Solutions of the review problems

1. Apply the algorithm to inversely ordered integers n,n-1,n-2,…,1. Fixing disorders of n with n-1,n-2,…,1 requires n-1 swap operations, fixing disorders of n-1 with n-2,n-3,…,1 requires n-2 swap operations, …, and fixing disorders of 2 with 1 requires 1 swap operation. The algorithm has to perform at least

   $$(n-1)+(n-2)+\ldots+1 = n*(n-1)/2$$

   swap operations.

2. Find an integer k such that the integer interval $[2^k,2^{k+1})$ contains n. Introduce level indices 0,1,2,…,k, with $k=[\log_2 n]$. Start the process with an array of inversely ordered integers n,n-1,n-2,…,1. For each run of *pushdown* swap operations are performed until the bottom of the tree is reached. Let M be the total number of swap operations that are performed.

   (i)

   $$M = \text{height}([n/2])+ \text{height}([n/2]-1)+ \text{height}([n/2]-2)+\ldots+ \text{height}(1).$$

   (ii) Counting swaps for the full tree with all levels 0,1,2,…,k filled up we get an estimate for M from above by the sum

   $$1*2^{k-1}+2*2^{k-2}+3*2^{k-3}+\ldots+k*2^0,$$

   and this sum with the help of the provided formula may be expressed as 2^{k+1}-k-2. We obtain an estimate for M from above by 2n. Counting swaps for the full tree with all levels 0,1,2,…,k-1 filled up leads to an estimate for M from below by the sum

   $$1*2^{k-2}+2*2^{k-3}+3*2^{k-4}+\ldots+(k-1)*2^0,$$

   and this sum by the same approach as above may be expressed as 2^k-(k-1)-2. We get an estimate for M from below by n/4, for sufficiently large n.
   (iii) The order of growth is linear.

3. Use a trie data structure for the process of sorting. First perform an insertion of all words into an initially empty trie. This operation takes O(n) time. Trie nodes will automatically place the letters of inserted words according to alphabetical ordering. Traversal operation combined with printing out paths corresponding to inputted words will take care of accessing the words in sorted order. Also this operation takes O(n) time.

4. (i) Recurrence equation

$T(1) = 1,$
$T(n) = 2*T(n-1)+1.$

Solution: $T(n)=2^n-1$.

(ii) Recurrence equation

$T(1) = c_1,$
$T(n) = 2*T(n/2)+c_2*n.$

Solution: $T(n)=O(n*\log_2 n)$.

5. (i) The first diagonal contains 1 number that requires 1 operation +. The second diagonal contains 2 numbers that require 1 operation + (each). The n^{th} diagonal contains n numbers that require 1 operation + (each of them). Altogether we get

$$1+2+3+\ldots+n = (n+1)*n/2.$$

(ii) Each diagonal contains 0 and 1 as its boundary points and a number of points of the interval (0,1). The first diagonal consists of (0,1/2,1). The second diagonal of (0,1/4,3/4,1). The third one of (0,1/8,1/2,7/8,1), etc. We observe that the n^{th} diagonal splits the interval [0,1] into n+1 intervals and that if we interpret lengths of these intervals as multiples of the length of the shortest interval, we obtain rows of Pascals triangle (1,1), (1,2,1), (1,3,3,1), …. If two relative consecutive lengths of the given diagonal have the form a,b, then in the next diagonal they contribute a+b (as an easy arithmetic computation shows) and this is exactly the recursive rule for the Pascal's triangle.