- Big O Complexity of delMax()
  - public int delMax(){
  - int data = heap_array[0]; O(1)
  - heap_array[0] = heap_array[nElems-1]; O(1)
  - heap_array[nElems-1] = 0; O(1)
  - nElems--; O(1)
  - sink(0); O(N Log 2 (n))
  - System.*out*.println("Max Deleted:");
  - printArray();
  - return data;
  - }
    - private void sink(int k){
    - int temp = heap_array[k]; O(1)
    - int child_node;
    - while (location(k,1)< nElems){ O(log base 2 (n))
    - child_node = findLargestChild(k); O(d)
    - if(heap_array[child_node] > temp){ O(1)
    - heap_array[k] = heap_array[child_node]; O(1)
    - }else
    - break;
    - k = child_node; O(1)
    - }
    - heap_array[k] = temp; O(1)
    - }
- The worst case for delMax() is O(d log $_2$ N) where d is equal to number of children
  - highlighted code causes this time complexity

- BIG O COMPLEXITY OF daryHeapsort()

```
public int[] daryHeapsort(){  // heapsort function
  int N = nElems-1;
  for (int i = N/2; i >=0; i--){              // O(log 2 N )
     sort(heap_array,i);   O( log 2 N )
  }
  int k = nElems;
  for (int i = k; i > 1; i--){                // O ( N)
     int temp = heap_array[0];    // set temp = to root      // O(1)
     heap_array[0] = heap_array[i-1]; // set root to most recent child     O(1)
     heap_array[i] = temp;  // most recent child = root        O(1)
     nElems--; // decrease nELems    O(1)
     sort(heap_array,0);           //o( Log 2 N     )
  }
  System.out.println("Sorted Array:");
  printArray();
  return heap_array;
}
private void sort(int[] arr, int index) {
  int left = 2 * index;  // left  = 2 * current root O(1)
  int right = 2 * index + 1;  // right = 2 * current root plus 1   O (1)
  int max = index; // set max = root index   O(1)
  if (left <= nElems - 1 && arr[left] > arr[index]) // if left child great than parent  O(1)
  {
     max = left; // max = left
  }
  if (right <= nElems - 1 && arr[right] > arr[max])  // if right child great than parent   O(1)
     {
     max = right; // max = right
     }
  if (max != index)  // only called if max =! root;
  {
     int temp = arr[max];  //swap max and index given O(1)
     arr[max] = arr[index]; O(1)
     arr[index] = temp;
     sort(arr, max); // recursive call sort with arr and new max  log(N)
  }
}
```

- Worst case for dary-heapsort is O (N log N)
  - This is cause by the for loop which iterates through the array and the recursive call to implement the heapsort.