

Mr C has to Sort things out

ESC101: Fundamentals of Computing

Purushottam Kar

What is Sorting?

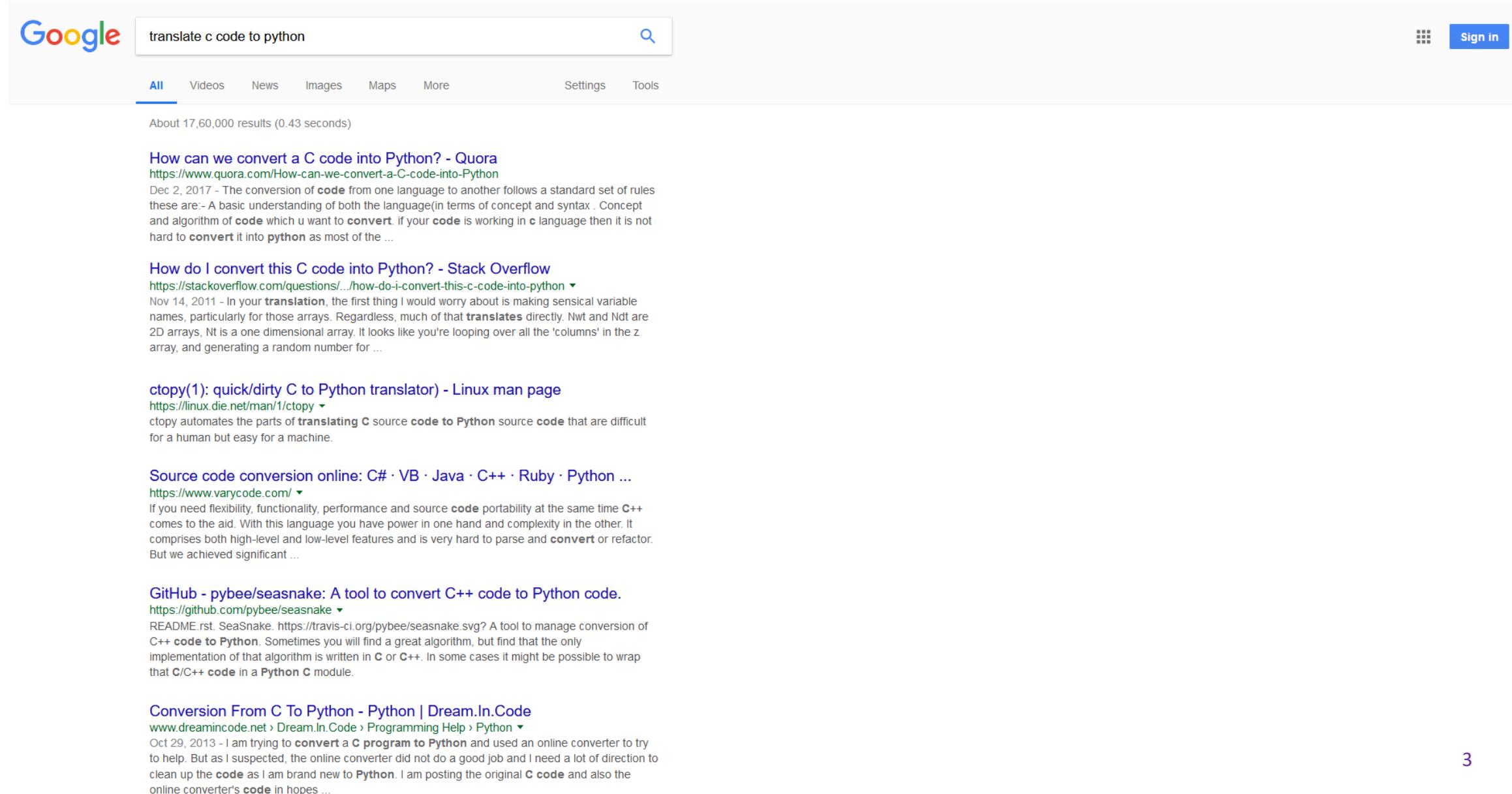
“

Sorting is the process of arranging items systematically, ordered by some criterion

”



Internet Searches



Google search results for "translate c code to python". The search bar shows the query. Below it, the "All" tab is selected, along with other categories like Videos, News, Images, Maps, and More. To the right are Settings and Tools. The results page indicates about 17,60,000 results found in 0.43 seconds.

How can we convert a C code into Python? - Quora
<https://www.quora.com/How-can-we-convert-a-C-code-into-Python>
Dec 2, 2017 - The conversion of **code** from one language to another follows a standard set of rules these are:- A basic understanding of both the language(in terms of concept and syntax . Concept and algorithm of **code** which u want to **convert**. if your **code** is working in **c** language then it is not hard to **convert** it into **python** as most of the ...

How do I convert this C code into Python? - Stack Overflow
<https://stackoverflow.com/questions/.../how-do-i-convert-this-c-code-into-python>
Nov 14, 2011 - In your **translation**, the first thing I would worry about is making sensical variable names, particularly for those arrays. Regardless, much of that **translates** directly. Nwt and Ndt are 2D arrays, Nt is a one dimensional array. It looks like you're looping over all the 'columns' in the z array, and generating a random number for ...

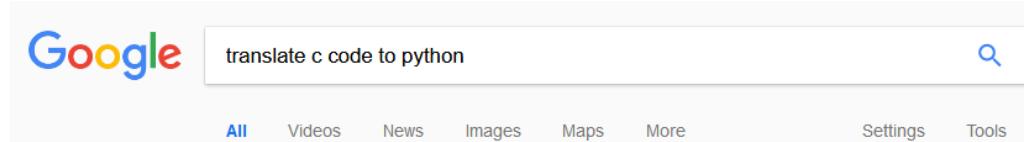
ctopy(1): quick/dirty C to Python translator) - Linux man page
<https://linux.die.net/man/1/ctopy>
ctopy automates the parts of translating C source **code** to Python source **code** that are difficult for a human but easy for a machine.

Source code conversion online: C# · VB · Java · C++ · Ruby · Python ...
<https://www.varycode.com/>
If you need flexibility, functionality, performance and source **code** portability at the same time **C++** comes to the aid. With this language you have power in one hand and complexity in the other. It comprises both high-level and low-level features and is very hard to parse and **convert** or refactor. But we achieved significant ...

GitHub - pybee/seasnake: A tool to convert C++ code to Python code.
<https://github.com/pybee/seasnake>
README.rst. SeaSnake. <https://travis-ci.org/pybee/seasnake.svg>? A tool to manage conversion of C++ **code** to Python. Sometimes you will find a great algorithm, but find that the only implementation of that algorithm is written in C or C++. In some cases it might be possible to wrap that C/C++ **code** in a Python C module.

Conversion From C To Python - Python | Dream.In.Code
www.dreamincode.net › Dream.In.Code › Programming Help › Python
Oct 29, 2013 - I am trying to **convert** a C program to Python and used an online converter to try to help. But as I suspected, the online converter did not do a good job and I need a lot of direction to clean up the **code** as I am brand new to Python. I am posting the original C **code** and also the online converter's **code** in hopes ...

Internet Searches



How can we convert a C code into Python? - Quora

<https://www.quora.com/How-can-we-convert-a-C-code-into-Python>

Dec 2, 2017 - The conversion of **code** from one language to another follows a standard set of rules these are:- A basic understanding of both the language(in terms of concept and syntax . Concept and algorithm of **code** which u want to **convert**. if your **code** is working in **c** language then it is not hard to **convert** it into **python** as most of the ...

How do I convert this C code into Python? - Stack Overflow

<https://stackoverflow.com/questions/.../how-do-i-convert-this-c-code-into-python> ▾

Nov 14, 2011 - In your **translation**, the first thing I would worry about is making sensical variable names, particularly for those arrays. Regardless, much of that **translates** directly. Nwt and Ndt are 2D arrays, Nt is a one dimensional array. It looks like you're looping over all the 'columns' in the z array, and generating a random number for ...

ctopy(1): quick/dirty C to Python translator) - Linux man page

<https://linux.die.net/man/1/ctopy> ▾

ctopy automates the parts of translating C source **code** to Python source **code** that are difficult for a human but easy for a machine.

Source code conversion online: C# · VB · Java · C++ · Ruby · Python ...

<https://www.varycode.com/> ▾

If you need flexibility, functionality, performance and source **code** portability at the same time **C++** comes to the aid. With this language you have power in one hand and complexity in the other. It comprises both high-level and low-level features and is very hard to parse and **convert** or refactor. But we achieved significant ...

GitHub - pybee/seasnake: A tool to convert C++ code to Python code.

<https://github.com/pybee/seasnake> ▾

README.rst. SeaSnake. <https://travis-ci.org/pybee/seasnake.svg>? A tool to manage conversion of C++ **code** to Python. Sometimes you will find a great algorithm, but find that the only implementation of that algorithm is written in C or C++. In some cases it might be possible to wrap that C/C++ **code** in a Python C module.

Conversion From C To Python - Python | Dream.In.Code

www.dreamincode.net › Dream.In.Code › Programming Help › Python

Oct 29, 2013 - I am trying to **convert** a C program to Python and used an online converter to try to help. But as I suspected, the online converter did not do a good job and I need a lot of direction to clean up the **code** as I am brand new to Python. I am posting the original C **code** and also the online converter's **code** in hopes ...

Google sorts webpages in decreasing relevance to the query you asked!

Internet Searches

A screenshot of a Google search results page. The search query is "translate c code to python". The results are as follows:

- How can we convert a C code into Python? - Quora**
<https://www.quora.com/How-can-we-convert-a-C-code-into-Python>
Dec 2, 2017 - The conversion of **code** from one language to another follows a standard set of rules these are:- A basic understanding of both the language(in terms of concept and syntax . Concept and algorithm of **code** which u want to **convert**. if your **code** is working in **c** language then it is not hard to **convert** it into **python** as most of the ...
- How do I convert this C code into Python? - Stack Overflow**
<https://stackoverflow.com/questions/.../how-do-i-convert-this-c-code-into-python>
Nov 14, 2011 - In your **translation**, the first thing I would worry about is making sensical variable names, particularly for those arrays. Regardless, much of that **translates** directly. Nwt and Ndt are 2D arrays, Nt is a one dimensional array. It looks like you're looping over all the 'columns' in the z array, and generating a random number for ...
- ctopy(1): quick/dirty C to Python translator) - Linux man page**
<https://linux.die.net/man/1/ctopy>
ctopy automates the parts of translating C source **code** to Python source **code** that are difficult for a human but easy for a machine.
- Source code conversion online: C# · VB · Java · C++ · Ruby · Python ...**
<https://www.varycode.com/>
If you need flexibility, functionality, performance and source **code** portability at the same time **C++** comes to the aid. With this language you have power in one hand and complexity in the other. It comprises both high-level and low-level features and is very hard to parse and **convert** or refactor. But we achieved significant ...
- GitHub - pybee/seasnake: A tool to convert C++ code to Python code.**
<https://github.com/pybee/seasnake>
README.rst. SeaSnake. <https://travis-ci.org/pybee/seasnake.svg>? A tool to manage conversion of C++ **code** to Python. Sometimes you will find a great algorithm, but find that the only implementation of that algorithm is written in C or C++. In some cases it might be possible to wrap that C/C++ **code** in a Python C module.
- Conversion From C To Python - Python | Dream.In.Code**
[www.dreamincode.net › Dream.In.Code › Programming Help › Python](http://www.dreamincode.net/Dream.In.Code/Programming Help/Python)
Oct 29, 2013 - I am trying to **convert** a C program to Python and used an online converter to try to help. But as I suspected, the online converter did not do a good job and I need a lot of direction to clean up the **code** as I am brand new to Python. I am posting the original C **code** and also the online converter's **code** in hopes ...

Most relevant webpage

Google sorts webpages in decreasing relevance to the query you asked!

Internet Searches

Google search results for "translate c code to python":

- How can we convert a C code into Python? - Quora**
<https://www.quora.com/How-can-we-convert-a-C-code-into-Python>
Dec 2, 2017 - The conversion of **code** from one language to another follows a standard set of rules these are:- A basic understanding of both the language(in terms of concept and syntax . Concept and algorithm of **code** which u want to **convert**. if your **code** is working in **c** language then it is not hard to **convert** it into **python** as most of the ...
- How do I convert this C code into Python? - Stack Overflow**
<https://stackoverflow.com/questions/.../how-do-i-convert-this-c-code-into-python>
Nov 14, 2011 - In your **translation**, the first thing I would worry about is making sensical variable names, particularly for those arrays. Regardless, much of that **translates** directly. Nwt and Ndt are 2D arrays, Nt is a one dimensional array. It looks like you're looping over all the 'columns' in the z array, and generating a random number for ...
- ctopy(1): quick/dirty C to Python translator) - Linux man page**
<https://linux.die.net/man/1/ctopy>
ctopy automates the parts of translating C source **code** to Python source **code** that are difficult for a human but easy for a machine.
- Source code conversion online: C# · VB · Java · C++ · Ruby · Python ...**
<https://www.varycode.com/>
If you need flexibility, functionality, performance and source **code** portability at the same time **C++** comes to the aid. With this language you have power in one hand and complexity in the other. It comprises both high-level and low-level features and is very hard to parse and **convert** or refactor. But we achieved significant ...
- GitHub - pybee/seasnake: A tool to convert C++ code to Python code.**
<https://github.com/pybee/seasnake>
README.rst. SeaSnake. <https://travis-ci.org/pybee/seasnake.svg>? A tool to manage conversion of C++ **code** to Python. Sometimes you will find a great algorithm, but find that the only implementation of that algorithm is written in C or C++. In some cases it might be possible to wrap that C/C++ **code** in a Python C module.
- Conversion From C To Python - Python | Dream.In.Code**
[www.dreamincode.net › Dream.In.Code › Programming Help › Python](http://www.dreamincode.net/Dream.In.Code/Programming Help/Python)
Oct 29, 2013 - I am trying to **convert** a C program to Python and used an online converter to try to help. But as I suspected, the online converter did not do a good job and I need a lot of direction to clean up the **code** as I am brand new to Python. I am posting the original C **code** and also the online converter's **code** in hopes ...

Most relevant webpage

Google sorts webpages in decreasing relevance to the query you asked!

Less relevant webpage

Email Clients

Folders
Last Refresh:
Tue, 12:33 am
[\(Check mail\)](#)

- **INBOX** (5161/9964)

Drafts

Sent

Trash (Purge)

Draft

[Folder Sizes](#)

Current Folder: INBOX

[Sign Out](#)

[Compose](#) [Addresses](#) [Folders](#) [Options](#) [Search](#) [Help](#) [Calendar](#) [Auto Response](#)

[Help \(LAN\)](#)

[Previous](#) | [Next](#) | [1](#) [2](#) [3](#) [4](#) [5](#) ... [14](#) [15](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#) [24](#) ... [196](#) [197](#) [198](#) [199](#) [200](#) | [Show All](#) | [Toggle All](#)

Viewing Messages: **901** to **950** (9964 total)

Move Selected To:

[INBOX](#) [Move](#) [Forward](#)

Transform Selected Messages:

[Flag](#) [Unflag](#) [Read](#) [Unread](#) [Delete](#)

[Thread View](#)

From	Date	Subject	Size	
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...	7.9 k	
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k	
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k	
<input type="checkbox"/> Yatindra Nath Singh	Aug 27, 2018	[acadstaff] Open Seminar Announcement.	10 k	
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava	Aug 27, 2018	[acadstaff] Re: IEEE MTT-S Lecture by Prof. Harsha...	12 k	
<input type="checkbox"/> asima@iitk.ac.in	Aug 27, 2018	[acadstaff] PhD Open Seminar of Pavan Kumar Roll N...	7.2 k	
<input type="checkbox"/> Subhajit Roy	Aug 27, 2018	[acadstaff] CSE Seminar: Fine-Pruning: Defending A...	4.7 k	
<input type="checkbox"/> Sounak Choudhury	Aug 27, 2018	[acadstaff] M.Tech. Thesis Defense in Mech. Engg. ...	5 k	
<input type="checkbox"/> DOAA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k	
<input type="checkbox"/> SANJEEV GARG	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k	
<input type="checkbox"/> eccc@weizmann.ac.il	Aug 27, 2018	[ECCC] New Paper published	5.6 k	
<input type="checkbox"/> Dean of R&D, IIT Kanpur	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k	
<input type="checkbox"/> gsaikat@iitk.ac.in	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k	
<input type="checkbox"/> Y N Mohapatra	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k	
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k	
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k	
<input type="checkbox"/> Sagar Chakraborty	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k	

Email Clients

I can sort by date of receipt

Last Refresh: Tue, 12:33 am (Check mail)

Folders

- INBOX (5161/5) Drafts Sent Trash (Purge) Draft Folder Sizes

Options Search Help Calendar Auto Response Sign Out Help (LAN)

5 16 17 18 19 20 21 22 23 24 ... 196 197 198 199 200 | Show All | Toggle All Viewing Messages: 901 to 950 (9964 total)

Transform Selected Messages: Flag Unflag Read Unread Delete

From	Date	Subject	Size
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...)	7.9 k
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k
<input type="checkbox"/> Yatindra Nath Singh	Aug 27, 2018	[acadstaff] Open Seminar Announcement.	10 k
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava	Aug 27, 2018	[acadstaff] Re: IEEE MTT-S Lecture by Prof. Harsha...	12 k
<input type="checkbox"/> asima@iitk.ac.in	Aug 27, 2018	[acadstaff] PhD Open Seminar of Pavan Kumar Roll N...	7.2 k
<input type="checkbox"/> Subhajit Roy	Aug 27, 2018	[acadstaff] CSE Seminar: Fine-Pruning: Defending A...	4.7 k
<input type="checkbox"/> Sounak Choudhury	Aug 27, 2018	[acadstaff] M.Tech. Thesis Defense in Mech. Engg. ...	5 k
<input type="checkbox"/> DOAA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k
<input type="checkbox"/> SANJEEV GARG	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k
<input type="checkbox"/> eccc@weizmann.ac.il	Aug 27, 2018	[ECCC] New Paper published	5.6 k
<input type="checkbox"/> Dean of R&D, IIT Kanpur	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k
<input type="checkbox"/> gsaikat@iitk.ac.in	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k
<input type="checkbox"/> Y N Mohapatra	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k
<input type="checkbox"/> Sagar Chakraborty	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k

Email Clients

I can sort by date of receipt

I can sort by sender

From	Date	Subject	Size
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...)	7.9 k
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k
<input type="checkbox"/> Yatindra Nath Singh		Seminar Announcement.	10 k
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava		[acadstaff] IITK MTT-S Lecture by Prof. Harsha...	12 k
<input type="checkbox"/> asima@iitk.ac.in		[acadstaff] Seminar of Pavan Kumar Roll N...	7.2 k
<input type="checkbox"/> Subhajit Roy		[acadstaff] Seminar: Fine-Pruning: Defending A...	4.7 k
<input type="checkbox"/> Sounak Choudhury		[acadstaff] Thesis Defense in Mech. Engg. ...	5 k
<input type="checkbox"/> DOAA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k
<input type="checkbox"/> SANJEEV GARG	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k
<input type="checkbox"/> eccc@weizmann.ac.il	Aug 27, 2018	[ECCC] New Paper published	5.6 k
<input type="checkbox"/> Dean of R&D, IIT Kanpur	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k
<input type="checkbox"/> gsaikat@iitk.ac.in	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k
<input type="checkbox"/> Y N Mohapatra	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k
<input type="checkbox"/> Sagar Chakraborty	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k

[Sign Out](#)

[Help \(LAN\)](#)

[Options](#) [Search](#) [Help](#) [Calendar](#) [Auto Response](#)

[5](#) [16](#) [17](#) [18](#) [19](#) [20](#) [21](#) [22](#) [23](#) [24](#) ... [196](#) [197](#) [198](#) [199](#) [200](#) | [Show All](#) | [Toggle All](#)

Viewing Messages: 901 to 950 (9964 total)

Transform Selected Messages:
[Flag](#) [Unflag](#) [Read](#) [Unread](#) [Delete](#)

Folders

Last Refresh:
Tue, 12:33 am
(Check mail)

- INBOX (5161/5)

Drafts

Sent

Trash (Purge)

Draft

[Folder Sizes](#)

Email Clients

Folders
Last Refresh: Tue, 12:33 am
[Check mail](#)

[Sign Out](#)
[Help \(LAN\)](#)

I can sort by date of receipt

INBOX (5161/5)
Drafts
Sent
Trash (Purge)
Draft
Folder Sizes

Options Search Help Calendar Auto Response

5 16 17 18 19 20 21 22 23 24 ... 196 197 198 199 200 | Show All | Toggle All

Viewing Messages: 901 to 950 (9964 total)

Transform Selected Messages:
Flag Unflag Read Unread Delete

From	Date	Subject	Size
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...)	7.9 k
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k
<input type="checkbox"/> Yatindra Nath Singh		Seminar Announcement	10 k
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava		1 MTT-S Lecture	12 k
<input type="checkbox"/> asima@iitk.ac.in		on Seminar of Pav	7.2 k
<input type="checkbox"/> Subhajit Roy		Seminar: Fine-Pruning	4.7 k
<input type="checkbox"/> Sounak Choudhury		Thesis Defense in M	5 k
<input type="checkbox"/> DOAA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k
<input type="checkbox"/> SANJEEV GARG	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k
<input type="checkbox"/> eccc@weizmann.ac.il	Aug 27, 2018	[ECCC] New Paper published	5.6 k
<input type="checkbox"/> Dean of R&D, IIT Kanpur	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k
<input type="checkbox"/> gsaikat@iitk.ac.in	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k
<input type="checkbox"/> Y N Mohapatra	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k
<input type="checkbox"/> Sagar Chakraborty	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k

I can sort by sender

I can sort by size of the mail

Page 4
ESC101

Email Clients

Folders
Last Refresh: Tue, 12:33 am (Check mail)

INBOX (5161/5161)
Drafts Sent Trash (Purge) Draft Folder Sizes

Options Search Help Calendar Auto Response
5 16 17 18 19 20 21 22 23 24 ... 196 197 198 199 200 | Show All | Help (LAN) Sign Out

I can sort by date of receipt

I can sort by subject

I can sort by sender

I can sort by size of the mail

From	Date	Subject	Size
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...)	7.9 k
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k
<input type="checkbox"/> Yatindra Nath Singh		Seminar Announcement	10 k
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava		[acadstaff] IITK MTT-S Lecture Series	12 k
<input type="checkbox"/> asima@iitk.ac.in		[acadstaff] Seminar of Pavitra Ramamurti	7.2 k
<input type="checkbox"/> Subhajit Roy		[acadstaff] Seminar: Fine-Pruning of Neural Networks	4.7 k
<input type="checkbox"/> Sounak Choudhury		[acadstaff] Thesis Defense in M...	5 k
<input type="checkbox"/> DOAA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k
<input type="checkbox"/> SANJEEV GARG	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k
<input type="checkbox"/> eccc@weizmann.ac.il	Aug 27, 2018	[ECCC] New Paper published	5.6 k
<input type="checkbox"/> Dean of R&D, IIT Kanpur	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k
<input type="checkbox"/> gsaikat@iitk.ac.in	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k
<input type="checkbox"/> Y N Mohapatra	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k
<input type="checkbox"/> Anand Kumar Jha	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k
<input type="checkbox"/> Sagar Chakraborty	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k

Email Clients

Folders
Last Refresh: Tue, 12:33 am (Check mail)

INBOX (5161/5)
Drafts
Sent
Trash (Purge)
Draft
Folder Sizes

Options Search Help Calendar Auto Response
15 16 17 18 19 20 21 22 23 24 ... 196 197 198 199 200 | Show All |
Pages: 901 to 950 (9964 total)
Transform Selected Messages: Flag Unflag Read Unread Delete

I can sort by date of receipt

I can sort by subject

I can sort by sender

I can sort by size of the mail

The same set of objects can be ordered by more than one criterion or “key”!

From	Date	Subject	Size
<input type="checkbox"/> Nitin Gupta	Aug 27, 2018	[acadstaff] Re: M.Tech. defense: Surbhit Wagle (BS...)	7.9 k
<input type="checkbox"/> Sameer Khandekar	Aug 27, 2018	[acadstaff] Notice: Ph. D. Oral Examination - Mr. ...	20 k
<input type="checkbox"/> Sumit	Aug 27, 2018	[acadstaff] Notice for PhD defence of Mr Rajeev Ku...	5.2 k
<input type="checkbox"/> Yatindra Nath Singh		Seminar Announcement	10 k
<input type="checkbox"/> Dr. Kumar Vaibhav Srivastava		[acadstaff] IITK MTT-S Lecture Series	12 k
<input type="checkbox"/> asima@iitk.ac.in		[acadstaff] Seminar of Pavitra Ramamurti	7.2 k
<input type="checkbox"/> Subhajit Roy		[acadstaff] Seminar: Fine-Pruning of Neural Networks	4.7 k
<input type="checkbox"/> Sounak Choudhury		[acadstaff] Thesis Defense in M...	5 k
<input type="checkbox"/> DOA	Aug 27, 2018	[All] [Fwd: Best student paper award]	6.6 k
	Aug 27, 2018	[acadstaff] PhD Oral Examination (10102073)	78 k
	Aug 27, 2018	[ECCC] New Paper published	5.6 k
	Aug 27, 2018	[All] Institute Lecture by Mr Arvind Gupta today @ ...	3.3 k
	Aug 27, 2018	[acadstaff] Physics Colloquium: 31st August (Frid...	4.3 k
	Aug 26, 2018	[faculty] [MSP] M Tech Thesis Defence: on solution ...	6.1 k
	Aug 26, 2018	[acadstaff] QuIC talk: Shreya P Kumar, Ulm Univers...	5 k
	Aug 26, 2018	[acadstaff] General Physics Seminar: Ish Dhand, Ul...	4.7 k
	Aug 26, 2018	[acadstaff] 30th Aug: General Physics Seminar by D...	5 k

Sagar Chakraborty

Email Clients

The screenshot shows an email inbox with several sorting options highlighted by orange and green callouts:

- I can sort by date of receipt**: Points to the "Date" column header.
- I can sort by subject**: Points to the "Subject" column header.
- I can sort by sender**: Points to the "From" column header.
- I can sort by size of the mail**: Points to the "Size" column header.

The same set of objects can be ordered by more than one criterion or “key”!

Sort criterion can be numeric (e.g. size, date, relevance) or lexicographic (e.g. alphabetic)

Key UI elements visible in the inbox:

- Header: Options, Search, Help, Calendar, Auto Response
- Page navigation: 5 16 17 18 19 20 21 22 23 24 ... 196 197 198 199 200 | Show All
- Message list headers: From, Date, Subject, Size
- Action buttons: INBOX, Move, Forward, Thread View, Flag, Unflag, Read, Unread, Delete

The same set of objects can be ordered by more than one criterion or “key”!

Sort criterion can be numeric (e.g. size, date, relevance) or lexicographic (e.g. alphabetic)

Applications of Sorting

- The ability to order objects (webpages, emails, movies, songs) according to varied and user-dictated criteria can be lucrative
- A subfield of machine learning called *ranking* is dedicated to this and has immense applications – recommendation systems (Amazon, Flipkart), internet search, personalized medicine
- Profs especially love sorting since it helps them assign grades!
- Sorting can also make other (less fancy) operations very simple
- Agenda for the next couple of lectures
 - Look at a few applications of sorting
 - Look at a few efficient techniques of sorting
 - Get introduced to the divide and conquer technique

Search in Sorted Arrays

Brute Force Search



- Is the element 4 present in the array?
 - Can search the array from left to right or right to left
 - `for(i=0;i<11;i++) if(a[i]==4) return i; return -1;`
 - `for(i=10;i>=0;i--) if(a[i]==4) return i; return -1;`
 - Searching from left seems faster for the query 4
- Is the element 3 present in the array?
- Is the element 5 present in the array?
- If there are N elements in the array we have to do at least N operations (to verify absence) - can we do any better?

Binary Search

1	1	1	2	3	4	4	6	8	8	9
---	---	---	---	---	---	---	---	---	---	---

- The above array is sorted in ascending order $a[i] \leq a[i + 1]$
- Can sort arrays in descending order too i.e. $a[i] \geq a[i + 1]$
- Now lets try searching again by exploiting sortedness
- Crucial insight: if we are searching for the element K and if we know $a[j] < K$ then we also know $a[i] < K$ for all $i < j$
 - Proof: $a[i] < a[j]$ since $i < j$ and array is sorted and we know $a[j] < K$
 - Similarly, if $a[j] > K$ then we also know $a[i] > K$ for all $i > j$
- We will use the above to eliminate vast swathes of the array

Binary Search



Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)
 - Continue search on $a[1:5]$ -- use the same trick again

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)
 - Continue search on $a[1:5]$ -- use the same trick again

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)
 - Continue search on $a[1:5]$ -- use the same trick again
- So a win-win situation – we either find the element or else reduce the search space to only half of the array

Binary Search



- Suppose we check $a[6] == K$? Three possible outcomes
 - Case 1: $a[6] == K$. Great we have found K. Go home and rest
 - Case 2: $a[6] < K$. The left half of the array can never contain K (e.g. $K = 5$)
 - Continue search on $a[7:11]$ -- use the same trick again
 - Case 3: $a[6] > K$. The right half of the array can never contain K (e.g. $K = 2$)
 - Continue search on $a[1:5]$ -- use the same trick again
- So a win-win situation – we either find the element or else reduce the search space to only half of the array
- Example of the *Divide and Conquer* technique – divide original problem into smaller instances of the same problem

Binary Search



Binary Search



- Lets take an example – search for the element 1 in the array

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0$, $R = N-1$

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0$, $R = N-1$
 - At every time step, we check the middle element of active range

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0$, $R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range

Binary Search



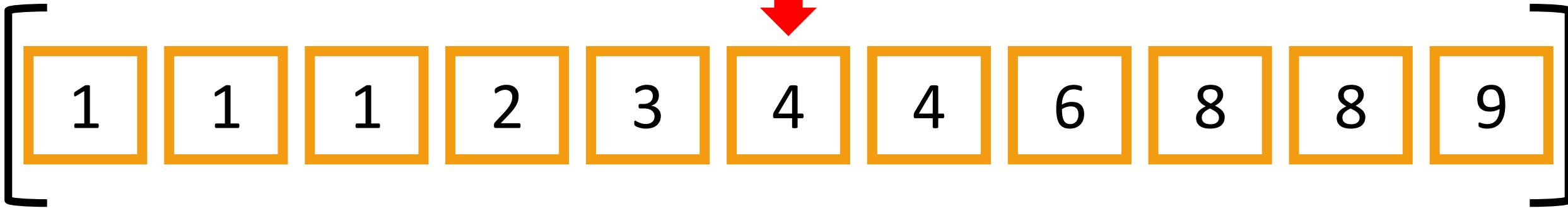
- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



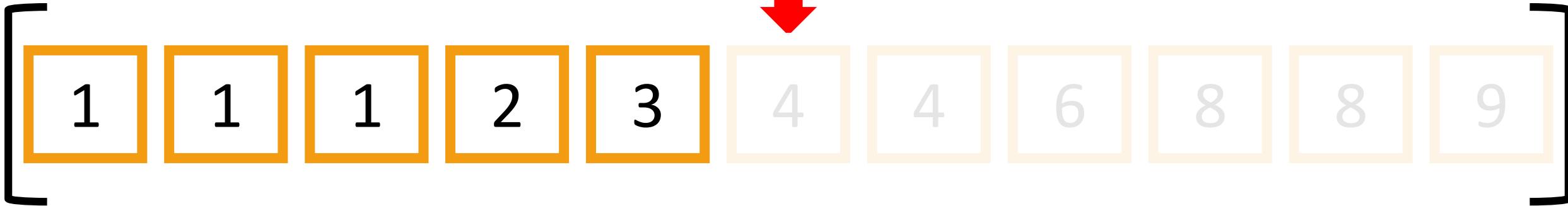
- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0$, $R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



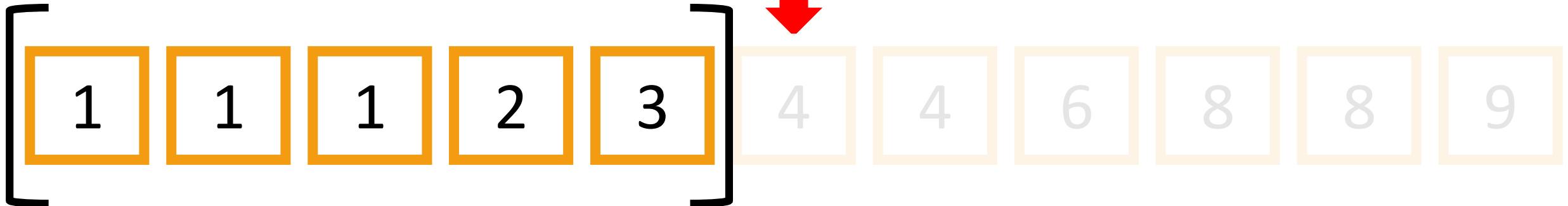
- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



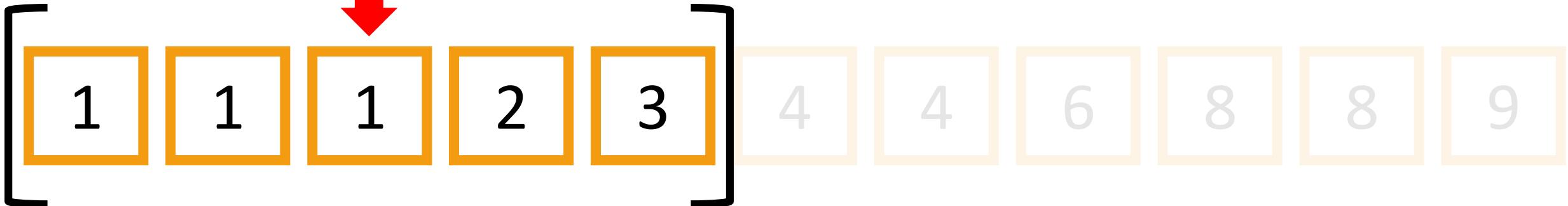
- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



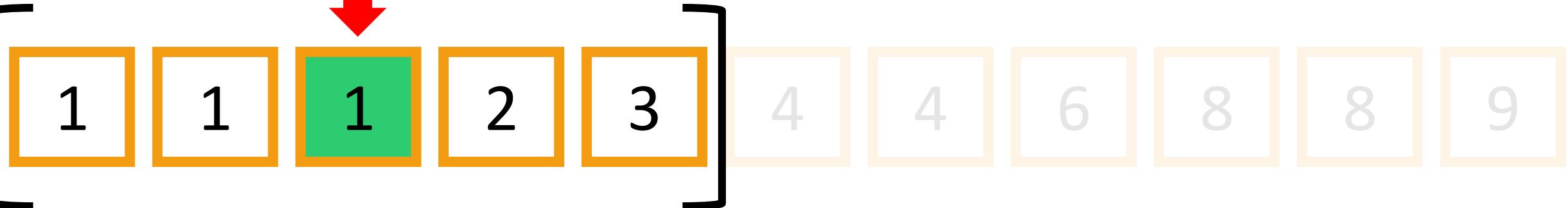
- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 1 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



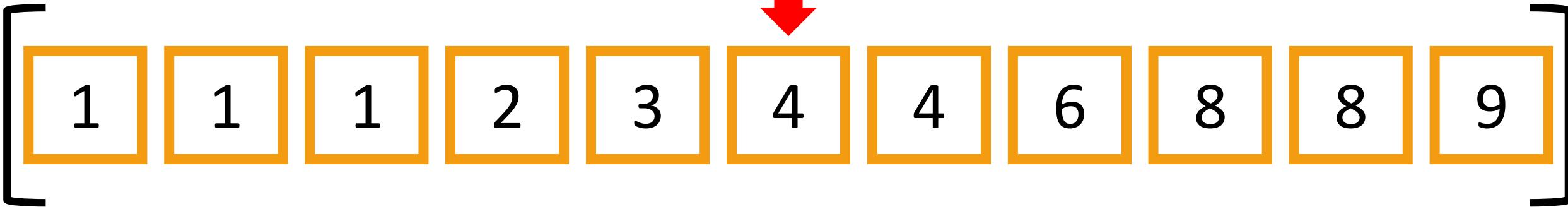
- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



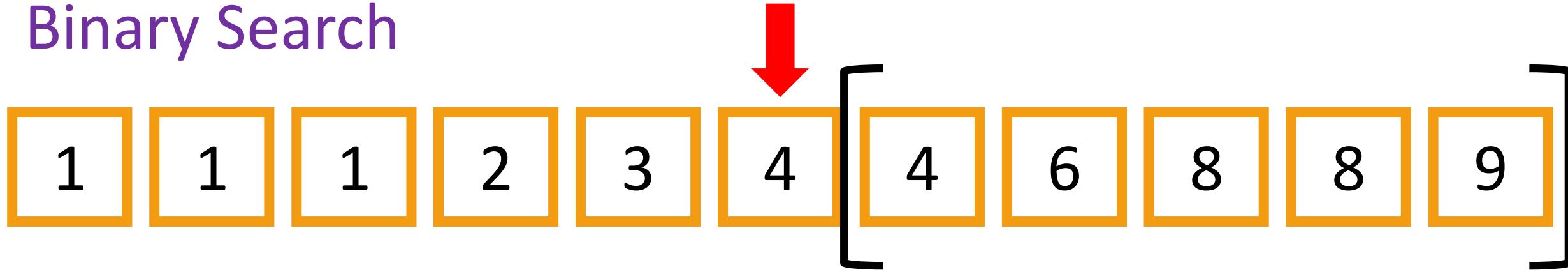
- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



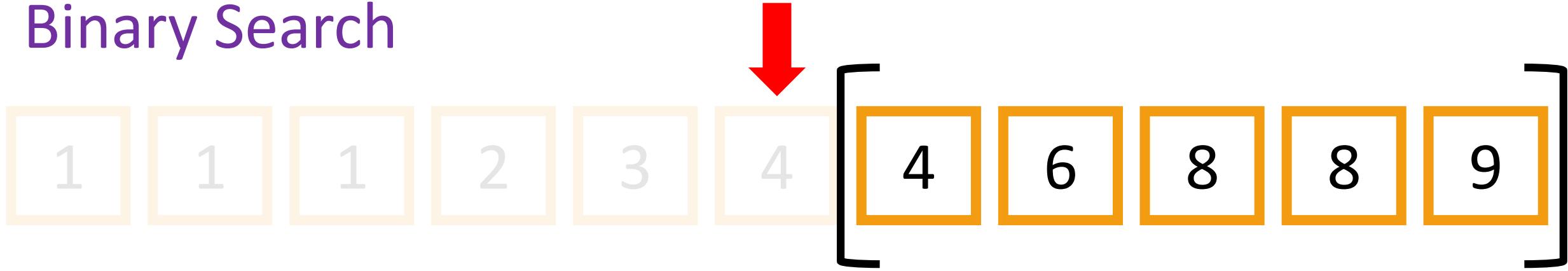
- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



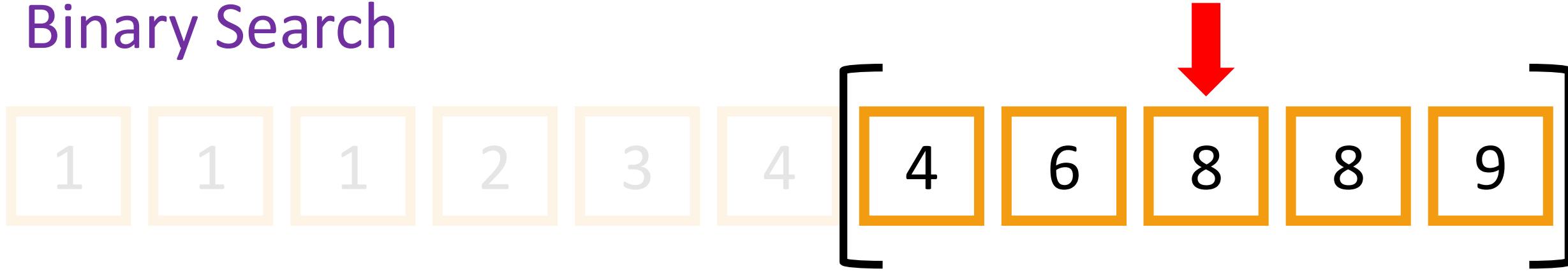
- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0$, $R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



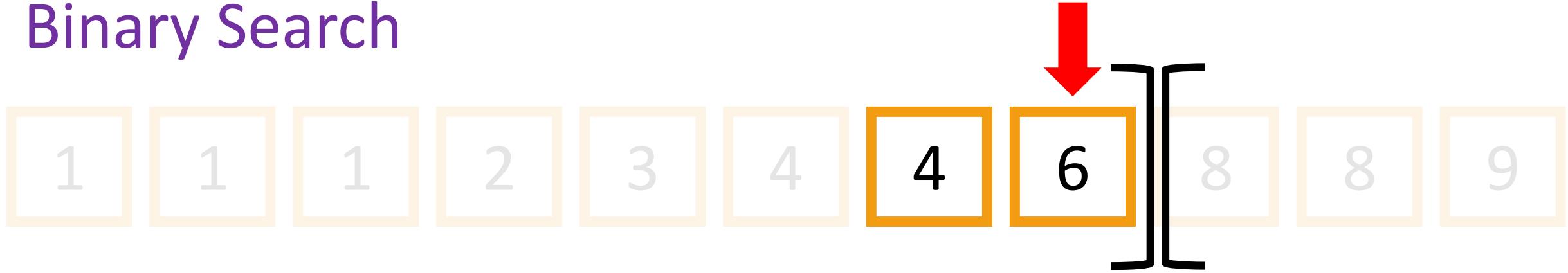
- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search



- Lets take an example – search for the element 7 in the array
- We will always maintain an *active range* $a[L:R]$ with $0 \leq L \leq R < N$
 - Initially the active range is entire array i.e. $L = 0, R = N-1$
 - At every time step, we check the middle element of active range
- Invariants: we will ensure two things
 - At all points of time, if the key we are searching for is at all present in the array, it must be present within our chosen active range
 - At every time step, we will halve the size of the active range
- Will need to be careful about termination criterion – more later

Binary Search

BINARY SEARCH

1. Given: Sorted array a with N elements, key to search K
2. Let $L \leftarrow 0$ and $R \leftarrow N - 1$ *//Initial active range is full array*
3. While $L \leq R$
 1. Let $M \leftarrow \text{ceil}((L + R)/2)$
 2. If $a[M] == K$, return M *//Found key, return location*
 3. If $a[M] > K$, set $R \leftarrow M - 1$ *//Right portion can't host K*
 4. If $a[M] < K$, set $L \leftarrow M + 1$ *//Left portion can't host K*
4. Return -1 *//We failed to find the key 😞*

Binary Search

BINARY SEARCH

1. Given: Sorted array a with N elements, key to search K
2. Let $L \leftarrow 0$ and $R \leftarrow N - 1$ *//Initial active range is full array*
3. While $L \leq R$
 1. Let $M \leftarrow \text{ceil}((L + R)/2)$
 2. If $a[M] == K$, return M *//Found key, return location*
 3. If $a[M] > K$, set $R \leftarrow M - 1$ *//Right portion can't host K*
 4. If $a[M] < K$, set $L \leftarrow M + 1$ *//Left portion can't host K*
4. Return -1 *//We failed to find the key 😞*

The above is often known as *pseudo code*, something that gives details of an algorithm but does not strictly follow rules of C or any other programming language

Binary Search

BINARY SEARCH

Exercise: convert this to proper C code

1. Given: Sorted array a with N elements, key to search K
2. Let $L \leftarrow 0$ and $R \leftarrow N - 1$ *//Initial active range is full array*
3. While $L \leq R$
 1. Let $M \leftarrow \text{ceil}((L + R)/2)$
 2. If $a[M] == K$, return M *//Found key, return location*
 3. If $a[M] > K$, set $R \leftarrow M - 1$ *//Right portion can't host K*
 4. If $a[M] < K$, set $L \leftarrow M + 1$ *//Left portion can't host K*
4. Return -1 *//We failed to find the key 😞*

The above is often known as *pseudo code*, something that gives details of an algorithm but does not strictly follow rules of C or any other programming language

Binary Search

Exercise: write a recursive version

Exercise: convert this to proper C code

BINARY SEARCH

1. Given: Sorted array a with N elements, key to search K
2. Let $L \leftarrow 0$ and $R \leftarrow N - 1$ *//Initial active range is full array*
3. While $L \leq R$
 1. Let $M \leftarrow \text{ceil}((L + R)/2)$
 2. If $a[M] == K$, return M *//Found key, return location*
 3. If $a[M] > K$, set $R \leftarrow M - 1$ *//Right portion can't host K*
 4. If $a[M] < K$, set $L \leftarrow M + 1$ *//Left portion can't host K*
4. Return -1 *//We failed to find the key 😞*

The above is often known as *pseudo code*, something that gives details of an algorithm but does not strictly follow rules of C or any other programming language

Asymptotic Time Complexity

- An effort to quantify the *speed* of algorithms in a manner that is independent of the computer on which they are executed
- Arguably binary search seems “faster” than brute force search
- We saw that in the worse case, brute force search on an unsorted array must check all N elements before answering
- Can binary search on sorted arrays also be forced to do so?
- Let $T(N)$ denote the time taken by binary search to search for a key in a sorted array with N elements
- We know that at every iteration of the while loop, binary search either discovers the element being searched or else reduces the length of the active range by a factor of 2

Asymptotic Time Complexity

- Thus, we must have $T(N) \leq c + T(N/2)$
 - c is the time taken to compare the middle element and update L, R
 - Note that c does not depend on N at all. Also note that $T(1) \leq c$
 - The above is a *recurrence relation*. It expresses T in terms of itself
- Applying the above to $T(N/2)$ gives us $T(N/2) \leq c + T(N/4)$ i.e. $T(N) \leq 2c + T(N/4) = 2c + T(N/2^2)$
- Repeating this gives us $T(N) < cm + T(N/2^m)$ for any $m > 0$
- However for $m \geq \text{ceil}(\log_2 N)$ we have $2^m \geq N$
- This means that $T(N) \leq c \cdot \text{ceil}(\log N) + T(1) \leq c \cdot \text{ceil}(\log N) + c$
- For all $N \geq 4$ we have $\text{ceil}(\log N) + 1 \leq 2 \log N$ which gives us
$$T(N) \leq 2c \cdot \log N$$

Big-Oh Notation

- Suppose we have two functions $f, g: \mathbb{R}_+ \rightarrow \mathbb{R}_+$ such that there exists a constant $c > 0$ so that for all “large” values of $x \in \mathbb{R}_+$ i.e. for all $x \geq M$ for some $M > 0$, we have

$$f(x) \leq c \cdot g(x)$$

Then we say that $f(x) = O(g(x))$

- Be careful that c must not depend on x for the above statement
- The above discussion shows that the runtime complexity of Binary search is $T(N) = O(\log N)$ since for some constant c that doesn't depend on N we have $T(N) \leq 2c \cdot \log N$ for all $N \geq 4$
- Exercise: show that the runtime of brute force search is $O(N)$

Exercises

- Given a array `int a[N]`; sorted in ascending order
 - Find the number of occurrences of a given number K in the array
 - Generalizes the search problem we just studied
 - Find the predecessor of a given number K in the array
 - Largest number in the array that is strictly smaller than K . Return `NULL` if none.
 - Be careful, the key K may itself occur say $N/2$ times in the array
 - Given a positive integer $M \leq N$, find the element of the array which is greater than or equal to exactly M elements of the array
 - $M = 1$ gives the smallest element, $M = N$ the largest element, $M = N/2$ the median
 - If you are interested, look up the term *quantile* on the internet for more info
- Make sure your algorithms take no more than $O(\log N)$ steps!
- Can you do the above operations as fast if the array is not sorted?



Exercises

- Given a (non-sorted) array `int a[N]` ; count the number of swaps A swap is a pair $0 \leq i \neq j < N$ such that $i < j$ but $a[i] > a[j]$
 - This problem is related to a ranking metric known as *area under the ROC curve*. Check it out if interested
- We have two arrays of N numbers `int P[N], F[N]` ; containing 12th marks of N students each who cleared and did not clear JEE
 - Find out the number of students who did not clear JEE but had 12th std marks more than at least 50% of students who did clear JEE
- Solve these problems faster than $O(N^2)$ time (Hint may involve sorting). Assume you have a routine that can sort N elements in $O(N \log N)$ time – will see such methods soon.

Sorting Algorithms

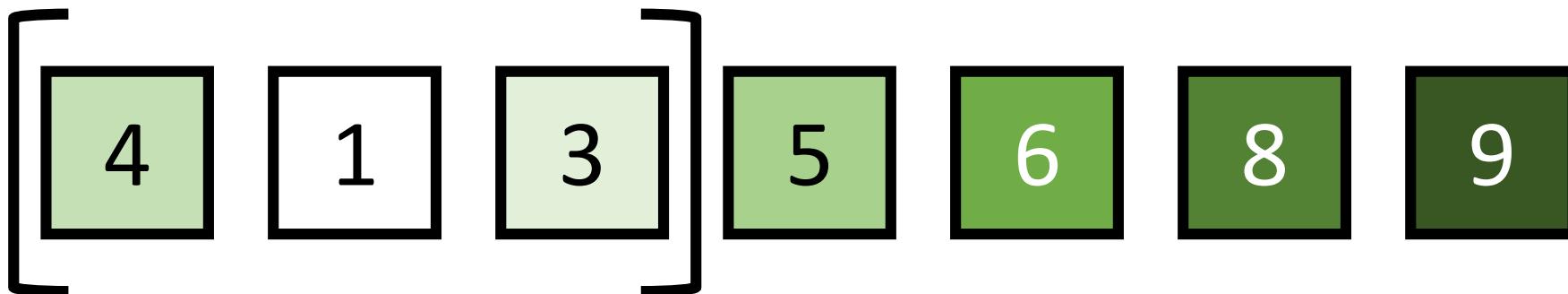
Selection Sort

Selection Sort

- One of the many (many) algorithms for sorting – very simple
- Like binary search, maintains *active range* $a[0:R]$ with $0 \leq R < N$
 - Initially the active range is entire array i.e. $R = N - 1$
- Invariants: we will ensure two things
 - At all points of time, the non-active portion will be sorted in ascending order i.e. for all $R \leq i < j$ we will ensure $a[i] \leq a[j]$
 - The non-active elements will never be smaller than the elements in the active range i.e. if $i \leq R < j$ then $a[i] \leq a[j]$
- The active region will shrink by one element at each step

Selection Sort

- One of the many (many) algorithms for sorting – very simple
- Like binary search, maintains *active range* $a[0:R]$ with $0 \leq R < N$
 - Initially the active range is entire array i.e. $R = N - 1$
- Invariants: we will ensure two things
 - At all points of time, the non-active portion will be sorted in ascending order i.e. for all $R \leq i < j$ we will ensure $a[i] \leq a[j]$
 - The non-active elements will never be smaller than the elements in the active range i.e. if $i \leq R < j$ then $a[i] \leq a[j]$
- The active region will shrink by one element at each step



Selection Sort

Selection Sort

- Notice that we never have to touch the non-active region 😊

Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region

Selection Sort

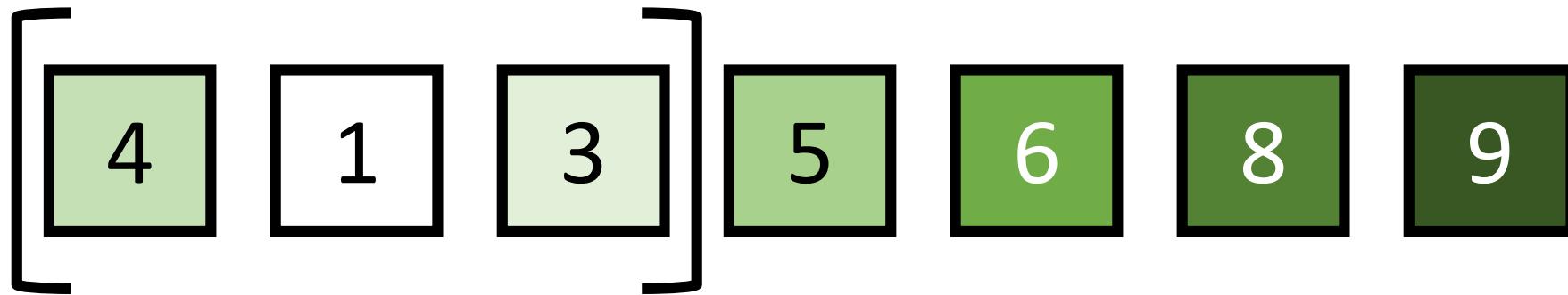
- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region

Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap

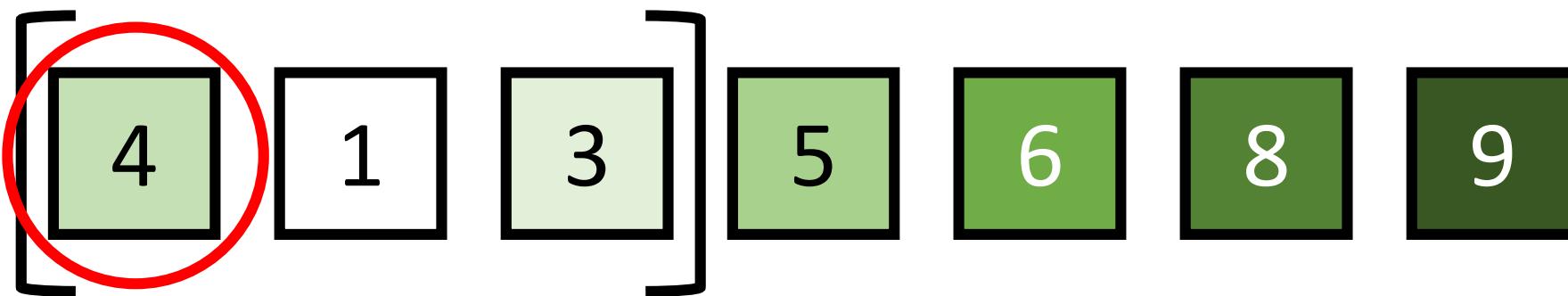
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



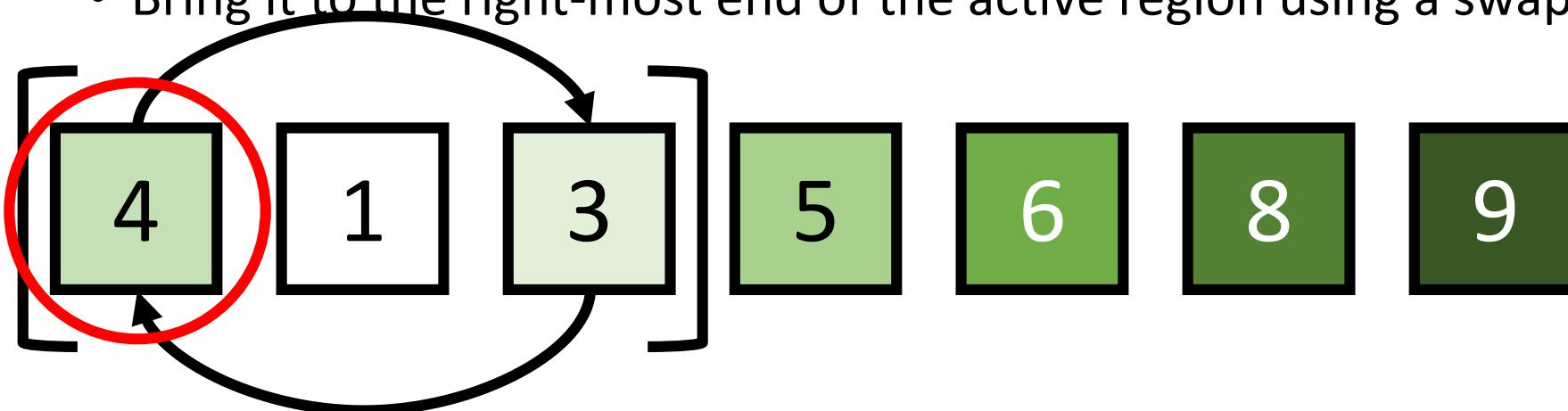
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



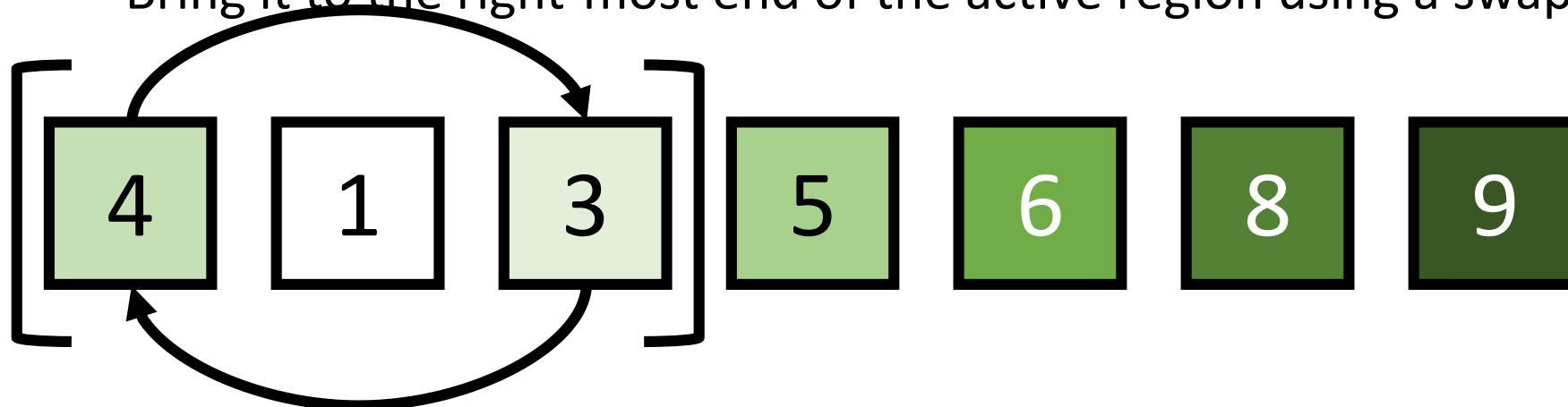
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



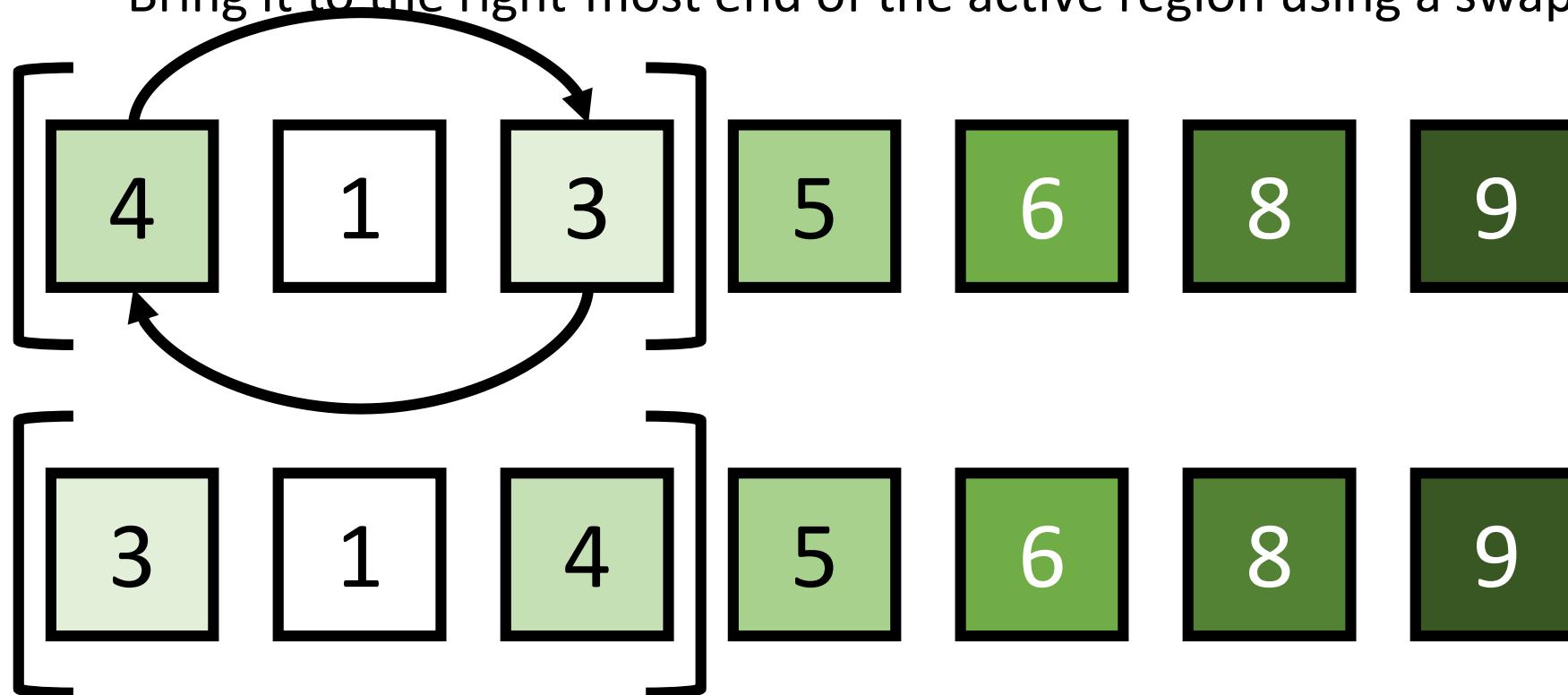
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



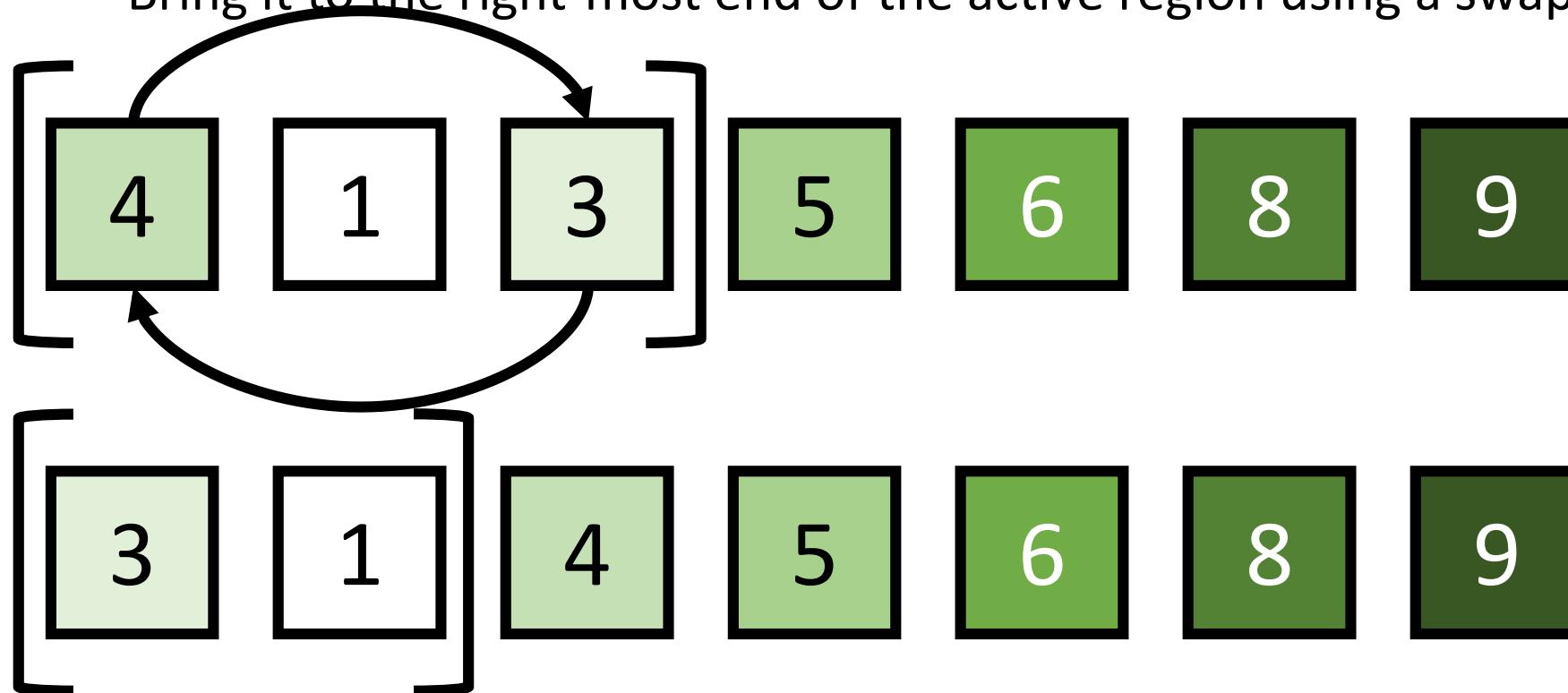
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



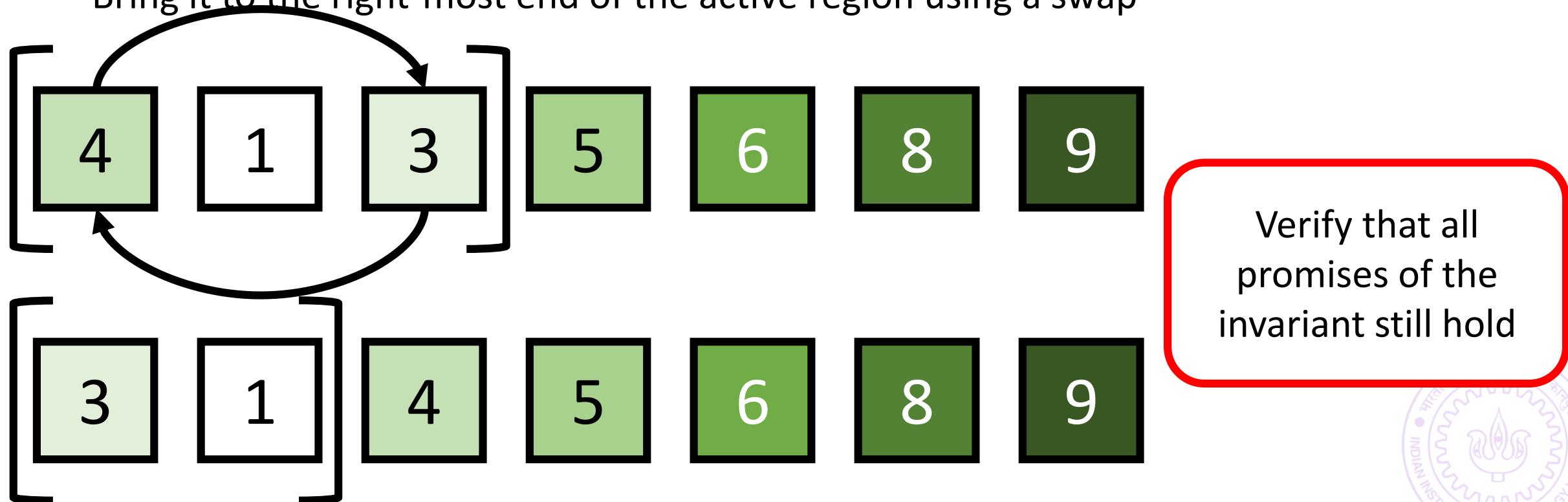
Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



Selection Sort

- Notice that we never have to touch the non-active region 😊
- To maintain the invariant and still shrink the active region
 - We search for the largest element in the active region
 - Bring it to the right-most end of the active region using a swap



Selection Sort

SELECTION SORT

1. Given: Array a with N elements
2. For $R = N - 1; R > 0; R --$ *//Initial active range is full array*
 1. $i \leftarrow \text{FINDMAX}(a, 0, R)$ *//Location of largest element in $a[0, R]$*
 2. $\text{SWAP}(a, i, R)$ *//Bring largest element to the end*

SWAP

1. Given: Array a , location i, j
2. Let $\text{tmp} \leftarrow a[i]$
3. Let $a[i] \leftarrow a[j]$
4. Let $a[j] \leftarrow \text{tmp}$

FINDMAX

1. Given: Array a , locations i, j
2. Let $k \leftarrow i, \text{max} = a[k]$
3. For $l = i; i \leq j; l ++$
 1. If $a[l] > \text{max}$, $\text{max} = a[l], k = l$
4. Return k

Selection Sort

SELECTION SORT

Exercise: convert this to proper C code

1. Given: Array a with N elements
2. For $R = N - 1; R > 0; R --$ *//Initial active range is full array*
 1. $i \leftarrow \text{FINDMAX}(a, 0, R)$ *//Location of largest element in $a[0, R]$*
 2. $\text{SWAP}(a, i, R)$ *//Bring largest element to the end*

SWAP

1. Given: Array a , location i, j
2. Let $\text{tmp} \leftarrow a[i]$
3. Let $a[i] \leftarrow a[j]$
4. Let $a[j] \leftarrow \text{tmp}$

FINDMAX

1. Given: Array a , locations i, j
2. Let $k \leftarrow i, \text{max} = a[k]$
3. For $l = i; i \leq j; l ++$
 1. If $a[l] > \text{max}$, $\text{max} = a[l], k = l$
4. Return k

Selection Sort

Exercise: write a recursive version

Exercise: convert this to proper C code

SELECTION SORT

1. Given: Array a with N elements
2. For $R = N - 1; R > 0; R --$ *//Initial active range is full array*
 1. $i \leftarrow \text{FINDMAX}(a, 0, R)$ *//Location of largest element in $a[0, R]$*
 2. $\text{SWAP}(a, i, R)$ *//Bring largest element to the end*

SWAP

1. Given: Array a , location i, j
2. Let $\text{tmp} \leftarrow a[i]$
3. Let $a[i] \leftarrow a[j]$
4. Let $a[j] \leftarrow \text{tmp}$

FINDMAX

1. Given: Array a , locations i, j
2. Let $k \leftarrow i, \text{max} = a[k]$
3. For $l = i; i \leq j; l ++$
 1. If $a[l] > \text{max}$, $\text{max} = a[l], k = l$
4. Return k

Time Complexity

- Let $T(N)$ be the time taken for selection sort to sort N elements
- Let $M(N)$ be the time taken to find location of max of N elements
- At any time step when active region is $[0: R]$, we do two things
 - Find the largest element within the active region – takes time $M(R + 1)$
 - Swap the largest element with the element at $a[R]$ - takes time c (const)
- Thus, we have $T(N) \leq M(N) + c + T(N - 1)$
- It is easy to show that $M(N) \leq d \cdot N$ for all N for some constant d
- Exercise: expand the recurrence as before and show that

$$T(N) \leq \mathcal{O}(N^2)$$

Assume $T(1) \leq c$

- Notice that selection sort doesn't need any extra memory (except a few tmp variables to store one integer each) – *in-place sorting*

Summary

Summary

- Applications of sorting: ranking, recommendation, internet search

Summary

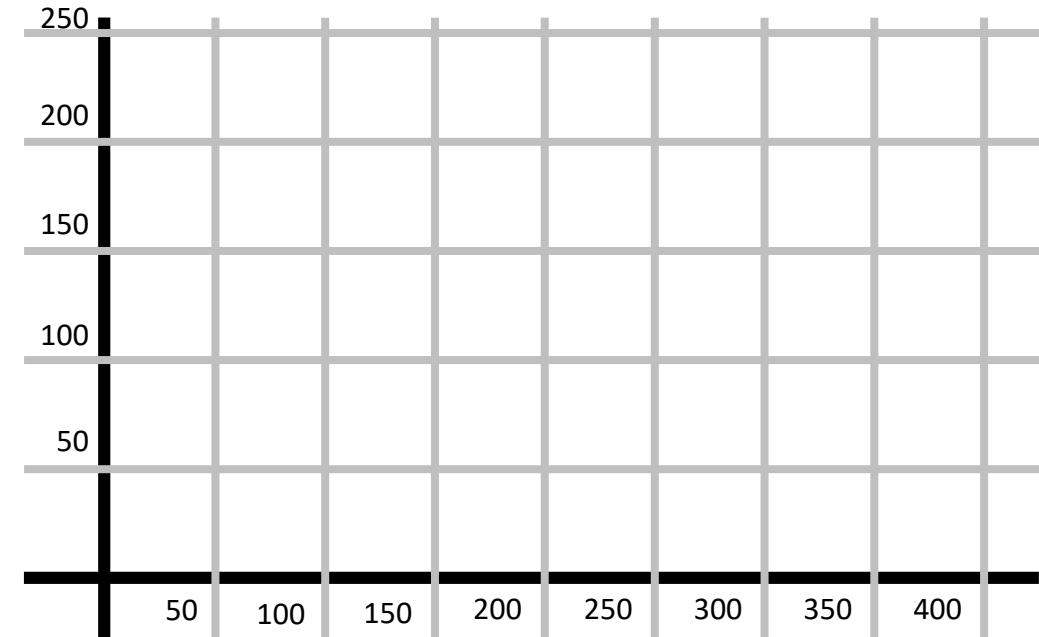
- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$

Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$

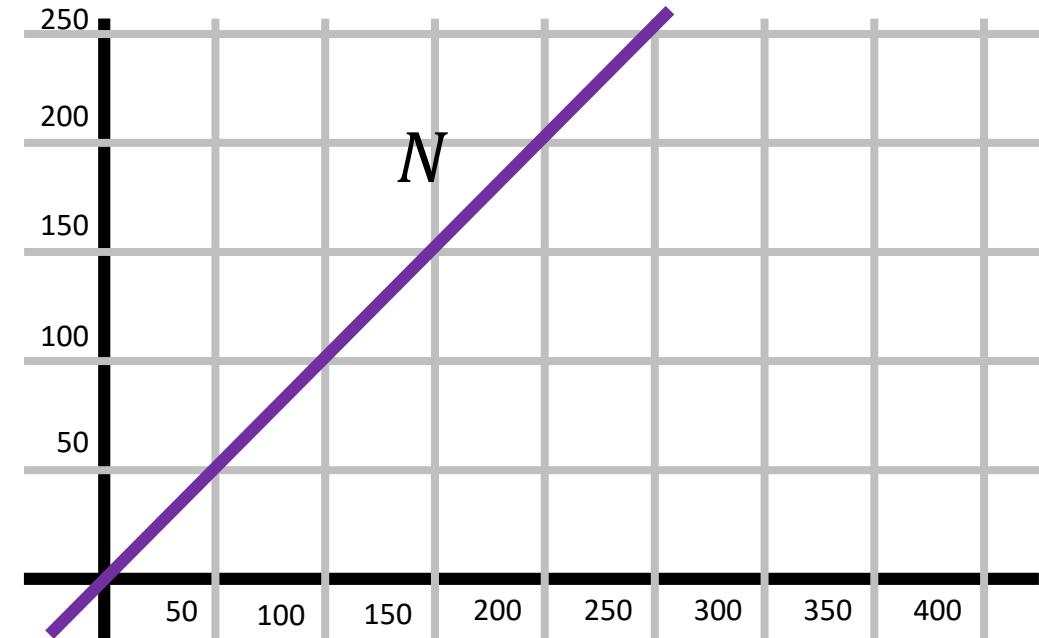
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$



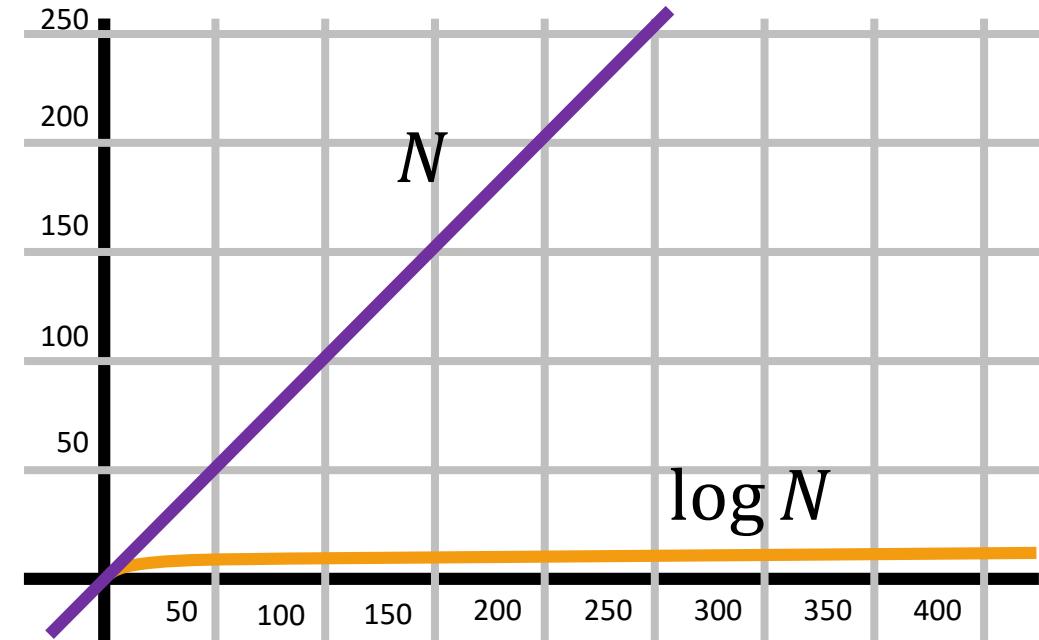
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$



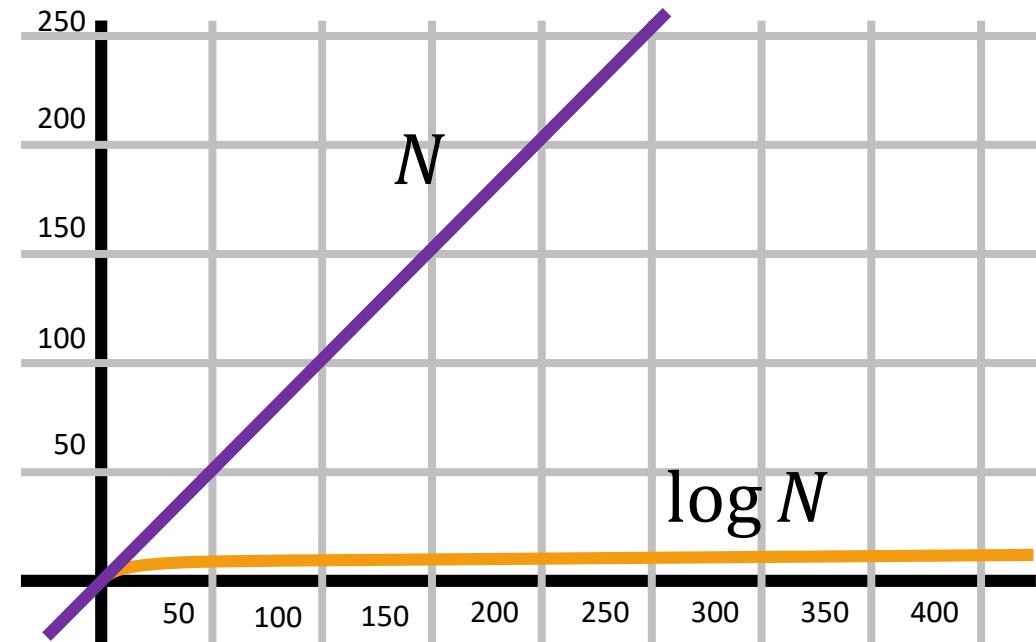
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$



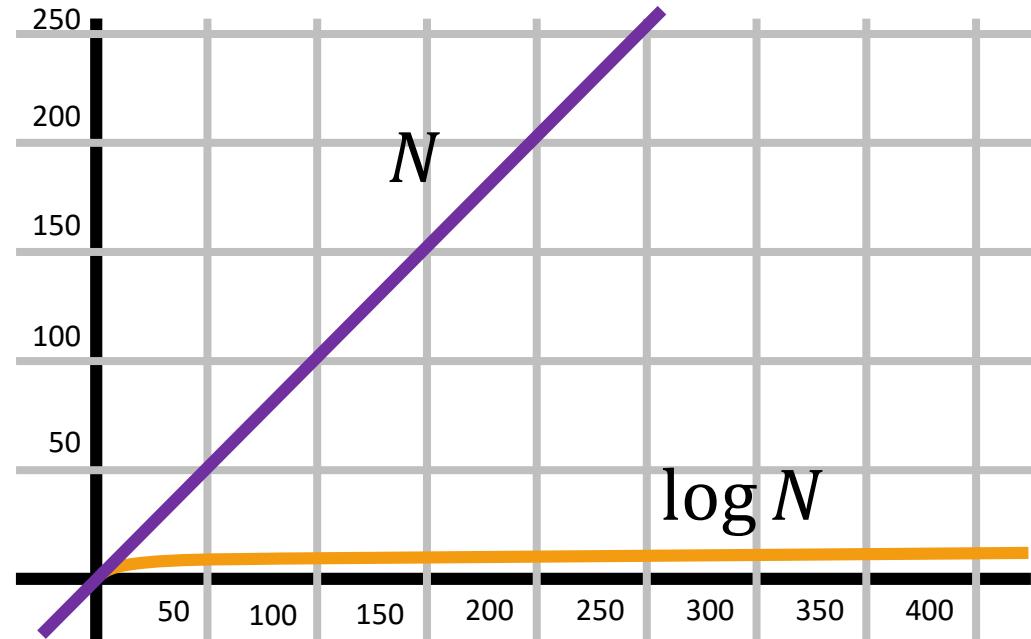
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$



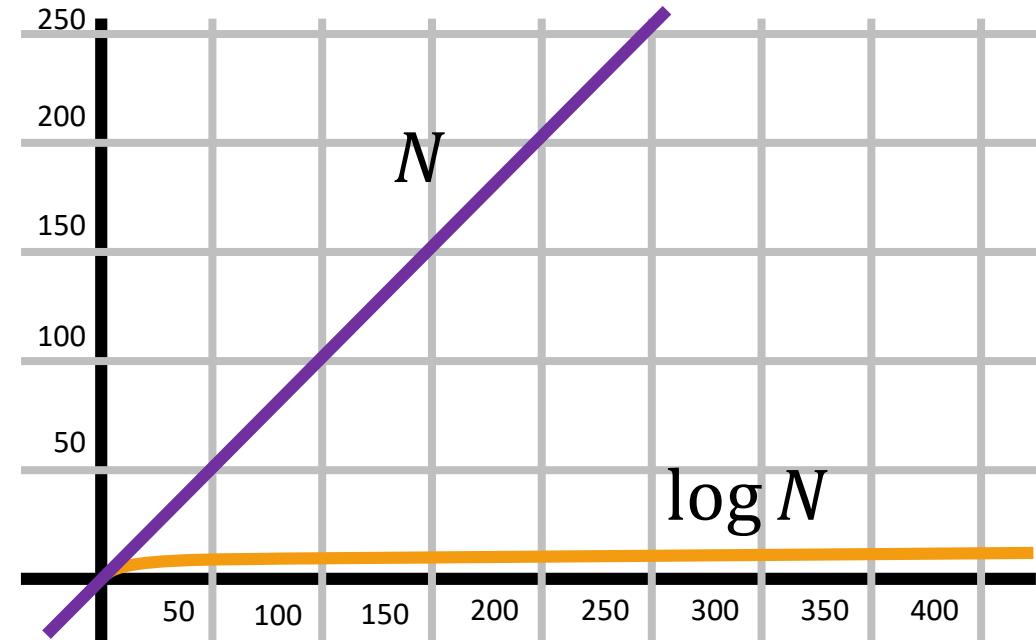
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$



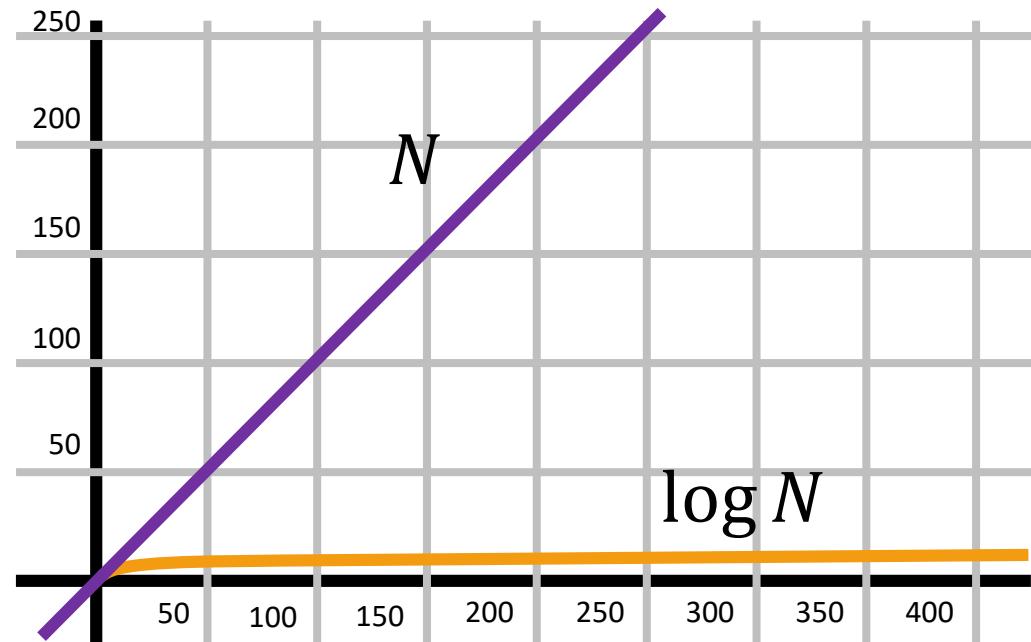
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$
 - Merge Sort



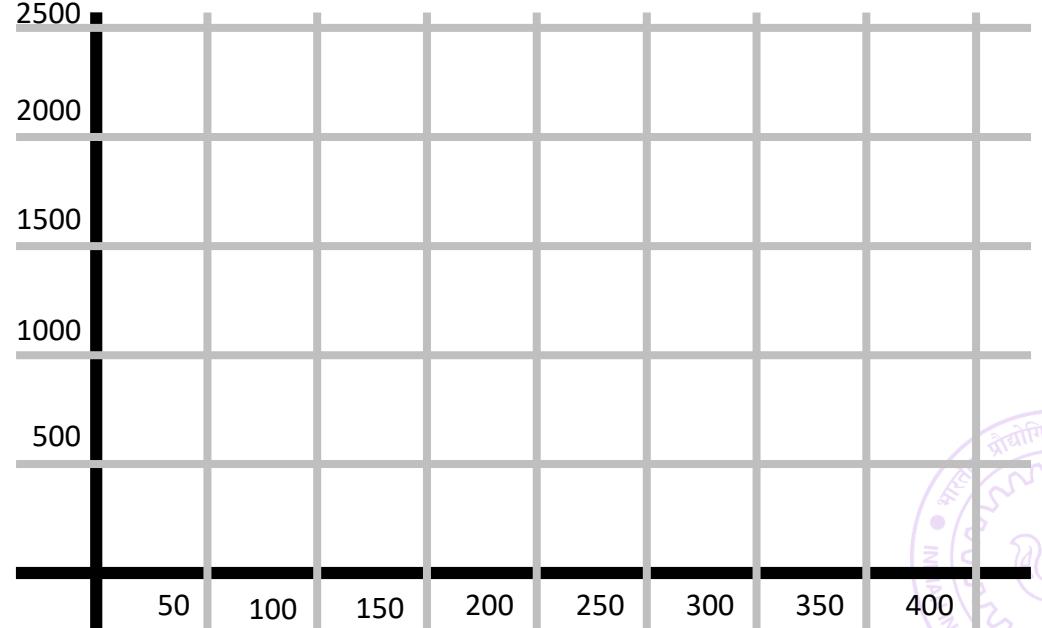
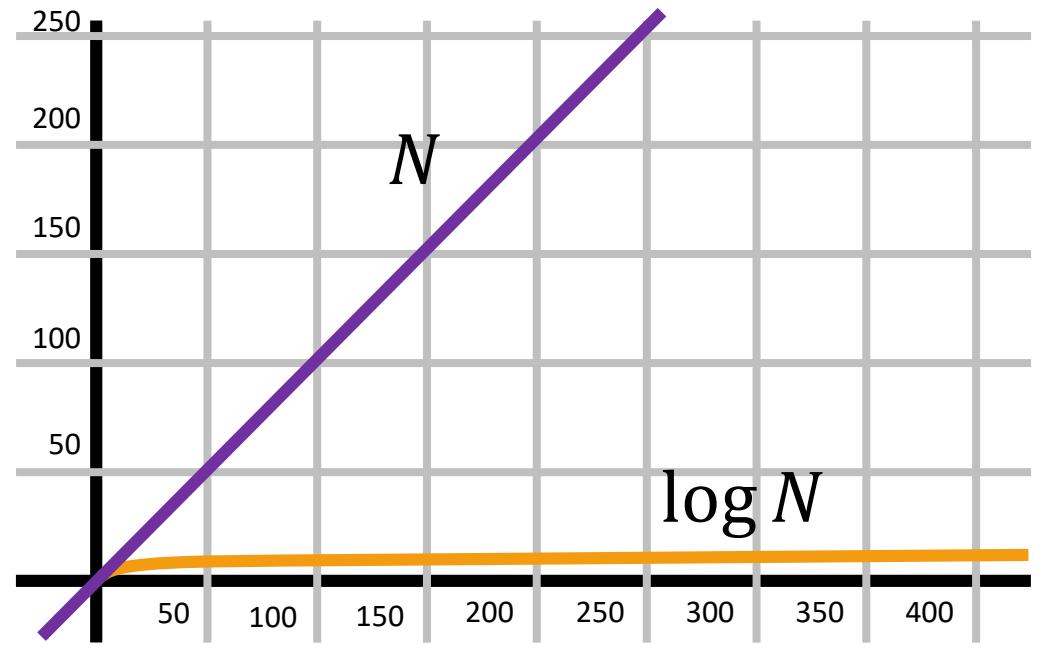
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$
 - Merge Sort
 - Quick Sort



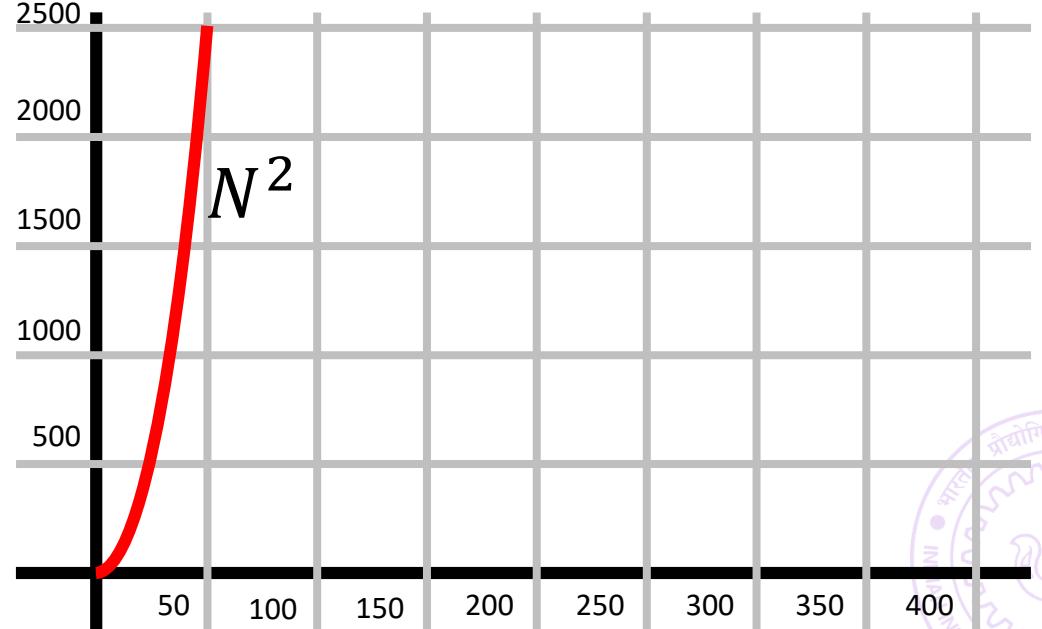
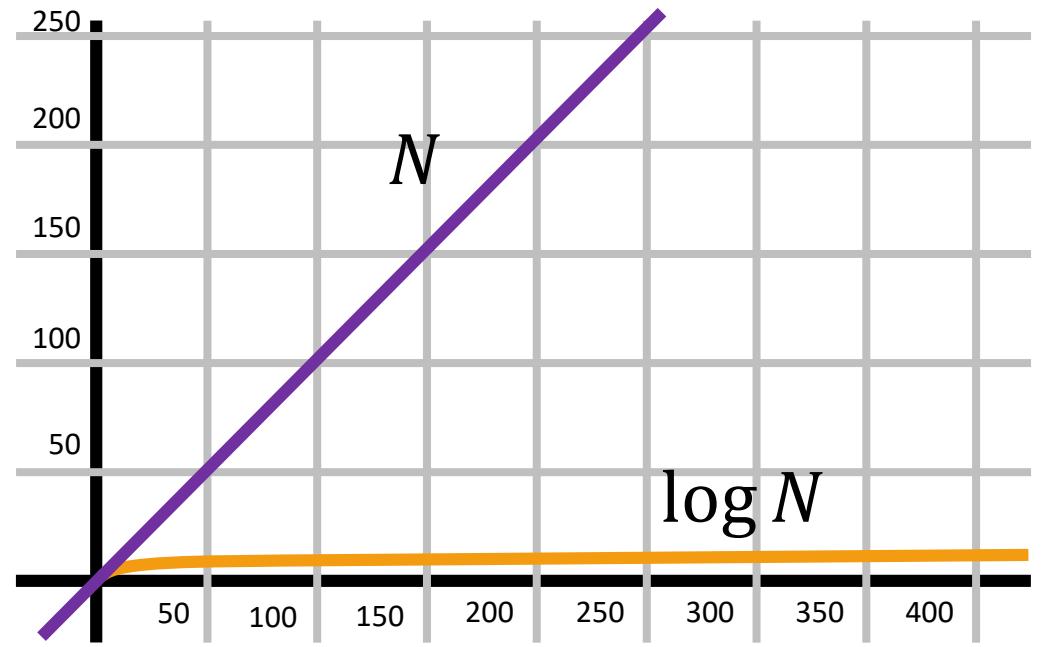
Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$
 - Merge Sort
 - Quick Sort



Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$
 - Merge Sort
 - Quick Sort



Summary

- Applications of sorting: ranking, recommendation, internet search
- Brute force search $O(N)$
- Fast searches on sorted arrays: binary search $O(\log N)$
- Selection sort $O(N^2)$
- Next: fast sorting $O(N \log N)$
 - Merge Sort
 - Quick Sort

