**Question 2 (15 Pts, SO-2)**

Compare and contrast between iterative and recursive solutions. When would you prefer iteration over recursion and vice-versa? Justify your answer by giving different examples than the ones which are provided in the lecture slides.

**Recursion**

* Involves a recursive function which calls itself repeatedly until a base condition is not reached .
* Keeps code short and simple.
* Terminates when a base case is recognized.
* Increases the processor’s operating time.
* Uses selection structure.
* Recursion is slower than iteration.

**Iteration**

* Involves the use of loops through which a set of statements are executed repeatedly until the condition is not false.
* Makes code longer.
* Terminates when the loop-continuation condition fails.
* Reduces the processor’s operating time.
* Uses repetition structure.
* Iteration is faster than recursion.

If method must solve all of the subproblems of the original problem, an iterative method is preferable than a recursive method since recursive method have a procedure call overhead that iterative method do not.

On the other hand, if not all subproblems must be solved (some subproblems may be skipped), a recursive method may be preferable because it enables subproblems to be skipped.

There is another important disadvantage. If the depth of the recursion is too large, then there is a chance of stack overflow. This problem doesn’t occur in iterative method.

**Example: Insertion Sort**

* **Recursive Code**

**void** **insertionSortRecursive**(**int** A[], **int** n) {

**if** (n <= 1)

insertionSortRecursive(A, n-1)

**int** last = A[n-1]

**int** j = n-2

**while** (j >= 0 && A[j] > last) {

A[j+1] = A[j]

j—

}

A[j+1] = last

}

* **Iterative Code**

**void** **insertionSort**(**int** A[], **int** n) {

**int** i, key, j

**for** (i = 1 to n-1) {

key = A[i]

j = i - 1

**while** (j >= 0 && A[j] > key) {

A[j + 1] = A[j]

j = j - 1

}

A[j + 1] = key

}

* Space Complexity of recursive code = O(n) (for recursion call stack)
* Space Complexity of iterative code = O(1)
* Here the recursive algorithm is difficult to analyse and less intuitive to think. It includes the overhead of function calls and recursion call stack.
* The Iterative approach looks intuitive, clean and easy to understand.