CS 252

M_J 22 April 2024

Python lists

```
a = []
for k in range(1000):
    a.append(k)
```

Does interpreter pre-allocate memory? If so, how many slots?

```
a = [ ]
for k in range(1000):
    a.append(k)
```

Pretend: preallocate 20 slots

← when k=20, we ran out of slots.

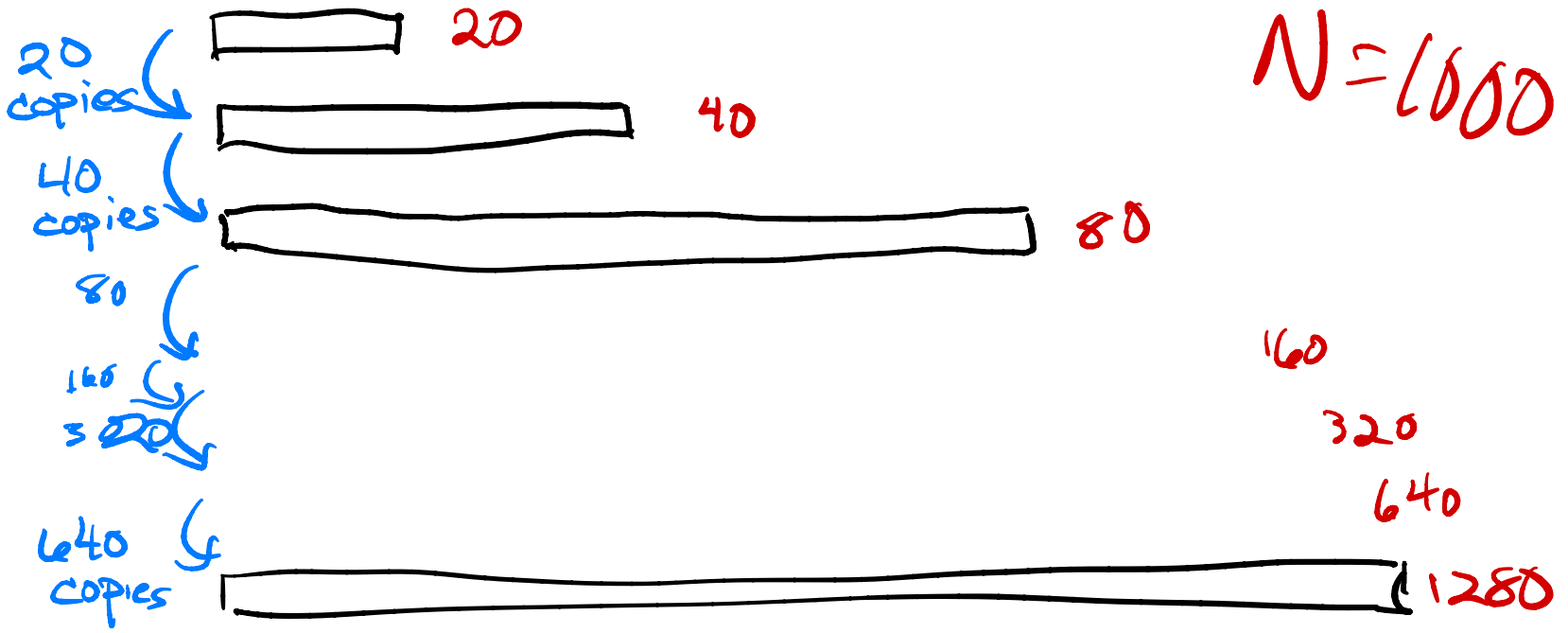
What happens?

Charlie suggests:

① allocate a new array, double-sized

② copy the data from old to new

③ deallocate old



$N = 1000$

Number of rounds $= \lceil \log_2 N/20 \rceil$

$$\#\text{ copy ops} = 20(1 + 2 + 4 + 8 + 16 + 32)$$
$$= 20(2^6 - 1) = 20(2^{\lceil \log_2 N/20 \rceil} - 1)$$

$2^{\lceil \log_2 N/20 \rceil}$ kinda like $2^{\log_2 N/20}$

Total runtime $= N/20$

is $O(N)$

What's the runtime of a.append? It depends.

Usually ~ O(1)

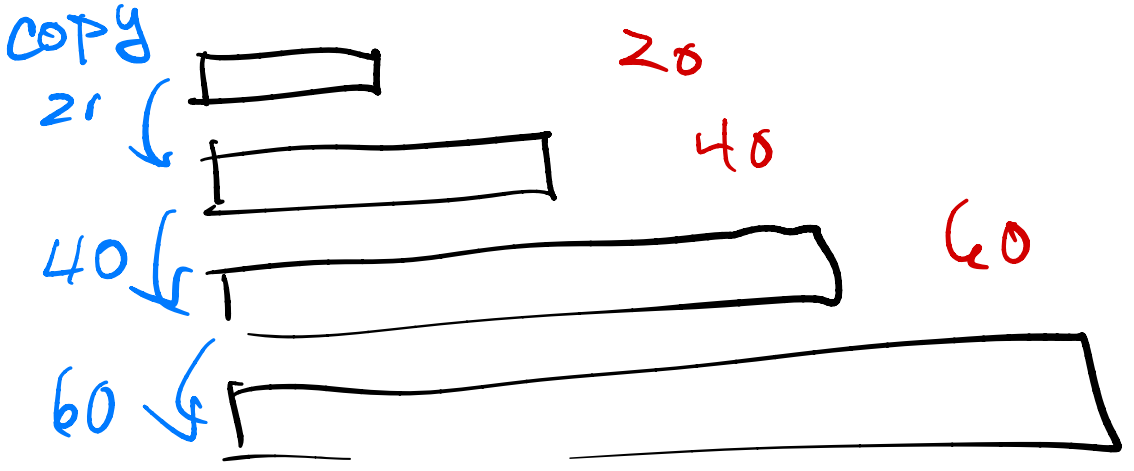
If you need to reallocate + copy O(N)

On average over N calls to append, O(1)

"Amortized constant time"

If we added fixed-size chunks instead of doubling

$O(N^2)$



$$\#\text{copy ops} = 20\left(1 + 2 + 3 + \ldots + \frac{N}{20}\right)$$