Proof by Induction

Claim: Let L be a list of n-numbers, where $n \geq 0$ and $n = 2^k$ for $k \geq 0$. Mergesort(L, n) requires kn comparisons.

Pf: Note that $T(0) = 1$, $T(1) = 0$ and for $n \geq 2$,

$$T(n) \leq 2T\left(\frac{n}{2}\right) + n.$$

We will prove by Induction on n that

$$T(n) = kn = n \log_2 n.$$

[Note that $n = 2^k \Leftrightarrow k = \log_2 n$]

Base Case: for $n = 0$ and $n = 1$,

$$T(0) = k \cdot 0 = 0$$

$$T(1) = k \cdot 1 = k = 0 \quad \text{as} \quad 1 = 2^0$$

I.H: Let $n \geq 2$ and assume that

$$T\left(\frac{n}{2}\right) \leq \frac{n}{2} \log_2 \left(\frac{n}{2}\right)$$

then

$$T(n) \leq 2T\left(\frac{n}{2}\right) + n$$

apply I.H for $T\left(\frac{n}{2}\right) \leq \frac{n}{2} \log_2 \left(\frac{n}{2}\right)$ as $\frac{n}{2} < n$.

We obtain $T(n) \leq 2T\left(\frac{n}{2}\right) + n$

$$\leq 2\left[\frac{n}{2} \log_2 \left(\frac{n}{2}\right)\right] + n$$

$$= n \log_2 \left(\frac{n}{2}\right) + n$$

$$= n\left[\log_2 n - \log_2 2\right] + n$$

$$= n \log_2 n - n + n \quad \text{as} \quad \log_2 2 = 1.$$

Thus

$$T(n) \leq n \log_2 n = nk. \quad \blacksquare$$

So far we have counted only number of comparisons in mergesort and obtained

$T(n) = 0$ for $n = 0, 1$

$T(n) = n \log n$ for $n = 2^k$, $k \geq 0$.

If we count all the operations performed by Mergesort including forming lists L_1, L_2, remove, append etc.,

all of them can be bounded in terms of number of comparisons

$\Rightarrow$ If $n = 2^k$, $k \geq 0$, the running time of Mergesort(L, n) is $O(n \log n)$.

For general values of n, the recurrence is

$T(n) = 0$ ~~or 1~~ for $n = 0, 1$

$T(n) \leq T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + n \quad \forall n \geq 2$.

and in this case also by induction it can be shown $T(n) = O(n \log n)$.

Theorem: for any list L of n numbers, the running time of algorithm Mergesort(L, n) is $O(n \log n)$. ∎

Set of Full Binary Trees

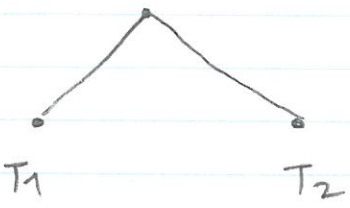
A FBT is defined recursively as

BASIS STEP: There is a FBT consisting only of single vertex r.

Recursive Step: If T_1 and T_2 are disjoint FBTs, then there is a FBT $T_1 \bullet T_2$ consisting of root r together with edges connecting the root to each of the roots of the left tree T_1 and the right tree T_2.

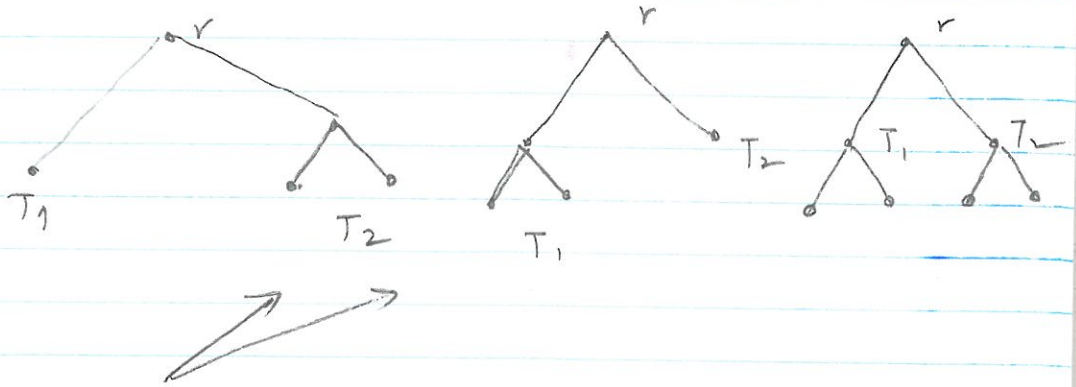
BASIS : • r

Step1 :



← recursion applied once

Step 2:



Trees obtained by applying recursion twice.

Define
$h(T)$ = height of FBT T.

BASIS STEP: the height of FBT consisting of only a root r is $h(T)=0$

Recursive Step: If T_1 & T_2 are FBTs, then

FBT $T = T_1 \cdot T_2$ has $h(T) = 1 + \max\{h(T_1), h(T_2)\}$

Define
$n(T)$ = # vertices in FBT T.

Basis Step: $n(T)$ of FBT consisting of only a root

r is $n(T)=1$.

Recursive Step: If T_1 & T_2 are FBTs, then

vertices in FBT $T = T_1 \cdot T_2$ is

$$n(T) = 1 + n(T_1) + n(T_2).$$

CLAIM: If T is a FBT then

$$n(T) \leq 2^{h(T)+1} - 1$$

Proof: Induction on ?

BASIS: For a FBT consisting of a single node r, $h(T)=0$.

$$n(T) = 1 \leq 2^{h(T)+1} - 1$$

$$= 2^{0+1} - 1 = 2 - 1 = 1$$

Recursive Step:

Let $n \geq 2$, assume it is true for all trees with at most $n-1$ nodes.

Let T_1 and T_2 be trees with at most $n-1$ nodes and we assume $n(T_1) \leq 2^{h(T_1)+1} - 1$

$$n(T_2) = 2^{h(T_2)+1} - 1$$

Then, for $T = T_1 \circ T_2$

$$n(T) = n(T_1) + n(T_2) + 1 \qquad \text{(Definition)}$$

$$\leq \left(2^{h(T_1)+1} - 1\right) + \left(2^{h(T_2)+1} - 1\right) + 1 \qquad \text{(by I.H)}$$

$$= 2^{h(T_1)+1} + 2^{h(T_2)+1} - 1$$

$$\leq 2 \cdot 2^{\max\{h(T_1), h(T_2)\}+1} - 1$$

$$= 2 \cdot 2^{h(T)} - 1$$

as $h(T) = 1 + \max\{h(T_1), h(T_2)\}$

$$= 2^{h(T)+1} - 1$$

∎

Tower of Hanoi

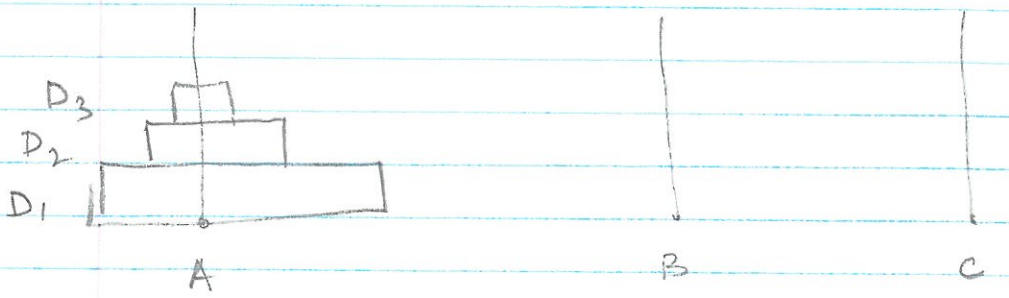
- n-discs of different radii
- 3 pegs : A, B, C.

Initially : All discs arranged in an increasing order of radii in Peg A.

Finally: All discs arranged in an increasing order of radii in Peg B.

Each move consists of moving a disc from one peg to another, ensuring that never a large disk is placed on top of small one.

Here is one way to solve for n = 3.

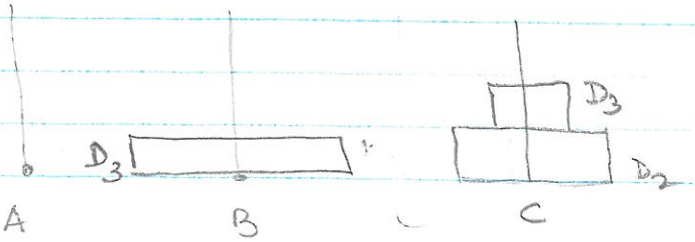


Move 1: Move D_3 to B.

2. Move D_2 to C

3. Move D_3 to C

4. Move D_1 to B



After Move 4, we have a similar problem.

We have to move $n=2$ disks from C to B.

Move 5 : Move D_3 to A

Move 6 : Move D_2 to B

Move 7 : Move D_3 to B.

The total # of moves for $n=3$ disks

$$M_3 = 7$$

Consider the following strategy for $n \geq 2$ disks.

Step 1 : Move the topmost $n-1$ disks from Peg A to Peg C, recursively.

Step 2 : Move the bottommost disk from Peg A to to Peg B in one step.

Step 3 : Move the $n-1$ disks from Peg C to Peg B, recursively.

Then for $n \geq 2$, $M_n = M_{n-1} + 1 + M_{n-1}$

$$M_n = 2M_{n-1} + 1.$$

Claim: $M_n = 2^n - 1$ for $n \geq 1$.

Pf: Base case: $n = 1$ $M_1 = 1$ as one more is enough.

I.H, Let $n \geq 2$ and assume that

$$M_{n-1} = 2^{n-1} - 1$$

Then,

$$M_n = 2M_{n-1} + 1$$

$$= 2\left(2^{n-1} - 1\right) + 1$$

$$= 2^n - 1. \quad \blacksquare$$