Data Engineering

Introduction

Logistics:

1. 2 weekly hours
2. Assignments – 20% of the final grade
3. Final Exam – 80%
4. Weekly reception hour
5. or.perets@shenkar.ac.il

Requirements:

1. Programming skills in Python
2. Design and implementation of SQL queries
3. Data structures: Hash table, binary tree, analyzing complexity time
4. Sorting algorithms: merge-sort, quick-sort
5. Searching algorithms: binary search
6. Basic probability theory: expectation, variance, conditional probability

What You Are Getting

1. Deep knowledge of cloud architecture and data engineering methods and techniques
2. Basic and advanced tools for cloud computing
3. **“From a single and local machine to the entire world”**
4. Technical skills: improvement of programming skills, queries, analysis and computational thinking

What You Are Giving

1. Homework exercises
2. Class participation
3. Final exam
4. **Smiles and fun ☺**

Introduction

What is Cloud?

Datacenter hardware and software that the vendors use to offer the computing resources and services.

What is Big Data?

Big data is a collection of large datasets that cannot be processed using traditional computing techniques. **It is not a single technique or a tool**.

Benefits

- Using the information kept in the social network like Facebook, twitter, etc.
- Using the information in the social media like preferences and product perception of their consumers.

Sources of Big Data

Social networking sites: Facebook, Google, LinkedIn

E-commerce site: Amazon, Flipkart, Alibaba

Weather Station: All the weather station and satellite gives very huge data which are stored and manipulated to forecast weather

Telecom company: Airtel, Vodafone

Share Market: Stock exchange across the world

3V's of Big Data

Velocity: The data is increasing at a very fast rate.

It is estimated that the volume of data will double in every 2 years.

Variety: Data is structured as well as unstructured.

structured – predefined schema

unstructured – each document has different properties

Volume: The amount of data which we deal with is of very large size of **Peta** bytes.

(1 Peta = 1024 TB)

Handling Big Data

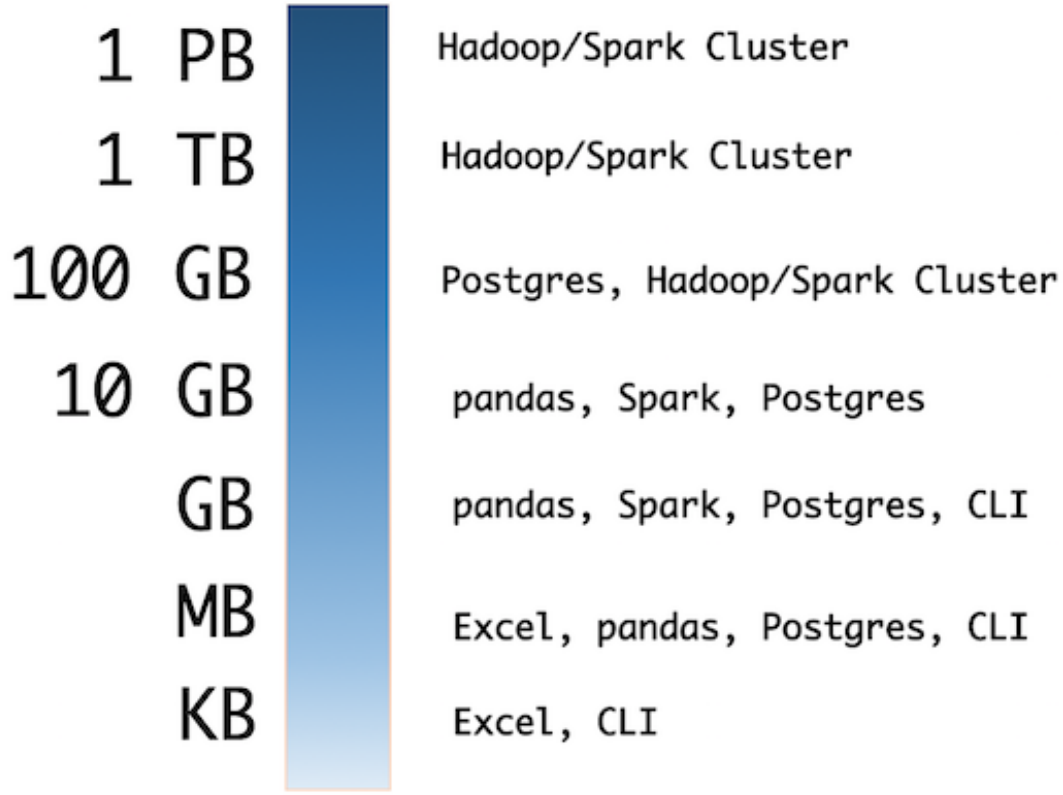
Excel

Pandas – Lecture 3

Hadoop – Lectures 4,5

Spark – Lectures 6,7

Streaming – Lectures 10,11

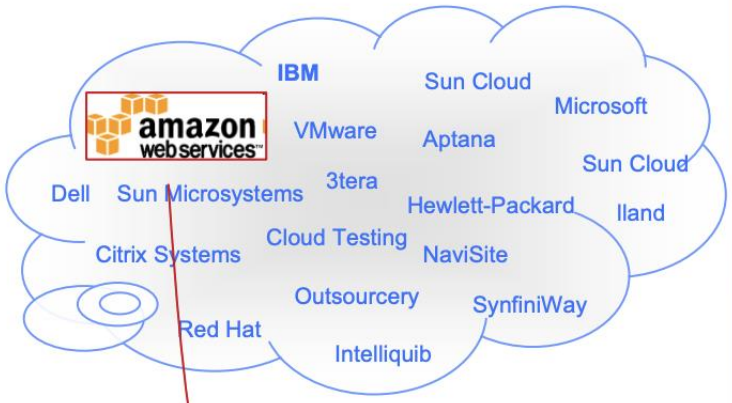


Cloud Computing

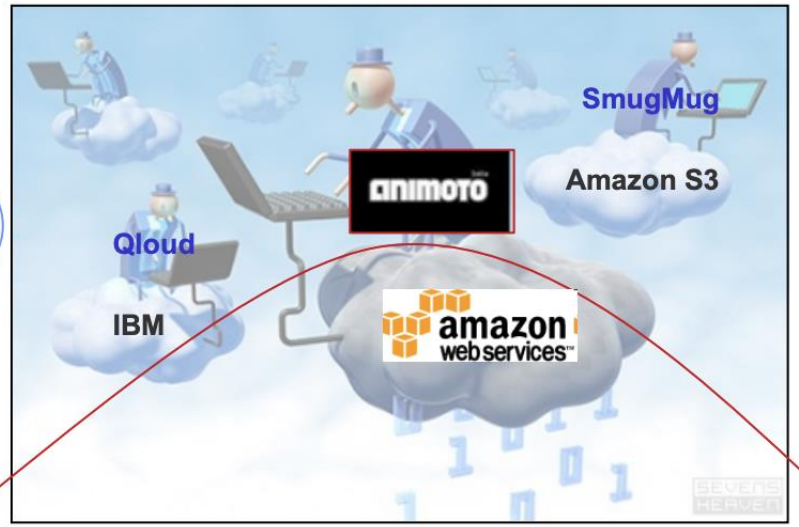
- Represents both the cloud & the provided services
- Why call it “cloud computing”?
  - Some say because the computing happens out there "in the clouds” ☺
  - Wikipedia: "the term derives from the fact that most technology diagrams depict the Internet or IP availability by using a drawing of a cloud."

Cloud Computing

Cloud providers



Cloud Users & Service Providers



Service Users

Users use it to produce video pieces from their photos, video clips and music.

Cloud Computing Services

- **S**oftware **a**s **a** **S**ervice (SaaS) – applications though the browser
- **P**latform **a**s **a** **S**ervice (PaaS) - Delivery of a computing platform for custom software development as a service
- **I**nfrastructure **a**s **a** **S**ervice (IaaS) - Deliver of computer infrastructure as a service
- And more ..

Data Scientist vs. Data Engineer

With an incredible 2.5 quintillion bytes of data generated daily,

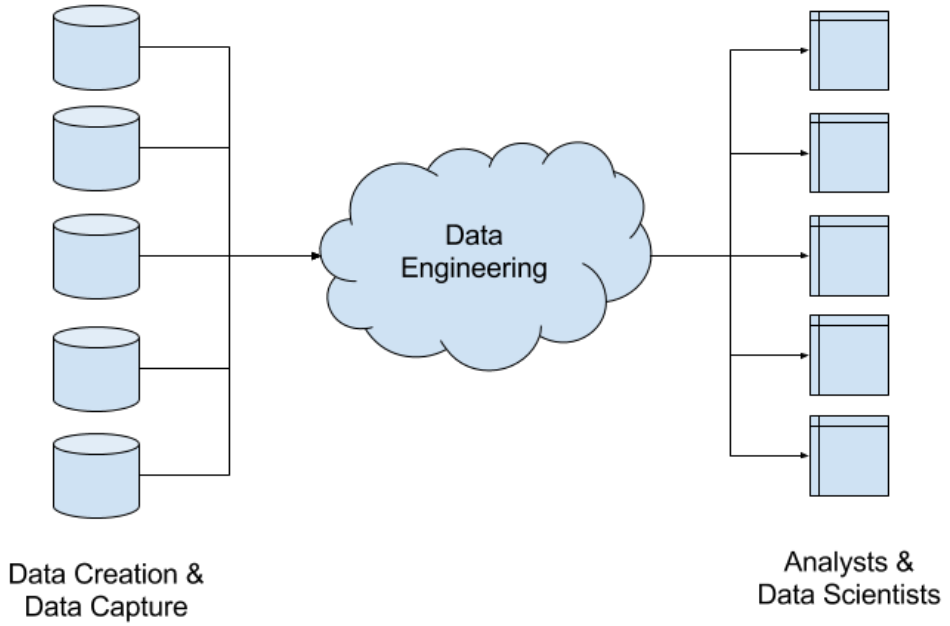
data scientists are busier than ever:

- Provides methods to make use of this data
- getting data for analysis to produce meaningful and useful insights

Data Science Moto: “The more information we have, the more we can do with it”

Data Scientist vs. Data Engineer

A **data engineer** is an engineering role within a data science team or any data related project that requires creating and managing technological infrastructure of a data platform.



Data Engineer Responsibilities

- Architecture design

data engineering entails designing the architecture of a data platform.

- Development of data related instances

customize and manage integration tools, databases, warehouses, and analytical systems.

- Data pipeline maintenance/testing

test the reliability and performance of each part of a system (or they can cooperate with the testing team).

Data Engineer Responsibilities

- Machine learning algorithm deployment

Machine learning models are designed by data scientists, data engineers are responsible for deploying those into production environments.

- Manage data and meta-data

A data engineer is managing the data stored and structuring it properly via database management systems (DBMS).

- Track pipeline stability

Monitoring the overall performance and stability of the system

Cloud Approaches

Deterministic

- Analysis of the entire database

- Advanced techniques for minimize the complexity time (lectures 2-9)

Stochastic

Streaming data analysis, sampling and estimations (lectures 10-12)

Recap – Algorithms

Algorithm

Definition: A finite set of statements that guarantees a solution in finite interval of time.

For example, instructions for coffee:

- Take a glass
- Add coffee
- Add hot-water
- Add sugar, if you want
- Enjoy ☺

Algorithm

Definition: A finite set of statements that guarantees a solution in finite interval of time.

Out goal is to find algorithm / solution to a given problem that guarantees an optimal solution (not just simple solution).

What is a “good” algorithm?

- Run in less time
- Consume less memory

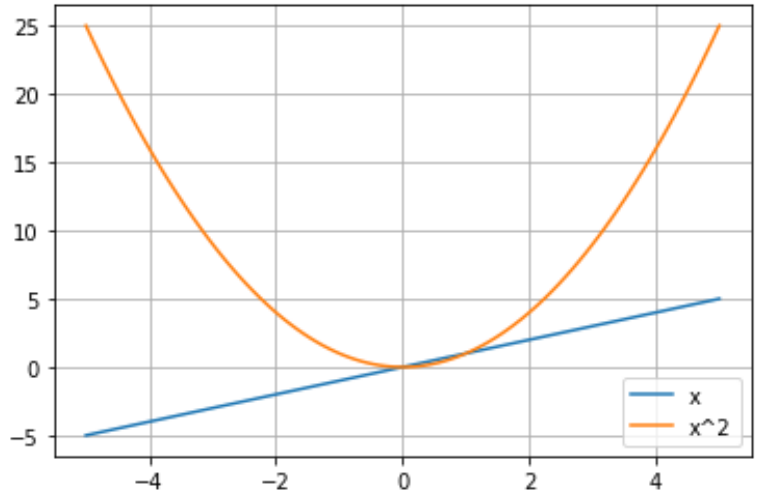
Running Time of an Algorithm

1. Depends upon - input size, nature of input
2. Generally, time grows with size of input, so running time of an algorithm is usually measured as function of input size.
3. Running time is measured in terms of number of steps/primitive operations performed
4. Independent from machine, OS

Asymptotic Notation

Most of the times, we do not need the exact number of operations. We can express the number of operations by:

1. O – upper bound
2. θ – tight bound
3. Ω – lower bound



For example, $f(x) = x \; and \; g(x) = x^2$, so $g(x)$ is the upper bound of $f(x)$

Comparing Complexity Functions

f(*n*)	Classification
1	***Constant***: run time is fixed, and does not depend upon n.
log n	***Logarithmic***: when *n* increases, so does run time, but much slower.
n	***Linear***: run time varies directly with *n*.
n log n	When *n* doubles, run time slightly more than doubles.
n^2	***Quadratic***: when *n* doubles, runtime increases fourfold.
n^3	***Cubic***: when n doubles, runtime increases eightfold.
2^n	***Exponential***: when n doubles, run time squares.

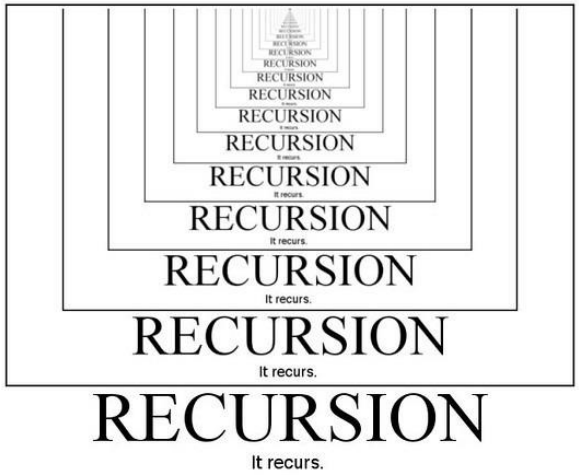
Comparing Complexity Functions

N	$\log_2 N$	$5N$	$N \log_2 N$	N^2	2^N
8	3	40	24	64	256
16	4	80	64	256	65536
32	5	160	160	1024	$\sim 10^9$
64	6	320	384	4096	$\sim 10^{19}$
128	7	640	896	16384	$\sim 10^{38}$
256	8	1280	2048	65536	$\sim 10^{76}$

Recursion

Sometimes, the best way to solve a problem is by solving a **smaller version** of the exact same problem first.

Recursion is such a technique that solves a problem by solving a smaller problems of the same type.



Recursion

For example, the **factorial** problem.

$$n! = 1 \cdot 2 \cdot 3 \cdots n$$

One way, we can implement it using loops.

Other way, we can use the recursive method:

$$4! = 4 \cdot 3!$$

$$3! = 3 \cdot 2!$$

$$2! = 2 \cdot 1!$$

$$1! = 1 \cdot 0!$$

When to stop?

Recursion

1. **Stop condition** – what is the smallest case we know?

2. **Recursive step** – how to divide the problem?

```
def factorial(number):
  if number == 0:
    return 1
  return number * factorial(number - 1)
```

Complexity Time of Recursive Algorithms

```python
def func(number):
  if number == 0:
    return 1
  return number * func(number // 2)
```

Let $T(n)$ the recurrence equation of this algorithm.

Each iteration, the algorithm performed $O(1)$ operations (if) and call to the same problem with $T\left(\frac{n}{2}\right)$, total:

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

How To Solve?

- If the recursion has division difference $\left(\frac{n}{b}\right)$: **Master theorem**

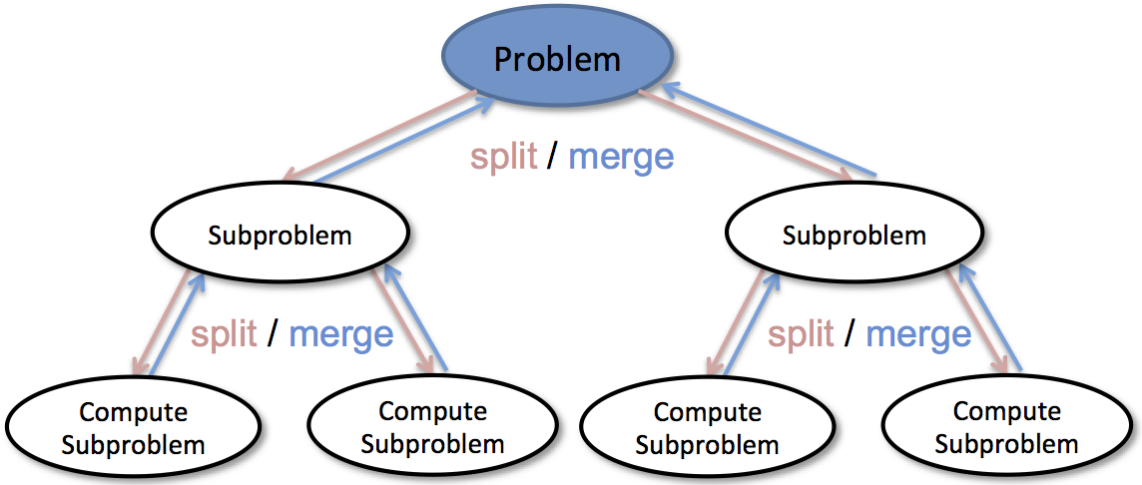
Let $T(n) = a \cdot T\left(\frac{n}{b}\right) + n^k$ be a recurrence relation, then:

$a > b^k$	$T(n) = \Theta\left(n^{\log_b a}\right)$
$a = b^k$	$T(n) = \Theta\left(n^k \cdot logn\right)$
$a < b^k$	$T(n) = \Theta\left(n^k\right)$

Divide and Conquer

A method of designing algorithms that (informally) proceeds as follows:

- Split the problem into several, sub-instances of the same problem.
- Independently solve each of the sub-instances and then combine the sub-instance solutions to yield a solution for the original instance.



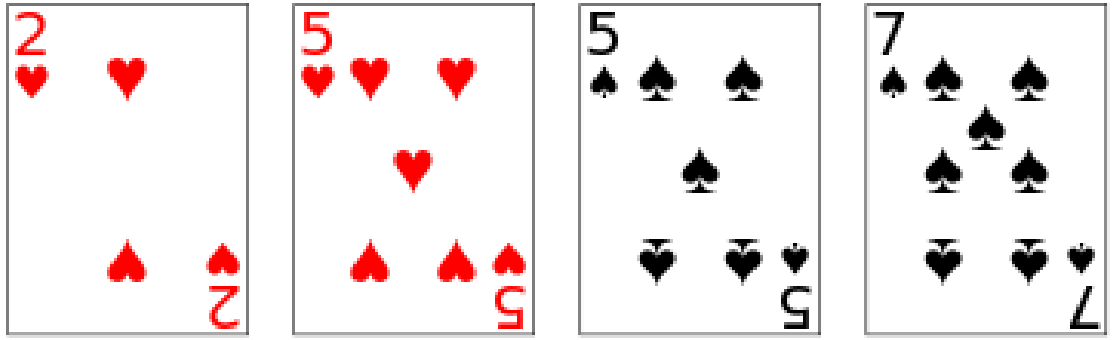
Divide and Conquer

Divide the problem into a number of sub-problems (similar to the original problem but smaller)

Conquer the sub-problems by solving them recursively (if a sub-problem is small enough, just solve it in a straightforward manner).

Combine the solutions for the sub-problems into a solution for the original problem

The Sorting Problem



The Sorting Problem

Arrangement of values from small to large **or** large to small.

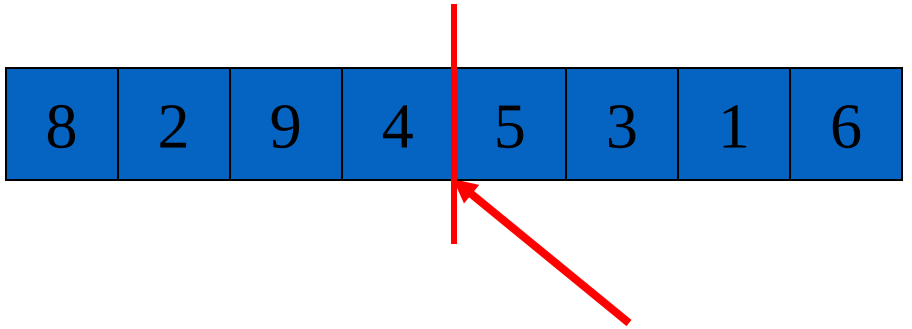
For a list of numbers, sorting in ascending order is sorting from smallest to largest.

For a list of letters, sorting in descending order is sorting from the last letter to the first letter.

For a list of words, we will sort them lexicographically (according to the dictionary).

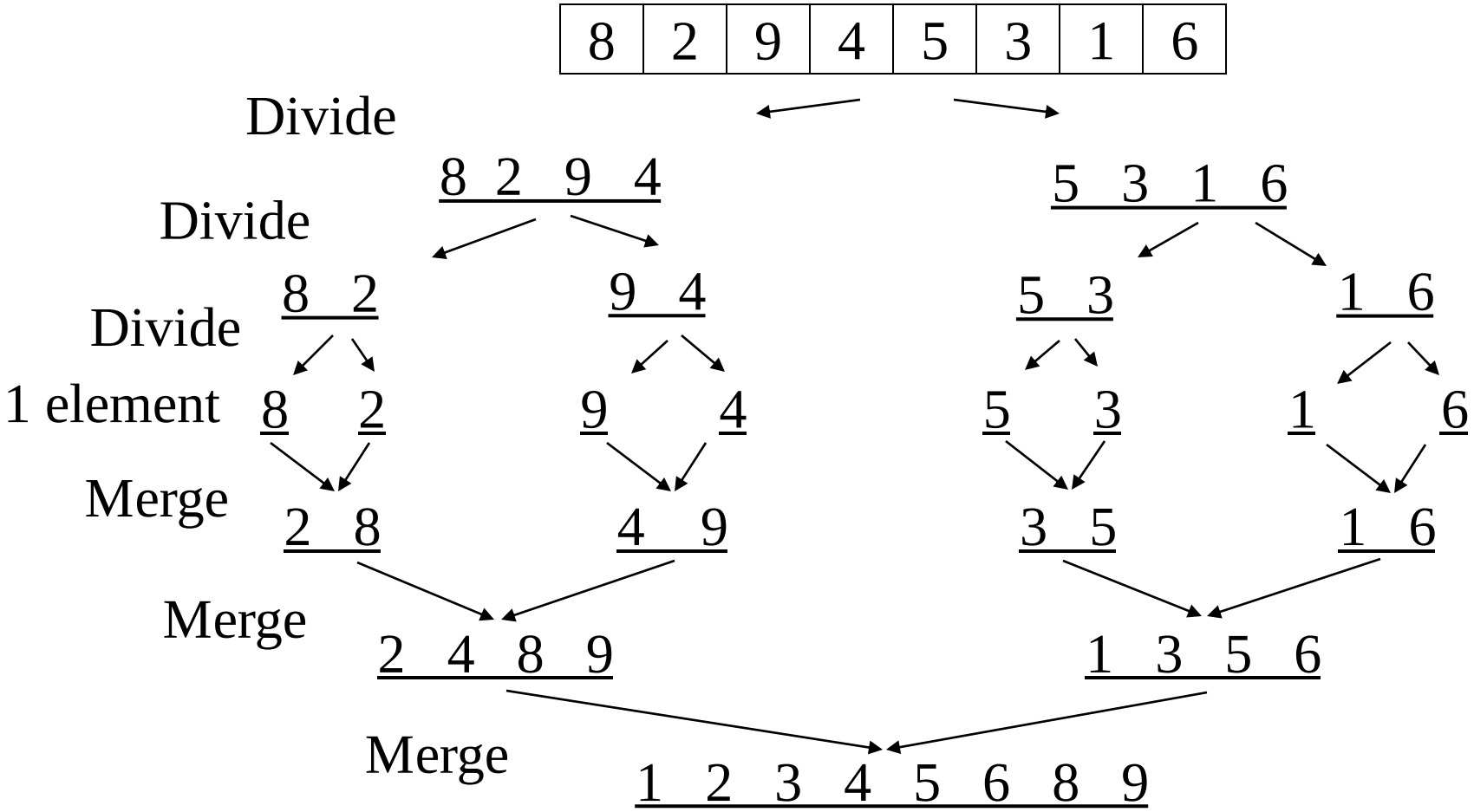
Merge Sort – Divide and Conquer

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves.



Process: divide it in two at the midpoint, conquer each side in turn (by recursively sorting), merge two halves together.

Merge Sort – Divide and Conquer



Merge-sort Analysis

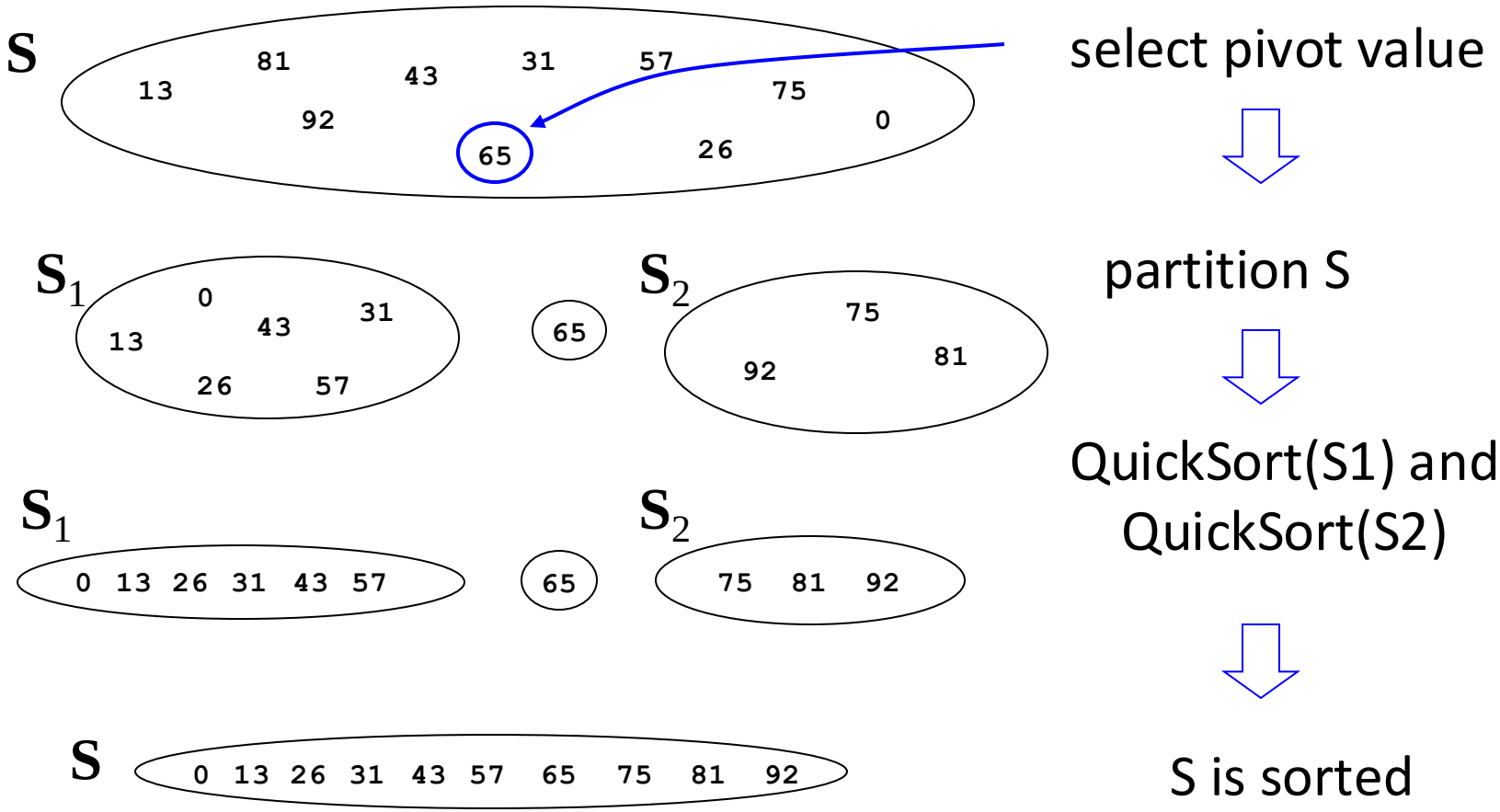
- Let $T(N)$ be the running time for an array of N elements
- Mergesort divides array in half and calls itself on the two halves.
- After returning, it merges both halves using a temporary array
- Each recursive call takes $T\left(\frac{N}{2}\right)$ and merging takes $O(N)$
- Total of $O(n \log n)$

Quicksort

Quicksort uses a divide and conquer strategy:

- Partition array into left and right sub-arrays
- Choose an element of the array, called pivot
- the elements in left sub-array are all less than pivot
- elements in right sub-array are all greater than pivot
- Recursively sort left and right sub-arrays
- Concatenate left and right sub-arrays in O(1) time

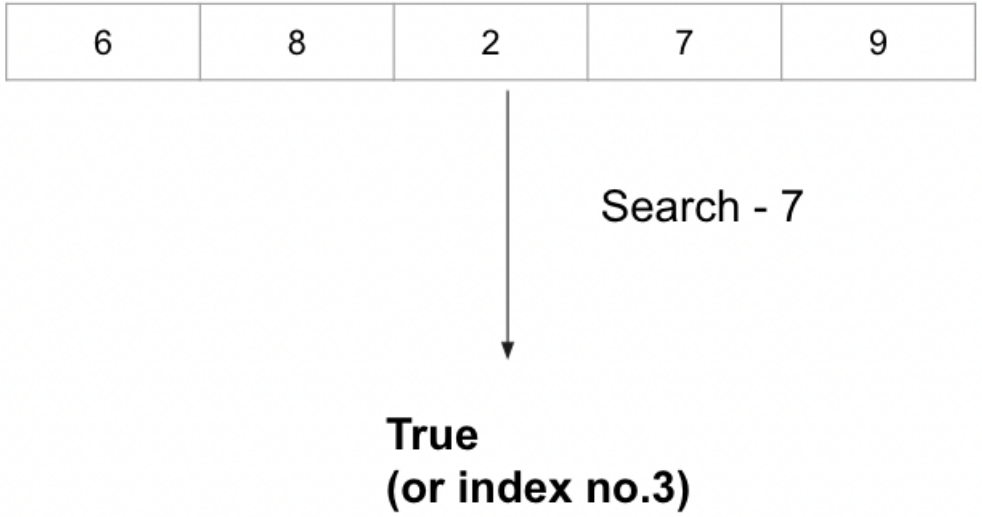
The steps of QuickSort



Searching

Searching is the process of looking for a particular value in a collection.

For example, a program that maintains a membership list for a club might need to look up information for a particular member – this involves some sort of search process.

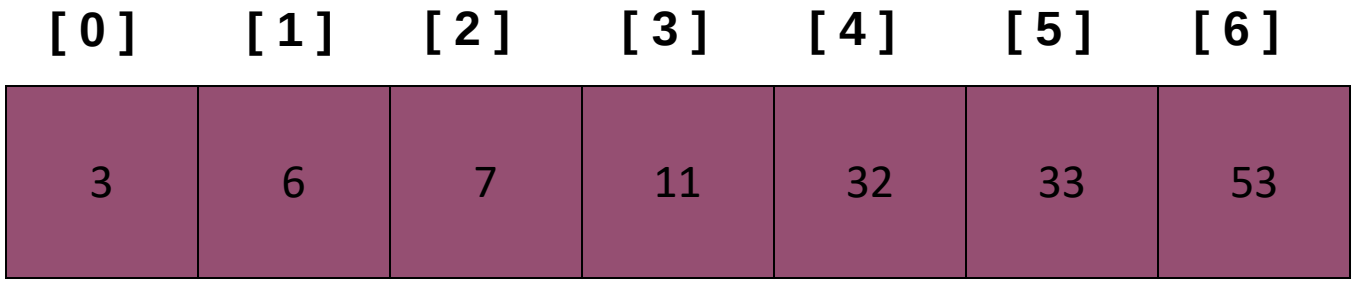


Binary Search

- Perhaps we can do better than O(n) in the average case?
- Assume that we are give an array of records that is sorted. For instance:
  - an array of records with integer keys sorted from smallest to largest (e.g., ID numbers), or
  - an array of records with string keys sorted in alphabetical order (e.g., names).

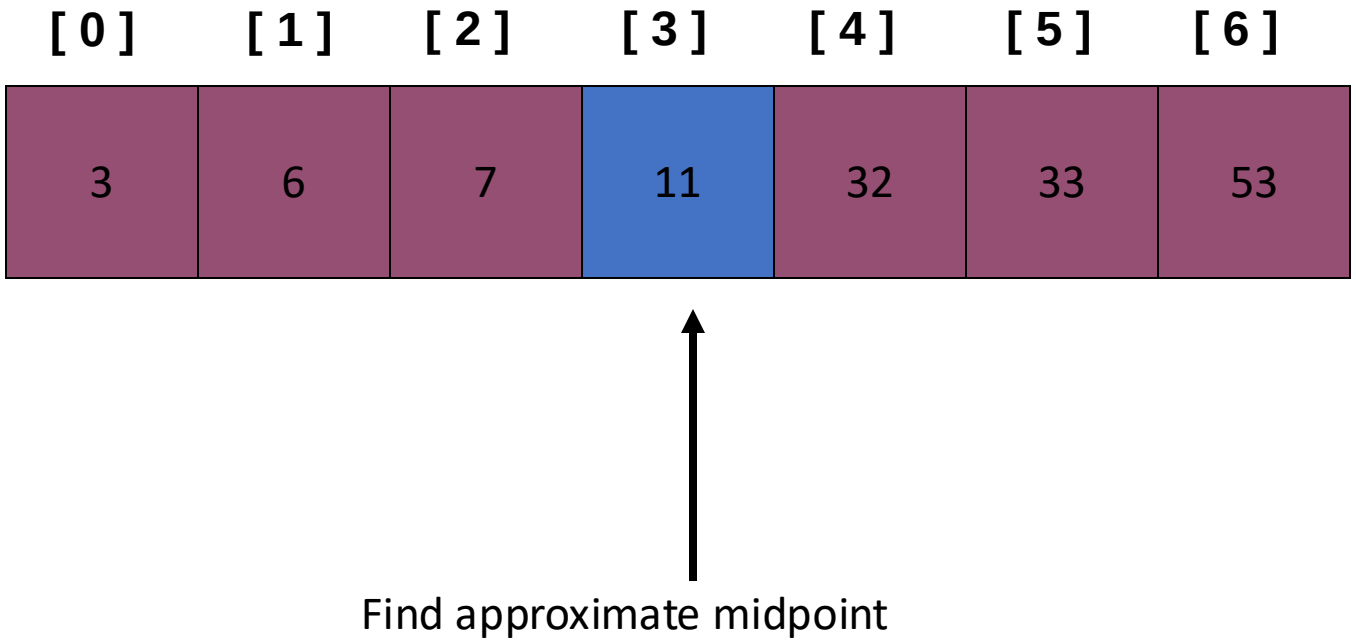
Binary Search

Example: sorted array of integer keys. Target=7.



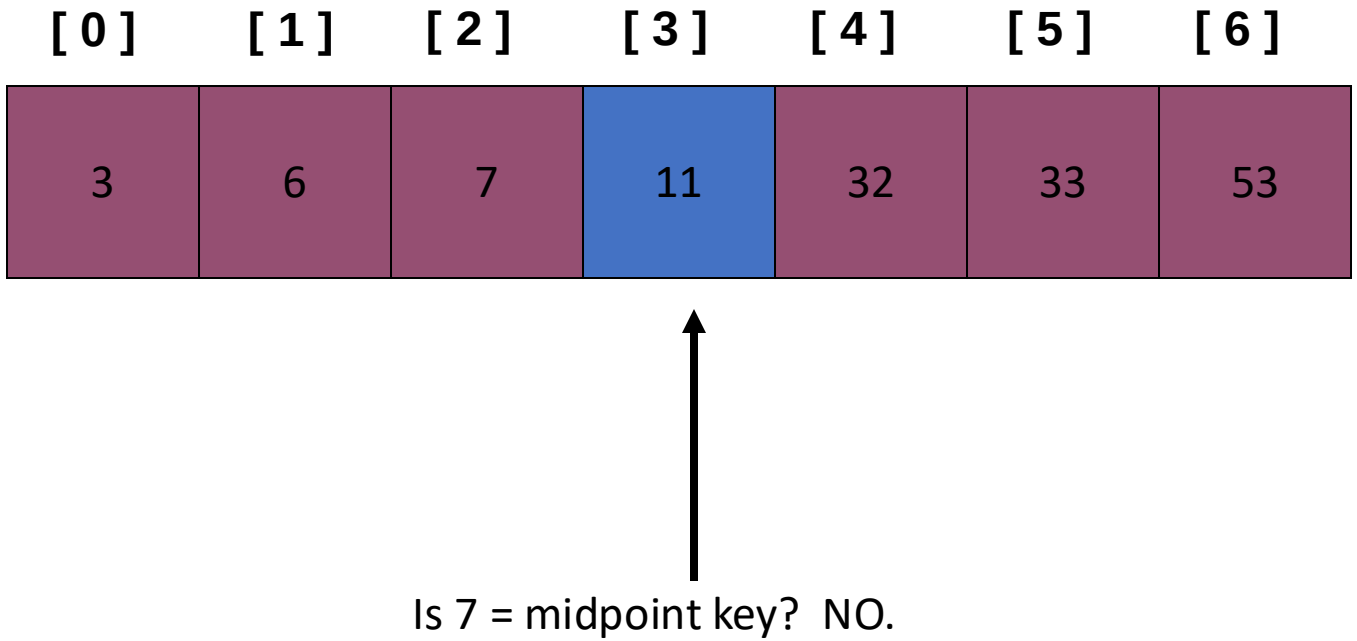
Binary Search

Example: sorted array of integer keys. Target=7.



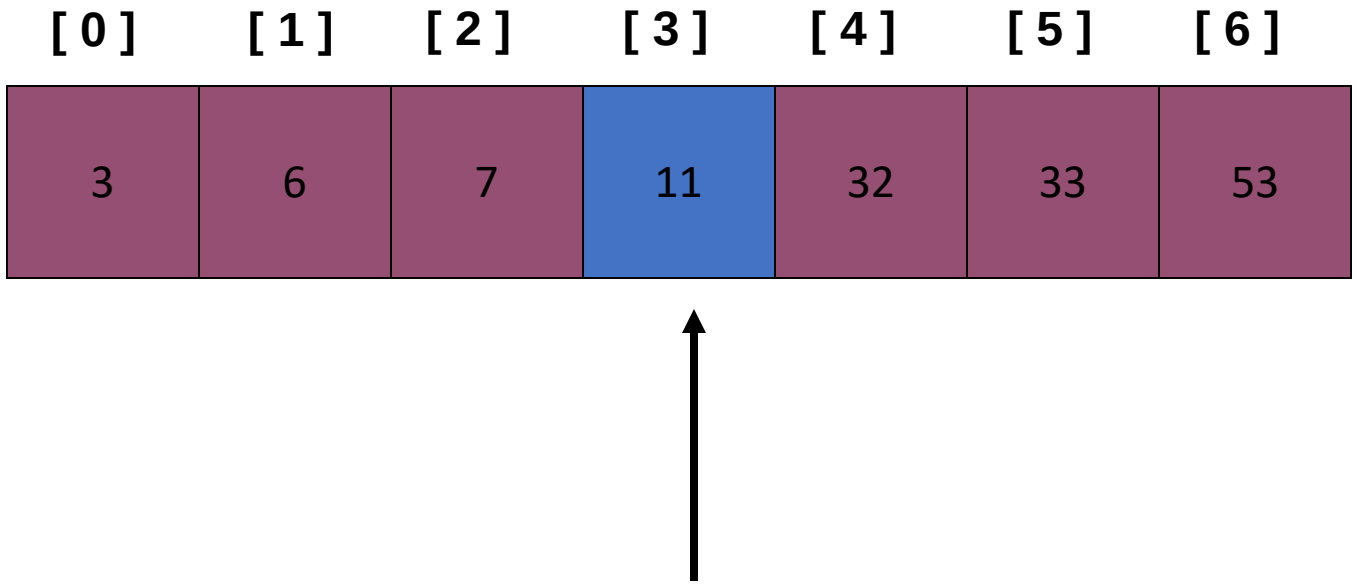
Binary Search

Example: sorted array of integer keys. Target=7.



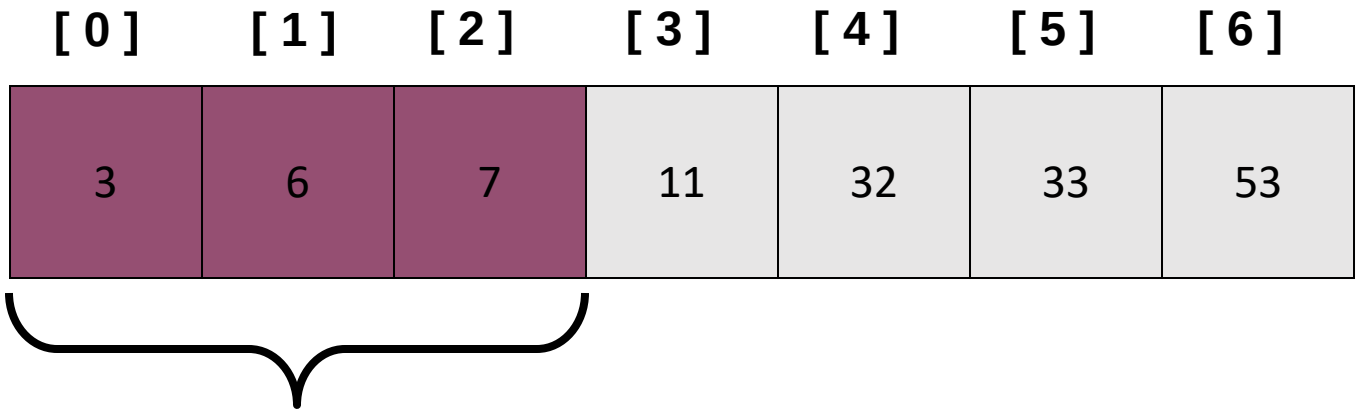
Binary Search

Example: sorted array of integer keys. Target=7.



Binary Search

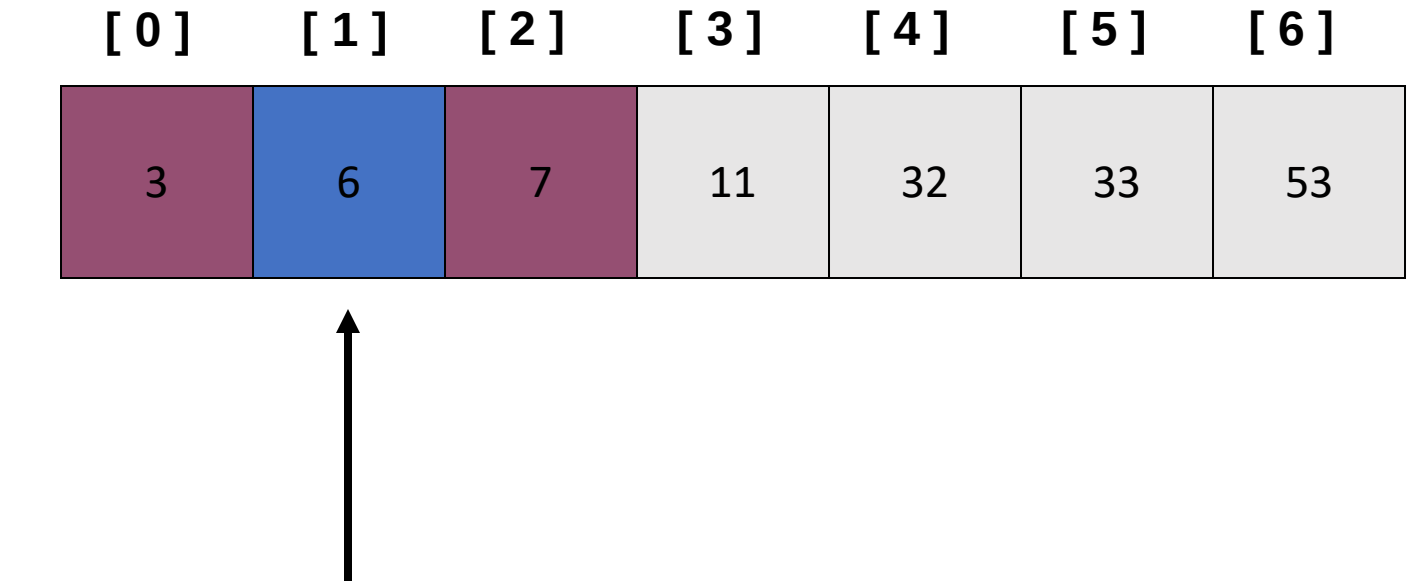
Example: sorted array of integer keys. Target=7.



Search for the target in the area before midpoint.

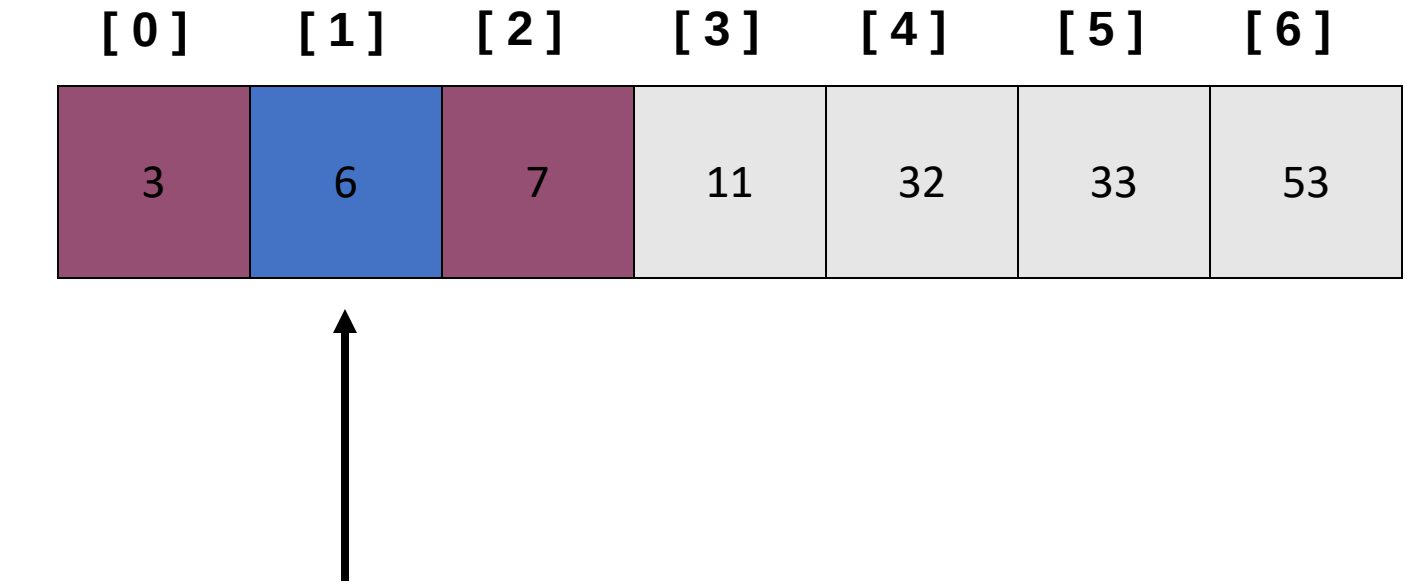
Binary Search

Example: sorted array of integer keys. Target=7.



Binary Search

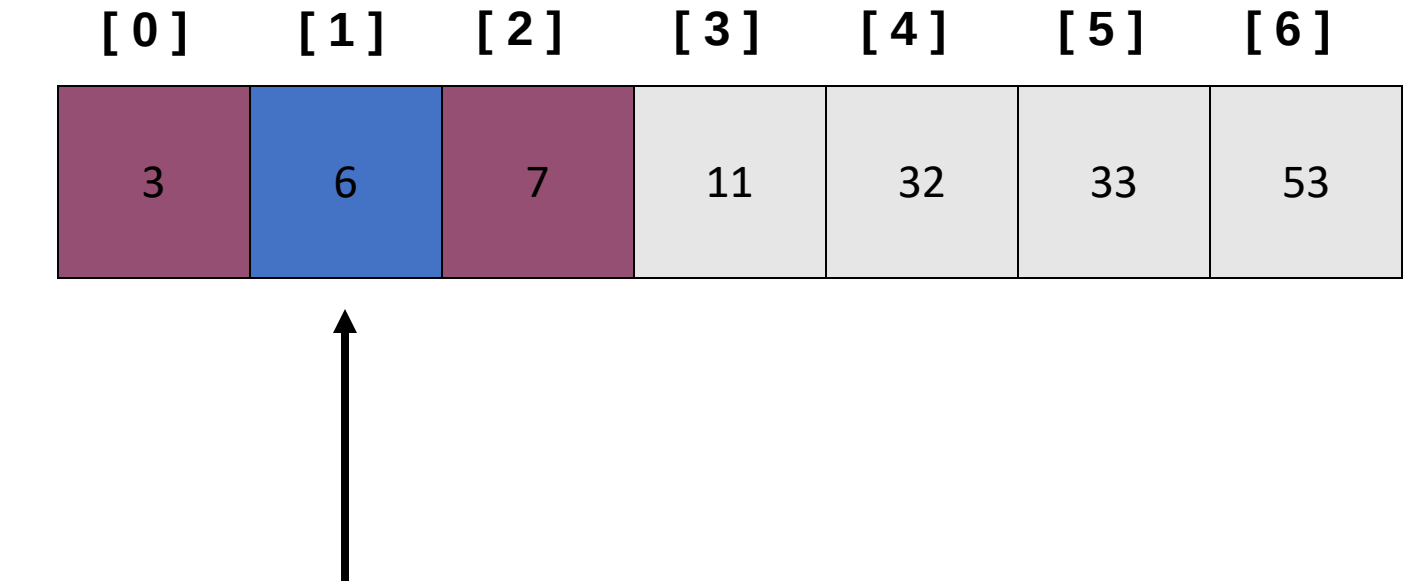
Example: sorted array of integer keys. Target=7.



Target = key of midpoint? NO.

Binary Search

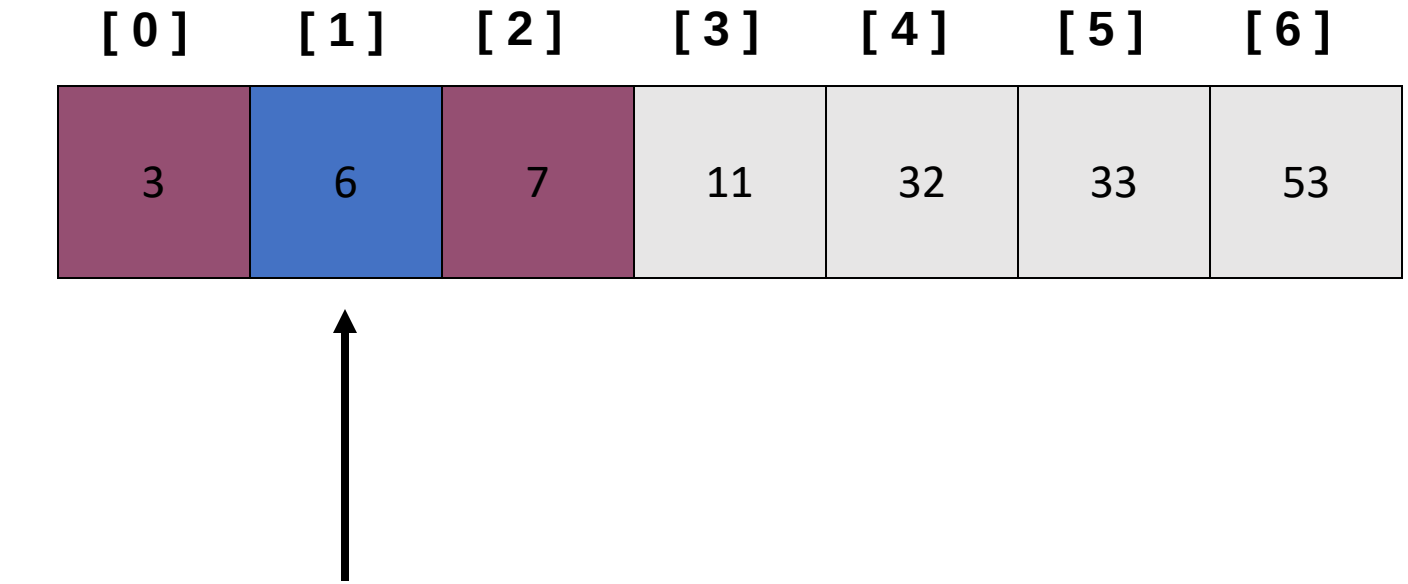
Example: sorted array of integer keys. Target=7.



Target < key of midpoint? NO.

Binary Search

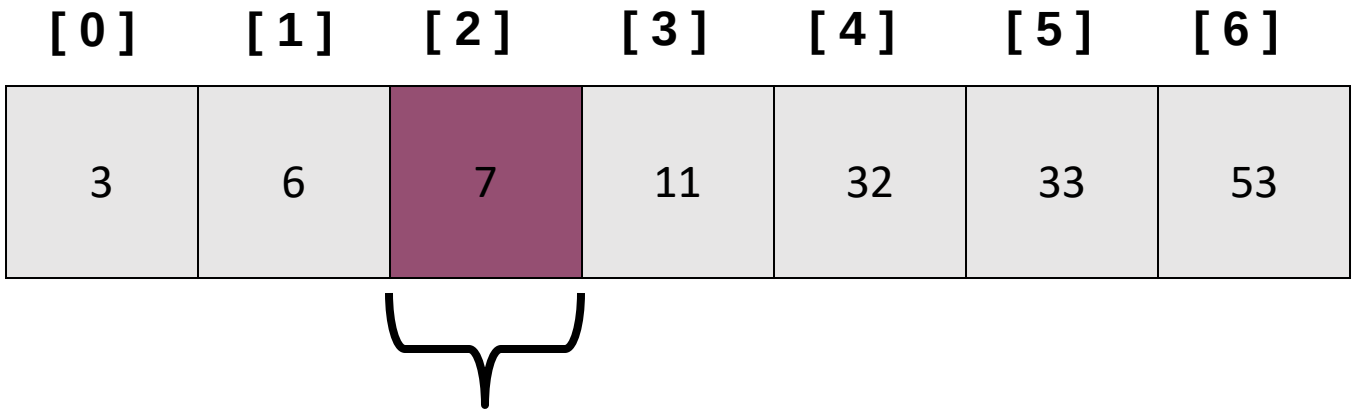
Example: sorted array of integer keys. Target=7.



Target > key of midpoint? YES.

Binary Search

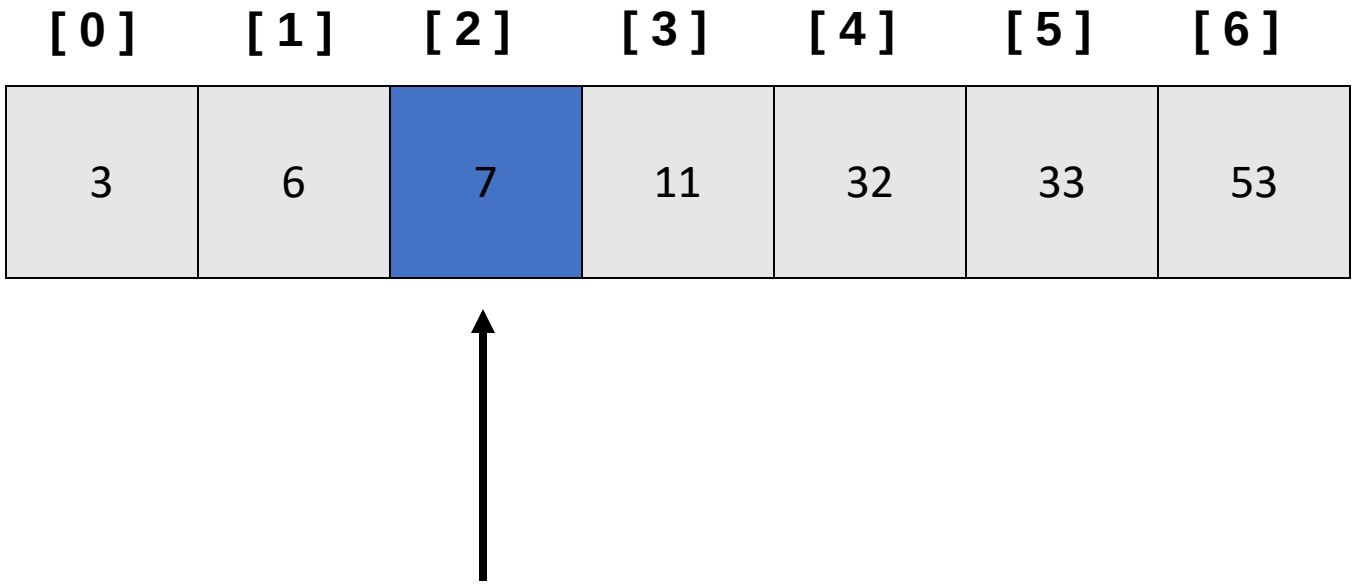
Example: sorted array of integer keys. Target=7.



Search for the target in the area after midpoint.

Binary Search

Example: sorted array of integer keys. Target=7.



Binary Search: Analysis

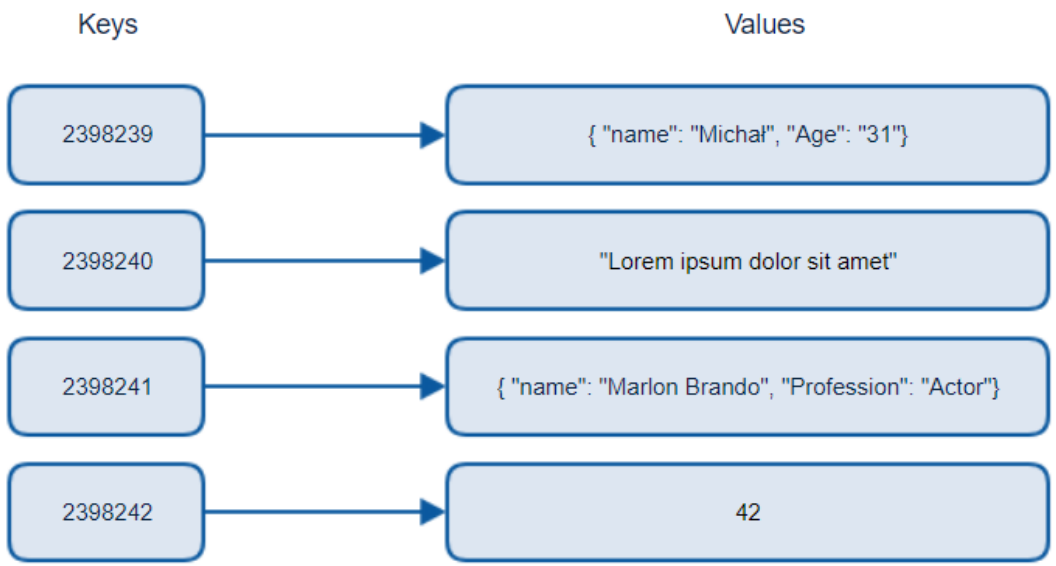
- Each level in the recursion, we split the array in half (divide by two), thus:

$$T(n) = T\left(\frac{n}{2}\right) + O(1)$$

- By Master theorem, $O(\log_2 n)$

Hash Tables

- Provides virtually direct access to objects based on a key (a unique String or Integer).
- Must have unique keys



Hash Tables: Runtime Efficient

Lookup time does not grow when n increases

- A hash table supports
  - fast insertion O(1)
  - fast retrieval O(1)
  - fast removal O(1)

Recap – Probability

Expectation

Let X be a discrete random variable with values $x_1, x_2, \ldots, x_n$.

$$E[X] = \sum_{i=1}^{n} x_i \cdot P(X = x_i)$$

Linearity: **ALWAYS**, no matter what are the relations between X and Y!

$$E[X + Y] = E[X] + E[Y]$$

$$E[cX] = cE[X]$$

Variance

Let X be a discrete random variable with values $x_1, x_2, \dots, x_n$.

Variance:

$$V[X] = E[(X - E[X])^2] = E[X^2] - (E[X])^2$$

$$V[cX] = c^2 V[X]$$

Independence

Let X, Y be random variables, then X, Y are independent if and only if

$$P[X = x_i, Y = y_i] = P[X = X_i] \cdot P[Y = y_i]$$

Conditional Probability

Let A, B two events, then:

$$P[A|B] = \frac{P[A \cap B]}{P[B]}$$

Union Bounds

Let $E_1, E_2, \ldots, E_n$ events, then:

$$P[E_1 \cup E_2 \cup \cdots \cup E_n] \leq P[E_1] + P[E_2] + \cdots + P[E_n]$$