Date __/__/__ Abhishikat

Binomial heap

Inserting a Node

Insert node x into binomial Heap H, assuming Node x has already been allocated & key[x] has already been filled in.

```
Binomial-heap-insert(H, x)

    H' ← Make-Binomial-Heap()
    p[x] ← NIL
    child[x] ← NIL
    sibling[x] ← NIL
    degree[x] ← 0
    head[H'] ← x
    H ← Binomial-heap-union(H, H')
```

// we make a one node binomial heap H' and unite it with H.

```
Node HeapMin(H) // Node with minimum key.
1. find the root x with the minimum key in the root list of H, and remove x from the root list of H.
2. H' ← Make-Binomial-Heap()
3. reverse the order of linked list of x's children and set head[H'] to point to the head of the resulting list.
4. H ← Binomial-Heap-Union(H, H')
5. return x
```

Date __/__/____

getMin (H) // Find Minimum Key

```
y ← NIL
x ← head[H]
min ← ∞
while x ≠ NIL
    do if key[x] < min
        then min ← key[x]
             y ← x
        x ← sibling[x]
return y.
```

Binomial-heap-Union (H_1, H_2)

1. H ← MakeBinomialHeap()
head[H] ← Merge (H_1, H_2)
free the objects H_1 and H_2 but not the lists they point to
if head[H] = NIL
    then return H
prev-x ← NIL
x ← head[H]
next-x ← sibling[x]
while next-x ≠ NIL
    do if (degree[x] ≠ degree[next-x]) or
        (sibling[next-x] ≠ NIL) and degree[sibling[next-x]]
        = degree[x])
    then prev-x ← x — Case 1 & 2
        x ← next-x — Case 1 & 2
    else if key[x] ≤ key[next-x]
        [redacted]
        then sibling[x] ← sibling[next-x] — Case 3
            BinomialLink(next-x, x) — Case 3

Date __/__/____

else if prev-x = NIL — Case 4
then Head[H] ← next-x — Case 4
Binomial-link (x, next-x) — Case 4
x ← next-x — Case 4
next-x ← Sibling[x] — Case 4
return H

// Case 1
// degree[x] ≠ degree[next-x]

// Case 2
// degree[x] = degree[next-x] = degree[Sibling[next-x]]

// Case 3 & 4
When x is first of two roots of equal degree.

Algo Merge(y, z)

P[y] ← z
Sibling[y] ← child[z]
child[z] ← y
degree[z] ← degree[z] + 1