Why Efficiency Matters

Madhavan Mukund

https://www.cmi.ac.in/~madhavan

Programming, Data Structures and Algorithms using Python

Week 1

A real world problem

- Every SIM card needs to be linked to an Aadhaar card

A real world problem

- Every SIM card needs to be linked to an Aadhaar card
- Validate the Aadhaar details for each SIM card

A real world problem

- Every SIM card needs to be linked to an Aadhaar card
- Validate the Aadhaar details for each SIM card
- Simple nested loop

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Every SIM card needs to be linked to an Aadhaar card
- Validate the Aadhaar details for each SIM card
- Simple nested loop
- How long will this take?
  - M SIM cards, N Aadhaar cards
  - Nested loops iterate $M \cdot N$ times

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Every SIM card needs to be linked to an Aadhaar card
- Validate the Aadhaar details for each SIM card
- Simple nested loop
- How long will this take?
  - M SIM cards, N Aadhaar cards
  - Nested loops iterate $M \cdot N$ times
- What are M and N
  - Almost everyone in India has an Aadhaar card: $N > 10^9$
  - Number of SIM cards registered is similar: $M > 10^9$

```
for each SIM card S:
  for each Aadhaar number A:
    check if Aadhaar details of S
    match A
```

A real world problem

- Assume $M = N = 10^9$

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes
  - $(1.67 \times 10^9)/60 \approx 2.8 \times 10^7$ hours

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes
  - $(1.67 \times 10^9)/60 \approx 2.8 \times 10^7$ hours
  - $(2.8 \times 10^7)/24 \approx 1.17 \times 10^6$ days

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes
  - $(1.67 \times 10^9)/60 \approx 2.8 \times 10^7$ hours
  - $(2.8 \times 10^7)/24 \approx 1.17 \times 10^6$ days
  - $(1.17 \times 10^6)/365 \approx 3200$ years!

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

A real world problem

- Assume $M = N = 10^9$
- Nested loops execute 10^{18} times
- We calculated that Python can perform 10^7 operations in a second
- This will take at least 10^{11} seconds
  - $10^{11}/60 \approx 1.67 \times 10^9$ minutes
  - $(1.67 \times 10^9)/60 \approx 2.8 \times 10^7$ hours
  - $(2.8 \times 10^7)/24 \approx 1.17 \times 10^6$ days
  - $(1.17 \times 10^6)/365 \approx 3200$ years!
- How can we fix this?

```
for each SIM card S:
  for each Aadhaar number A:
     check if Aadhaar details of S
     match A
```

Guess my birthday

- You propose a date

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?
- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes
- Interval shrinks from $365 \to 182 \to 91 \to 45 \to 22 \to 11 \to 5 \to 2 \to 1$

Guess my birthday

- You propose a date
- I answer, *Yes*, *Earlier*, *Later*
- Suppose my birthday is 12 April
- A possible sequence of questions
  - September 12? Earlier
  - February 23? Later
  - July 2? Earlier
  - ...
- What is the best strategy?

- Interval of possibilities
- Query midpoint — halves the interval
  - June 30? Earlier
  - March 31? Later
  - May 15? Earlier
  - April 22? Earlier
  - April 11? Later
  - April 16? Earlier
  - April 13? Earlier
  - April 12? Yes
- Interval shrinks from $365 \to 182 \to 91 \to 45 \to 22 \to 11 \to 5 \to 2 \to 1$
- Under 10 questions

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes
- From 3200 years to 50 minutes!

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes
- From 3200 years to 50 minutes!
- Of course, to achieve this we have to first sort the Aadhaar cards

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes
- From 3200 years to 50 minutes!
- Of course, to achieve this we have to first sort the Aadhaar cards
- Arranging the data results in a much more efficient solution

A real world problem

- Assume Aadhaar details are sorted by Aadhaar number
- Use the halving strategy to check each SIM card
- Halving 10 times reduces the interval by a factor of 1000, because $2^{10} = 1024$
- After 10 queries, interval shrinks to 10^6
- After 20 queries, interval shrinks to 10^3
- After 30 queries, interval shrinks to 1
- Total time $\approx 10^9 \times 30$

```
for each SIM card S:
  probe sorted Aadhaar list to
  check Aadhaar details of S
```

- 3000 seconds, or 50 minutes
- From 3200 years to 50 minutes!
- Of course, to achieve this we have to first sort the Aadhaar cards
- Arranging the data results in a much more efficient solution
- Both algorithms and data structures matter