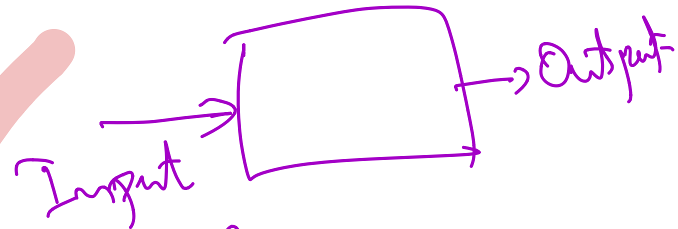
Algorithmic Problems

Table



1. Schedule time-table for courses at IITH.

2. Given a positive integer n, check if n is a prime.

Yes | No

3. Given a sequence of names/numbers, arrange them in lexicographic/non-decreasing order.

→ ordering of items or names,

4. Given two number x and y compute $x \cdot y$: Number.

Input Length

1. Schedule time-table for courses at IITH. (Set of courses, slots and constraints)

2. Given a positive integer n, check if n is a prime. # bits in the binary representation of n.

3. Given a sequence of names/numbers, arrange them in lexicographic/non-decreasing order. # numbers/names.

4. Input x, y ; Output is x.y.
bits in x + # bits in y
or # digits in x + # ~~bits~~ digits in y

5. DFS, BFS : # vertices + # edges.

RAM Model

$n \to$ $T(n)$ - ~~no~~ basic operation it takes.

$S(n)$ - additional space/memory location

1. Random access: Unit cost for read/write of a memory location

2. Each word of data is limited in size (#bits)

3. Each instruction is unit cost: arithmetic operations (+,-,*,/,%), comparison.

$A[i] \leftarrow x$

$a \geq b$

$a \leq b$

$a = b$

How efficient the algorithm when the the input length increases.

$T(n), S(n) \sim$ input length

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

$(n-1)$

$(n-1)$

$(n-1)$

$1 \le \#\text{time} \le i$

$1+2+3+\cdots n = \frac{n(n-1)}{2}$

$n \le \#\text{time} \le \frac{n(n-1)}{2}$

n

- Worst case running time
- Loop invariant

- reverse sorted $\le \frac{3}{2} n^2 \le 10 n^2$
- sorted. $\le n \le 4n$

Input: Given n ~~numbers~~ distinct integers

Output: Sort it in the increasing order...

A[1 . . . . n]

Sorting

Insertion

Input: 5, 8, 0, -3, 100

Output: -3 0 5 8 100

5	8	0	-3	100

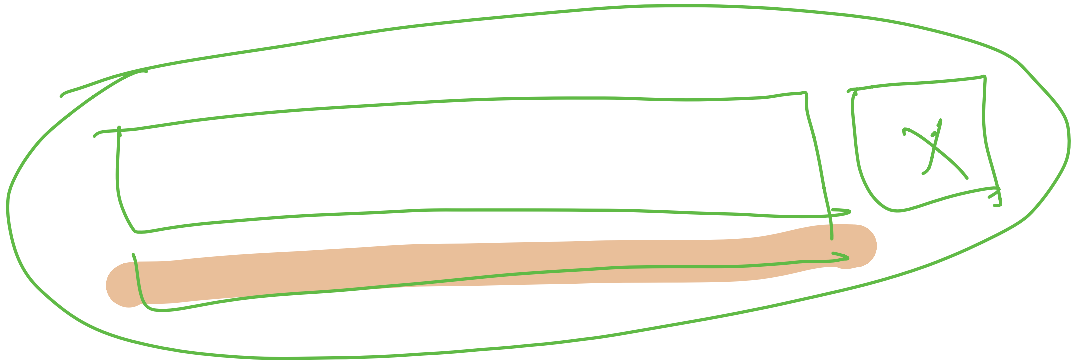
A[. .]

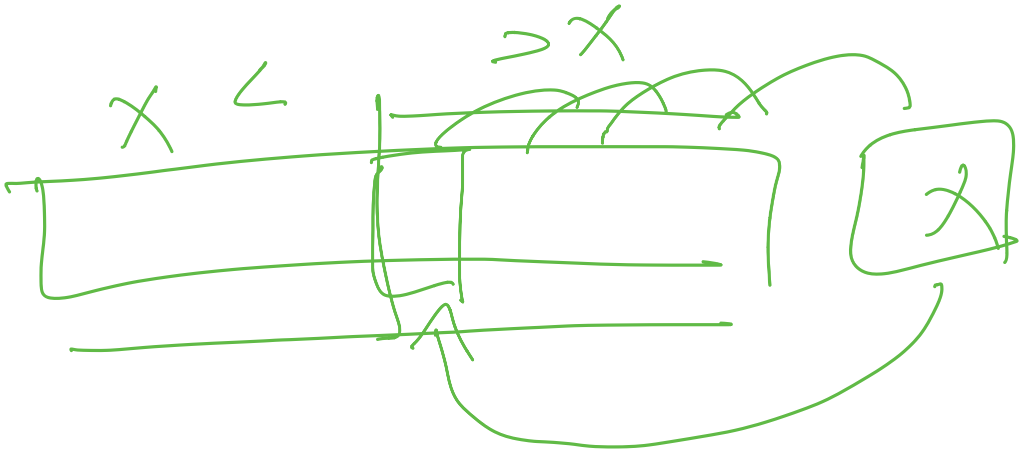
5	8

key ← 0

5	8	8	

0	5	8	





Insertion Sort

- Inverse sorted $3n^2 \le R(n) \le 10n^2$
- Sorted $3n \le R(n) \le 4n$

Worst-case Running time.

$T : \mathbb{N} \to \mathbb{N}$

$T(n)$ = maximum # of steps/basic operation the algorithm takes on an input of length n.

$$\frac{3}{2} n^2 \le T(n) \le 10 n^2$$

$$T(n) = O(n^2)$$

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

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

$$T_{in}(n) \le 10 n^2$$

$$T_{heap}(n) \le 100\, n \log_{10} n$$

$$\frac{10\cdot\left(10^{10}\right)^{2}}{10^{6}} = \frac{10^{21}}{10^{6}} = \underline{\underline{10^{15}\text{ sec}}}$$

$$\frac{10\cancel{0}\cdot 10^{10}\cdot 1\cancel{0}}{10^{\cancel{4}}} = \underline{\underline{10^{9}\text{ sec}}} \checkmark$$

= Worst-case running time.

INSERTION-SORT(A)

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

loop invariant

Given a sorted starts

$A[1 \ldots q]$

$A[1 \ldots q+1]$ — sorted

Loop-invariant.

At the end of q^{th} iteration of the for loop.

$A[1 \ldots q+1]$ is a sorted sequence of the numbers $\{A[1], A[2] \ldots A[q+1]\}$.

Base case

$q=0$. $A[1]$ — sorted.

Induction.

At the end of $(q-1)^{st}$ iteration $A[1 \ldots q]$ is sorted. (I. H)

Want to prove: At the end of q^{th} iteration $A[1 \ldots q+1]$ is sorted.

---

* Input length
* RAM-model
* Worst-case runtime
* Loop invariant | Induction.

Asymptotic analysis.