The Heapsort Algorithm

void max-heapify( T[] A, Integer n, Integer i)
    p = i;
    while ( 2p ≤ n )
            m = 2p+1;
        else
            m = 2p;
            swap( A[p], A[m] );
            p = m;
        else
            return;
    void build-max-heap( T[] A)
        n = A.length;
        for ( i = \lceil n/2 \rceil, \lceil n/2 \rceil – 1, ..., 1 )
            max-heapify( A, n, i)
    void sort-max-heap( T[] A)
        n = A.length;
        for ( i = n, n–1, ..., 2 )
            swap( A[1], A[i] );
            max-heapify( A, i–1, 1);
    void heapsort( T[] A)
        build-max-heap( A);
        sort-max-heap( A);

Initially: A is an array of size at least n, and 1 ≤ i ≤ n. The max-heap property holds everywhere in the subtree of A[1..n] rooted at A[i], except possibly at A[i] itself.

Upon return: The subtree of A[1..n] rooted at A[i] is a max-heap. The rest of A is unchanged.

Comparisons: at most 2h, where h is the height of the subtree.
This height is
1 if \lceil n/2 \rceil+1 ≤ i ≤ \lceil n/2 \rceil,
2 if \lceil n/2 \rceil+1 ≤ i ≤ \lceil n/2 \rceil,