Algorithm for Fibonacci Heap Operations (from CLR text)

Make-Fibonacci-Heap()

```
n[H] := 0
min[H] := NIL
return H
```

Fibonacci-Heap-Minimum(H)

```
return min[H]
```

Fibonacci-Heap-Link(H,y,x)

```
remove y from the root list of H
make y a child of x
degree[x] := degree[x] + 1
mark[y] := FALSE
```

CONSOLIDATE(H)

```
for i:=0 to D(n[H])
     Do A[i] := NIL
for each node w in the root list of H
    do x:= w
       d:= degree[x]
       while A[d] <> NIL
           do y:=A[d]
              if key[x]>key[y]
                then exchange x<->y
              Fibonacci-Heap-Link(H, y, x)
              A[d]:=NIL
             d:=d+1
       A[d]:=x
min[H]:=NIL
for i:=0 to D(n[H])
    do if A[i]<> NIL
          then add A[i] to the root list of H
               if min[H] = NIL or key[A[i]]<key[min[H]]
                  then min[H]:= A[i]
```

Fibonacci-Heap-Union(H1,H2)

```
H := Make-Fibonacci-Heap()
min[H] := min[H1]
Concatenate the root list of H2 with the root list of H
if (min[H1] = NIL) or (min[H2] <> NIL and min[H2] < min[H1])
   then min[H] := min[H2]
n[H] := n[H1] + n[H2]
free the objects H1 and H2
return H
```

Fibonacci-Heap-Insert(H,x)

```
degree[x] := 0
p[x] := NIL
child[x] := NIL
left[x] := x
right[x] := x
mark[x] := FALSE
concatenate the root list containing x with root list H
if min[H] = NIL or key[x]<key[min[H]]
        then min[H] := x
```

```
n[H]:= n[H]+1
```

Fibonacci-Heap-Extract-Min(*H*)

```
z:= min[H]
if x <> NIL
        then for each child x of z
             do add x to the root list of H
                p[x]:= NIL
             remove z from the root list of H
             if z = right[z]
                then min[H]:=NIL
                else min[H]:=right[z]
                     CONSOLIDATE(H)
             n[H] := n[H]-1
return z
```

Fibonacci-Heap-Decrease-Key(*H*,*x*,*k*)

```
if k > key[x]
   then error "new key is greater than current key"
key[x] := k
y := p[x]
if y <> NIL and key[x]<key[y]
   then CUT(H, x, y)
        CASCADING-CUT(H,y)
if key[x]<key[min[H]]
   then min[H] := x
```

CUT(*H*,*x*,*y*)

```
Remove x from the child list of y, decrementing degree[y]
Add x to the root list of H
p[x]:= NIL
mark[x]:= FALSE
```

CASCADING-CUT(*H*,*y*)

```
z:= p[y]
if z <> NIL
  then if mark[y] = FALSE
       then mark[y]:= TRUE
       else CUT(H, y, z)
            CASCADING-CUT(H, z)
```

Fibonacci-Heap-Delete(*H*,*x*)

```
Fibonacci-Heap-Decrease-Key(H,x,-infinity)
Fibonacci-Heap-Extract-Min(H)
```