02/13/2019

MIT 6006.

Lecture #3. Sorting

~~1. Insertion Sort~~

Find a median

array $A[0:n] \rightarrow$ $B[0:n]$

Binary Search:

Not obvious apps:
data compression.
computer graphics

1. Insertion Sort.

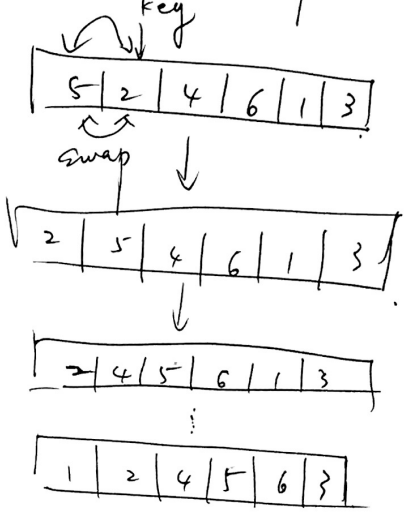
For $i = 1, 2 \dots, n$.

☆ insert $A[i]$ into sorted array $A[0:i-1]$.

by pairwise swaps down to the correct position.

compare >> swap

$O(n^2)$



binary search?

$O(n \log n)$.

asymptotic complexity

O, Ω.

2. Merge Sort.

Divide & Conquer.

e.g. L' R'

20 12

13 11

7 9

2 1

two finger algo

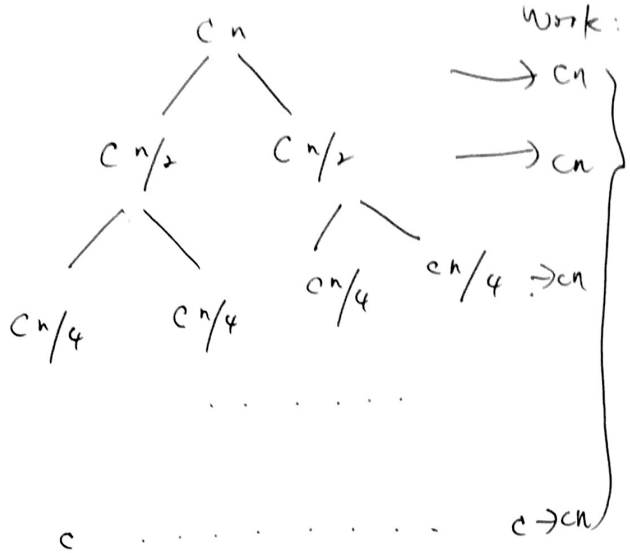
1 2 7 9 11 12 13 20

$O(n \times \log(n))$

it's a recursive algo.

Complexity:

$$T(n) = \underbrace{c_1}_{\text{divide}} + \underbrace{2T(n/2)}_{\text{recursion}} + \underbrace{c \cdot n}_{\text{merge}}$$



$\Rightarrow T(n) = (1+\log n) \cdot cn$

$= O(n \log n)$

What is the advantage of insertion sort over merge sort?

in merge sorting, you have to make some copies of the original array.

↓

$O(n)$.

"trade space for time complexity"

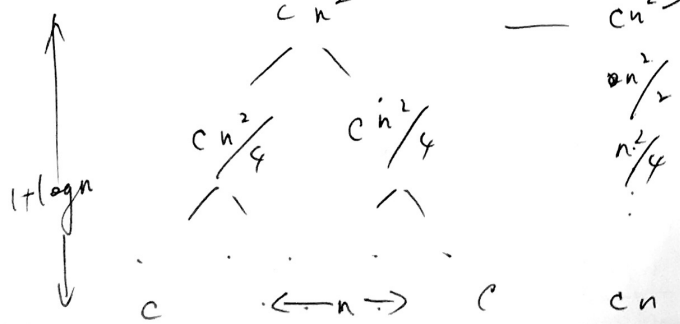
in insertion sorting, only $O(1)$ auxiliary space.

(an in-place sorting)

- merge sort in python: $2.2 n \log n$ μs
- insertion sort in python: $0.2 n^2$ μs.
- C $0.001 n^2$ μs

Recursive works: examples:

① $T(n) = 2T(n/2) + cn^2$



② $T(n) = 2T(n/2) + c$