Name: Rohit Kudache

USN: 1BM18CS083

Page No.

Date

1. Create Binomial tree node with data, degree as integer datatype
child, siblings, parent as pointer to the Node.

2. New node creation Function
Node* newNode(int key) it will assign corresponding child, parent and siblings to NULL

3. Insert Function - inserting a key into the binomial heap
insert (list<Node*>_head, int key)
- Create new Node temp
- Call insertToheap Function.

Insert to Heap Function - Inserting a Binomial tree to the binomial heap

insert to Heap Function (_heap, tree)
- creating a new heap temp
- inserting Binomial tree to temp heap
as temp.push_back(tree);
- Union operation to insert Binomial tree to original heap
temp = UnionBionomialHeap(_heap,temp)

4. Get min Function - return pointer to of minimol value Node present in the Binomial heap
Node* GetMin(list<Node*>_heap)
- we use iterator :: it as heap begin
- assign temp tNode to it pointer
- check it != _heap.end()
- if *it → data < temp → data
then temp = *it and it++

1BM18CS083

Page No.
Date

5. ExtractMin Function (takes heap as argument)
   - Create list <Node*> For new-heap, lo;
   - Create porater temp.
   - Assign temp to store minimal value element in heap (getMin (- heap))
   - list <Node*> :: iterator it;
   - Assign it to _heap.begin()
   - check For it != _heap.end()
   - if (* it != temp)
   - insert all Binomial tree into new Binomial heap Expect getmin
   [new-heap.push_back (*it);]
   - increment it
   - Assign Lo to the Function which remove min value From the tree And return to Binomial heap
   - Create new-heap which stores the value f output f the Function UnionBinomial Heap.
   - adjust the new-heap and again Store it in new-heap which will be consider as Final One.

6. Print the binomial heap

Parul
09/12/20.