GPS 5 due Mon 6th

HW 5 due Tue 7th

Midterm 1 questions + solutions tomorrow.

Algorithms:

correctness > speed

> (space)

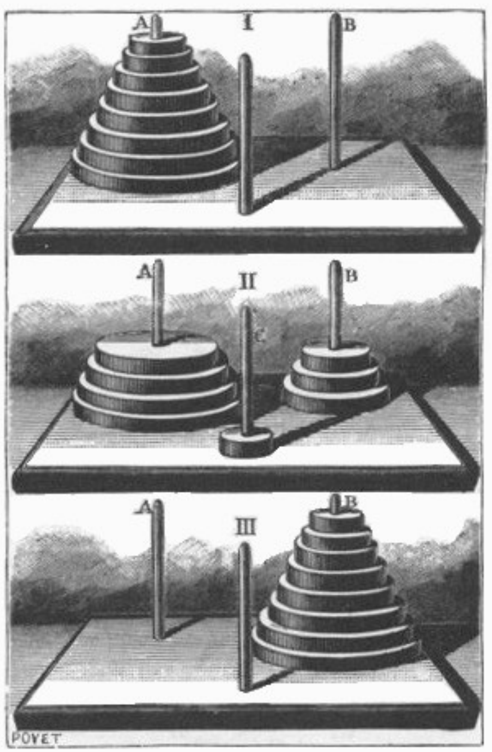
RAM model

Reductions: Solve problem A using by solving some problem B. Don't need to know B's algorithm.

Recursion: Reduce an instance of problem A to a smaller instance of the same problem A.

1883 Lucas: Tower of Hanoi

Fig. 15.



La tour d'Hanoi.

Three pegs.

Stack of disks on one peg: A.

Can move topmost disk of a peg to top of another peg.

Cannot place bigger disk on smaller disk.

Goal: Move stack from A to B

A B C

Can reduce n disks to the smaller n-1 disks.

Move n disks from A to B (via C):

If $n \geq 1$

Move n-1 disks from A to C

Correctness via induction.

Move (biggest) disk n from A to B

Move $n-1$ disks from C to B (via A)

$T(n)$: # moves with n-disk Tower of Hanoi

n	0	1	2	3	4	
$T(n)$	0	1	3	7	15	...

If $n=0$, $T(n)=0$

If $n\geq 1$, $T(n)=2T(n-1)+1$

$T(n)=2^n-1$?

Proof by induction: Let $n\geq 0$. Assume $T(k)=2^k-1$ for $k<n$.

If $n=0$, $T(n)=T(0)=0=2^0-1=2^n-1$ ✓

If $n\geq 1$,
$$\begin{aligned} T(n) &= 2T(n-1)+1 \\ &= 2\cdot(2^{n-1}-1)+1 \\ &= 2^n-1 \quad ✓ \end{aligned}$$

Given $A[1..n]$ of, say, letters.
Want to rearrange A so it is sorted.

von Neumann '45: merge sort

Input:	S	O	R	T	I	N	G	E	X	A	M	P	L
Divide:	S	O	R	T	I	N	G	E	X	A	M	P	L
Recurse Left:	I	N	O	R	S	T	G	E	X	A	M	P	L
Recurse Right:	I	N	O	R	S	T	A	E	G	L	M	P	X
Merge:	A	E	G	I	L	M	N	O	P	R	S	T	X

i j

Sort $A[1..n]$ "in place".

```
MergeSort(A[1..n]):
  if n > 1
      m ← ⌊n/2⌋
      MergeSort(A[1..m])        ⟨⟨Recurse!⟩⟩
      MergeSort(A[m+1..n])      ⟨⟨Recurse!⟩⟩
      Merge(A[1..n], m)
```

Sorts $A[1..n]$ assuming $A[1..m]$ + $A[m+1..n]$ are sorted.

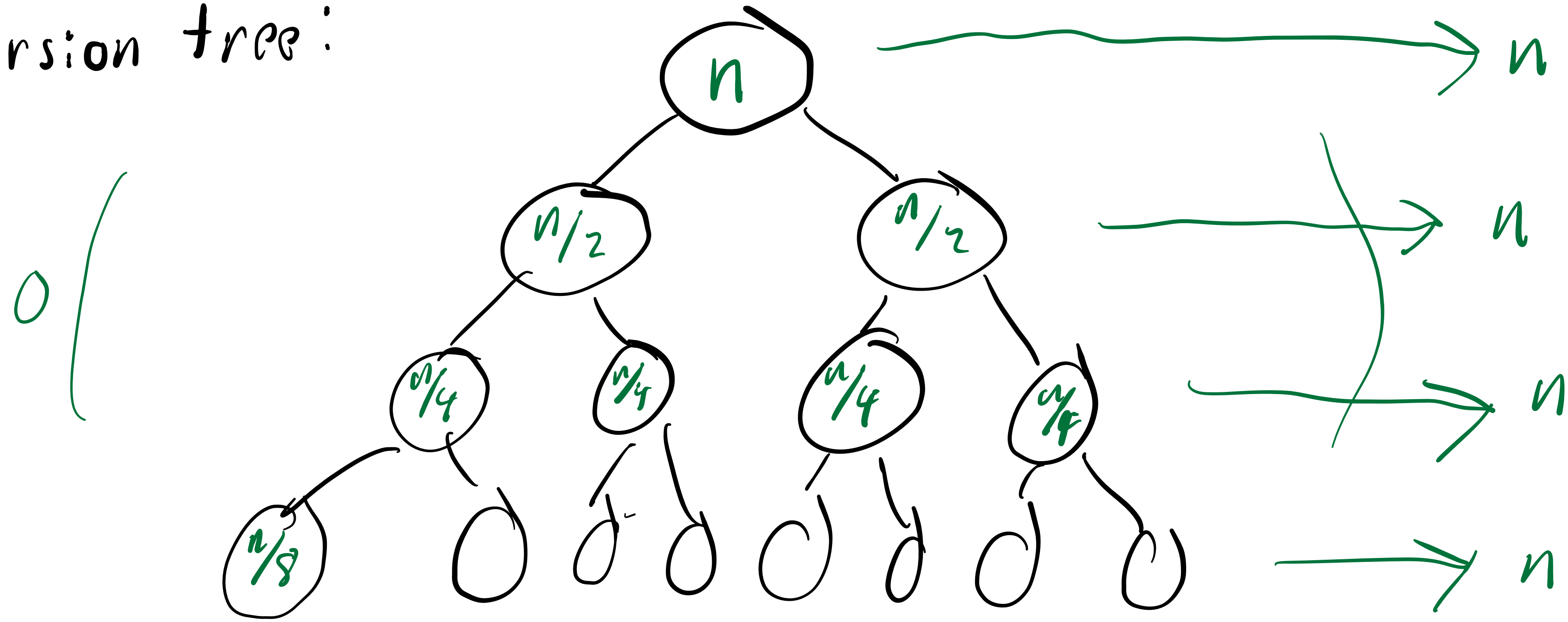
```
Merge(A[1..n], m):
  i ← 1; j ← m+1
  for k ← 1 to n
      if j > n
          B[k] ← A[i]; i ← i+1
      else if i > m
          B[k] ← A[j]; j ← j+1
      else if A[i] < A[j]
          B[k] ← A[i]; i ← i+1
      else
          B[k] ← A[j]; j ← j+1

  for k ← 1 to n
      A[k] ← B[k]
```

$T(n)$: time for n-element merge sort

$$T(n) = O(n) + T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil)$$

Ignore floors & ceilings: $T(n) = 2T(n/2) + O(n)$

Recursion tree:



$T(n)$ = sum of all node values.

depth k: $n/2^k = 1$

$\iff n = 2^k \iff k = \log_2 n$

$T(n) = O(n \log_2 n)$

$\underline{= O(n \log n)}$

Hoare '61: quicksort

Input:	S	O	R	T	I	N	G	E	X	A	M	P	L
Choose a pivot:	S	O	R	T	I	N	G	E	X	A	M	$\boxed{P}$	L
Partition:	A	G	O	E	I	N	L	M	$\boxed{P}$	T	X	S	R
Recurse Left:	A	E	G	I	L	M	N	O	$\boxed{P}$	T	X	S	R
Recurse Right:	A	E	G	I	L	M	N	O	$\boxed{P}$	R	S	T	X

Sort A[1..n] in place.

```
QuickSort(A[1..n]):
  if (n > 1)
      Choose a pivot element A[p]
      r ← Partition(A, p)
      QuickSort(A[1..r − 1])      ⟨⟨Recurse!⟩⟩
      QuickSort(A[r + 1..n])      ⟨⟨Recurse!⟩⟩
```

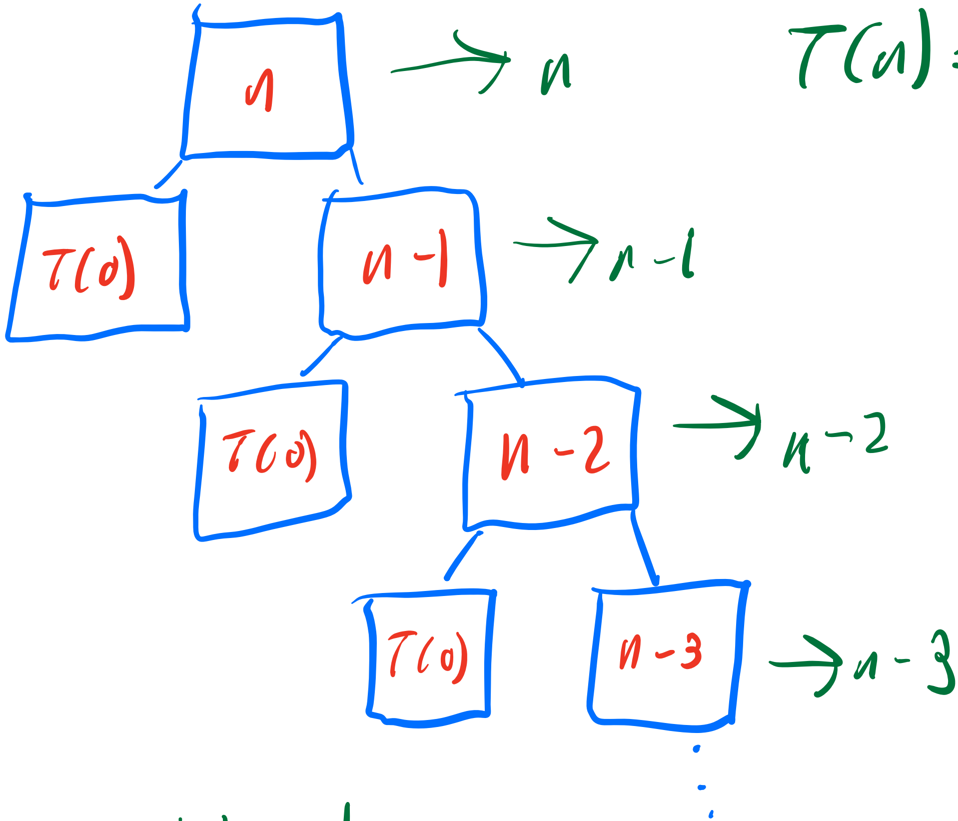
Partitions around A[p] + returns new index of p, its rank.

```
Partition(A[1..n], p):
  swap A[p] ↔ A[n]
  ℓ ← 0                 ⟨⟨#items < pivot⟩⟩
  for i ← 1 to n − 1
      if A[i] < A[n]
          ℓ ← ℓ + 1
          swap A[ℓ] ↔ A[i]
  swap A[n] ↔ A[ℓ + 1]
  return ℓ + 1
```

$$T(n) = \Theta(n) + \max_{1 \le r \le n} \left(T(r-1) + T(n-r) \right)$$

worst-case

If $r = 1$ or n... $T(n) \ge \Omega(n) + T(0) + T(n-1)$



depth k is n

$$T(n) \geq \Omega\left(\sum_{i=0}^{n} n - i\right) = \Omega(n^2)$$

$$T(n) = \Theta(n^2)$$

worst-case

divide-and-conquer:

- divide input into smaller instances that can be solved independently
- delegate to Recursion Fairy
- conquer to build instance solution from recursive