Dual Hanoi Problem

Shalin Patel

3 December, 2018

1 Preliminary Notions

We begin with the assumption that the colors of the disks involved are white and black for simplicity. Then the first peg will designated as the black goal and the second peg will be designated the white goal with the final peg being referred to as the aux peg. This problem can be broken into a few steps that together will form the bulk of the algorithm.

The algorithm can be broken into a few subparts. The first is the double tower phase. In this part of the algorithm, we take a tower with $2n$ disks such that the first two are the same size, the next two are of the same size, and so on until all $2n$ are accounted for. The next phase is the merge problem in which two stacks of size n are to be merged into a single stack. These two stacks are identical when compared element-wise except for their coloring. Finally is the split phase in which a stack of size $2n$ is split into two stacks of size n that are identical except for the coloring.

2 Double Tower Phase

In this phase, a stack of size $2n$ must be moved from a starting peg to a goal peg. The stack is given such that the colors alternate and that any two disks of the same size must be adjacent to each other. The following algorithm defines this phase as efficiently as possible. The parameter fix is a boolean that will make sure that the polarity of the bases remains the same as the beginning.

Algorithm 1 Double Tower

```
1: procedure DoubleTower(start, goal, aux, n, fix)
2:     if n == 1 then
3:         Move(start, goal)
4:         Move(start, goal)
5:     else
6:         if fix then
7:             DoubleTower(start, goal, aux, n − 1)
8:             Move(start, goal)
9:             Move(start, goal)
10:            DoubleTower(goal, aux, start, n − 1) ▷ Polarity is flipped on the base
11:            DoubleTower(aux, goal, start, n) ▷ Fixes polarity and onto right peg
12:        else
13:            DoubleTower(start, aux, goal, n − 1)
14:            Move(start, goal)
15:            Move(start, goal)
16:            DoubleTower(aux, goal, start, n − 1)
```

Note that in this algorithm, the tower is first moved to the aux peg as the polarity on the base pair is flipped. Then the algorithm is run again to move the stack onto the goal peg as it is able to reverse the polarity again to get it correct.

3 Merge Phase

We can now use the Double Tower algorithm to solve the merge problem. Again, assume that there is a start peg, and we wish to move all the n disks onto the goal peg which has the other n disks that are to be merged with the n on the starting peg. The algorithm is relatively simple in that we first merge $n-1$ disks onto the aux. Finally, the tower on the aux peg is moved back to goal after the base pair is merged on the goal. Note that since two Double Tower operations are done, there is no need to fix the polarity at any step as it will fix itself. It is detailed as follows

Algorithm 2 Merge

```
1: procedure Merge(start, goal, aux, n)
2:     if n == 1 then
3:         Move(start, goal)
4:     else
5:         Merge(start, goal, aux, n − 1)
6:         DoubleTower(goal, aux, start, n − 1, false)
7:         Move(start, goal)
8:         DoubleTower(aux, goal, start, n − 1, false)
```

Note that the merge algorithm is as efficient as possible because it is required to move

$2n-2$ pegs to the aux peg in order to combine the base pairing.

4 Split Phase

This is one of the final steps! In this phase, we take a tower of $2n$ disks as described in the Double Tower phase and want to perform the reverse of the merge phase. In this phase the stack starts on a start peg and we want to move all pegs of the top color to the goal peg with an aux peg, as always. This algorithm is quite simple as well. We know that the top $2n-2$ disks must be moved to the aux peg in order for the base pair to separate. Then this stack can be moved back to the start peg and the split can be executed again. Note that since two Double Tower operations are done, there is no need to fix the polarity at any step as it will fix itself. This is detailed in the algorithm below.

Algorithm 3 Split

```
1: procedure SPLIT(start, goal, aux, n)
2:     if n == 1 then
3:         MOVE(start, goal)
4:     else
5:         DOUBLETOWER(start, aux, goal, n − 1, false)
6:         MOVE(start, goal)
7:         DOUBLETOWER(aux, start, goal, n − 1, false)
8:         SPLIT(start, goal, aux, n − 1)
```

Note that a procedure in which the split takes place on the aux peg could work, but this would mean that $n-1$ disks would need to move from aux to start and this procedure would be more inefficient as it requires the use of merge and double tower. In this algorithm, we only have to use double tower once, whereas the other would require multiple uses.

5 The Final Solution

The following is the final algorithm for the dual hanoi problem. To see the solution, we begin by noting that the bottom disks must be swapped. Hence, the other $2n-2$ disks must first be merged. Then the bottom disk of the start peg will move to the aux peg. At this point, using double tower, the stack of $2n-2$ disks are moved to the aux peg. The disk at goal will move to the start and the $2n-2$ tower will move to the start peg. Then the base disk at the aux peg moves to the goal. Finally, the split can be performed. This is detailed in the algorithm below.

Algorithm 4 DualHanoi

```
1: procedure DualHanoi(start, goal, aux, n)
2:     if n == 1 then
3:         Move(start, aux)
4:         Move(goal, start)
5:         Move(aux, goal)
6:     else
7:         Merge(start, goal, aux, n − 1)
8:         Move(start, aux)
9:         DoubleTower(goal, aux, start, n − 1, false)        ▷ Polarity flipped
10:        Move(goal, start)
11:        DoubleTower(aux, start, goal, n − 1, false)        ▷ Polarity corrected
12:        Move(aux, goal)
13:        DoubleTower(start, goal, aux, n − 1, true)         ▷ Polarity maintained
14:        Split(goal, start, aux, n − 1)
```

With this the problem should be solved as efficiently as possible.