Analysis of Algorithms Practice Session – II

(2010-2011, Spring Term)

G. Selda UYANIK

(seldauyanik@itu.edu.tr)

Question:

« A *binary tree* is a rooted tree in which each node has at most two children.»

- Show by *induction* that

H: in any binary tree the number of nodes with two children is exactly one less than the number of leaves.

Answer:

Notations:

- The # of *leaf* nodes in a tree T : n_0(T)
- The # of nodes in a tree T with 2 children: n_2(T)

Proof:

1. Basis: tree with single node: H is true : $n_0(T_1)$ = 1,
   $n_2(T_1)$ = 0 , $n_2(T_1) = n_0(T_1) - 1$ ✓

2. Assume true for an ordinary tree T_2:
   v: is a leaf node, not root since T_2 has more than one node, it has paret u.

$n_0(T_2)$ = m, $n_2(T_2) = n_0(T_2) - 1$ = m-1

Answer:

Proof [cont.]:

3. Show H is also true for tree obtained from T_2:

Delete v: new tree obtained: T_3, Show H is also true on T_3:

$n_0(T_3) = ?$, $n_2(T_3) = ?$

- Parent u

→ if no other child, it becomes, leaf (case 1)

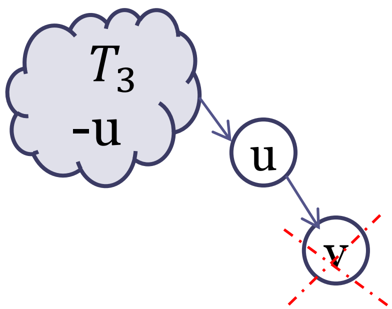
→ if has other child, it is no more one of two children nodes. (case 2)

Answer:

Proof [cont.]:

3. Show H is also true for tree obtained from T_2:

• Case 1:

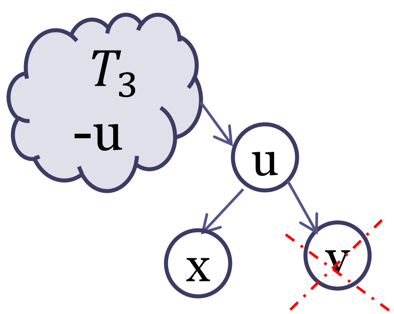


$n_0(T_3) = n_0(T_2) =$ m

$n_2(T_3) = n_2(T_2) =$ m - 1

H holds !

• Case 2:



$n_0(T_3) = n_0(T_2)$ - 1 = m -1

$n_2(T_3) = n_2(T_2)$ - 1= m – 1 -1

H holds !

Background:

- You're helping a group of ethnographers analyze some oral history data they've collected by interviewing members of a village to learn about the fives of people who've lived there over the past two hundred years.
- From these interviews, they've learned about a set of n people: P_1....... P_n (all have died)
- They also collected facts about when these people lived relative to one another.
- Each fact has one of the following two form:
  - For some i and j, person P_i died before person P_j was born; or
  - For some i and j, the life spans of P_i and P_j overlapped at least partially.

Question:

- Naturally, they're not sure that all these facts are correct; memories
- determine whether the data they've collected is at least *internally consistent*, in the sense that *there could have existed a set of people for which all the facts they've learned simultaneously hold.*

- Give an efficient algorithm to do this: either it should produce proposed dates of birth and death for each of the n people so that all the facts hold true, or it should report (correctly) that no such dates can exist

—that is, the facts collected by the ethnographers are not internally consistent.

Formulation:

- Directed graph G:

  V: represent each person with 2 nodes: b_i & d_i

  E: edges directed from earlier evet to the preceding event.

- If G is constructed, problem transforms into deciding whether G is acyclic or not !

➢ How to consturct G ?

➢ Why cycle in G represents inconsistency of the facts ethnographers collected ?

Solution:

➢ How to consturct G:

- For every fact been told :
  - If it is said person P_i died before person P_j was born: add $\boldsymbol{d_i \rightarrow b_j}$ edge to G.
  - If it is said the life spans of P_i and P_j overlapped at least partially: add $\boldsymbol{b_i \rightarrow d_j}$ and $\boldsymbol{b_j \rightarrow d_i}$ edges to G !

Solution:

- Why cycle in G represents inconsistency of the facts ethnographers collected ?

- Any cycle: Remember: edges should represent preceding of times of events.

  Cycle prevents putting a particular event as first ! :

  Which one (e.g.n^{th}) you choose => you'll get inconsistency. ($(n-1)^{th}$) has edge toward it: should have taken place earlier)

- No cycle: its topological order represents the birth and death times of people consistently with the given facts !

Question:

METU-UPEM Day1:
http://www.ceng.metu.edu.tr/contest/upem

- M lanes in METU, connects 2 of the N crossroads
- There is no pair of crossroads connected by more than one lane
- it is possible to pass from each crossroad to each other crossroad by a path composed of one or more lanes.
- A cycle of lanes is simple when passes through each of its crossroads exactly once.

➢ UPEM organization wants to put pictures of the winners of UPEM 2010 contest, on the lanes of the longest simple loop on METU streets.

➢ Fortunately, each lane is participating in no more than one simple cycle !

Question:

- Input: 1^{st}line: positive integers N & M,

  Each of latter M lines: consist 2 integers representing which crossroads each lane combines.
- Output: An integer that is the length of the longest simple loop on ODTU streets.

Sample input

7	8
3	4
1	4
1	3
7	1
2	7
7	5
5	6
6	2

→

Sample output

4

Formulation:

- How to represent crossroads and streets?

Formulation:

- How to represent crossroads and streets?
- Nodes: Crossroads
- Edges: Lanes
- Why ? (each lane combines exactly 2 crossroads but crossroads has no such restirction…)
- Problem ?

Formulation:

- How to represent crossroads and streets?
- Nodes: Crossroads
- Edges: Lanes
- Why ? (each lane combines exactly 2 crossroads but crossroads has no such restirction…)
- Problem?

Decomposes into finding longest cycle!

Solution: Simply traverse the graph, note the cycles and their lengths, choose longest! DFS or BFS.

Solution:

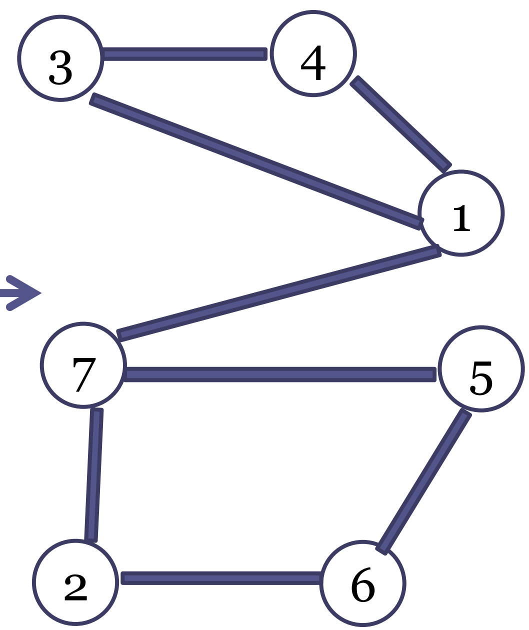
- Sample run:

Sample input

N	M
7	8
3	4
1	4
1	3
7	1
2	7
7	5
5	6
6	2

4

Sample output



Solution:

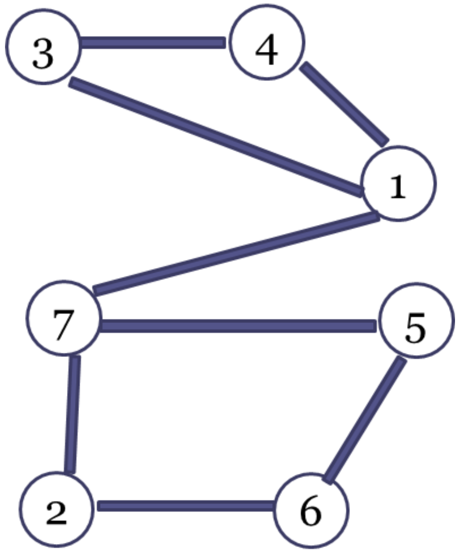
- With BFS:

BFS algorithm.

- $L_0 = \{ s \}$.
- L_1 = all neighbors of L_0.
- L_2 = all nodes that do not belong to L_0 or L_1, and that have an edge to a node in L_1.
- L_{i+1} = all nodes that do not belong to an earlier layer, and that have an edge to a node in L_i.

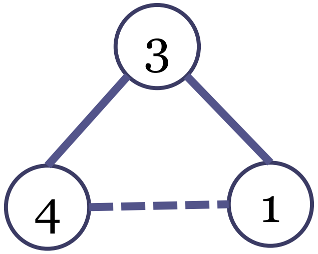
Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j or in same layer (i=j) but have an edge in G => cycle!

update length variable : how?
Find common anchestor of i & j e.g. At level k
Cycle length: i-k + j-k + 1 (... edge)
Keep maximum length value.



Solution:

Max. Length: BFS tree



L_0: {3} 0

L_1: {4,1} 3:

(Common anchestor of 4&1:3 at level 0,

1-0 + 1-0 + 1 = 3)

Solution:

Max. Length: BFS tree

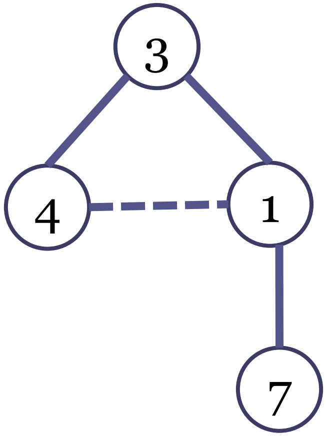
L_0: {3} 0

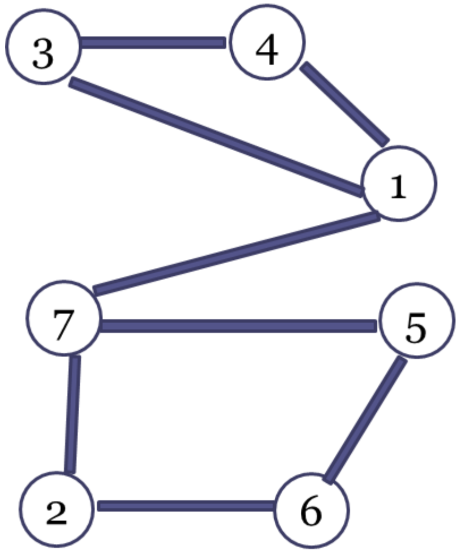
L_1: {4,1} 3:

(Common anchestor of 4&1:3 at level 0,

1-0 + 1-0 + 1 = 3)

L_2: {7} 3





Solution:

Max. Length: BFS tree

L_0: {3} 0

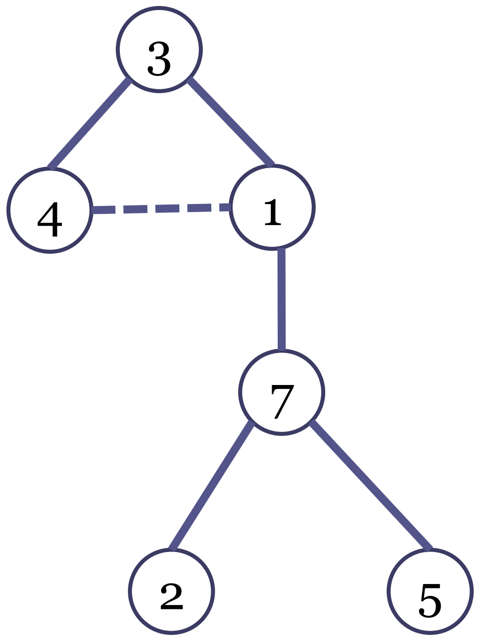
L_1: {4,1} 3:

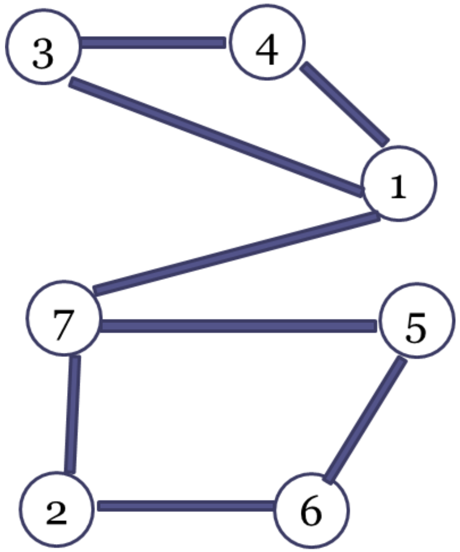
(Common anchestor of 4&1:3 at level 0,

1-0 + 1-0 + 1 = 3)

L_2: {7} 3

L_3: {2,5} 3

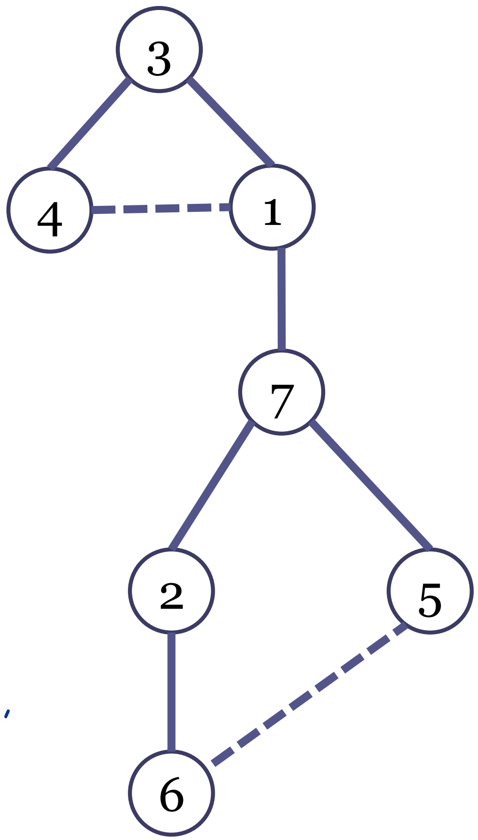


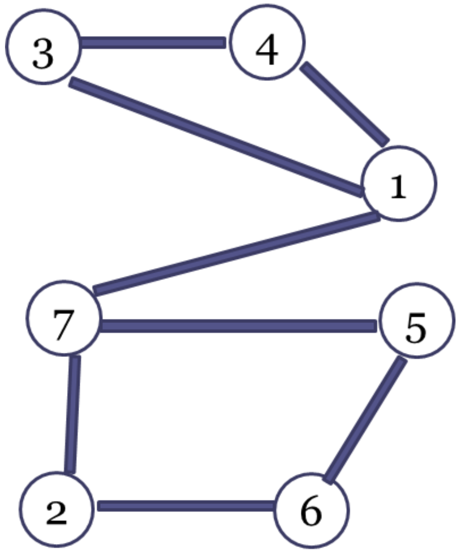


Solution:

Max. Length:

BFS tree





L_0: {3} 0

L_1: {4,1} 3:

(Common anchestor of 4&1:3 at level 0,

1-0 + 1-0 + 1 = 3)

L_2: {7} 3

L_3: {2,5} 3

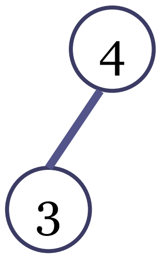
L_4: {6} 4:

(Common anchestor of 6&5 :7 at level 2,

4-2 + 3-2 + 1 = 4)

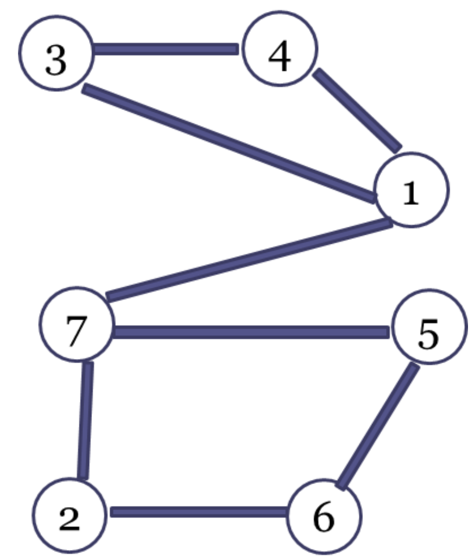
Solution:

- With DFS:



- $L_0: \{4\}$
- $L_1: \{3\}$

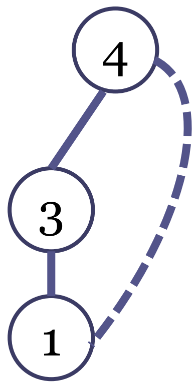
Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle!
update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.



Solution:

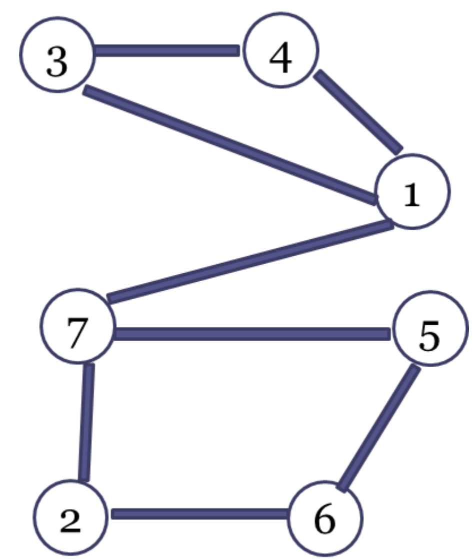
- With DFS:

- L_0: {4}
- L_1: {3}
- L_2: {1}



Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle!
update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.

cycle length: 2 – 0 + 1 = 3

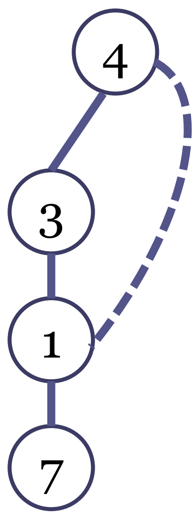


Solution:

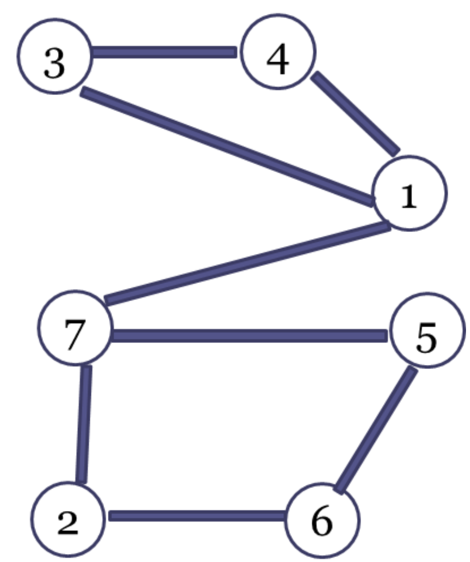
- With DFS:

Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle!
update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.

- L_0: {4}
- L_1: {3}
- L_2: {1}
- L_3 : {7}



cycle length: 2 – 0 + 1 = 3

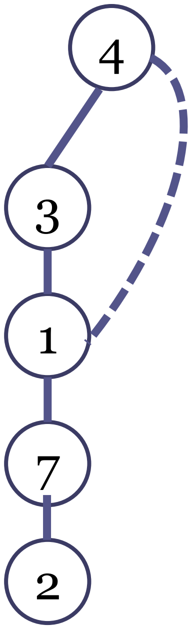


Solution:

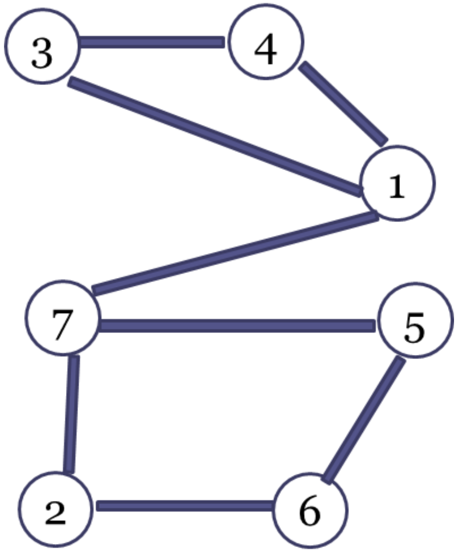
Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle! update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.

- With DFS:

- L_0: {4}
- L_1: {3}
- L_2: {1}
- L_3 : {7}
- L_4 : {2}



cycle length: 2 – 0 + 1 = 3

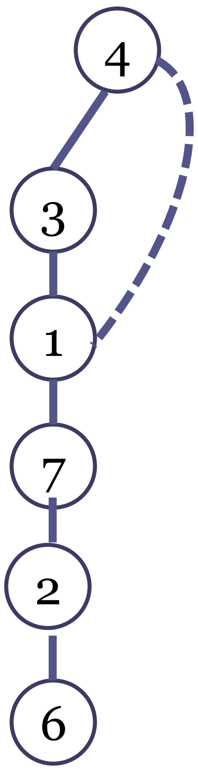


Solution:

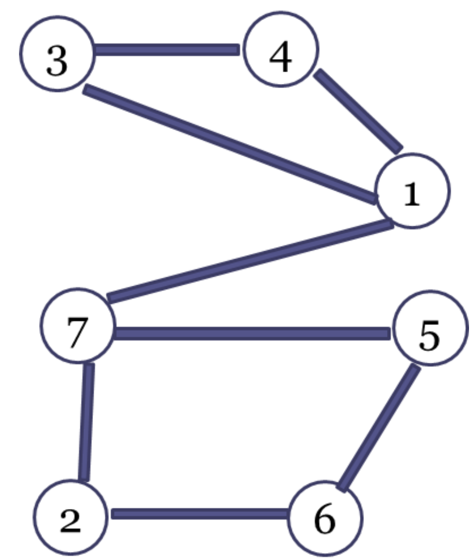
- With DFS:

Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle!
update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.

- L_0: {4}
- L_1: {3}
- L_2: {1}
- L_3 : {7}
- L_4 : {2}
- L_5 : {6}



cycle length: 2 – 0 + 1 = 3

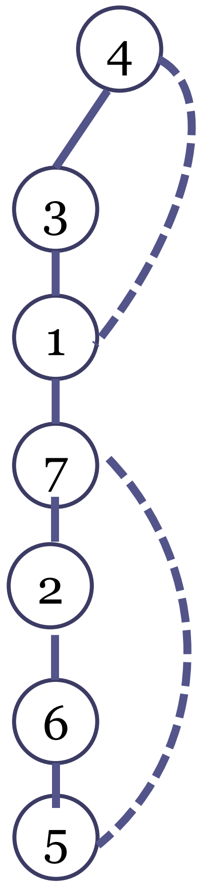


Solution:

- With DFS:

Addition: If node at layer i, has a dotted edge (edge in G not in T) to node at layer j => cycle! update length variable : how?
Cycle length: i - j + 1 ,
Keep maximum length value.

- L_0: {4}
- L_1: {3}
- L_2: {1}
- L_3 : {7}
- L_4 : {2}
- L_5 : {6}
- L_6 : {5}



cycle length: $2 - 0 + 1 = 3$

cycle length: $6 - 3 + 1 = 4$