Scientific Theory in Informatics A1N

Algorithmic Strategies Exercises

All groups should try to solve all exercises!

1. *Merge sort* is a sorting algorithm that is based on the algorithmic strategy of *divide and conquer* (D&Q). Consider the following sorting algorithm, *A*, which also uses D&Q:

   A first divides a large list into two smaller sub-lists: the low elements and the high elements. *A* then recursively sort the sub-lists. The steps are:

   1. Pick an element, called a *pivot*, from the list.
   2. Reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final (correct) position.
   3. Recursively apply the above steps to the sub-list of elements with smaller values and separately the sub-list of elements with greater values.

   The base cases of the recursion are lists of sizes zero or one, which never need to be sorted.

   In pseudocode:

   ```
   function A(array)
       if length(array) ≤ 1
           return array  // array is already sorted
       select and remove an element pivot from array
       create empty lists less and greater
       for each x in array
           if x ≤ pivot then append x to less
           else append x to greater
       return concatenate(A(less), [pivot], A(greater))
   ```

   A. As examples of D&Q algorithms, what is the fundamental difference between *merge sort* and *A*?

   B. As sorting methods, what are the similarities and differences between *merge sort* and *A*?

2. Consider the following alternative recursive *definition* of the factorial of a non-negative integer *n*:

$$fac(n) = \begin{cases} 1, & n = 0 \\ \dfrac{fac(n+1)}{n+1}, & n \geq 1 \end{cases}$$

   How could this definition be turned into a recursive *algorithm*?

3. Binary search can be used for finding the position of an element (key) in a sorted list of items:

```
function BinarySearch(array, key)
    BS(array, key, 0, length(array)−1)

function BS(array, key, low, high) {
    if (high < low)     // array is empty, so key not found
        return KEY_NOT_FOUND
    else
        // calculate midpoint to divide list in half
        middle = midpoint(low, high)
        if (array[middle] > key)
            // key is in lower sublist
            return BS(array, key, low, middle−1)
        else if (array[middle] < key)
            // key is in upper sublist
            return BS(array, key, middle+1, high)
        else
            // key has been found
            return middle
```

   Is *BinarySearch* as given above an example of a D&Q algorithm?

4. Consider the description of the song "Hey Jude" on slide 3. Is this an algorithm in the *formal* sense?

5. Redefine the definitions of *odd* and *even* on slide 19 so that they give *working recursive definitions* of a number being odd and even, respectively.

6. How can a general *greedy algorithm* be formulated as a *state space search algorithm*, as given on slide 38?