FIT1045 Algorithmic Problem Solving – Tutorial 7.

Objectives

The **objectives of this tutorial** are:

- To become familiar with the divide and conquer technique.
- To introduce recursion.
- To introduce recursive sorting algorithms that use divide and conquer techniques.
- To introduce binary search.

Task 1

The Tower of Hanoi is a puzzle that consists of three rods or spindles and n discs of different sizes. Initially these discs are stacked in increasing order in terms of diameter on a single rod. The aim of the puzzle is to move the stack of discs from one rod to another rod while obeying the following rules:

- Each move consists of moving the top disc on one rod to another rod.
- Only one disc can be moved at a time.
- No disc can be placed on top of a smaller disc.

The aim is to find the minimum number of moves required to move n discs from one rod to another.
Discuss the algorithm for solving the Tower of Hanoi. Step through the algorithm for the $n = 3$ disc case.
How many moves does it take to solve the 1 disc case?
How many moves does it take to solve the 2 disc case?
How many moves does it take to solve the 3 disc case.
Modify the algorithm so it would find the minimum number of moves for the n disc case.

Task 2

In groups of 2–3, use the `Mergesort` algorithm to sort a pile of unsorted cards into ascending numerical order.

Use `Mergesort` to sort the list [”cat”, ”goat”, ”sheep”, ”ant”, ”zebra”, ”dog”, ”horse”, ”flea”] into alphabetical order.

Task 3

The Partition Algorithm

The following is an algorithm to partition the list L:

```
pivot = L[0]
index =0
k=1
while (k<length(L))
{
      if(L[k]<pivot)
      {
            index=index+1
            swap the items L[index] and L[k] in L
      }
      k=k+1
}
swap the items L[0] and L[index] in L.
```

Give an invariant for the loop in this algorithm.

Trace the partition algorithm with input $L = [4, 2, 6, 1, 5, 7]$. Fill in Table by giving the values of the variables L, index and pivot every time the loop performs the if block.

L	index	pivot

Quicksort

In groups of 2–3, use the `Quicksort` algorithm to sort a small pile of unsorted cards into ascending numerical order.

Use `Quicksort` to sort the list [”cat”, ”goat”, ”sheep”, ”ant”, ”zebra”, ”dog”, ”horse”, ”flea”] into alphabetical order.

Discuss the differences and similarities of the the Mergesort and Quicksort algorithms. Compare these algorithms to the Selection Sort and Insertion Sort.

Task 4

Guess the number: You must guess the number your partner has selected from the numbers 1 to 100. Each time you guess, your partner tells you if you are correct, or if the number you guessed is too large or too small. What is the maximum number of guesses you need to find the number?

Discuss a strategy for solving this problem.

A similar strategy is used in binary search to determine if an item is in a sorted list. Write an algorithm for binary search.

Puzzle of the week

Crazy Chip Puzzle: Professor Moris has supposedly identical n VLSI chips that in principle are capable of testing each other. The professor's test jig accommodates two chips at a time. When the jig is loaded, each

chip tests the other and reports whether it is good or bad. A good chip always reports accurately whether the other chip is good or bad, but the answer of a bad chip cannot be trusted. Thus, the four possible outcomes of a test are as follows:

Chip A says	**Chip B says**	**Conclusion**
B is good	A is good	both are good, or both are bad
B is good	A is bad	at least one is bad
B is bad	A is good	at least one is bad
B is bad	A is bad	at least one is bad

The professor want to determine which chips are good using a strategy based on this kind of pairwise test. Assume that the bad chips can conspire to fool the professor.

- Show that if more than $n/2$ chips are bad, the professor cannot necessarily determine which chips are good.
- Suppose more than $n/2$ chips are good. What is the minimum number of pairwise tests needed to find one good chip?