1) Exercise 19.3-1 on page 522:
A node is marked if one of its children have been promoted. So, assume that x is the min root (value x = 20) and it has two children with larger values (left child = 30, right child =40). Now, we preform FIB-HEAP-DECREASE-KEY on the left child so that it's value (left child = 15) becomes smaller than the value of x. This (left child = 15 and root x = 20) violates the min heap property. So, the left child will be cut off causing the parent root x to become marked. Then CASCADING-CUT is performed on the cut child and it cascades up and joins the root list. Therefore, it doesn't matter to the analysis that x is a marked root since none of the operations depend on this.
Citations: 1) http://paul.rutgers.edu/~abasit/cs513/hw3_sol.pdf, Author: Abdul Basit, Title: Homework 3 Solutions, Date: unknown.
2) https://github.com/klutometis/clrs/blob/master/20.3/20.3-1.txt, Author: Peter Danenberg, Title: clrs/20.3/20.3-1.txt, Date: Oct. 27, 2008.
3) http://www.chegg.com/homework-help/suppose-root-x-fibonacci-heap-marked-explain-x-came-marked-r-chapter-20.3-problem-1e-solution-9780070131514-exc, Author: unknown, Title: CH20.3,1E, Date: unknown.

2) Problem 19-3 on page 529: More Fibonacci-heap operations
a) Fib-Heap-Change-Key(H,x,k)

```
if k>key[x]
  then Fib-Heap-Delete(H,x)            //O(lg n)
       y ← new node w/ key=k
       Fib-Heap-Insert(H,y)            //O(1)
else Fib-Heap-Decrease-Key(H,x,k)      //O(1)
```

Amortized running time:
k <= key[x]: O(1) Since in a Fibonacci Heap the Decrease-Key function has only O(1) actual cost and amortized cost and this is the only function we need to use when k<=key[x].
k>key[x]: O(lg n)+O(1) = O(lg n) This is because Delete function depends on Decrease-Key=O(1) and Extract-Min=O(lg n) functions, so, Delete has amortized cost of O(lg n). Then along with Delete we also use Insert function which has actual and amortized cost=O(1). Therefore, by taking the sum of both function's amortized cost we get O(lg n) amortized runtime when k>key[x].

b) Fib-Heap-Prune(H,r).

```
i ← 1
while i <= r
      do get the i-th leaf x from leaf list
      y←p(x)
      Fib-Heap-Delete(H,x,y)
      Fib-Heap-Cascading-Cut(H,y)
      i++
```

We can delete nodes from leaves, so no rearrangements of the heap data structure or operations are necessary. Each single node deletion is, in the best case=O(1) and in the worst case=O(lg n) if we need to cascade up the heap to the top.

Amortized analysis: Let t(H)=number of trees in the root list of H, m(H)=number of marked nodes in H, n(H)=number of nodes in the heap, $\Phi(D_i)$=potential of Fibonacci heap after deletion, and $\Phi(D_{i-1})$=potential of Fibonacci heap before deletion.
So, $\Phi(D_i)$= t(H) + 2m(H) + n(H) is the original potential of the heap given on page 509 with number of nodes in H added. Now we have,
s = min(r,n[H]) which means s=number of nodes deleted. Then for the difference we get,
$\Phi(D_i) - \Phi(D_{i-1}) = -s$, this means that the difference in the potential after deletion is the number of nodes deleted. So, the difference in potential is a constant.
Therefore, we calculate the amortized cost as the sum of the actual cost=O(1) of pruning and the difference in potential as follows.
Amortized cost = $c_i + \Phi(D_i) - \Phi(D_{i-1})$
= c_i *O(1) + $\Phi(D_i) - \Phi(D_{i-1})$ where c=number of prunes is a constant
= s*O(1) – s , where s=number of nodes deleted is a constant
= O(s) , where in s is a constant number therefore this can be written as O(1)
Thus, the amortized cost of FIB-HEAP-PRUNE will be of constant time O(1).

Citations:

1)https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&cad=rja&uact=8&ved=0ahUKEwjI7o_q9ufLAhWhmIMKHQ5PBSAQFgg5MAU&url=http%3A%2F%2Fwww.cs.uml.edu%2F~buford%2F91.503%2Fasn2_soln.doc&usg=AFQjCNGg_6kuUPK80SpIttyk_fuGuGmQMA, Author: unknown, Title: Sample Solutions for 91.503 Assignment #2, Date: unknown.
2) http://www.chegg.com/homework-help/wish-augment-fibonacci-heap-h-support-two-new-operations-wit-chapter-20.p-problem-2p-solution-9780070131514-exc, Author: unknown, Title: CH20.P,2P, Date: unknown.
3) Chapter 19.1 thru 19.3 pages 509 to 530, Author: CLRS, Title: Introduction to Algorithms 3E, Date: 2009.
4) http://cs.iit.edu/~cs430/scribbling/mar23.pdf, Author: Edward M. Reingold, Title: in class scribblings (Mar 23), Date: March 23, 2016.