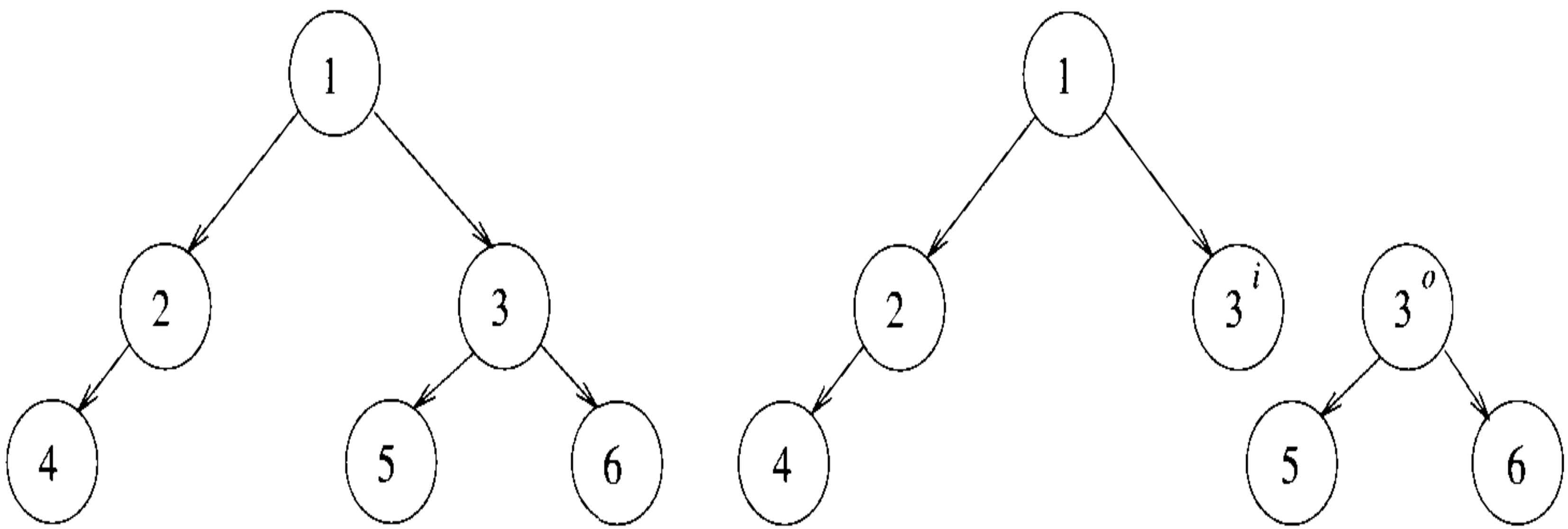
Tree vertex splitting

Dr. Bibhudatta Sahoo,
National Institute of Technology Rourkela

Tree vertex splitting

- Given a network and a loss tolerance level, the tree vertex splitting problem is to determine the optimal placement of boosters.



- In the **vertex splitting** problem, the objective is to determine a minimum number of vertices to **split** so that the resulting dag has no path of length > δ.
- A linear time algorithm is obtained for the case when the dag is a **tree**.

Directed and weighted binary tree

- Consider a network of power line transmission
- The transmission of power from one node to the other results in some loss, such as drop in voltage
- Each edge is labeled with the loss that occurs (edge weight)
- Network may not be able to tolerate losses beyond a certain level
- You can place boosters in the nodes to account for the losses

The Greedy Method : or Greedy heuristic

- Algorithms for optimization problems typically go through a sequence of steps, with a set of choices for each step.
- For many optimization problems, using dynamic programming to determine the best choices is overkill; simpler, more efficient algorithms are sufficient.
- A *greedy algorithm* always makes a choice that is locally optimal in the hope that it will lead to a globally optimal solution.

Defining Tree vertex splitting

- Let T = (V; E ; w) be a weighted directed tree
- V is the set of vertices
- E is the set of edges
- w is the weight function for the edges
- w_{ij} is the weight of the edge **$< i, j> \in E$**
- We say that $\mathbf{w_{ij}} = \infty$ if **$< i, j> \notin E$**
- A vertex with in-degree zero is called a **source vertex**
- A vertex with out-degree zero is called a **sink vertex**
- For any path $P \in T$, its delay d(P) is defined to be the sum of the weights (w_{ij}) of that path, or
- d(P) = $\displaystyle\sum_{<i,j> \in P} w_{ij}$
-
- Delay of the tree T, d(T) is the **maximum of all path delays**

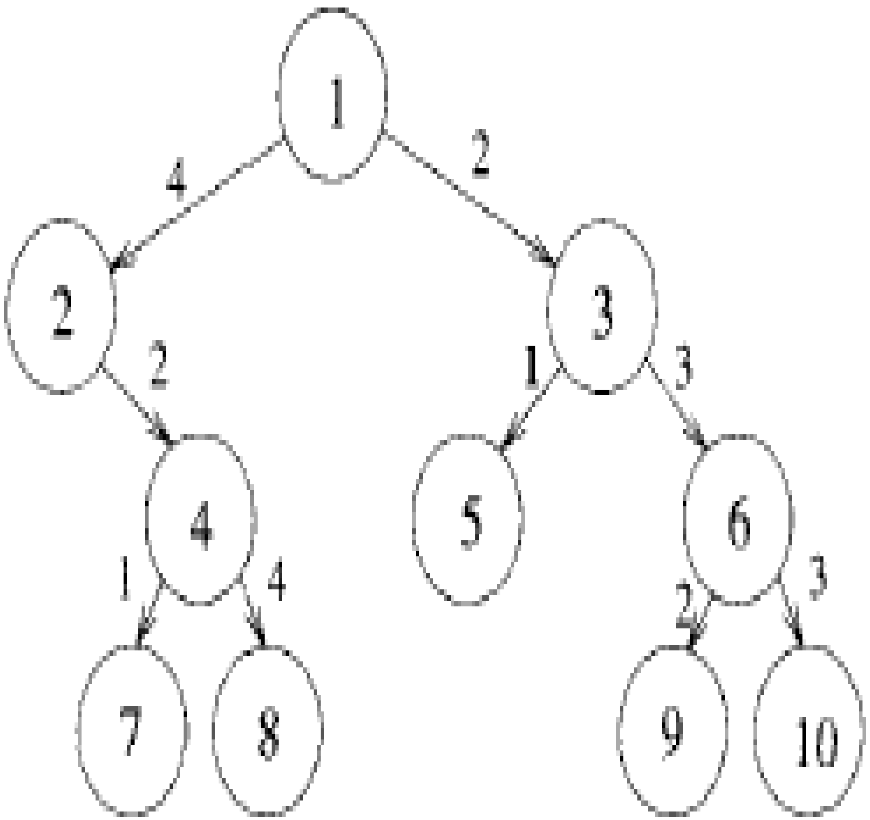
Defining Tree vertex splitting

- Given a weighted tree T(V, E, w) and a tolerance limit δ, any subset **X** of V is a feasible solution if d(T/X) < δ.
- Given an X, we can computed(T/X) in O(|V|)time.
- A trivial way of solving the TVSP is to computed (T/X) for each possible subset X of V.
- But there are $2^{|v|}$ such subsets **[exponential solution space]**
- A better algorithm can be obtained using the greedy method

A greedy approach to solving TVSP

- For the TVSP, the quantity that is optimized(minimized) is the number of nodes in X.
- A greedy approach to solving this problem is to compute for each node u∈ V, ,the maximum delay d(u) from u to any other node in its subtree.
- If u has a parent v such that $d(u) + w(v,u) > \delta$, then the node **u** gets split and **d(u)** is set to zero.
- Computation proceeds from the leaves toward the root.

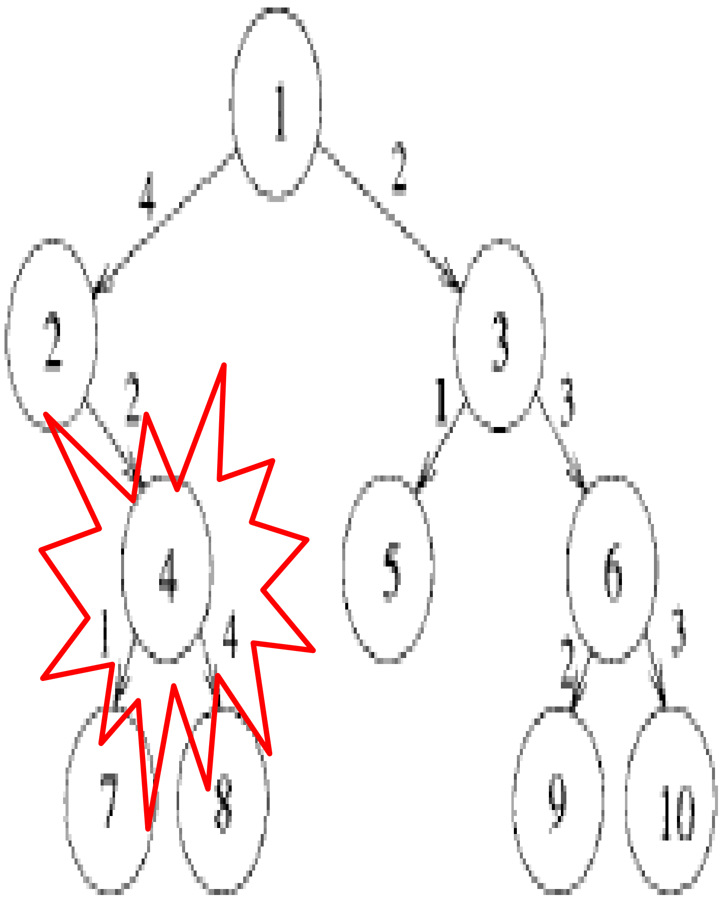
The TVSP: input; output :1



- let δ = 5.
- For each of the leaf nodes7, 8,5, 9, and 10 the delay is **zero**.
- Let u be any node and C(u) be the set of all children of u.
- Then d(u) is given by

$$d(u) = \max_{v \in C(u)} \{d(v) + w(u, v)\}$$

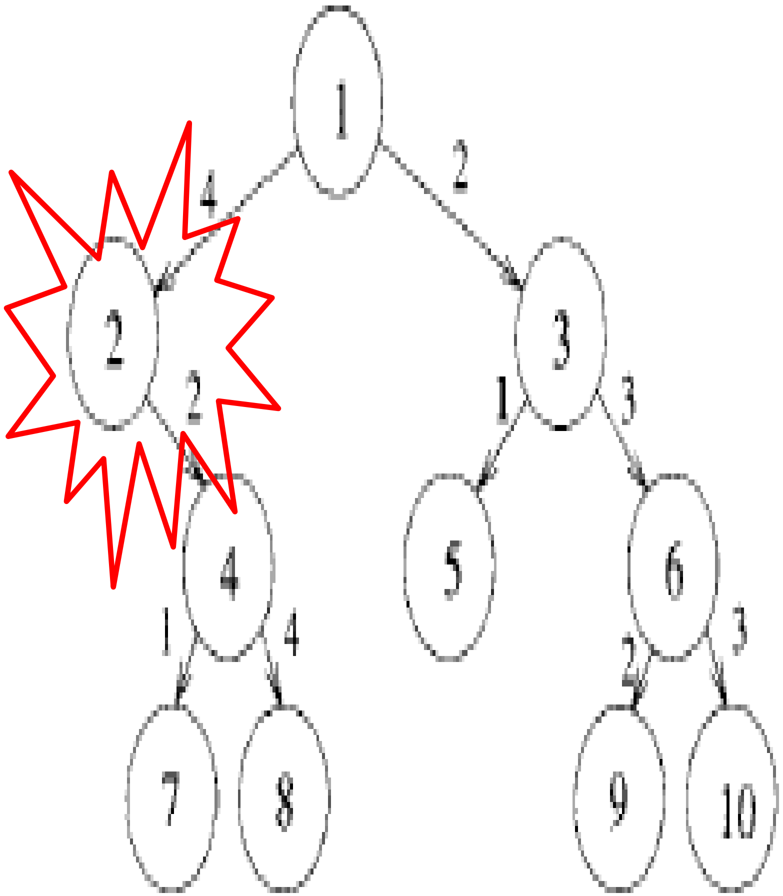
The TVSP: input: output : 2



$$d(u) = \max_{v \in C(u)} \{d(v) + w(u,v)\}$$

- d(7)=d(8)=d(5)=d(9)=d(10)=0
- let δ = 5.
- d(4)= 4
- 4 has a parent : **2**
- So d(u) + w(v,u)
- is 4 + 2 = 6 > δ**=5**
- $\Rightarrow$ **Split the node 4**
- $\Rightarrow$ d(4) = 0
  - Node 4 gets split.
  - We set d(4) = 0 **and continue with node 2**

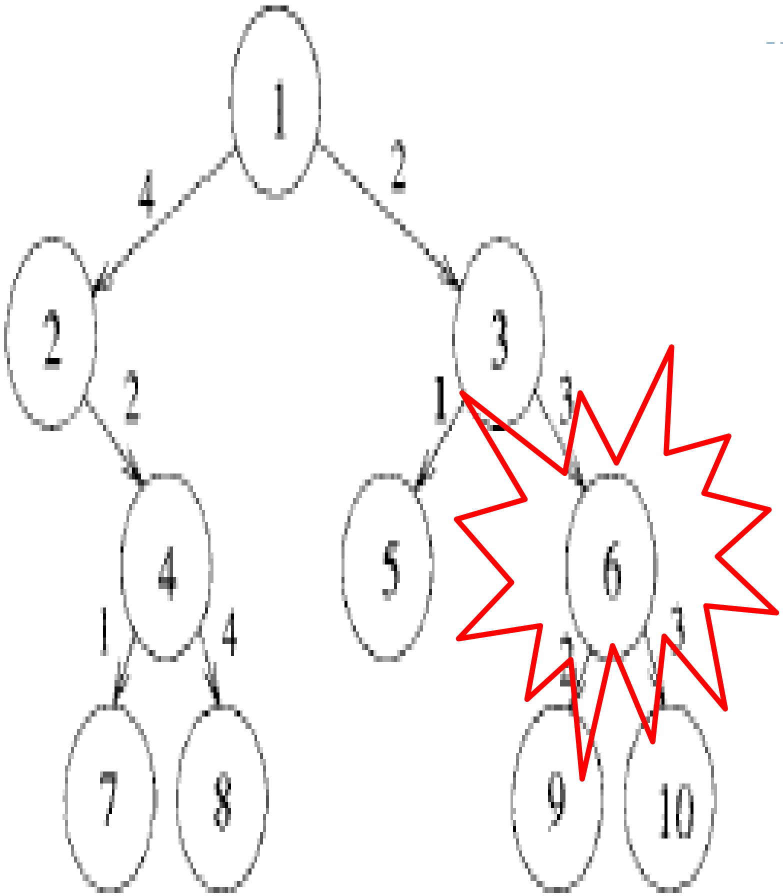
The TVSP: input; output: 3



$$d(u) = \max_{v \in C(u)} \{d(v) + w(u,v)\}$$

- Whether node 2 will split?
- d(4) = 0, d(2)= 2
- 1 is parent of node 2
- So d(u) + w(v, u)
- is 2+ 4 = 6 > δ**=5**
- ⇒ **Split the node 2**
- ⇒ d(2) = 0
  - Node 2 gets split.
  - We set d(2) = 0 **and continue with node 1**

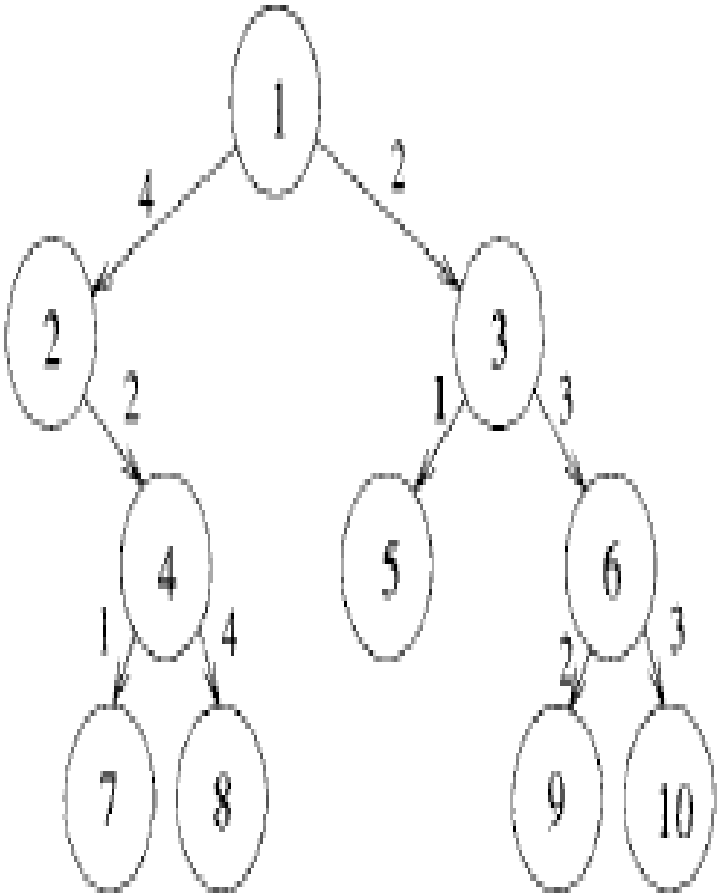
The TVSP: input; output: 4



$$d(u) = \max_{v \in C(u)} \{d(v) + w(u, v)\}$$

- Whether node 6 will split?
- d(9) = d(10)= 0, d(6)= 3
- 3 is parent of node 6
- So d(u) + w(v, u)
- is 3 + 3 = 6 > δ**=5**
- ⇒ **Split the node 6**
- ⇒ d(6) = 0
  - Node 6 gets split.
  - We set d(6) = 0 **and continue with node 3**

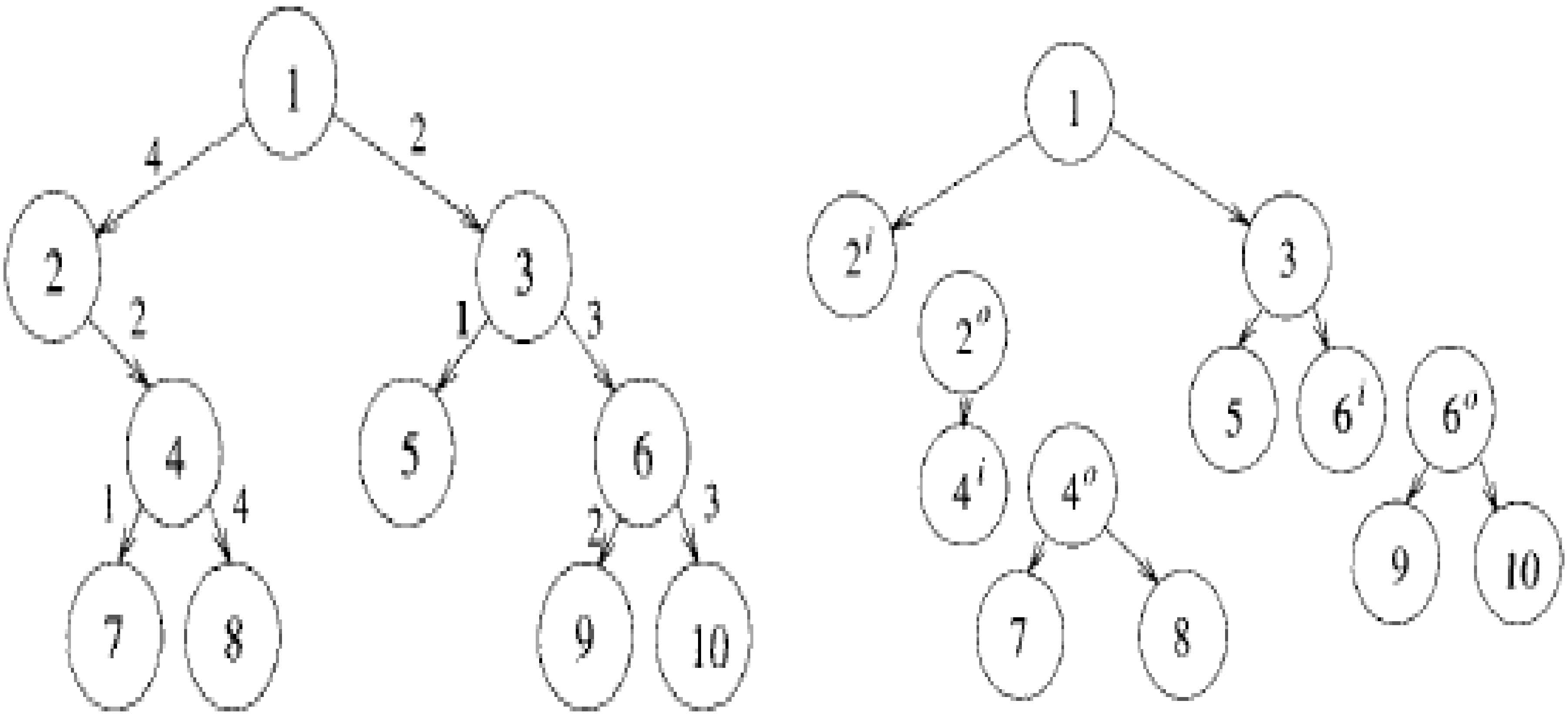
The TVSP: input; output: 5



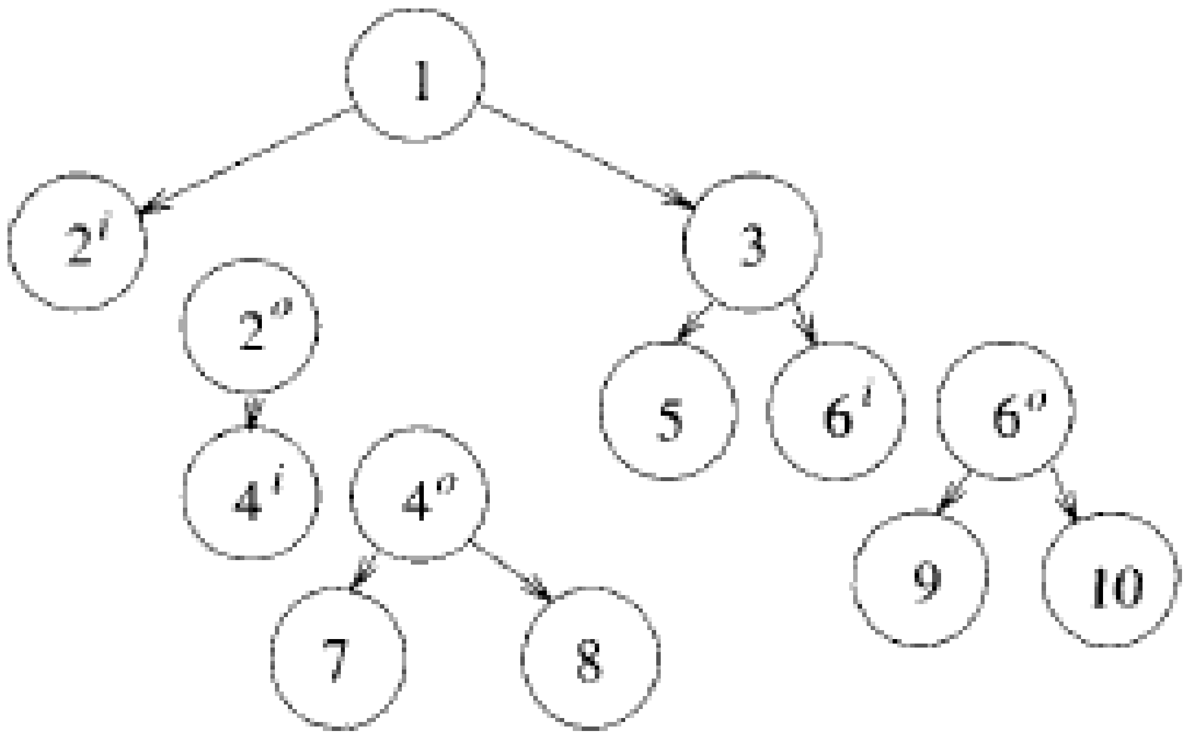
$$d(u) = \max_{v \in C(u)} \{d(v) + w(u, v)\}$$

- Whether node 3 will split?
- d(6) = 0, d(3)= 3
- 1 is parent of node 3
- So d(u) + w(v, u)
- is 3 + 2 = 5 > δ=5
- ⇒ **No Split of the node 3**

The TVSP: input; output : 6



The final tree that results after splitting the nodes 2,4, and 6.



The tree vertex splitting algorithm

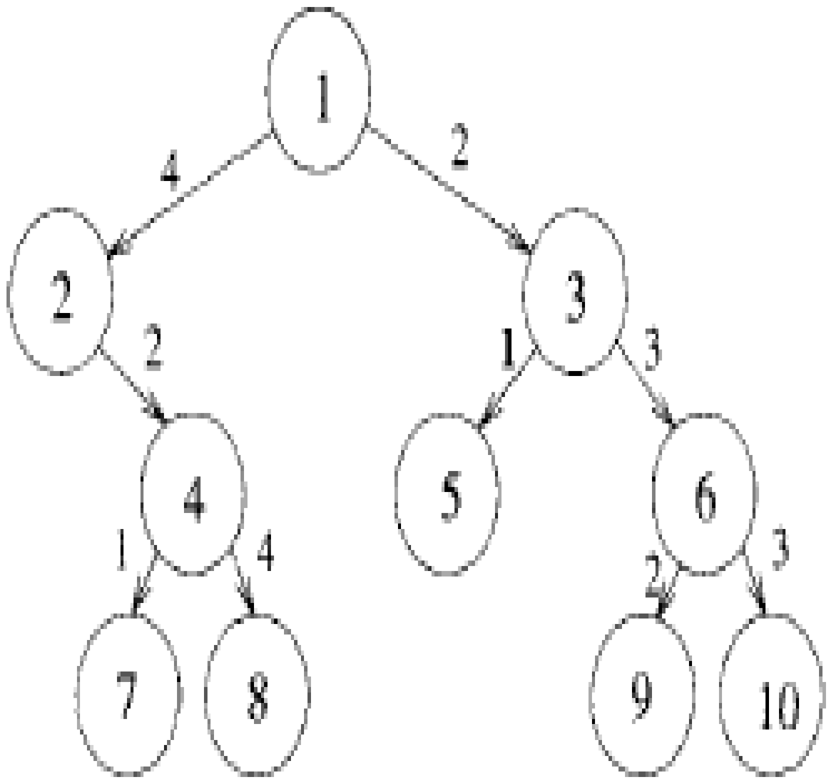
```
Algorithm TVS(T, δ)
// Determine and output the nodes to be split.
// w() is the weighting function for the edges.
{
    if (T ≠ 0) then
    {
        d[T] := 0;
        for each child v of T do
        {
            TVS(v, δ);
            d[T] := max{d[T], d[v] + w(T, v)};
        }
        if ((T is not the root) and
                (d[T] + w(parent(T), T) > δ)) then
        {
            write (T); d[T] := 0;
        }
    }
}
```

The tree vertex splitting algorithm

- Algorithm TVS(T,δ) is a recursive algorithm.
- TVS is called only once on each node T in the tree.
- When TVS is called on any node T, only a constant number of operations are performed (excluding the time taken for the recursive calls)
- Algorithm TVS takes **Θ(n)** time, where n is the number of nodes in the tree.

Different representation for directed binary tree

Different representation for directed binary tree



If tree[i] has a tree node, the weight of the incoming edge from its parent is stored in weight[i]

tree	weight
1	0
2	4
3	2
0	0
4	2
5	1
6	3
0	0
0	0
7	1
8	4
0	0
0	0
9	2
10	3

Different representation for directed binary tree

- Binary directed tree is represented as linear data structure: array .
- The tree is stored in the array **tree** with the root at **tree[1]**.
- Edge weights are stored in the array **weight[]**. If **tree[i]** has a tree node, the weight of the incoming edge from its parent is stored in **weight[i]**.
- The delay of node i is stored in **d[i]**.
- The array **d[ ]** is initialized to zero at the beginning.
- Entries in the arrays **tree[]** and **weight[]** corresponding to non-existence nodes will be zero

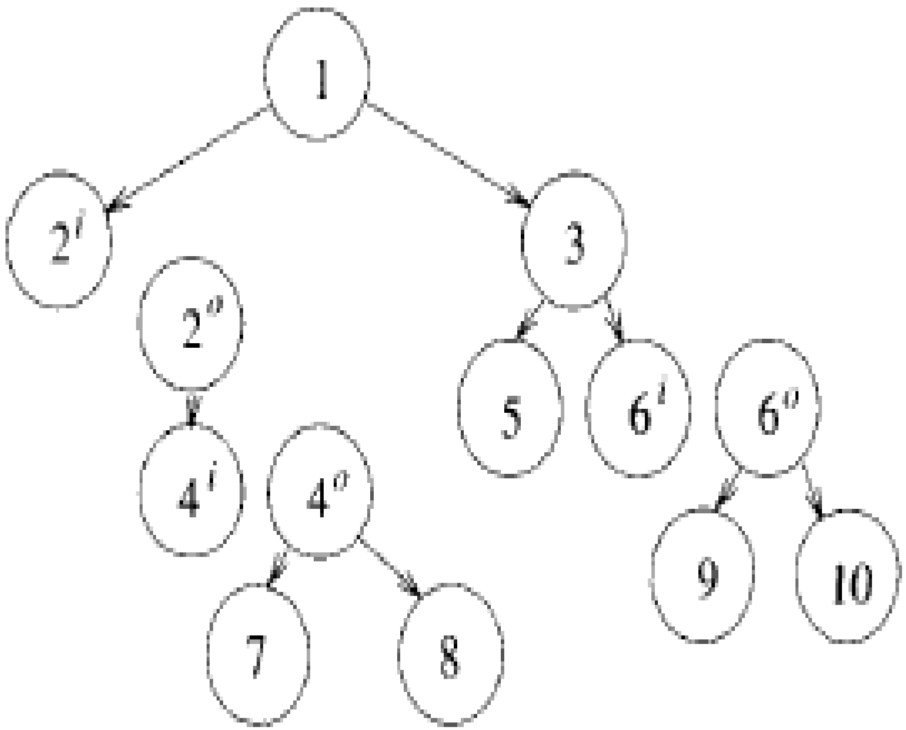
Algorithm 4.4 TVS for the special case of binary trees

```
Algorithm TVS(i, δ)
// Determine and output a minimum cardinality split set.
// The tree is realized using the sequential representation.
// Root is at tree[1]. N is the largest number such that
// tree[N] has a tree node.
{
    if (tree[i] ≠ 0) then // If the tree is not empty
        if (2i > N) then d[i] := 0; // i is a leaf.
        else
        {
            TVS(2i, δ);
            d[i] := max(d[i], d[2i] + weight[2i]);
            if (2i + 1 ≤ N) then
            {
                TVS(2i + 1, δ);
                d[i] := max(d[i], d[2i + 1] + weight[2i + 1]);
            }
        }
        if ((tree[i] ≠ 1) and (d[i] + weight[i] > δ)) then
        {
            write (tree[i]); d[i] := 0;
        }
}
```

TVS for the special case of binary trees

- The algorithm is invoked as TVS(1, δ).

Different representation for directed binary tree



tree	weight	d
1	0	
2	4	
3	2	
0	0	
4	2	
5	1	
6	3	
0	0	
0	0	
7	1	0
8	4	0
0	0	
0	0	
9	2	0
10	3	0

TVS for the special case of binary trees

- The algorithm is invoked as TVS(1, δ).
- Now, Show that Algorithm TVS (i,δ) will always split a minimal number of nodes.

- **Theorem 4.2**: Algorithm TVS outputs a minimum cardinality set U such that $d(T/U) \leq \delta$ on any tree T, provided no edge of T has weight $> \delta$.

Theorem 4.2

Proof by induction:

Base case. If the tree has only one node, the theorem is true.

Induction hypothesis. Assume that the theorem is true for all trees of size $\leq n$.

Induction step. Consider a tree T of size $n+1$

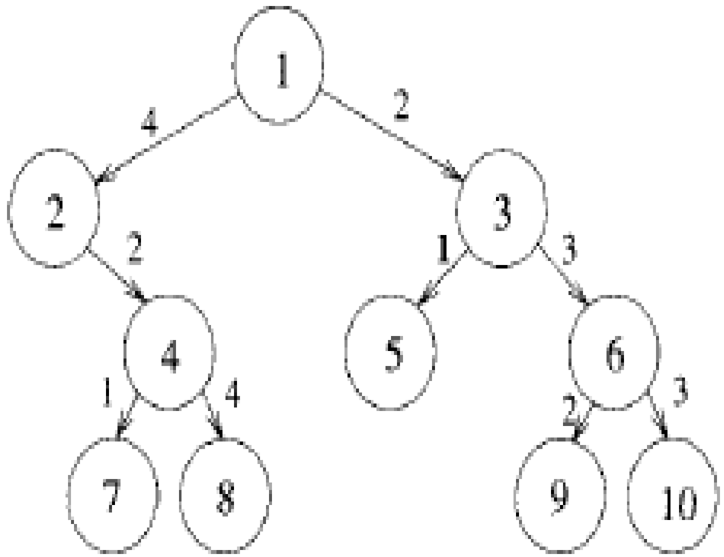
- Let U be the set of nodes split by `tvs`
- Let W be a minimum cardinality set such that $d(T/W) \leq \delta$
- We need to show that $|U| \leq |W|$
- If $|U| = 0$, the above is indeed true

Theorem 4.2

- Otherwise
  - Let x be the first vertex split by `tvs`
  - Let T_x be the subtree rooted at x
  - Let $T' = T - T_x + x$ // Delete T_x from T except for x
  - W has to have at least one node, y, from T_x
  - Let $W' = W - \{y\}$
  - If $\exists W*$ such that $|W*| < |W'|$ and $d\left(\frac{T'}{W*}\right) \leq \delta$, then since $d\left(\frac{T}{W*+\{x\}}\right) \leq \delta$, W is not minimum cardinality split set for T
  - Thus, W' has to be a minimum cardinality split set such that $d\left(\frac{T'}{W'}\right) \leq \delta$
- If `tvs` is run on tree T', the set of split nodes output is $U - \{x\}$
- Since T' has $\leq n$ nodes, $U - \{x\}$ is a minimum cardinality set split for T'
- This means that $|W'| \geq |U| - 1$, or $|W| \geq |U|$

Thanks for Your Attention!

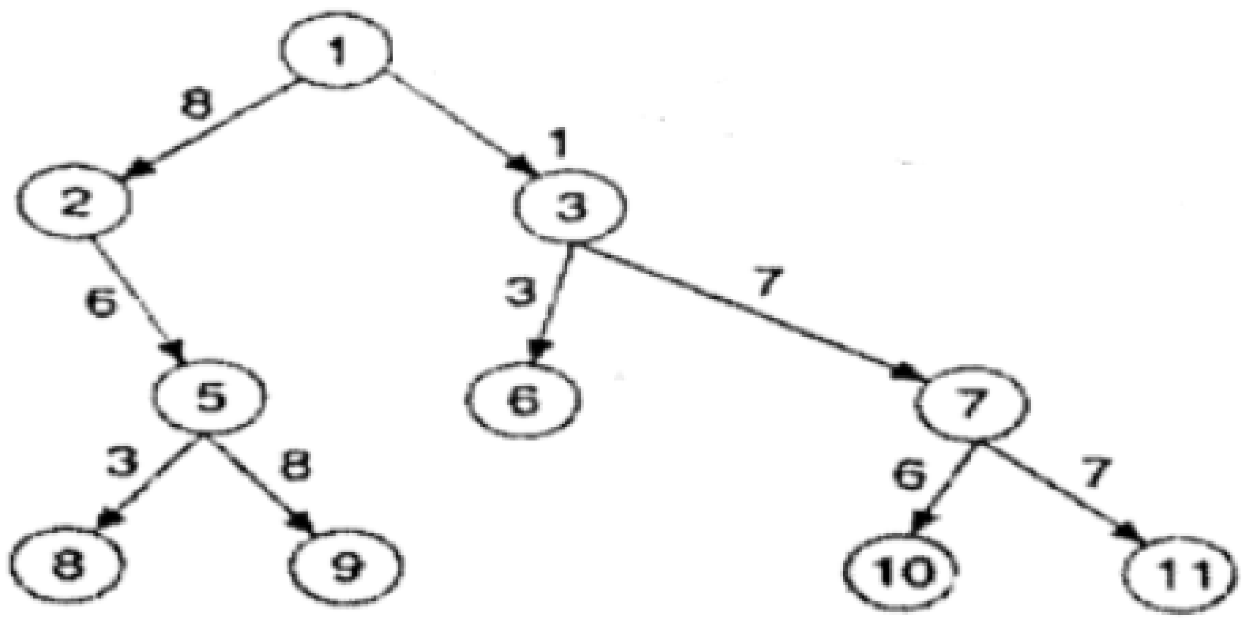
Exercises



1. For the tree shown above , solve the TVSP when (a) $\delta = 4$ and (b) $\delta = 6$.
2. Rewrite TVS Algorithm for general trees. Make use of pointers.

Exercises

- What is the tree vertex splitting problem? Solve the following tree vertex splitting problem for $\delta = 10$.



Reference

- **International Journal of Foundations of Computer ScienceVol. 09, No. 04, pp. 377-398 (1998)**No Access
- **VERTEX SPLITTING IN DAGS AND APPLICATIONS TO PARTIAL SCAN DESIGNS AND LOSSY CIRCUITS**
- DOOWON PAIK, , SUDHAKAR REDDY, and SARTAJ SAHNI
- https://doi.org/10.1142/S0129054198000301