Illinois Institute of Technology
Department of Computer Science

Solutions to Homework Assignment 6

CS 430 Introduction to Algorithms
Spring Semester, 2017

Solution:

1. As $A_k(j)$ requires $j \geq 1$. We only try to prove the cases when $j \geq 1$. We can prove this by induction. When $j = 1$, $tower(1) = 2^{tower(0)} = 2$. According to Lemma 21.3 on page 573 of CLRS, we have $A_3(1) = A_2(A_2(1)) = A_2(7) = 2^8 * 2^3 - 1 = 2^{11} - 1$. Thus we have $A_3(1) > tower(1)$. Suppose $A_3(j) > tower(j)$ for $j = p - 1$, where $p > 1$. We just need to prove the case when $j = p$. $A_3(p) = A_2^{p+1}(p) = A_2(A_2^p(p))$. As $A_3(p-1) = A_2^p(p-1)$ and $A_2(x)$ is obviously an increasing function, we have $A_3(p) > A_2(A_2^p(p-1)) = A_2(A_3(p-1))$. By induction hypothesis, we have $A_3(p) > A_2(A_3(p-1)) > A_2(tower(p-1)) = 2^{tower(p-1)+1}(tower(p-1)+1) - 1 > 2^{tower(p-1)} = tower(p)$. Thus we have proved $A_3(j) > tower(j)$.

2. A root x in the heap came to be marked because there exist a child with a decreased key.

   And it doesn't matter to the analysis that x is marked. It is because that we only check a node is marked or not in line 3 of CASCADING-CUT, which is only run on the nodes with non-NIL parent. That is, this check never run on the marked root due to its NIL parent. Thus, this markness doesn't lead any more actual work. It perhaps made the potential function to be larger, but this larger part will never be used in following processes.

3. **a.** According to the analysis in CLRS page 519, the amortized running time of operation below takes $O(\lg(n))$.

Algorithm 1 FIB-HEAP-CHANGE-KEY(H, x, k)

```
if k < x.key then
    FIB-HEAP-DECREASE-KEY(H, x, k)
elsek > x.key
    FIB-HEAP-DELETE(H, x)
    FIB-HEAP-INSERT(H, x, k)
end if
```

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 in algorithm 2.

Algorithm 2 FIB-HEAP-PRUNE(H, r)

```
for i ← 1 to min(r, H.n) do
    x ← random leaf node
    remove x from parent's children list
    remove x from leaves list
end for
```

Due to such structural modification, we add one more cost $s(H)$ to 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 page 521(CLRS), 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 $c + q + 4 - c - q = 4 = O(1)$