HW 3

Problem 1

$f(n) = n \log n$. The algorithms run time is $(n-2)\log n$ since it is dividing $n-2$ elements $\log 2$ times.

$$\lim_{n\to\infty} \frac{(n-2)\log n}{n \log n} \Rightarrow \lim_{n\to\infty} \frac{(n-2)}{n} \Rightarrow \lim_{n\to\infty} 1 - \overset{\to 0}{\frac{2}{n}} = 1.$$

By limit theorem the running time of the algorithm is $\Theta(n \log n)$.

Problem 2

(a) Divide the rungs into groups of $\lfloor \sqrt{n} \rfloor$ rungs. Starting at the lowest group, drop a jar from the highest rung in each group until it shatters. Then drop a jar starting from the lowest rung in the current group and move up until it shatters. The highest safe rung will be the rung right before the rung the jar fell from and shattered.
$2\lfloor \sqrt{n} \rfloor - 1$ drops at most $= O(\sqrt{n})$. $f(n) = \sqrt{n}$

(b) Divide the rungs into groups of $\lfloor n^{\frac{k-1}{k}} \rfloor$ rungs. Starting at the lowest group, drop a jar from the highest rung in each group until it shatters. Then drop a jar starting from the lowest rung in the current group and move up until it shatters. The highest safe rung will be the rung right before the rung the jar fell from and shattered.
$2k\lfloor n^{1/k} \rfloor - 7$ drops at most.

Problem 3:

```
i=1
findSmallerKeys(X, i)
    if (i >= heap size)
        return
    if (heap(i) >= X)
        return
    print heap(i)
    findSmallerKeys(X, 2i)
    findSmallerKeys(X, 2i+1)
```

The time complexity of the algorithm is $O(n)$ because the worst case scenario is every element in the heap of size n is smaller than X so n comparisons would be made.