Advanced Recursion

Inst. Nguyễn Minh Huy

Contents

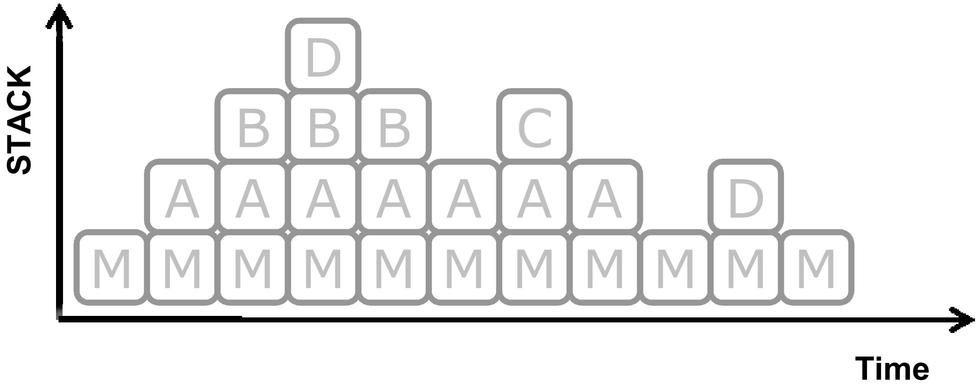
- Recursion analysis.
- Popular recursion problems.

Contents

- **Recursion analysis.**
- Popular recursion problems.

Recursion analysis

- Call stack:
  - Memory stores states of recursive function.
  - State information:
    - Function arguments.
    - Local variables.
    - Current statement.
    - …



```
void main()
{
        A();
        D();
}

void A()                void C()
{                       {
        B();
        C();
}                       }

void B()                void D()
{                       {
        D();
}                       }
```

Recursion analysis

- Stack overflow:
  - Call stack if full.
  - Cannot put more recursive function!!
  - Causes:
    - Do not have base case.
    - Too many recursive calls.
  - Solutions:
    - Use loop.
    - Use user-defined call stack.

Recursion analysis

- Advantages of recursion:
    - Do not look for solution, just define the problem!
    - Make program shorter and easier.
    - Elegant approach.

Recursion analysis

- Dis-advantages of recursion:
  - Stack overflow.
  - Slow performance.
  - Use more resources.
  - Some problems cannot be solved recursively.

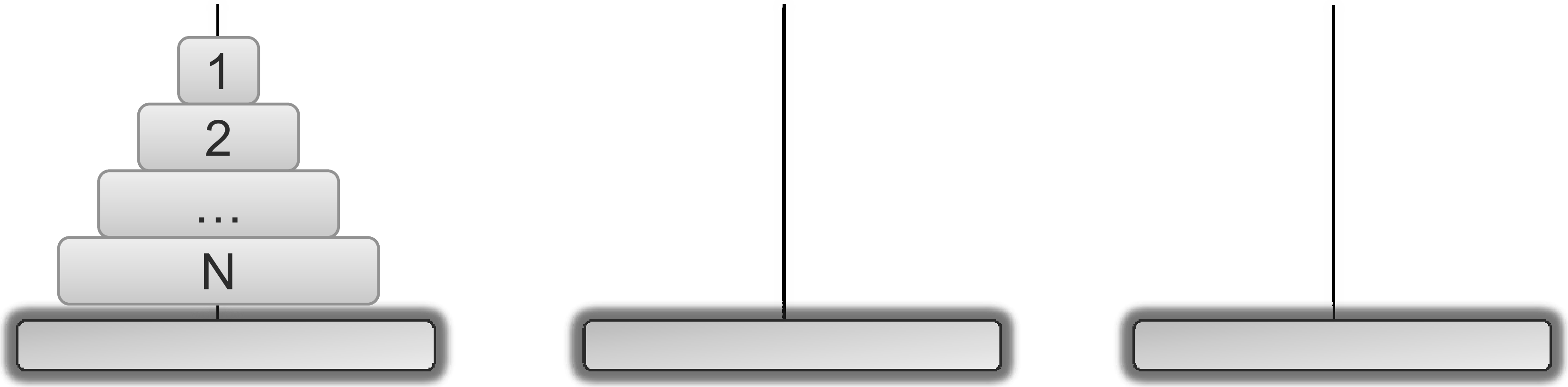
  ➔ Recursion is not “holy grail”!!

Contents

- Recursion analysis.
- **Popular recursion problems.**

Popular recursion problems

- Hà Nội Tower:
  - Problem:
    - There are 3 rods #A, #B, #C.
    - Rod #A contains stack of N disks in ascending order of size.
    - Objective: move disk stack from #A to #C.
      - Move 1 disk at a time.
      - Place smaller disk on top of bigger one.
      - Use #B for temporary rod.



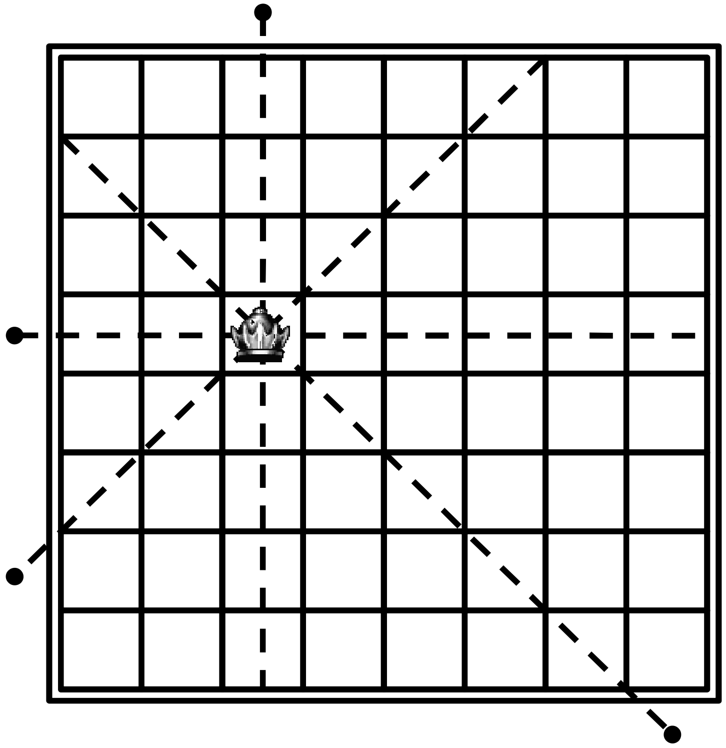
Popular recursion problems

- Hà Nội Tower:
  - Divide-and-conquer:

```
Move( N disks, source #A, dest #C, temp #B )
{
        if ( N == 1 )
                Move top disk #A to #C;
        else
        {       // Split N disks: N – 1 smaller disks and 1 bottom disk.
                Move( N – 1 disks, source #A, dest #B, temp #C );
                Move top disk #A to #C;
                Move( N – 1 disks, source #B, dest #C, temp #A );
        }
}
```

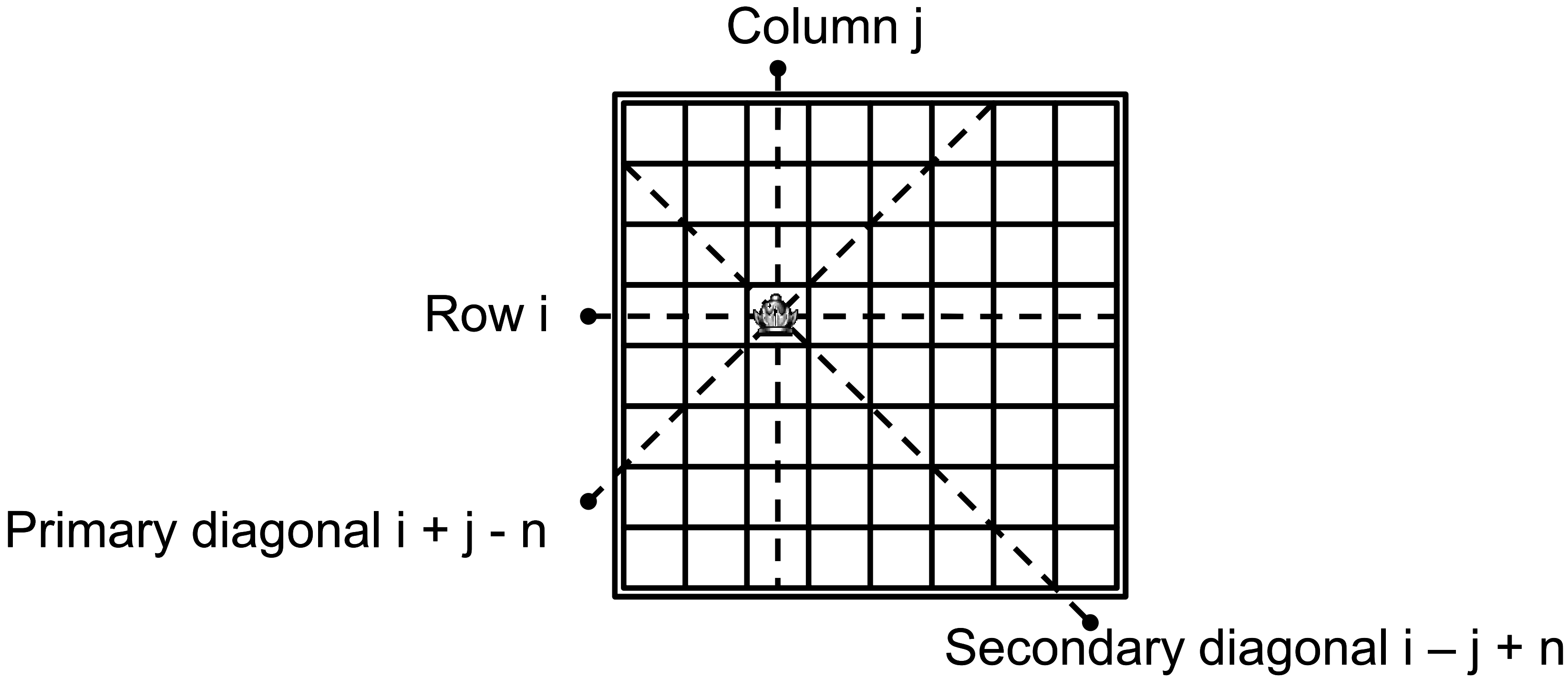
Popular recursion problems

- Eight Queens:
  - Problem:
    - Chessboard 8 x 8 cells.
    - Try to put 8 queens on board.
    - The queens do not capture each other:
      - Not in same row.
      - Not in same column.
      - Not in same primary diagonal.
      - Not in same secondary diagonal.



Popular recursion problems

- Eight Queens:
  - Analysis:
    - Can only put a queen on un-captured cells.
    - If queen is put at (i, j), which cells are captured?



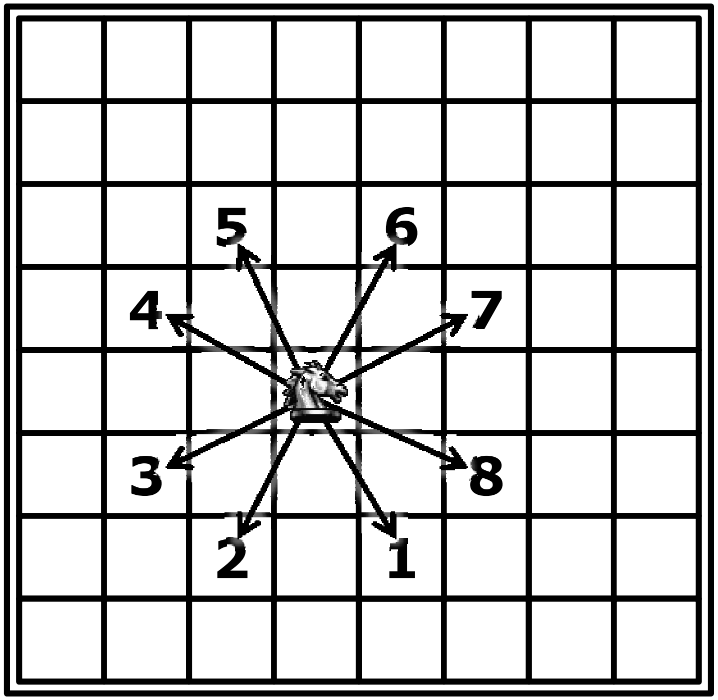
Popular recursion problems

- Eight Queens:
  - Backtracking:

```
TryQueen ( cell (i, j), rowFlag, colFlag, pDiaFlag, sDiaFlag  )
{
    if ( cell (i, j) is captured )
        return;

    Update captures at the cell;

    if ( i is last row )
        Print result;
    else
        for (int k = 0; k < 7; k++)
            TryQueen(cell (i+1, k),rowFlag,colFlag,pDiaFlag, sDiaFlag);

    Roll back captures at the cell;
}
```

Popular recursion problems

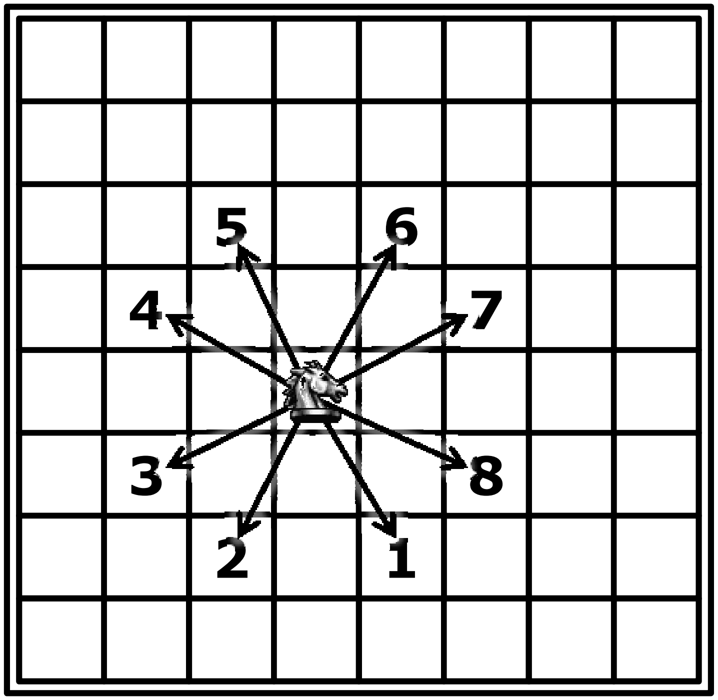
- Knight Route:
  - Problem:
    - Chessboard 8 x 8 cells.
    - Put a knight at a cell.
    - Find route for the knight:
      - Move through all board cells.
      - Stop once at each cells.



Popular recursion problems

- Knight Route:
  - Analysis:
    - Can only move to unoccupied cells.
    - If knight at (i, j), which cells can move next.

1: (i + 2, j + 1)
2: (i + 2, j – 1)
3: (i + 1, j – 2)
4: (i – 1, j – 2)
5: (i – 2, j – 1)
6: (i – 2, j + 1)
7: (i – 1, j + 2)
8: (i + 1, j + 2)



Popular recursion problems

- Knight Route:
  - Backtracking:

```
TryKnight( cell (i, j), board state, step )
{
    if ( cell (i, j) is occupied )
        return;

    Update board state;

    if ( is last step )
        Print result;
    else
        TryKnight( cell (i + 2, j + 1), board state, step + 1 );
        TryKnight( cell (i + 2, j – 2), board state, step + 1 );
        …
    Roll back board state;
}
```