ADS Lab-10

Triveni.y

1BM19CS411

9/12/2020

Write a program to implement the functions of a Binomial Heap:-

(i) Insert (H, K)

(ii) getMin (H)

(iii) extractMin (H)

```
Node* insertATreeinHeap(Node* _heap, Node *tree)
{
    list <Node *> temp;
    temp.push_back(tree);
    temp = unionBinomialHeap(_heap, temp);
    return adjust(temp);
}

<list> Node * Insert ( list <Node*> _head, int key)
{
    Node* temp = newNode(key);
    return insertATreeinHeap(_head, temp);
}

list<Node*> UnionBinomialHeap (list <Node*> l1,
                               list <Node*> l2)
{
    list <Node*> _new;
    list <Node*> :: iterator it = l1.begin();
    list <Node*> :: iterator ot = l2.begin();
```


Y. Tejy

```
while (it1 != l1.end() && it2 != l2.end())
{
    if ((*it1)->degree <= (*it2)->degree)
    {
        new.push_back(*it1);
        it1++;
    }
    else
    {
        new.push_back(*it2);
        it2++;
    }
}
while (it1 != l1.end())
{
    new.push_back(*it1); it1++;
}
while (it2 != l2.end())
{
    new.push_back(*it2); it2++;
}
```

getMin

```
Node* getMin(list<Node*> _heap)
{
  list<Node*>::iterator it = _heap.begin();
  Node* *temp = *it;
  while (it != heap.end())
  {
    if ((*it)->data < temp->data)
        temp = *it;
    it++;
  }
  return temp;
}
```

Extract-Min

```
list <Node*> extractMin (list <Node*> _heap)
{
   list <Node*> new_heap, lo;
   Node* temp;
   temp = getMin (_heap);
   list <Node*> :: iterator it;
   it = _heap.begin();
   while (it != _heap.end())
   {
      if (*it != temp)
      {
         new_heap.push_back(*it);
      }
      it++;
   }
   lo = removeMinFromTree (temp);
   new_heap = unionBinomialHeap (new_heap, lo);
   new_heap = adjust (new_heap);
   return new_heap;
}
```


Y. Tree