**Short answer questions [4\*5=20 Marks]**

**Please use examples and well labeled diagram(s) wherever appropriate.**

**i. Explain the concept of exceptions and exception handling, when they are used and how to catch exceptions.**

An exception refers to a problem that occurs in the program running time. The exception will be defined as an event being thrown. There are many types of exceptions, and they all inherit from SystemException. These types of exceptions are roughly divided into the following part

1. Related to array collection

2. Related to member access

3. Related to parameters

4. Related to arithmetic

5. Related to IO Of course, there are other abnormalities.

Move tr C# exception handling, it is based on four keywords: try, catch, finally and throw.

Try: The try block identifies which specific exceptions of the code block will be activated. It is followed by one or more catch blocks.

Catch: The program catches the exception, and the problem to be dealt with, where the program exception is handled.

Finally: The finally block is executed no matter what type of exception is thrown. For example, if a file is opened, it must be closed if the exception is raised.

Throw: A program throws an exception when a problem occurs. By using a throw keyword.

**ii. Differentiate between ‘class’ and ‘struct’ in C#.**

**iii. Explain the possible reason a class in C# can only inherit from a single class, while being able to implement multiple interfaces. Explain the sequence of constructors that will be called when you create an object that is an instance of a derived class**

**iv. Write short note on Big-O Analysis of Algorithms.**

**v. Explain briefly dynamic programming and, backtracking.**

1)

In my opinion, the core of the dynamic programming method is a list. The step of solving a complex problem should be to solve this problem hierarchically, then a series of small problems will arise, but among these small problems, some problems are solved in the same way. If the traditional recursive method is used, small problems with the same solution will be repeatedly executed, which will waste time.

The little story below will tell you what the dynamic programming looks like.

Question: 1 + 1 + 1 + 1 + 1 =?

A: The answer that we are assumed to be “5!”

Q: if we add one more “1 +” at the left side of equation, what is the answer?

A: You can easily figure out the answer is ”6!”

But why you can get the answer without calculated by your brain, that because you have already remembered the first equation which is 1 + 1 + 1 + 1 + 1 = 5!, your brain had record the first value of the equation which “5!” at the same time, you don’t need do calculate repeatedly. So, from my understanding the core of the dynamic programming is to remember the solutions of sub-problems that have been solved.

Move to C#, we still can find some example to explain what the dynamic programming is? I give some code below

class Program

{

static void Main(string[] args)

{

Console.WriteLine("This is not dynamic programming");

Console.WriteLine(getFib(5));

Console.ReadLine();

//Involving array to record the fib value

Console.WriteLine("#####################################");

Console.WriteLine("This is basic dynamic programming");

Console.WriteLine(getFibDynamic(5));

Console.ReadLine();

}

static int getFib(int n) {

if (n == 1)

return 1;

if (n == 2)

return 1;

return getFib(n - 1) + getFib(n-2);

}

static int[] fib = new int[6];

static int getFibDynamic(int n)

{

if (n == 1)

return fib[1]=1;

if (n == 2)

return fib[2]=1;

// if fib[n] existed, return directly

if (fib[n] != 0)

{

return fib[n];

}

// not existed, using recursive method, and record the value

else {

return fib[n]= getFibDynamic(n - 1) + getFibDynamic(n - 2);

}

}

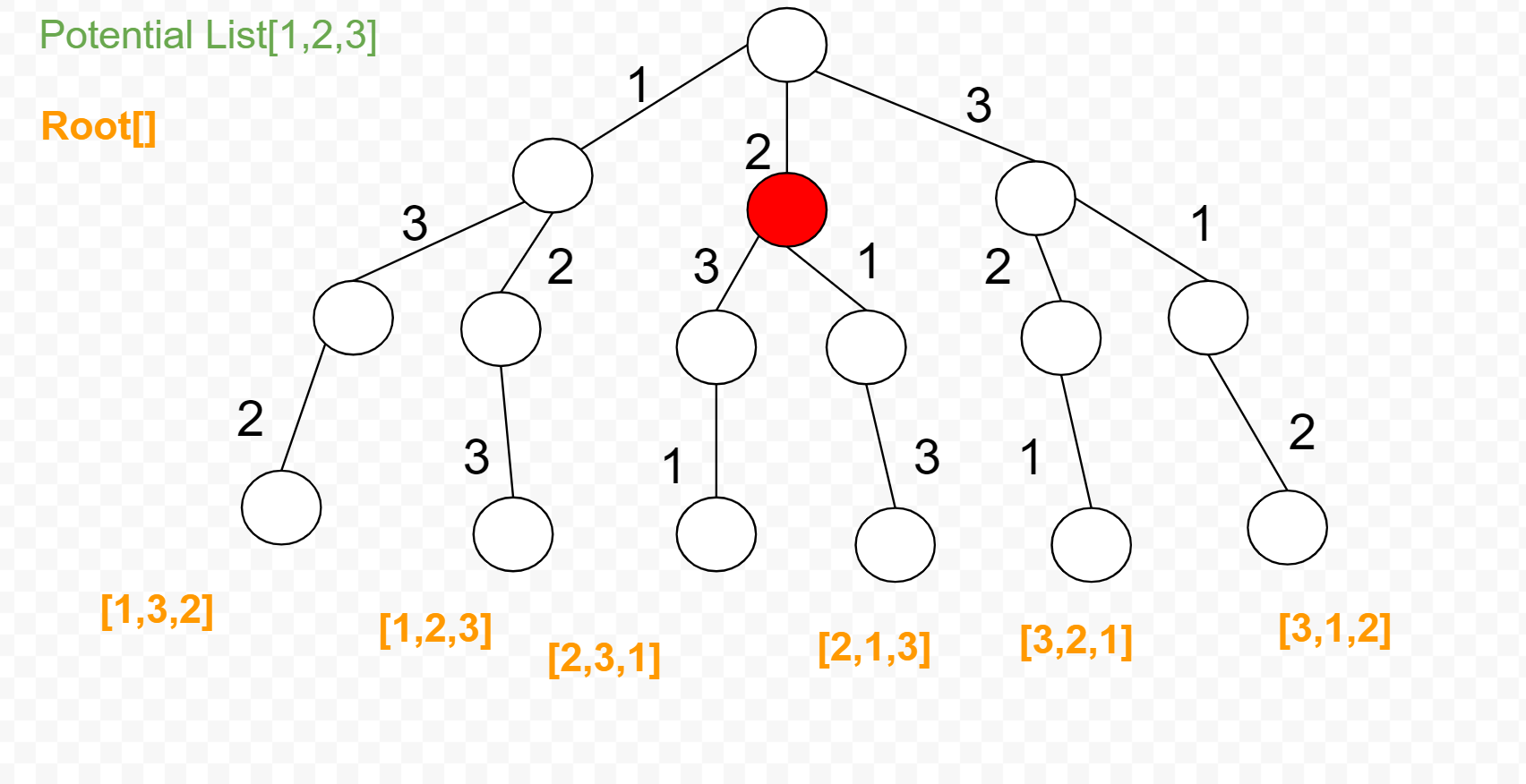
}

We can find the two solutions to get the value of fib, the example above passed 5 into getFib method, in order to get the value, first we need go through fib(4) + fib(3), then fib(3) + fib(2)..etc, the fib(3) will be executed twice, whereas we use a list int[] fib to record the fib value while being executed, like the second method getFibDynamic, which will be save much time to finish the process.

2)

The backtracking algorithm is violent enumeration. All possible calculations are performed at each step, and the calculation returns when the end point is reached. The backtracking algorithm also has the possibility of repeated calculations same as dynamic programming.

In order to better understanding, we can think of backtracking as a permutation problem, for example we decided to find all combinations of list [1,2,3], each element of the new combination should not be same.



We can clearly see from the picture above, each node of the branch is making a decision, for example, the node marked in red, it faces two choices, one is 3 and the other is 1, because the root node is 2, so 2 is no longer Is an option. The backtracking method present the same thing, every decision is a recursion. As long as we make a choice before recursion and cancel the previous choice after recursion, we can get the potential list and root(combinations) correctly.