COL 202: DISCRETE MATHEMATICAL STRUCTURES

LECTURE 35

QUIZ 4

APR 12, 2024 | ROHIT VAISH

Problem 1 [24 points]

Consider the following variant of the Towers of Hanoi problem: There are *four* pegs numbered $1, 2, 3, 4$ from left to right. There are n discs of distinct sizes arranged on peg 1 in increasing order of their sizes from top to bottom. The objective is to move all n discs to peg 4 while following the same rules as in the three-peg problem, namely,

- only the topmost disc on each peg can be moved, and
- a larger disc cannot be placed on top of a smaller disc.

Your goal is to design an algorithm for the four-peg problem that is *asymptotically faster* than the three-peg algorithm discussed in class. You may find it helpful to read the problem statements of all three parts (a), (b), and (c) before starting to write your solution.

PROBLEM 1(a)

(a) **[10 points]** Describe your algorithm for the four-peg problem.

For $n=1$, a single move suffices. (peg 1 $\longrightarrow$ peg 4)

For $n=2$, three moves suffice

small disc : peg 1 $\longrightarrow$ peg 2

large disc : peg 1 $\longrightarrow$ peg 4

small disc : peg 2 $\longrightarrow$ peg 4.

PROBLEM 1(a)

(a) **[10 points]** Describe your algorithm for the four-peg problem.

For $n \geq 3$, fix $1 < k < n$. (We will later show that $k=2$ works.)

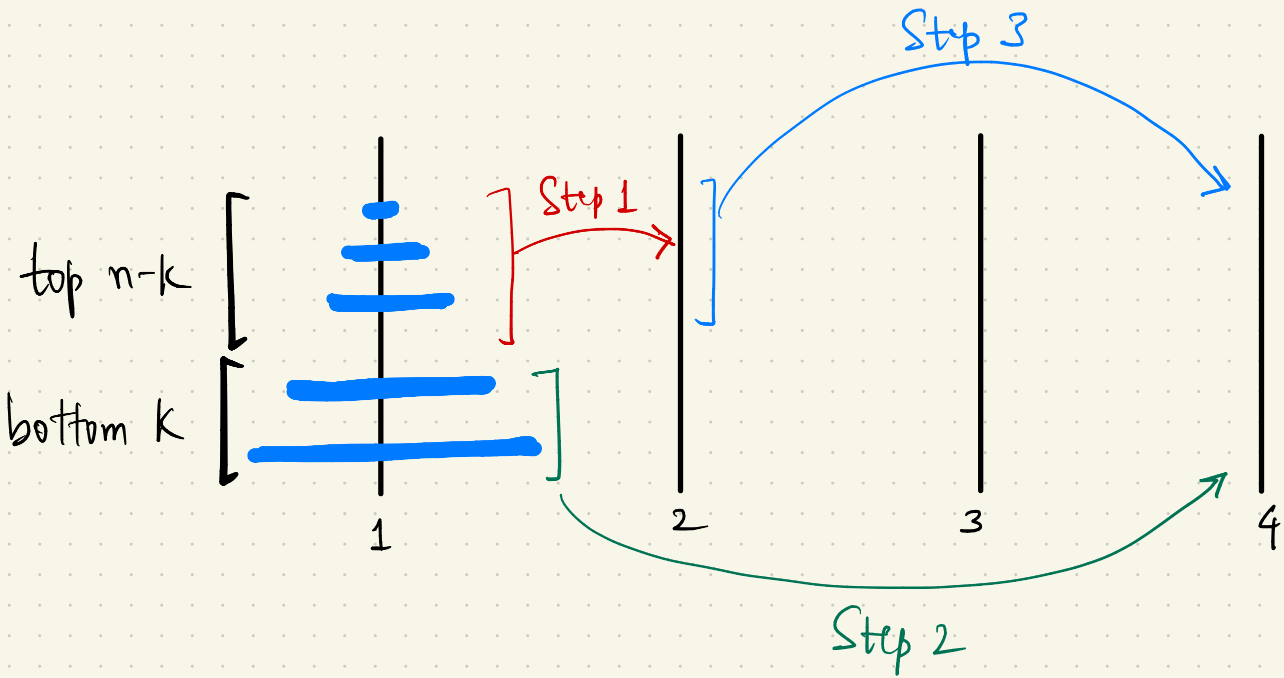
1. Recursively move the top $n-k$ discs from peg 1 to peg 2 (keeping the bottom k discs fixed) allowing the use of all four pegs.

2. Recursively move the k discs on peg 1 to peg 4 by using the three-peg algorithm for pegs 1, 3, and 4.

3. Recursively move the $n-k$ discs on peg 2 to peg 4 (keeping the bottom k discs on peg 4 fixed) using all four pegs.

PROBLEM 1(a)

(a) **[10 points]** Describe your algorithm for the four-peg problem.



PROBLEM 1(a) [10 points]

The algorithm should be well-defined for all $n \geq 0$ —— 2 pts

Clear distinction between steps that use 3 pegs and those that use 4 pegs. —— 4 pts

The algorithm should correctly solve the 4 peg problem —— 4 pts
(a formal proof of correctness is not necessary as long as correctness is evident from the description)*

* In general, you should provide a proof of correctness.

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Let $T_4(n)$ = No. of moves made by the algorithm when starting with n discs on peg 1 in the 4-peg problem.

$T_3(n)$ = ——— 3-peg problem.

We know that $T_3(n) = 2^n - 1$.

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

$$T_4(n) = \underbrace{T_4(n-k)}_{\text{Step 1}} + \underbrace{T_3(k)}_{\text{Step 2}} + \underbrace{T_4(n-k)}_{\text{Step 3}}$$

$$= 2T_4(n-k) + T_3(k)$$

$$= 2T_4(n-k) + 2^k - 1.$$

Note that $k=1$ recovers the 3-peg Towers of Hanoi recurrence.

We need an asymptotically faster algorithm.

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Say $k=2$. By the plug-and-chug method:

$$T_4(n) = 2T_4(n-2) + 2^2 - 1$$

$$\overset{\text{Plug}}{=} 2\left[2T_4(n-4) + 2^2 - 1\right] + 2^2 - 1$$

$$\overset{\text{Chug}}{=} 2^2 T_4(n-4) + 2^3 + 2^2 - 2 - 1$$

$$\overset{\text{Plug}}{=} 2^2\left[2T_4(n-6) + 2^2 - 1\right] + 2^3 + 2^2 - 2 - 1$$

$$\vdots$$

$$= 2^i T_4(n-2i) + 2^{i+1} + 2^i + 2^{i-1} + \cdots 2^2 - 2^{i-1} - 2^{i-2} \cdots - 2^1 - 2^0$$

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

$$T_4(n) = 2^i\, T_4(n-2i) + 2^{i+1} + 2^i + 2^{i-1} + \cdots 2^2 - 2^{i-1} - 2^{i-2} \cdots - 2^1 - 2^0$$

If n is even, $i = n/2 - 1$:

$$T_4(n) = 2^{\frac{n}{2}-1}\, \underbrace{T_4(2)}_{=3} + 2^{n/2} + 2^{n/2-1} + 2^{\frac{n}{2}-2} + \cdots + 2^2 - 2^{\frac{n}{2}-2} \cdots - 2^1 - 2^0$$

$$= \frac{3}{2}\, 2^{n/2} + 2^{n/2} + 2^{n/2-1} - 2^1 - 2^0$$

If n is odd, $i = (n-1)/2$:

$$T_4(n) = 2^{\frac{n-1}{2}}\, \underbrace{T_4(1)}_{=1} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} + 2^{\frac{n-3}{2}} + \cdots + 2^2 - 2^{\frac{n-3}{2}} \cdots - 2^1 - 2^0$$

$$= 2^{\frac{n-1}{2}} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 2^1 - 2^0$$

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Verify: $T_4(n) = \frac{3}{2} 2^{n/2} + 2^{n/2} + 2^{n/2 - 1} - 2^1 - 2^0$ for even n

(by induction) For even n, $P(n)$: $T_4(n) =$ as above.

Base case
$n = 2$

$$T_4(2) = 2T_4(2-2) + 2^2 - 1 = 3 \quad \left(\text{since } T_4(0) = 0\right)$$

$$\frac{3}{2} 2^1 + 2^1 + 2^0 - 2^1 - 2^0 = 3 \quad ✓$$

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Verify: $T_4(n) = \frac{3}{2} 2^{n/2} + 2^{n/2} + 2^{n/2 - 1} - 2^1 - 2^0$

Induction Step. Will show $P(n) \Rightarrow P(n+2)$.

$$T_4(n+2) = 2T_4(n) + 2^2 - 1$$

$$= 2\left[\frac{3}{2} 2^{n/2} + 2^{n/2} + 2^{n/2-1} - 2^1 - 2^0\right] + 2^2 - 1$$

$$= 3 \cdot 2^{n/2} + 2^{\frac{n}{2}+1} + 2^{n/2} - 2^2 - 2^1 + 2^2 - 1$$

which satisfies $P(n+2)$.

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Verify: $T_4(n) = 2^{\frac{n-1}{2}} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 2^1 - 2^0$ for odd n

(by induction) For odd n, $P(n)$: $T_4(n)$ = as above.

Base case
$n = 3$

$T_4(1) = 2T_4(3-2) + 2^2 - 1 = 5$ (since $T_4(1) = 1$)

$2^1 + 2^2 + 2^1 - 2^1 - 2^0 = 5$ ✓

PROBLEM 1(b)

(b) [**12 points**] Derive a bound on the number of moves taken by your algorithm as a function of n. (You do not need to provide a matching lower bound.)

Verify: $T_4(n) = 2^{\frac{n-1}{2}} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 2^1 - 2^0$ for odd n

Induction Step. Will show $P(n) \Rightarrow P(n+2)$.

$$T_4(n+2) = 2T_4(n) + 2^2 - 1$$

$$= 2\left[2^{\frac{n-1}{2}} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 2^1 - 2^0\right] + 2^2 - 1$$

$$= 2^{\frac{n+1}{2}} + 2^{\frac{n+3}{2}} + 2^{\frac{n+1}{2}} - 2^2 - 2^1 + 2^2 - 1$$

which satisfies $P(n+2)$.

PROBLEM 1(b) [12 points]

Correct recurrence —— 4 pts

Guessing closed form via plug-and-chug —— 4 pts

Verification via induction for even n —— 2 pts

Verification via induction for odd n —— 2 pts

PROBLEM 1(c)

(c) **[2 points]** Show that your algorithm is asymptotically faster than the algorithm for the three-peg problem. Specifically, show that the number of moves taken by your algorithm is little-o (o) of that of the three-peg algorithm discussed in class.

For even n, $$\frac{T_4(n)}{T_3(n)} = \frac{\frac{3}{2}\, 2^{n/2} + 2^{n/2} + 2^{n/2-1} - 2^1 - 2^0}{2^n - 1} \longrightarrow 0$$

as $n \to \infty$

For odd n, $$\frac{T_4(n)}{T_3(n)} = \frac{2^{\frac{n-1}{2}} + 2^{\frac{n+1}{2}} + 2^{\frac{n-1}{2}} - 2^1 - 2^0}{2^n - 1} \longrightarrow 0$$

as $n \to \infty$

In both cases, $T_4(n) = o(T_3(n))$.

PROBLEM 1(c) [2 points]

Demonstration for even n ———————— 1 pt

Demonstration for odd n ———————— 1 pt

An elegant solution proposed by Sanjay L (PH1)

1. Recursively move $\lfloor n/2 \rfloor$ discs from peg 1 to peg 2 (keeping the bottom $\lceil \frac{n}{2} \rceil$ discs fixed) using three pegs (1, 2, 3).

2. Recursively move the $\lceil \frac{n}{2} \rceil$ discs on peg 1 to peg 4 by using the three-peg algorithm for pegs 1, 3, and 4.

3. Recursively move the $\lfloor \frac{n}{2} \rfloor$ discs on peg 2 to peg 4 (keeping the bottom $\lceil \frac{n}{2} \rceil$ discs on peg 4 fixed) using three pegs 2, 3, 4.

$$\# \text{ moves} = 2^{\lfloor n/2 \rfloor} - 1 + 2^{\lceil n/2 \rceil} - 1 + 2^{\lfloor n/2 \rfloor} - 1$$