1. C
2. A
3. A, C, A
4. B, D, A
5. C
6. C
7. C

1. A simple divide-and-conquer algorithm with parallel recursive calls working on A[i+1..n+1].
2. Running time is n^2. A simple counterexmmple is [1,4,2,3]. When constructing a subsequence starting at 1, the algorithm greedily picks 4, which pecludes it from discovering 1,2,3 - the longest increasing subsequence.
3. Hints:
   One simple possible solution is to consider a binary choice of including the first element of a subproblem in the subsequence or not. Then the subproblem is defined by two parameters: the starting index i and the index of the previously picked element j (j < i). The subproblem is to find the length of the longest increasing subsequence in A[i..n], such that its elements are lerger than A[j]. The whole problem is the subproblem (i=1, j=0), assuming that A[0] = minus infinity. The resulting dynamic-programming algorithm will have n^2 running time and will use n^2 space.

   Another possible solution:
   Assume that the subproblem is to construct in A[i..n] an increasing subsequence that starts with A[i]. Then the choice is k-nary: what is the index of the next element to include after A[i]. We consider all elements from A[i+1..n] that are larger than A[i]. The solution to the whole problem is then the maximum of the solutions to all the possible subproblems i = 1,2,..n.
   The resulting algorithm will still have n^2 running time, but will use n space.