1BM19CS062
Rajwal Patil N

ADS LAB-9

i) Create Binomial tree with data, degree as integer datatypes.
child, sibling, parent as pointer to the node.

ii) New node creation function
Node * Newnode(int key) it would assign corresponding child, parent & sibling to NULL

iii) Insert function - inserting a key into the binomial heap.
insert (list<node*> -head, int key)
→ create new node temp.
→ call insert to heap 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 = union binomial heap (_heap, temp)

(4) Get min Function - return pointer to minimal value node present in the binomial heap.

Node* get min (list<node*> -heap)
- we use iterator :: it as heap begin
- assign temp Node to it pointer.
- check it != -heap.end()
- if *it -> data < temp -> data
then temp = *it and it++

(5) Extraction function (takes heap of argument)
- create list<Node*> for new-heap, lo;
- create pointer 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)
- unset all binomial tree into new binomial heap except 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 of output of the function.
union binomial heap
→ adjust the new-heap and again store it in the new heap which will be considered as the final one
6. print the binomial heap.