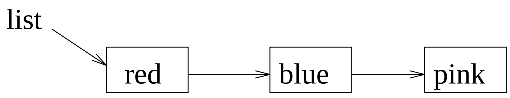
CPS 100 - Program Design and Analysis II
Dr. Susan Rodger
Section: More Recursion (handout)

**Recursion Review**

- clone
- base case
- simpler
- slower

**Example:**




```
void Print(Node * list)
{
   if (list != NULL)
   {
      cout << list->info << " ";
      Print(list ->next);
   }
}
```

What is the output?

What is the output if cout and Print are swapped?

**Example**

```
Node * Copy (Node * list)
// postcondition: returns a copy of the list
{
   if (list == NULL)
      return NULL;
   else
   {
      Node * temp = new Node(list->info);
      temp->next = Copy(list->next);
      return temp;
   }
}
```

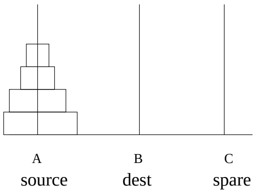
**Towers of Hanoi**

N≥1 blocks on rod A.

Move the N blocks to rod B.

Rules:

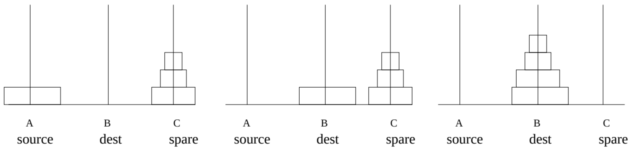
- Can't put large block on a small block.
- Move one block at a time.
- "move" = from one rod to another.




Towers(Count, Source, Dest, Spare)

- Count = number of blocks to move
- Source = The blocks are currently here
- Dest = Want to move all the blocks here
- Spare = The third rod

- Towers(1,A,B,C)
- Towers(N,A,B,C)




Tree of Recursive calls

Trace Towers(6,A,B,C) (partially shown)

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | T(6,A,B,C) | | | |
| | T(5,A,C,B) | | T(1,A,B,C) | | T(5,C,B,A) | |
| T(4,A,B,C) | T(1,A,C,B) | T(4,B,C,A) | | T(4,C,A,B) | T(1,C,B,A) | T(4,A,B,C) |
| ... | | ... | | ... | | ... |

**Code**

```
void Towers(int Count, char Source, char Dest, char Spare)
{
   if (Count == 1)
   {  cout << "Move disk from pole " << Source
           << " to pole " << Dest << endl;
   }
   else
   {  Towers(Count-1, Source, Spare, Dest);  // Point X
      Towers(1, Source, Dest, Spare);        // Point Y
      Towers(Count-1, Spare, Dest, Source);  // Point Z
   }  // end if
}  // end Towers
```

**Cost of Towers of Hanoi**

T(N) is the time it takes to move N blocks from source to dest.

- T(1) =
- T(N) =

- T(N) = 2T(N-1) + 1
  = 2[2 T(N-2) + 1] + 1
  = 2 [2 [2 T(N-3) + 1] + 1] + 1
  = $2^3$ T(N-3) + 4 + 2 + 1
  = ...

**Prove by induction**

T(N) = $2^N - 1$

- Basis: T(1)
- I.H. Assume T(k) = $2^k - 1$ for all $k \leq N$
- I.S.

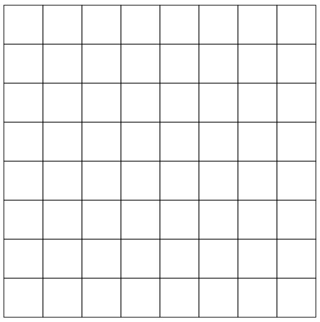
**BackTracking**

Retrace steps in reverse order

**8 Queens Problem**

8x8 chess board

Place each of 8 queens on the board so that none of them can capture another in one move. That means for each queen, there cannot be another queen in the same row, same col, or same diagonal.



**Idea:**

Call: PlaceNextQueen(1)

```
PlaceNextQueen(column)
 row = 1
 if column > 8
   done
 else
   while (row <= 8 and not done yet)
     put Q in next valid row
     PlaceNextQueen(column+1)
     if (not done)
        remove Q in column, row
```

see queen.cc

**Divide and Conquer**

Divide the problem into pieces and conquer the pieces.

Sometimes you need to put the individual solutions together.

**Example: Binary Search**

Divide the problem in half and conquer the appropriate side (recursively).

Here there is no putting back together of pieces. We'll see an example of that when we talk about sorting.