Unit -5

Dynamic Graphs

Page No.
Date

Imp

Que- Explain Link cut tree with operations and Example, Advantage, Disadvantge, application.

⇒

Link cut tree :-

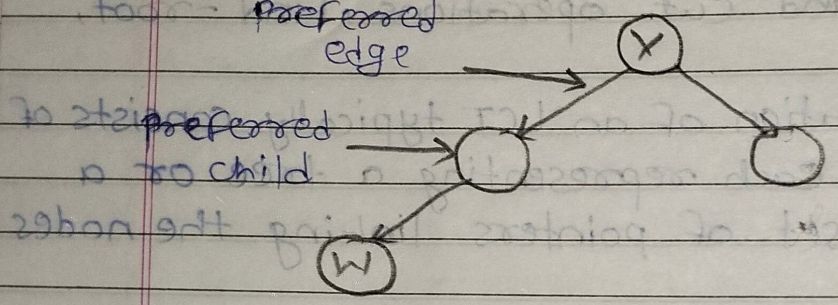
- Link cut Trees (LCT) is a data structure that allow for efficient dynamic maintenance of trees.
- It is a type of self-adjusting data structure that allows for efficient manipulation of trees, such as link and cut operations, find-root, and access.
- The implementation of an LCT typically consists of a set of nodes, each representing a tree or a subtree, and a set of pointers linking the nodes together.
- Each node contains two pointers, one pointing to its parent and one pointing to its child and a value associated with the node.
- The link-cut tree was invented in 1982 by Daniel Dominic sleator and Robert Endre Tarjan.
- The Link-cut trees store a collection of vertex disjoint rooted trees subject to the following operations :-
  - makeTree() :- It makes a new vertex and puts a singleton tree.
  - getRoot(V) :- It returns the root of the tree containing V.
  - cut(V) :- It destroys the edge {V, Parent(V)}.
  - link(V,W) :- Assumes V is the root of its tree and V and W are in different trees, then makes V a child of W.

• Terminologies :-

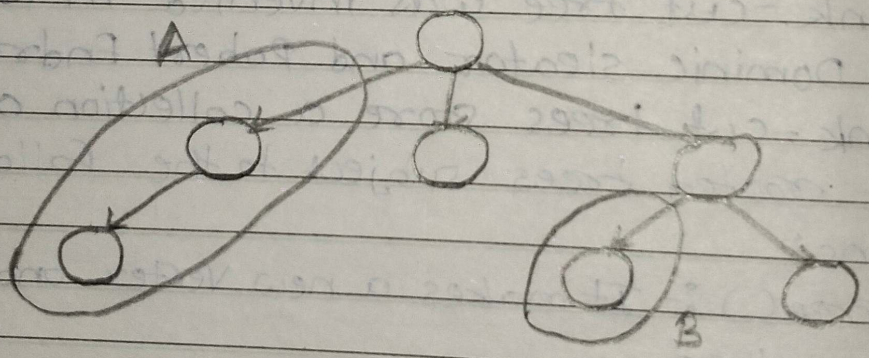
1) Preferred child :- For some node V, the child containing subtree which contains the last accessed node (say W) in V's subtree.

Note :- A node might not have a preferred child if it was the last accessed node in its subtree.

2) Preferred edge :- An edge leading to a Preferred child is called a preferred edge.



3) Preferred Path :- It is a maximal chain of Preferred edges, which we will represent using splay trees.



• operations :-

1) getRoot (v) :-

- First access(v) and return the lowest depth element in v's auxiliary tree, by repeatedly going to the left subtree of V.

Page No.
Date

2) access(v) :-
- Let's say we access a node v, then it no longer has any preferred child, we splay v, which brings it to the root of the auxiliary tree, we then detach the right subtree of v, so now v has no right subtree and it might have a path-pointer to another vertex w which is in another auxiliary tree.
- Now we are going to splay w and detach its right subtree, after which we will make v its right subtree, lastly, we will splay v again.

3) cut(v) :-
- access (v) and detach its left subtree (if it exists).

4) link(v, w) :-
- Assuming, w is the child and v is the parent, access(w) and then access(v), now we attach v as w's left child and make w, v's parent.

- The time complexity of all these operations is O(log n)

Advantages :-
1) Efficient Dynamic operations
2) Amortized Efficiency
3) Versatility in Applications
4) Path and Tree Queries
5) Flexibility in Representation.

• Disadvantages :-
1) Complexity of Implementation
2) overhead of splay operation
3) Learning curve
4) Memory overhead

• Applications :-
1) Dynamic connectivity in Networks
2) Dynamic minimum Spanning Tree
3) Dynamic Graph Algorithms
4) Network Flows
5) Dynamic programming on Trees

Imp
Que- Explain the Preferred Path Decompasition.
⇒

Preferred Path Decomposition :-
- Preferred Path Decomposition is also known as Heavy-Light Decomposition.
- Preferred path decomposition is a technique used in graph theory and data structures to manage and manipulate trees efficiently.
- The idea is to decompose a tree into Paths in a way that enables efficient operations like queries and updates.

1) Preferred child :- For each node in the tree, the Preferred child is the child that has the longest subtree.

2) Preferred Path :- Starting from the root, follow the Preferred child to form a Path.

- This Path continues until it cannot follow any further because there is no child or all remaining children have smaller subtrees.

3) Decomposition :- The entire tree is decomposed into several Preferred paths. once a preferred Path is Formed, the Process repeats for each node that is not on the current Path.
- These nodes starts new Preferred Paths.

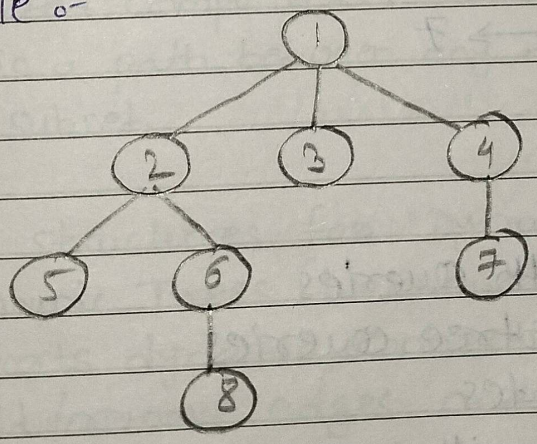
4) Heavy Path :- Preferred Paths are sometimes referred to as heavy paths because they follow the "heaviest" or longest subtree at each step.

• operations :-

1) Path Queries :- operations that involve querying or updating values along a path from the root to a node can be optimized using this decompasition.

2) Subtree queries :- Preferred path decomposition helps break down the problem into manageable segments, making subtree queries more efficient.

Example :-



Steps :-

1) Identify Preferred child :-

- For node 1, the children are 2, 3 and 4. Node 2 has the largest subtree (containing nodes 5, 6 and 8).
- For node 2, the children are 5 and 6. Node 6 has the largest subtree (containing nodes
- For node 4, the child is 7.
- other nodes either have no children or a single child, making the Preferred child choice straightforward.

2) Form Preferred Paths :-

- Starting from node 1, follow the Preferred child to form the first Path : 1 → 2 → 6 → 8.
- After forming the first Path, start a new Path from each node not on this Path.
- Start from node 3, from a new Path : 3.
- Start from node 4, from a new path : 4 → 7
- Start from node 5, from a new Path :- 5

So, the tree is decomposed into the following Preferred Paths :-

- Path 1 :- 1 → 2 → 6 → 8
- Path 2 :- 3
- Path 3 :- 4 → 7
- Path 4 :- 5

- Advantages :-

1) Efficient Path queries
2) Efficient subtree queries
3) Dynamic updates
4) Improved Algorithm performance.

• Disadvantages :-
1) complexity of Implementation
2) space overhead
3) Non-optimal for All Trees
4) complex query Handling
5) Limited Applicability.

• Application :-
1) Dynamic Trees
2) Heavy-Light Decomposition
3) Lowest common Ancestor (LCA)
4) Network Design and Analysis

Que- Explain the Dynamic Connectivity :-
⇒

Dynamic Connectivity :-
-Dynamic connectivity in dynamic graphs refers to the problem of maintaining information about the connected components of a graph as it undergoes updates, such as the addition or removal of edge.
• Dynamic Graph :- A graph that allows for modifications, such as adding or removing edges.
• Connected components :- subsets of the graph where there is a path between any two vertices within the same subset.

• Data structures for Dynamic connectivity :-
1) Dynamic Trees (Link-cut Trees) :-
-supports dynamic connectivity with operations to add/remove edges and query connectivity.
-Allows for efficient tree manipulations and Path queries.

- Particularly useful for maintaining a forest of trees.

2) Euler Tour Trees :-
- uses an Euler tour representation of a tree to maintain dynamic connectivity.
- Efficiently supports edge insertions and deletions in the context of trees.

• Applications :-
1) Network Reliability
2) Dynamic minimum spanning Tree
3) Real-time Navigation system.
4) social networks
5) Transportation system
6) computer Networks

• Advantages :-
1) Efficiency
2) Scalability
3) Flexibility
4) Robustness
5) cost-Effective

• Disadvantages :-
1) Complexity
2) performance overhead
3) Memory usage
4) Debugging and Testing
5) Scalability issues.

Page No.
Date

top

que- Explain Euler Tour Trees with example.

⇒

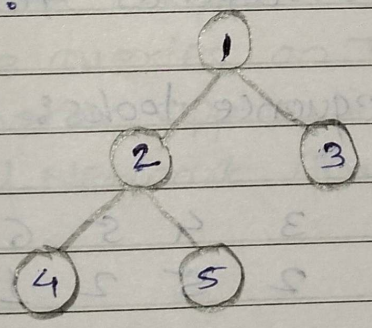
Euler Tour tree :-

- Euler tour is defined as a way of traversing tree such that each vertex is added to the tour when we visit it (either moving down from parent vertex or returning from child vertex).
- It requires exactly 2*N-1 vertices to store Euler tour.

Euler Tour concept :-

- An Euler tour in a tree is a traversal that visits each edge exactly twice : once when traversing from a parent to a child, and once when returning from the child to the parent.
- This traversal creates a sequence of nodes (edges) that captures the structure of the tree in linear form.

Example :-



Steps :-

1. Start at the root (1) :-
   - Visit 1 (begin tour at the root)

2. Traverse to the first child (2) :-
   - Visit 2 (from 1 to 2)

Date

3] Traverse to the first child of 2 (4):-
- visit 4 (from 2 to 4)
- Return to 2 (from 4 to 2)

4] Traverse to the second child of 2 (5):-
- visit 5 (from 2 to 5).
- Return to 2 (from 5 to 2)
- Return to 1 (From 2 to 1).

5] Traverse to the second child of 1 (3):-
- visit 3 (From 1 to 3)
- Return to 1 (From 3 to 1)

The Euler tour for this tree would be:-
$1 \rightarrow 2 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 2 \rightarrow 1 \rightarrow 3 \rightarrow 1$

- Euler Tour Tree Representation:-
-This sequence can be represented in a data structure, typically a balanced binary tree, where each node corresponds to an entry in the Euler tour.
-Here's how the sequence looks:-

Index:-	0	1	2	3	4	5	6	7	8
Node:-	1	2	4	2	5	2	1	3	1

- Dynamic operations:-

1] Link (Adding an Edge):- connect two trees by adding an edge.
2] Cut (Removing an Edge):- Disconnect a tree into two separate trees by removing an edge.

3] Query (connectivity or subtree information):-
-check if two nodes are in the same tree or gather information about a subtree.

- Advantages:-
1) Efficient Dynamic operations
2) compact Representation
3) simplified Traversal
4) Flexibility

- Disadvantages:-
1) complexity of Implementation
2) specific use cases
3) overhead for small Data
4) Memory usage
5) overhead for specific queries.

Applications:-
1) Dynamic connectivity
2) Network Design
3) Range queries on Trees
4) Geometric applications
5) social Network Analysis