papergrid

Date: / /

Binomial Heap

Siadhama

```
func Insert (head, key):
{
    Node *temp = new Node(key)
    list <Node *> t
    t.push_back (temp)
    t = unionBH(head, key)
    return adjust (t)
}

adjust (list <Node> heap)
{   if (heap.size < 1) return heap
    list <Node> newheap
    ark it1, it2, it 4
    it  N: it 2 = it = heap.begin ()
    if ( heap.size () == 2)
  {   it 2 = it 1
      it 2 ++
      it 3 = heap.end()
  }
    else
  { it2 ++
    it3 = it 2
    it3 ++
  }
  while (it1 != heapend ())
  { if (it2) == heapend()) it1 ++
    else if (it1->degree < it2 -> degree)
    { it (it1 ; it2 ++;)
       if (it3 != heap ()) it 3++
}
```

Scanned with CamScanner

papergrid

Date: / /

```
else if (it3 != heap.end() && it1->degree == it2->degree &&
        it->degree == it3->degree)
        it1++, it2++, it3++;
    }
    return temp
}
func getmin(list<Node*> heap)
{ auto it = heap.begin()
    while (it != heap.end())
    { if (it->data < temp->data) temp = it
        it++
    }
return temp;
}
func extractmin(list<Node, heap)
{ heap<Node*> new heap, lo, Nodetemp;
  while (it != dep.end())
  { if (it != temp) newheap.pushback(it)
      it++
  }
  lo = len(rp)
  pushheap = union BH(newtemp, lo)
  new tmp = adjust(newtmp)
  return new heap
}
```