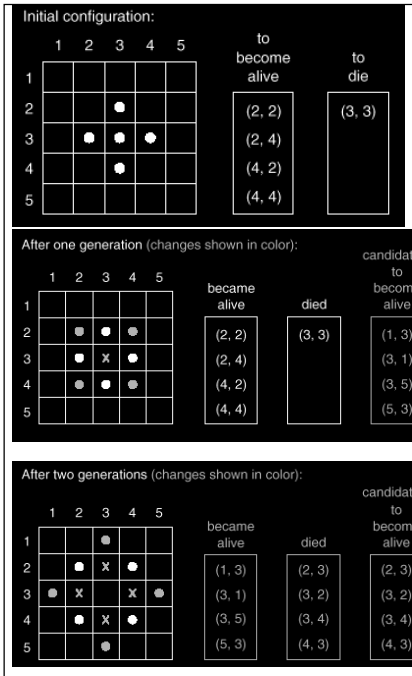


What did we learned in the last class?

Flashback: Two versions of Game of Life..

Lesson from Last Class

Program Analysis and Program Design are closely interrelated. A good computer engineer must know both.

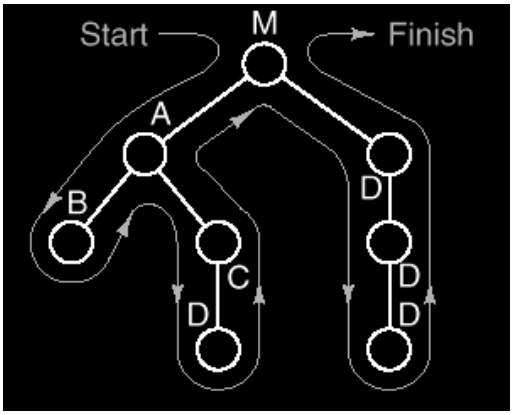
In this course we will learn a host of new powerful programming techniques. Along with we will learn more formal methods for analyzing their performance.

Technique of Recursion

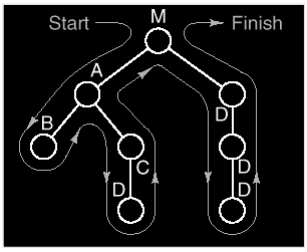
3

Concept of Recursion

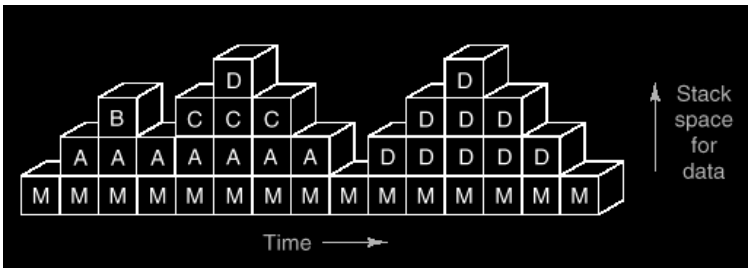
- Let us consider a set of nested subroutines...



Program Stack



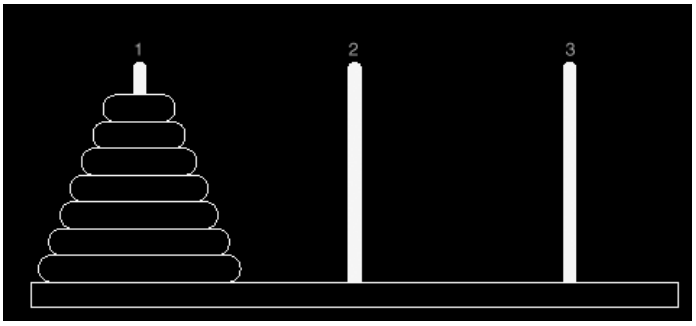
In recursive program, instead of one routine calling a different routine, one routine can repeatedly call itself.



Recursion

- Recursion is a powerful tool which can make the solution of many difficult problem astonishingly easy.
- It is a powerful tool to divide and conquer complex problems.
- However, it is also very important to carefully analyze a recursive solution.
- In this class we will see two examples of recursive solutions, and will learn techniques how to analyze recursive programs.

Tower of Hanoi



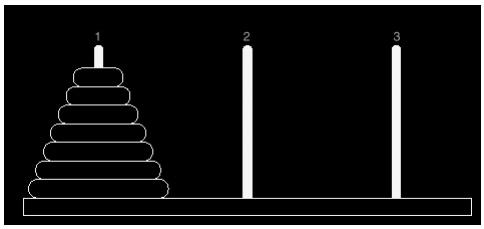
This is task which is underway at the Temple of Brahma. At the creation of the world, the priest were given a brass platform on which were 3 diamond needles. On the first needle were stacked 64 golden disks, each one slightly smaller than the one under it. The priest were assigned the task of moving all the golden disks from the first needle to the third. The end of the task will signify the end of the world.

Solution

- Solution:

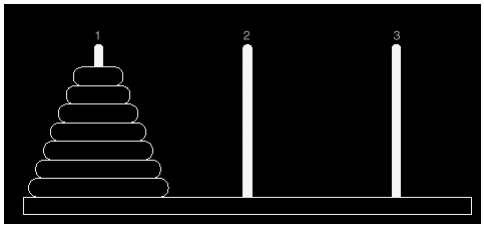
Move(64,1,3,2)

- Meaning: Move 64 disks from tower 1 to tower 3 using tower 2 as temporary.



Solution (Divide and Conquer)

- Step 1:
  - Move (63,1,2,3)
  - printf("Move disk #64 from tower 1 to tower 3\n");
  - Move(63,2,3,1)
- Step 2 ?



Structure of Recursive Program

Every recursive process consists of two parts:

1. A smallest, base case that is processed without recursion; and
2. A general method that reduces a particular case to one or more of the smaller cases, thereby making progress toward eventually reducing the problem all the way to the base case.

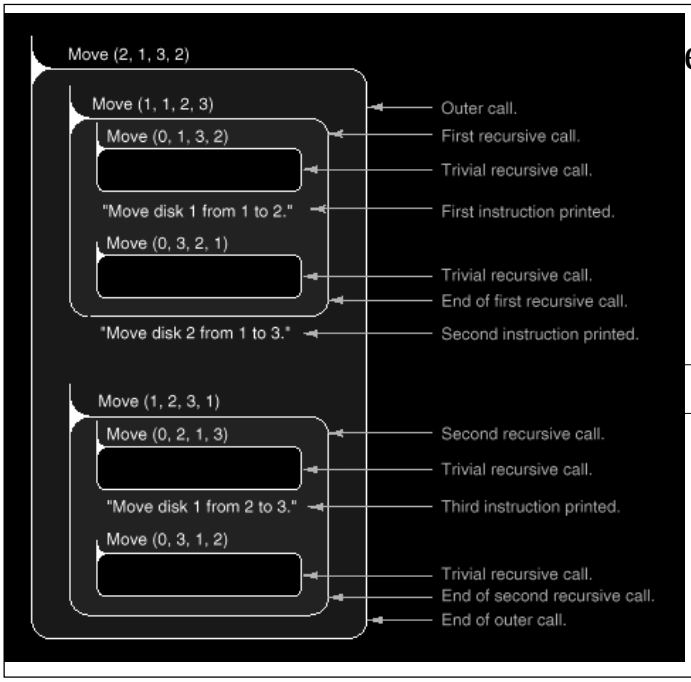
Solution

int Move(int count, int start, int finish, int temp);

Pre: There are at least count disks on the tower start. The top disk (if any) on each of towers temp and finish is larger than any of the top count disks on tower start.

Post: The top count disks on start have been moved to finish; temp (used for temporary storage) has been returned to its starting position.

```
/* Move: moves count disks from start to finish using
temp
for temporary storage. */
void Move(int count, int start, int finish, int temp)
{
    if (count > 0) {
        Move(count-1, start, temp, finish);
        printf("Move a disk from %d to %d.\n", start,
finish);
        Move(count-1, temp, finish, start);
    }
}
```

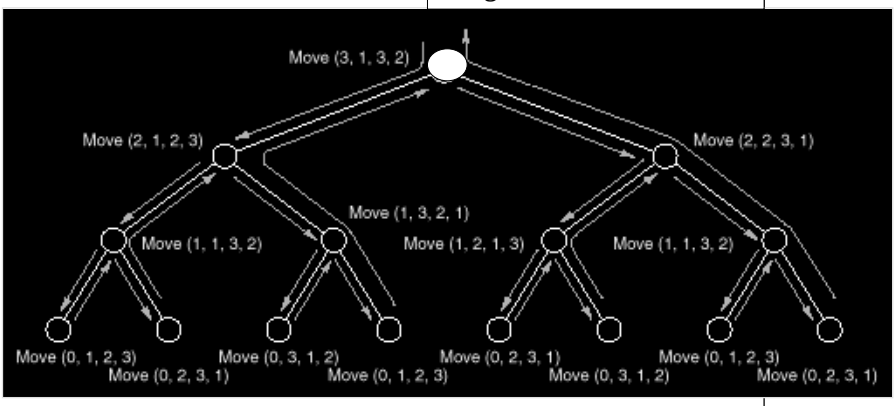


Move(2,1,3,2)

Analysis

- Recursion Tree

Height & Number of Nodes



- Number of Nodes= 1+2+4+…….. $2^{63} = 2^{64} - 1$

(derive on board)

How Large is this number?

- $10^3 \approx 2^{10}$
- Let the priest can perform
  - one move per second then it will take:
  - $2^{64} = 2^4 \cdot 2^{60} = 16 \times 10^{18}$ secs
- There are about:
  - 3.2×10^7 seconds in a year.
  - The life of universe is believed to be 20 billion years.
  - It will take 25 times longer time to complete the task!
- Computers will fail
  - because of time.
  - How much space will be required?