Massachusetts Institute of Technology
Department of Mechanical Engineering

2.003J/1.053J Dynamics & Control I

Fall 2007

Homework 5 Solution

Problem 5.1 : Calculating the factorial of non-negative integer with recursion

$n!$ is associated with $(n-1)!$ as below:

$$n! = n\times(n-1)\times\cdots\times 2\times 1 = n\times(n-1)!$$

Therefore, the function of calculating $n!$ is achieved by calling itself for $(n-1)!$ and multiplying n. m-file is shown as below:

```
function o=fctrlrc(n)
%
% Problem 5.1 : Calculating the factorial of non-negative integer with
recursion
%

if n==0
   o=1;          % fctrlrc(0)=1
else
   o=n*fctrlrc(n-1); % Use recursion
end
```

The output of $125!$ will be displayed as below.

```
>> fctrlrc(125)     % Calculate 125!
ans =
  1.8827e+209
```

Problem 5.2 : Developing the Algorithm for the Solution of the Tower of Hanoi

i) Now, you have 2 different size disks, the small and the big one, in the tower of Hanoi. First, top disk (the small one) on the starting post can be moved to either intermediate post or ending post. Since ending post should be reserved for the big disk, the small one is moved to intermediate post (from 1 to 2). Next, remaining disk (the big one) on the staring post should be stacked on the ending post since there is no disk on this post. (from 1 to 3). Finally, the disk placed on the intermediate post is also stacked on the ending post.

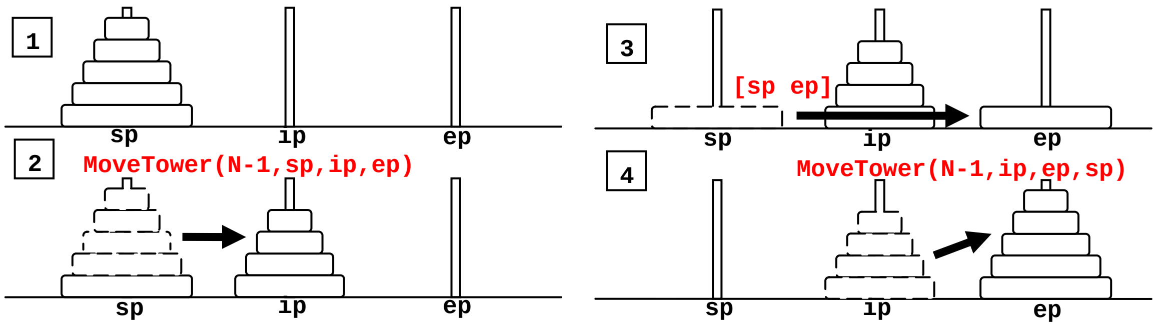
1: from 1 to 2

2: from 1 to 3

3: from 2 to 3

ii) Assume that you have N different size disks for the tower of Hanoi. They are located on the starting post. Now, we define function `MoveTower(N,sp,ep,ip)` to move whole N disks from the starting post (`sp`) to ending post (`ep`) with using intermediate disk (`ip`). This function can be rewritten with `MoveTower` associated with (N-1) disks movement.

To move whole disks from one to another, at least the biggest disk should move from `sp` to `ep`. To achieve this task, (N-1) disks on top of the biggest should move from `sp` to `ip` at first. After moving the biggest disk, (N-1) disk on the intermediate post (`ip`) should be stacked to the ending post (`ep`). This procedure for recursive algorithm is also described in below figures.



In addition to general case, special case N=0 should be considered. Since `MoveTower(0,sp,ep,ip)` means that there is no disk to move, nothing will be done.

Therefore, following description is for the recursive algorithm for the solution of the tower of Hanoi with N different size disks

1. If N is smaller than 1, just return **(in the special case of MoveTower(0,sp,ep,ip))**
2. Output ← MoveTower(N-1,sp,ip,ep) **(move N-1 disks from sp to ip)**
3. Output ← [sp ep] **(move the biggest disk from sp to ep)**
4. Output ← MoveTower(N-1,ip,ep,sp) **(move N-1 disks from ip to ep)**

Problem 5.3 : Solving the Tower of Hanoi with recursive algorithm

By using your own algorithm, the function to solve the tower of Hanoi is shown as below. Please, see the comments for each line.

```
function o=hanoi(n)
%
% Problem 5.2 Solve the tower of Hanoi with n different disks
%           with recursion algorithm
%
% Input argument n: number of disks
% Output argument o: solution matrix
%     The first column: post number which disk moves from
%     The second column: post number which disk moves to
%

% Number all the posts
sp=1;   % starting post are numbered to '1'
ep=3;   % ending post are numbered to '3'
ip=2;   % intermediate post are numbered to '2'

% Calculate Hanoitower wirh reccursion
% Move all the disks on the leftmost side post to the rightmost side post
o=MoveTower(n,sp,ep,ip);
```

```
function o=MoveTower(n,sp,ep,ip)
%
% The solution of whole n disk tower movement one post to the other post
% It can be also expressed with whole (n-1) disk tower movent
%
% n: number of disk in the start post
% a: start post number
% b: end post number
% c: immediate post number

% Define output matrix
o=[];

% if there is no disk, simply return without movement
% o is still empty
if n<=0, return; end

% Move whole disks except for the bottom one
% from starting post to temporary post
o=[o ; MoveTower(n-1,sp,ip,ep)];

% Move the remaining disk on starting post
% to ending post
o=[o ; sp ep];

% Move whole disks on the temporary post
% to ending post
o=[o ; MoveTower(n-1,ip,ep,sp)];
```

The solution of the tower of Hanoi with 5 different size disks is represented as below:

```
>> hanoi(5)
ans =
    1    3
    1    2
    3    2
```

```
     1     3
     2     1
     2     3
     1     3
     1     2
     3     2
     3     1
     2     1
     3     2
     1     3
     1     2
     3     2
     1     3
     2     1
     2     3
     1     3
     2     1
     3     2
     3     1
     2     1
     2     3
     1     3
     1     2
     3     2
     1     3
     2     1
     2     3
     1     3
```

Problem 5.4 : (Optional) Comparison between solving problem with and without recursion.

Two m-code works same in terms of the output of calculating the factorials. However, they are not quite different in the view of computer and programming performance. First, the program with the recursive algorithm is very concise, and easy to understand. Recursive algorithm requires so many function calls, and it needs stack memories to keep current position of program counter and some variables for each function call. In addition, function call needs much more

CPU time than normal operations. Eventually, running the code with recursion gives the value too late where large number of recursion is necessary. In case of programming without recursion, program itself can be a little complicated, but it runs faster than recursive version. However, it is sometimes much more useful to solve the problem recursively than sequentially like the tower of Hanoi.