

if not

- add to queue

}

BREADTH FIRST SEARCH

Program structure: O0nQE0 - Online Python3 Interpreter & Debugging Tool - Ideone.com

---

```
#Any figure represented in spanning tree

#DECLERATION
#Represent figure in dictionary list
#declare Visited DICT > Frequency array using DICT
from queue import Queue
discovered={} ## Resist value duplication > if the value is already in queue > don't put again
visited={}  ## To look if this vertex visited before or not
queue=Queue()
adj_list={
      'A':['B','D'],
      'B':['A','C'],
      'C':['B'],
      'D':['A','E','F'],
      'E':['D','F','G'],
      'F':['D','E','H'],
      'G':['E','H'],
      'H':['G','F']
      }  ## Represent figure as Dictionary list

#INITIALIZATION
for node in adj_list.keys():
   visited[node]=False
   discovered[node]=False
init_val=input("Initial Value: ")
queue.put(init_val)
disocovered[init_val]=True

#LOOPING
while not queue.empty():
   u=queue.get()
   if not visited[u]:   #If the vertix not visited > Operate
      visited[u] = True
      #Push for adjacent
      for v in adj_list[u]:
         if not visited[v]:
            if not discovered[v]: #Test if it found on queue or not > if found > don't put again
               queue.put(v)
               discorvered[v]=True
```