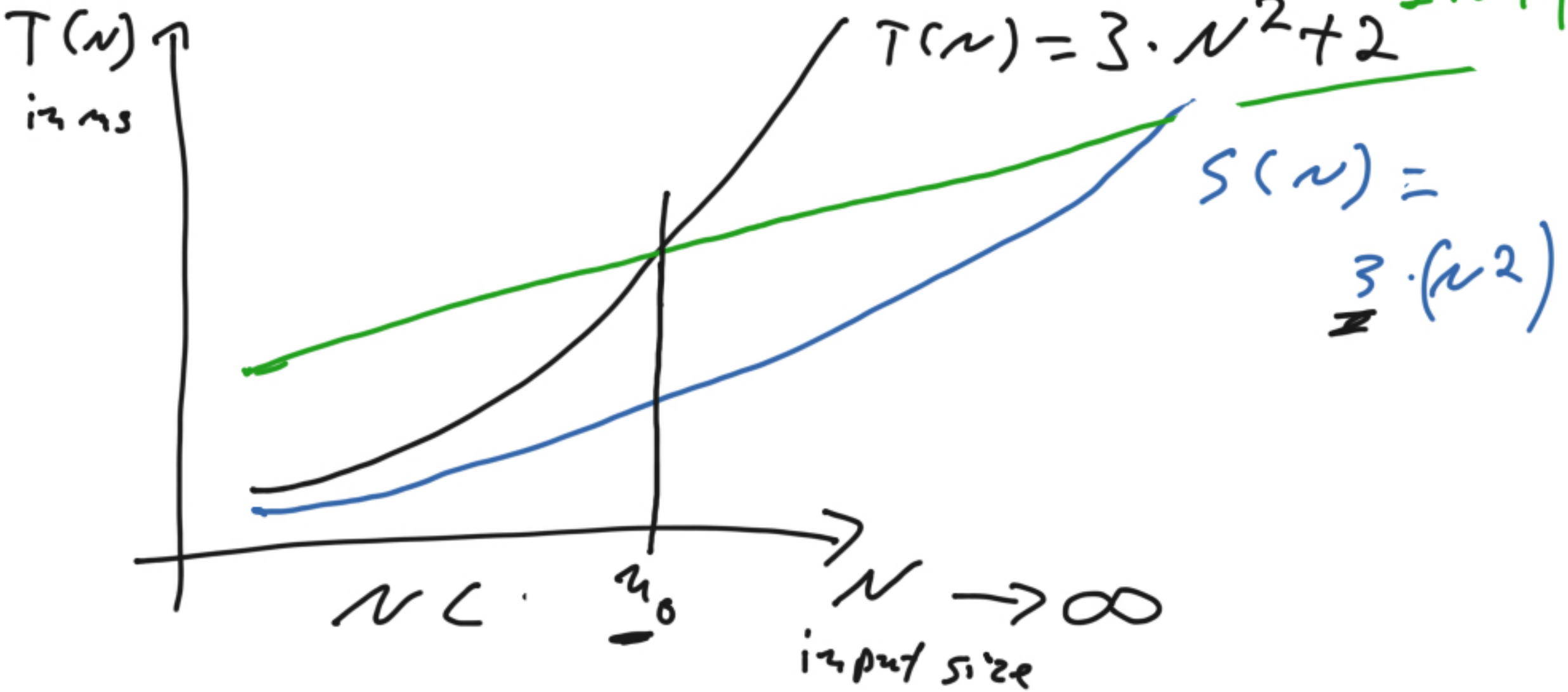
Big-O notation



$R(n) = O(T(n))$

$f(n) = O(g(n))$ if

$f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

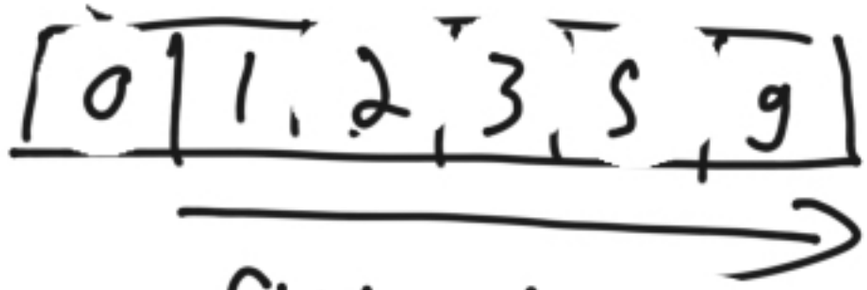
for some constant c and n_0.

$f(n) = \Omega(g(n))$ if

$f(n) \geq c \cdot g(n)$ for all $n \geq n_0$.

Comparison based sorting is $\Omega(n \log n)$

Selection Sort



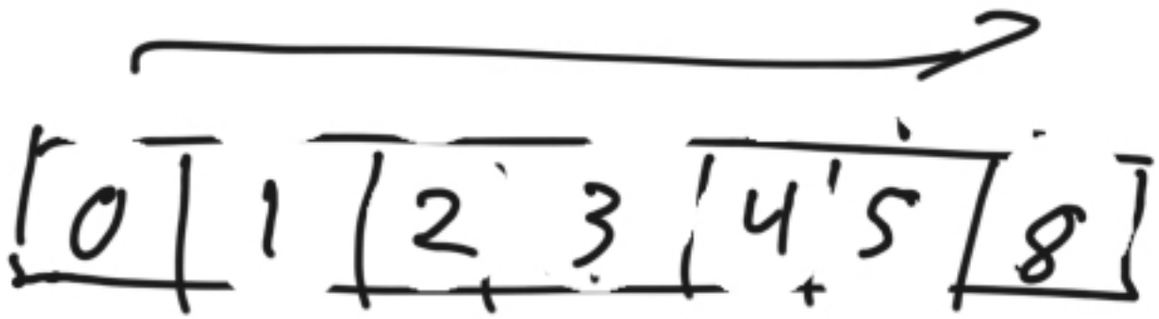

```
for p in range(0... n):
    for i in range(p, n):
        ....
```

find min to the right of p
swap element at p with that min

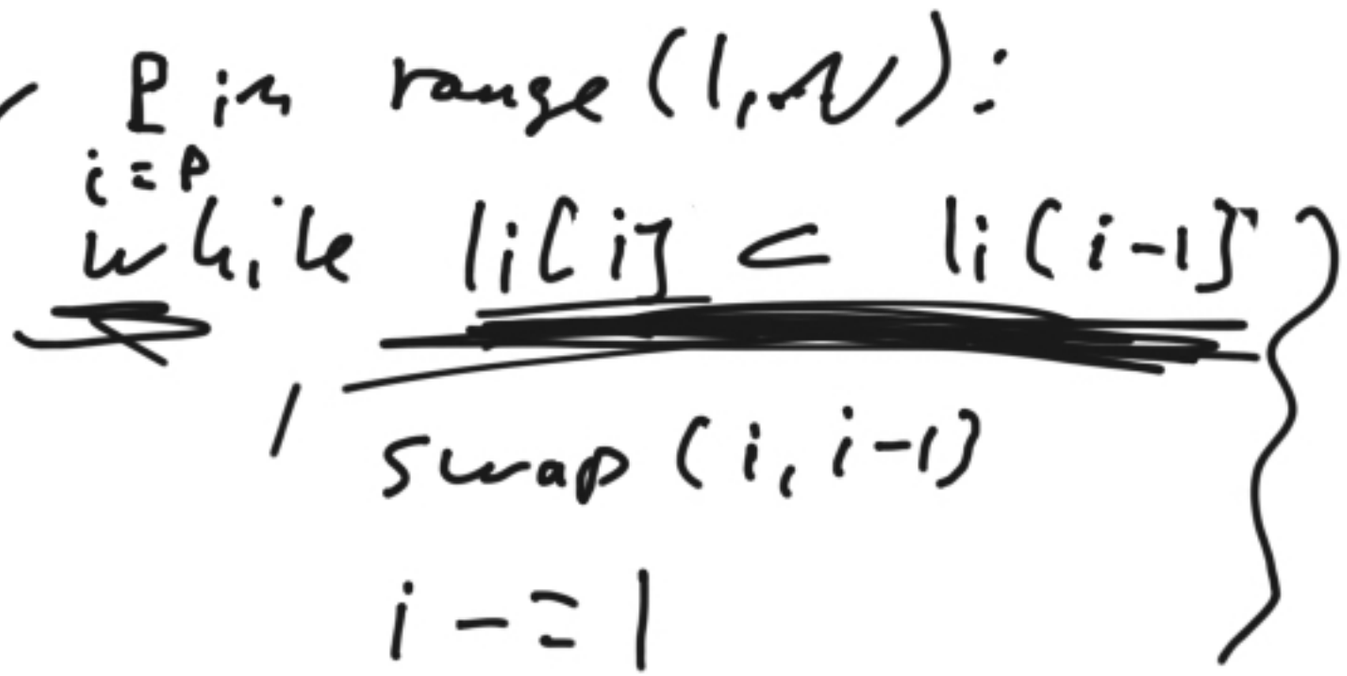
$$\frac{n + (n-1) + (n-2) + \cdots + 1}{\frac{n \cdot (n+1)}{2}} = \frac{n^2 - n}{2}$$

$$= O(n^2)$$

Insertion Sort



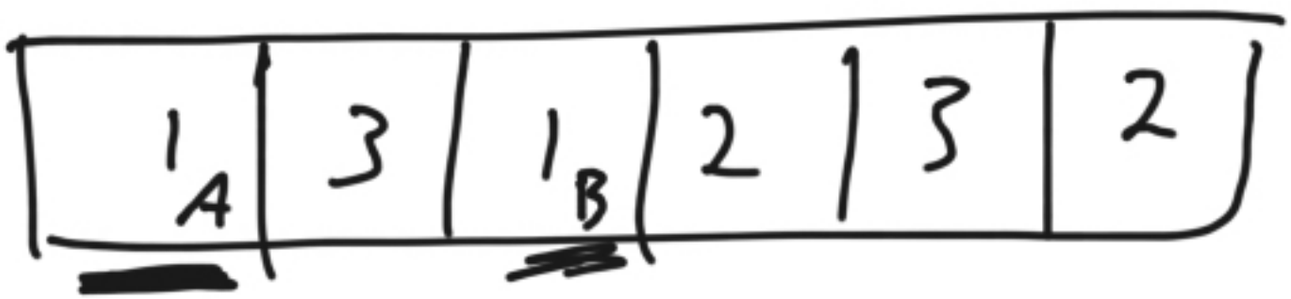
for P in range(1, n):

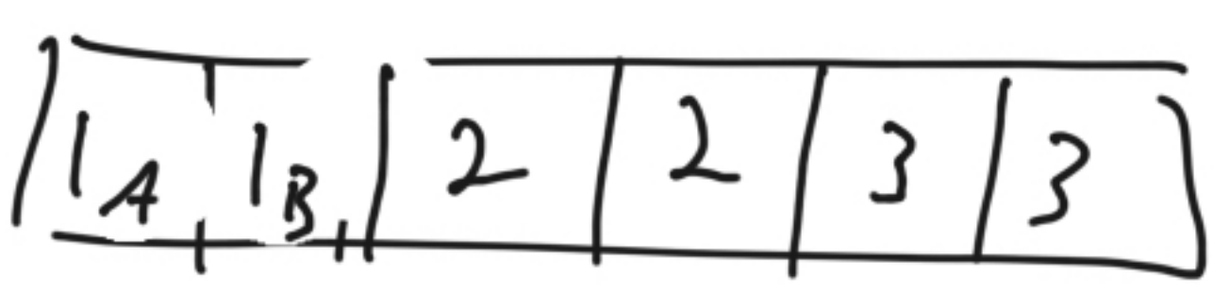


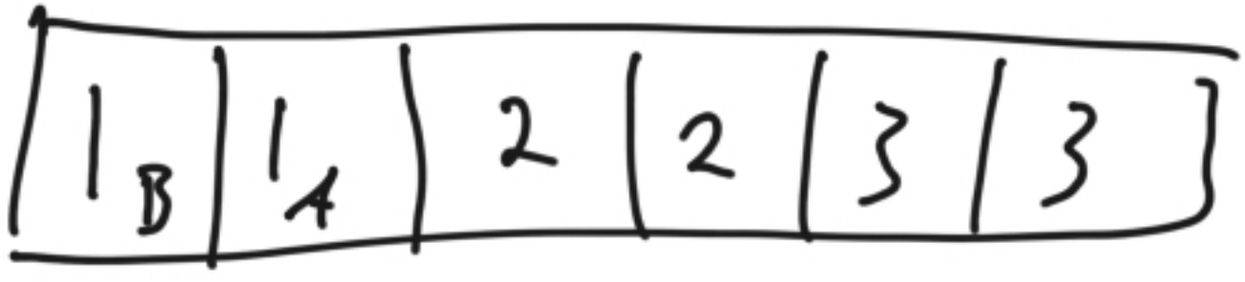
Worst case: $1 + 2 + 3 + \cdots + n = O(n^2)$

Best case: $O(n)$

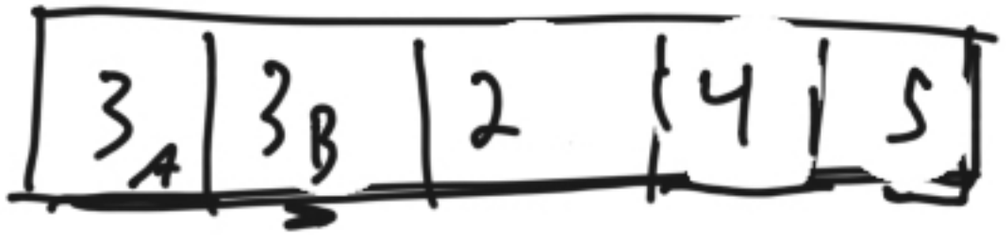
Sorting Stability







Selection Sort is <u>not</u> stable



Stacks and Queues

list: [ ]

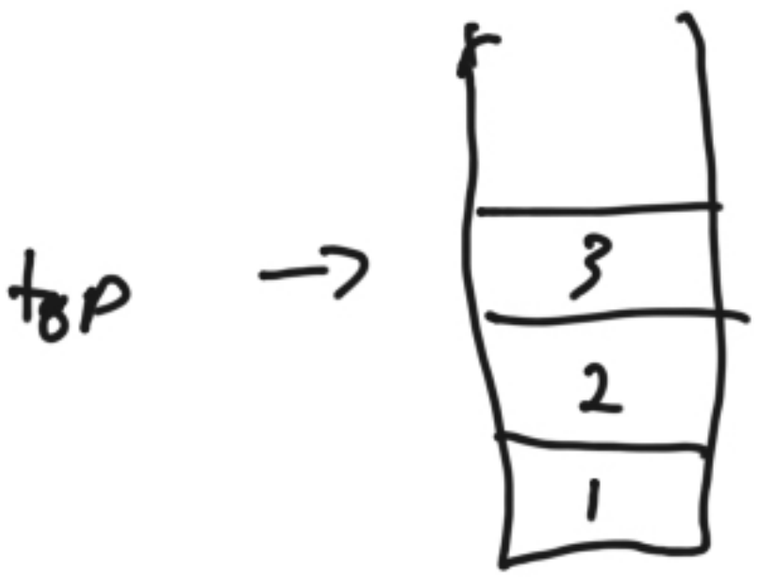
Python lists are arrays!

append, insert(e, x), remove(e)

list[e] = x

get [e]

Stack:



push(element)

pop() -> element

Stacks

Last in first out storage

LIFO

push(1)

push(2)

push(3)

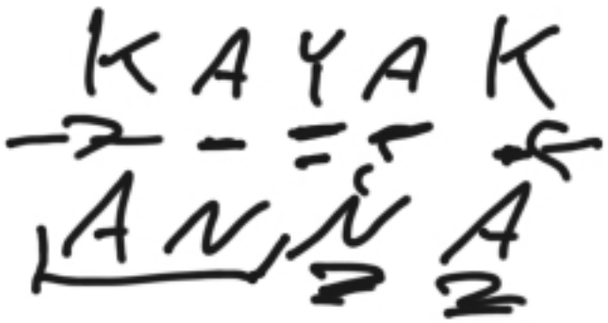
pop() → 3

pop() → 2

pop() → 1

Palindromes

K A Y A K



Pop() → K

Pop() → A

Queues

FIFO storage

First in
First out

enque(x): add x at the end

deque(): returns and removes element at the front

enque(1)
enque(2)
enque(3)

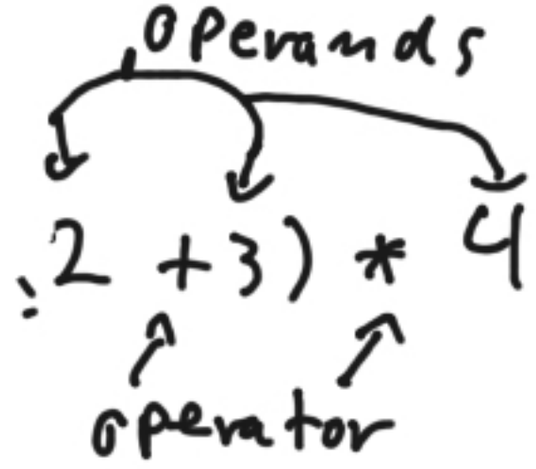
front → 1 → 2 → 3 ← end

deque() → 1

deque() → 2

deque() → 3

Reverse Polish Notation



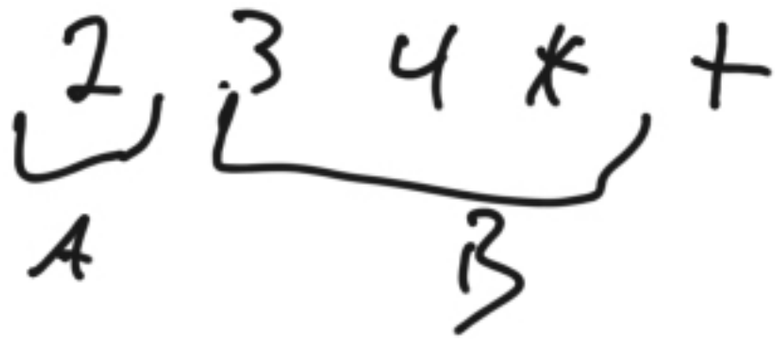
operand$_1$ operator operand$_2$

→ 2 + 3 * 4

2 3 + 4 *

operand$_1$ operand$_2$ operator

2 3 4 * +



Evaluating RPL expressions

S = 2 3 + 4 *
        i

if S[i] is a number:
    stack.push(S[i])
else: # operator
    operand 2 = stack.pop()
    operand 1 = stack.pop()
    stack.push(operator(operand1, operand2))
return stack.pop()