Illinois Institute of Technology
Department of Computer Science

Solutions to Homework Assignment 6

CS 430 Introduction to Algorithms
Spring Semester, 2018

Solution:

1. As defined, for integers $k \geq 0$ and $j \geq 1$,

$$A_k(j) = \begin{cases} j+1, & \text{if } k = 0, \\ A_{k-1}^{(j+1)}(j), & \text{if } k \geq 1, \end{cases}$$

we only consider cases when $j \geq 1$ here.

- When $j = 1$, $A_3(1) = A_2^{(2)}(1) = A_2(A_2(1)) = 2047$, $tower(1) = 2^{tower(0)} = 2$. Obviously we have $A_3(1) > tower(1)$.
- Assume when $j = i - 1$, we have $A_3(i-1) \geq tower(i-1)$; now we prove for $j = i$, we still have $A_3(i) \geq tower(i)$. According to **Lemma 21.3**, we have $A_2(j) = 2^{j+1}(j+1) - 1$. Also we know the function $A_k(j)$ strictly increases with both j and k. So:

$$\begin{aligned} A_3(i) = A_2^{(i+1)}(i) = A_2(A_2^{(i)}(i)) > A_2(A_2^{(i)}(i-1)) = A_2(A_3(i-1)). \\ \geq A_2(twoer(i-1)) = 2^{tower(i-1)+1}(tower(i-1)+1) - 1 > 2^{tower(i-1)} = tower(i) \end{aligned}$$

By induction hypothesis, we have proved that $A_3(j) \geq tower(j), \forall j \geq 1$.

2. Binomial-Heap-Extract-Min(H)

```
x = Binomial-Heap-Minimum(H)
Remove x from the root list of H
H' = Make-Binomial-Heap()
Reverse the order of the linked list of x's children, and set H'.head to point to the head of the
   resulting list.
H = Binomial-Heap-Union(H, H')
return x
```

Binomial-Heap-Minimum(H)

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

```
BINOMIAL-HEAP-UNION(H1, H2)
H = MAKE-BINOMIAL-HEAP()
H.head = merges the root lists of H1 and H2 into a single linked list that is sorted by degree into monotonically increasing order.
Free the objects H1 and H2 but not the lists they point to
if H.head = NIL then
    return H
prevx = NIL
x = H.head
nextx = x.sibling
while nextx ≠ NIL do
    if (x.degree ≠ nextx.degree) or (nextx.sibling ≠ NIL and nextx.sibling.degree = x.degree) then
        prevx = x
        x = nextx
    else
        if x.key ≤ nextx.key then
            x.sibling = nextx.sibling
            BINOMIAL-LINK(nextx, x)
        else
            if prevx = NIL then
                H.head = nextx
            else
                prevx.sibling = nextx
            BINOMIAL-LINK(x, nextx)
            x = nextx
    nextx = x.sibling
return H
```

```
BINOMIAL-LINK(y, z)
y.p = z
y.sibling = z.child
z.child = y
z.degree = z.degree + 1
```

```
BINOMAL-HEAP-DECREASE-KEY(H, x, k)
if k > x.key then
    error "new key is greater than current key"
x.key = k
y = x
z = y.p
while z ≠ NIL and z.key > y.key do
    exchange z with y
    y = z
    z = y.p
```

BINOMIAL-HEAP-DELETE(H, x)
BINOMIAL-HEAP-DECREASE-KEY$(H, x, -\infty)$
BINOMIAL-HEAP-EXTRACT-MIN(H)

3. **Problem 19-3 on page 529**

a. The algorithm is given below. According to the analysis in CLRS, page 519, the amortized running time of the implementation of FIB-HEAP-CHANGE-KEY is still $O(\lg(n))$

FIB-HEAP-CHANGE-KEY(H, x, k)
if $k < x.key$ **then**
    FIB-HEAP-DECREASE-KEY(H, x, k)
else$k > x.key$
    FIB-HEAP-DELETE(H, x)
    FIB-HEAP-INSERT(H, x, k)

b. In order to implement the FIB-HEAP-PRUNE(H, r), we modify the structure by adding double linked list among all the leaf nodes, which helps us easily extract any leaf node. To do the prune operation, we randomly choose a leaf node, and remove it from both the leaves list and its parent's list, as shown below:

FIB-HEAP-PRUNE(H, r)
for $i \leftarrow 1$ to $\min(r, H.n)$ **do**
    $x \leftarrow$ random leaf node
    remove x from parent's children list
    remove x from leaves list

Due to such structural modification, we add one more cost $s(H)$ to the original potential function, which is related to the size of heap. Thus, the new potential function is

$$\Phi'(H) = \Phi(H) + s(H) = t(H) + 2m(H) + s(H)$$

Assume we remove $q = \min(r, H.n)$ nodes in this operation, and removing such q nodes brings c times cascading cuts. Similar to decrease key analysis in CLRS, page 521, the actual cost in this operation should be $c + q$: c times cascading cut and adding new leaf nodes, q times removing nodes.

Thus, the potential change of original part $\Phi(H)$ is still $4 - c$, while the additional potential change is $-q$.

Thus, the amortized cost should be $c + q + 4 - c - q = 4 = O(1)$.