

Sample Answers

The exercises

33. Write an algorithm to classify all edges of an undirected graph, so that depth-first tree edges can be distinguished from back edges.

Answer: Here is how we can classify the edges:

```
function CLASSIFYEDGES( $\langle V, E \rangle$ )  
  mark each node in  $V$  with 0  
  for each  $v$  in  $V$  do  
    if  $v$  is marked 0 then  
      DFSEXPLOR( $v$ )
```

```
function DFSEXPLOR( $v$ )  
  mark  $v$  with 1  
  for each edge  $(v, w)$  do  
    if  $w$  is marked with 0 then  
      classify  $(v, w)$  (and thereby  $(w, v)$ ) as “tree edge”  
      DF  
    else  
      if  $(v, w)$  is a back edge then  
        classify  $(v, w)$  (and thereby  $(w, v)$ ) as “back edge”
```

34. Show how the algorithm from the previous question can be utilised to determine if an undirected graph is cyclic.

Answer: Once we know the difference between tree and back edges, this is easy. We just change depth-first exploration slightly:

```
function ISCYCLIC( $\langle V, E \rangle$ )  
  mark each node in  $V$  with 0  
  for each  $v$  in  $V$  do  
    if  $v$  is marked 0 then  
      if DFSEXPLOR( $v$ ) = True then  
        return True  
  return False
```

```
function DFSEXPLOR( $v$ )  
  mark  $v$  with 1  
  for each edge  $(v, w)$  do  $\triangleright w$  is  $v$ 's neighbour  
    if  $w$  is marked with 0 then  
      if DFSEXPLOR( $w$ ) then  
        return True  
    else  
      if  $(v, w)$  is a back edge then  
        return True  
  return False
```

35. Explain how one could also use breadth-first search to see whether an undirected graph is cyclic. Which of the two traversals, depth-first and breadth-first, will be able to find cycles faster? (If there is no clear winner, give an example where one is better, and another example where the other is better.)

Answer: A graph has a cycle iff its breadth-first forest contains a cross edge. Sometimes depth-first search finds a cycle faster, sometimes not. Below, on the left, is a case where depth-first search finds the cycle faster, before all the nodes have been visited. On the right is an example where breadth-first search finds the cycle faster.



36. Design an algorithm to check whether an undirected graph is 2-colourable, that is, whether its nodes can be coloured with just two colours in such a way that no edge connects two nodes of the same colour. Hint: Adapt one of the graph traversal algorithms.

Answer: An undirected graph can be checked for two-colourability by performing a DFS traversal.

This begins by first as two possible “colo

Then traverse each (via DFS) each neighbour with the opposite colour. If we encounter a two-colouring is not possible.

```

function ISTWOCOLOURABLE( $G$ )
  let  $\langle V, E \rangle = G$ 
  for each  $v$  in  $V$  do
     $colour[v] \leftarrow 0$ 
  for each  $v$  in  $V$  do
    if  $colour[v] = 0$  then
      DFS( $v, 1$ )
  return True

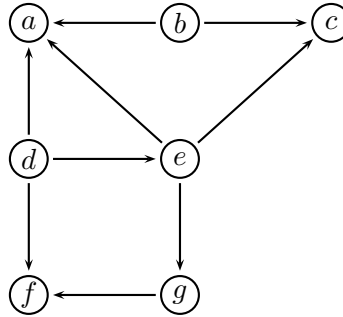
function DFS( $v, currentColour$ )
   $colour[v] \leftarrow currentColour$ 
  for each node  $u$  in  $V$  adjacent to  $v$  do
    if  $u$  is marked with  $currentColour$  then
      return False
    if  $u$  is marked with 0 then
      DFS( $u, 3 - currentColour$ )
  
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

37. Apply the DFS-based topsort algorithm to linearize the following graph:



Answer: Here is how the traversal stack develops:

$f_{7,4}$
 $g_{6,5}$
 $c_{3,2} \quad e_{5,6}$
 $a_{1,1} \quad b_{2,3} \quad d_{4,7}$

The reverse of the popping order is: d, e, g, f, b, c, a .

38. Apply insertion sort to the list S, O, R, T, X, A, M, P, L, E.

Answer: Have fun :)

39. For what kind of array

Answer: Insertion sort is an $O(n^2)$ algorithm. It is almost-sorted". The definition of an almost-sorted array is that it can be sorted in $O(n)$ where n is the size of the array. In next week's exercises we define inversions properly and use an algorithm to count their number.

40. Trace how interpolation search proceeds when searching for 42 in an array containing (in index positions 0..21)

20, 20, 21, 23, 25, 26, 26, 27, 29, 29, 29, 30, 32, 33, 34, 36, 38, 40, 41, 43, 43, 45

Answer: The first probe will be at position $x = 0 + \lfloor \frac{42-20}{45-20}(21-0) \rfloor = \lfloor \frac{22}{25} \cdot 21 \rfloor = 18$. At index 18 we find 41, so the entire segment from 0 to 18 is discarded. Interpolation sort is now ready to repeat the process for 43, 43, 45. However, it is essential to first check the extreme elements. Since $43 > 42$, the process must stop straight away, reporting failure.

41. (Optional.) For evenly distributed keys, interpolation search is $O(\log \log n)$. Show that $\log \log n$ has a smaller order of growth than $\log n$. Hint: Differential calculus tells us that $(\log x)' = \Theta(\frac{1}{x})$ and the chain rule says that $(f \circ g)'(x) = f'(g(x))g'(x)$.

Answer: We want to show that $\lim_{n \rightarrow \infty} \frac{\log \log n}{\log n} = 0$. By l'Hôpital's rule, it suffices to show that $\lim_{n \rightarrow \infty} \frac{1/n}{1/\log n} = 0$. But $\frac{1/n}{1/\log n} = \frac{\log n}{n}$, which approaches 0 (albeit slowly) as n approaches ∞ .