... 

{ 
    
    Heapifier.Singleton.stiftdown(A, 0, cur, hi) 
}  

for (int cur = hi - 10 + 1; cur <= 10; cur--) 

) 

public HeapSorter(int[] A, int 1, int hi) 

public class HeapSorter extends Asorter 

Initializing the Heap: HeapSorter()
\[(I) \quad \sum_{\substack{u \in \Omega \setminus \{1\} \text{ } \text{ with } u \text{ } \text{ an even number}}} \left( \frac{\eta \bar{z}}{u} \right)^{2^{(n-1)}} = (\eta)O \left[ \frac{\bar{z}^{2}}{u} \right] \sum_{\substack{u \in \Omega \setminus \{1\} \text{ } \text{ with } u \text{ } \text{ an even number}}} \left( \frac{\eta \bar{z}}{u} \right)^{2^{(n-1)}}\]

The time required by \texttt{stddown()} when called on a node of height \(h\) is

- The time analysis relies on the property that in an \(n\)-element heap, there are at most \(\left\lfloor \frac{\eta \bar{z}}{u} \right\rfloor \) nodes of height \(h\).
- The tighter analysis relies on the property that in the tree, and the heights of most nodes are small.
- We can derive a tighter bound than \(O \left( \log u \right)\) by observing that the running time of \texttt{HeapSorter()} is

\[O(n \log \log n)\]
\[ (4) \quad z = \frac{z_2 / (z_1 - 1)}{z_1} = y \underbrace{\int_0^\infty}_{\gamma} \]

in which \( x = z_2 / z_1 \) is substituted to yield

\[ (3) \quad \frac{z (x - 1)}{x} = y \underbrace{\int_0^\infty}_{\gamma} x \]

to obtain

\[ (2) \quad \frac{x - 1}{x} = y \underbrace{\int_0^\infty}_{\gamma} \]

\((I > |x| \text{ for both sides of the infinite geometric series})\)

The last summation can be evaluated by differentiating and multiplying by

**Analysis of HeapSorter**

(cont.)
(5) \[ (u)O = (uZ)O = \left( \frac{\pi}{\eta} \right)^{0=\gamma} u)O = \left( \frac{\pi}{\eta} \right)^{0=\gamma} u)O \]

as

Thus, the running time of the Heapsorter() constructor can be bounded.
Example of string()
Consider removing the node with the key 29 from the following heap:

Removal Of A Non-Root Node From A Heap
public void enqueue(EnqueueQueue) {
    public void dequeue(DequeueQueue) {
    public void void dequeue(Dequeue data) {
    public interface IPriorityQueue {
        import ordered.Ordered;
        import java.util.PriorityQueue;
        
        package queues;
}

    IPriorityQueue