Fast data structures and APIs

Jannusch Bigge

19.12.2023 and 09.01.2024

Fast data structures

Fast data structures

By now you should know the following data structures:

- Lists
- Tuples
- Sets
- Dictionaries

Fast data structures

By now you should know the following data structures:

- Lists
- Tuples
- Sets
- Dictionaries

These are all handy, but sometimes you need something special.

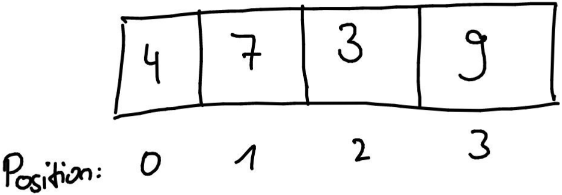
The Problem

The Problem

We have a list of numbers.

The Problem

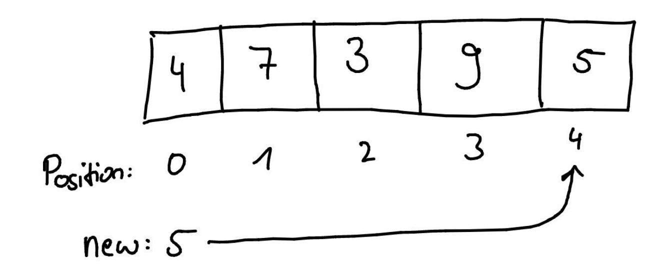
We have a list of numbers.



	4	7	3	9
Position:	0	1	2	3

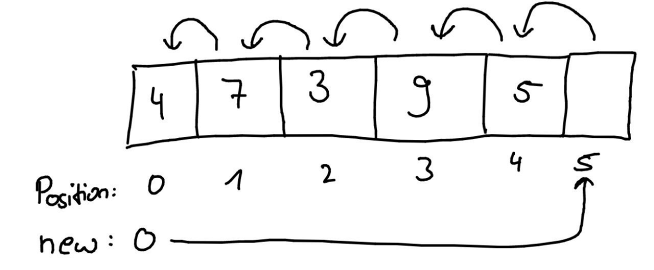
The Problem

We have a list of numbers.
And we want to add something to the end.



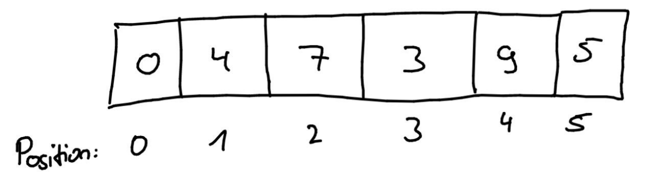
The Problem

We have a list of numbers.
Now we want to add something to the begin.



The Problem

We have a list of numbers.
With some more numbers now.



	0	4	7	3	9	5
Position:	0	1	2	3	4	5

The Problem

Adding something to the end is easy.

The Problem

Adding something to the end is easy.

- We just add it to the end.

The Problem

Adding something to the end is easy.

- We just add it to the end.
- This takes $O(1)$ time.

The Problem

Adding something to the end is easy.

- We just add it to the end.
- This takes $O(1)$ time.

Adding something to the begin is hard.

The Problem

Adding something to the end is easy.

- We just add it to the end.
- This takes $O(1)$ time.

Adding something to the begin is hard.

- We have to move all the other elements.

The Problem

Adding something to the end is easy.

- We just add it to the end.
- This takes $O(1)$ time.

Adding something to the begin is hard.

- We have to move all the other elements.
- This takes $O(n)$ time.

The Problem

Adding something to the end is easy.

- We just add it to the end.
- This takes $O(1)$ time.

Adding something to the begin is hard.

- We have to move all the other elements.
- This takes $O(n)$ time.
- pop(0) is the same.

Just reverse the list, stupid!

deque

There is a data structure that can do both in $O(1)$ time: deque

deque

There is a data structure that can do both in $O(1)$ time: **deque**

- It is short for **double ended queue**.
- It is a list that can be appended and prepended in $O(1)$ time.
- It is implemented as a doubly linked list.
- It is in the **collections** module.
- It is a bit slower than a list.

deque

Some examples:

```
from collections import deque
d = deque(range(1, 5))
d.append(5)     # append to the end
d.appendleft(0) # append to the begin
list(d)         # returns list of deque
d.rotate(1)     # rotate the deque
```

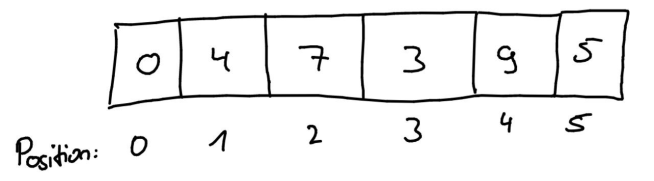
And an other one

New Problem

We have a list of numbers.

New Problem

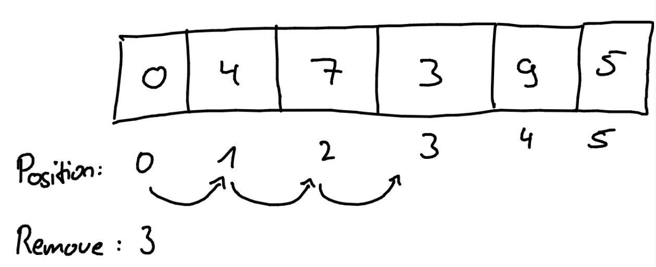
We have a list of numbers.



	0	4	7	3	9	5
Position:	0	1	2	3	4	5

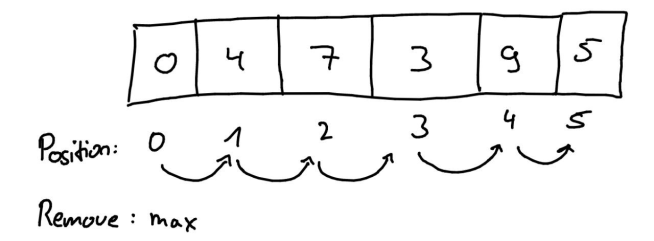
New Problem

We have a list of numbers.
And we want to find 3.



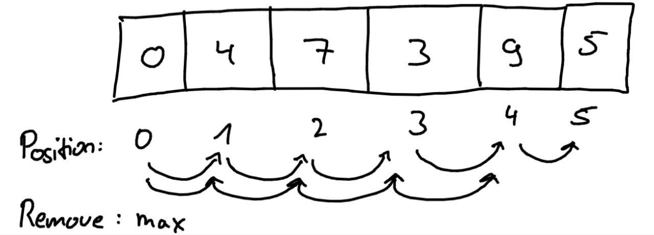
New Problem

We have a list of numbers.
Now we want to find the max element.

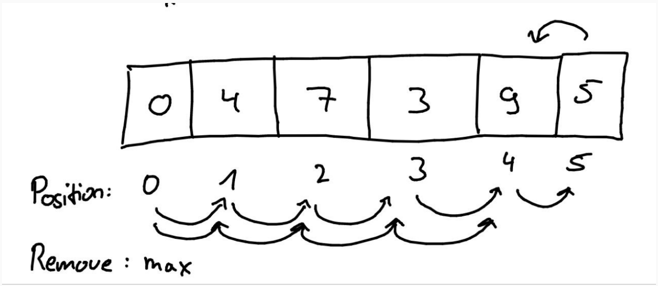


New Problem

We have a list of numbers.
Now we want to find the max element.



We have a list of numbers.
Finaly we have to shrink the list.



heapqueue

There is a data structure that can delete in $O(\log n)$ and find even in $O(1)$ time: **heapqueue**

heapqueue

There is a data structure that can delete in $O(\log n)$ and find even in $O(1)$ time: **heapqueue**

- It is a list that can be appended and prepended in $O(\log n)$ time.
- It is implemented as a binary heap.
- It is in the **heapq** module.
- It is a bit slower than a list.

heapqueue

Some examples:

```
from heapq import heappush, heappop, heapify
h = []
heappush(h, 5) # append to the end
heappush(h, 0) # append to the begin
heappop(h)     # pop the smallest element
```

Next Weeks

Next Weeks

1. APIs (the main way to get data from the internet)

Next Weeks

1. APIs (the main way to get data from the internet)
2. Pandas (we have to store and sort our data)

Next Weeks

1. APIs (the main way to get data from the internet)
2. Pandas (we have to store and sort our data)
3. Tensorflow (we want to do some machine learning)

APIs

REST API

We want to get Data from a Website.

REST API

We want to get Data from a Website.The owner of the site is nice and provides an API.

REST API

We want to get Data from a Website.The owner of the site is nice and provides an API.

- We use **requests** to get the data.
- We use **json** to parse the data.

REST API

Some examples:

```
import requests
import json
r = requests.get(’https://api.github.com/events’)
r.status_code # returns status code
r.json() # returns json
```

What happens here?

The requests module sends a GET request to the server.

What happens here?

The **requests** module sends a **GET** request to the server.
We have different types of requests:

- **GET** - get data
- **POST** - send data
- **PUT** - update data
- **DELETE** - delete data

This requests are called **HTTP** requests.

The response

The server responds with a **status code** and some **data**.

The response

The server responds with a **status code** and some **data**.
This status codes are common:

- **200** - OK
- **404** - Not Found
- **500** - Internal Server Error

The response

First we should check the status code:

```
r = requests.get('https://api.github.com/events')
r.status_code # returns status code
```

The response

First we should check the status code:

```
r = requests.get(’https://api.github.com/events’)
r.status_code # returns status code
```

Then we can parse the data:

```
r.text # returns the data (aka text) as string
r.json() # returns a json object of the text
```

json

Json is a why to represent data as a string.

json

Json is a why to represent data as a string. It is a bit like a dictionary. And therefor there is a module that can parse it.

json

Json is a why to represent data as a string. It is a bit like a dictionary. And therefor there is a module that can parse it.

- **json.loads** - parse a string
- **json.dumps** - create a string

APIs for advanced users

APIs for advanced users

Maybe there is a nice website which contains a lot of data.

APIs for advanced users

Maybe there is a nice website which contains a lot of data.
And maybe there is no API or only a payed API.

scrapy

scrapy

Scrapy is a framework to scrape data from websites.

scrapy

Scrapy is a framework to scrape data from websites.

- It is a bit like a web browser.
- It can parse html.
- It can follow links.
- It can save the data.

Simple HTML parsing

We want to extract some information from a html document.

Simple HTML parsing

We want to extract some information from a html document.
We need a parser

Simple HTML parsing

We want to extract some information from a html document.
We need a parserand a way to identify the information.

Simple HTML parsing

We want to extract some information from a html document.
We need a parserand a way to identify the information.
We can use scrapy to parse the html and we will use XPATH to identify the information.

Simple HTML parsing

Some examples:

```
document = ”Some␣html␣document”
from scrapy.selector import Selector
sel = Selector(text=document)
sel.xpath(’//title/text()’).extract()
sel.xpath(’//title/*’).get_all()
```

Task

Task

- Read the data of a river
- https://www.pegelonline.wsv.de/webservice/ueberblick
- Create a plot of the water level over time

And if you are done:

Task

- Read the data of a river
- https://www.pegelonline.wsv.de/webservice/ueberblick
- Create a plot of the water level over time

And if you are done:
Do the same with:
https://www.umwelt.sachsen.de/umwelt/infosysteme/hwims/portal/web/wasserstand-pegel-501010