

# Greedy Algorithms: Main Ideas

Michael Levin

Higher School of Economics

Algorithmic Toolbox  
Data Structures and Algorithms

# Outline

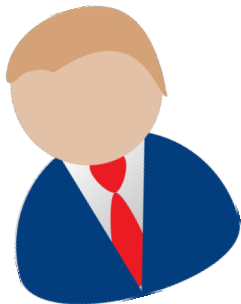
- 1 Largest Number
- 2 Car Fueling
- 3 Implementation and Analysis
- 4 Main Ingredients

# Learning objectives

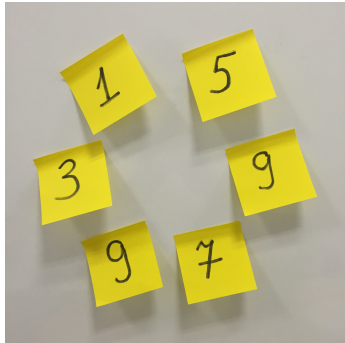
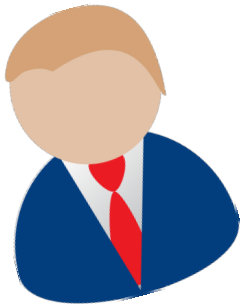
- Come up with a greedy algorithm yourself

# Job Interview

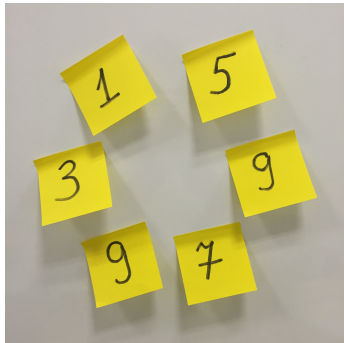
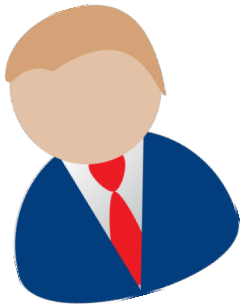
# Job Interview



# Job Interview



# Job Interview



# Largest Number

## Toy problem

What is the largest number that consists of digits 3, 9, 5, 9, 7, 1? Use all the digits.



# Largest Number

## Toy problem

What is the largest number that consists of digits 3, 9, 5, 9, 7, 1? Use all the digits.

## Examples

359179, 537991, 913579, ...

Correct answer

997531

# Greedy Strategy

5	7	3	9	1	9
---	---	---	---	---	---

 →

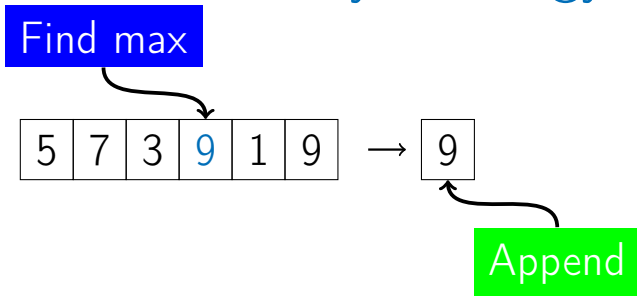
# Greedy Strategy

Find max



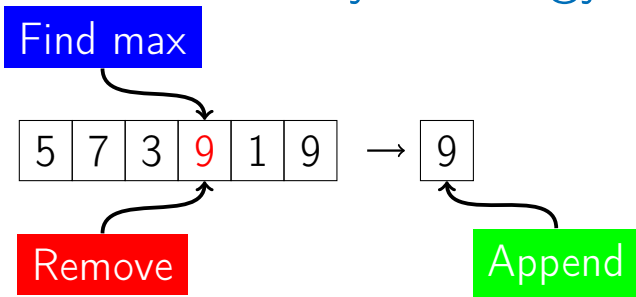
- Find max digit

# Greedy Strategy



- Find **max** digit
- **Append** it to the number

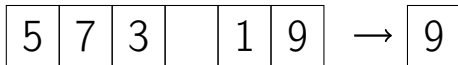
# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits

# Greedy Strategy

Find max



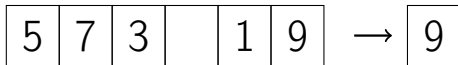
Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits

# Greedy Strategy

Find max



Remove

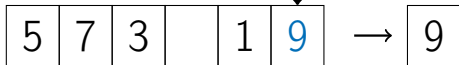
Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list



# Greedy Strategy

Find max

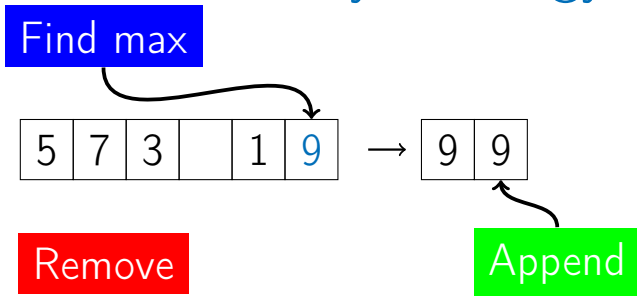


Remove

Append

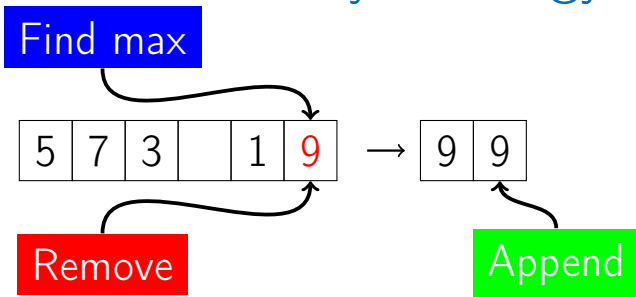
- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

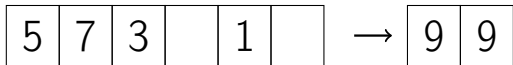
# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max



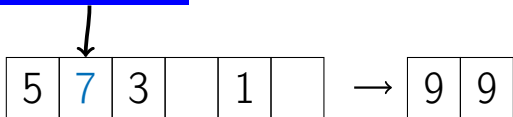
Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max

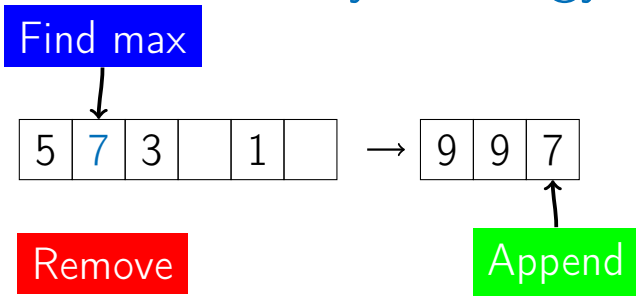


Remove

Append

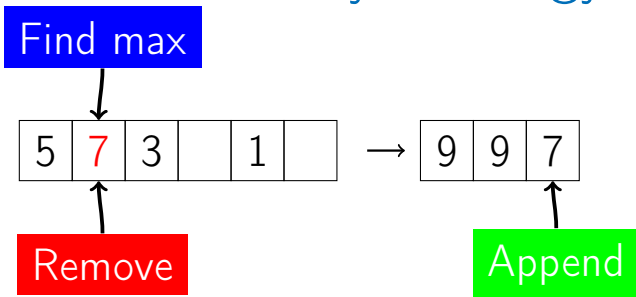
- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

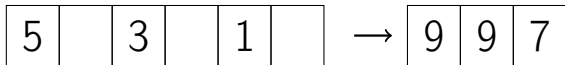
# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max



Remove

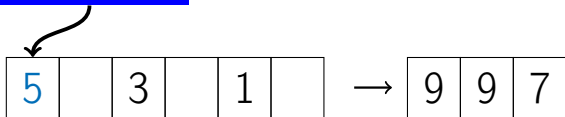
Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list



# Greedy Strategy

Find max



Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max

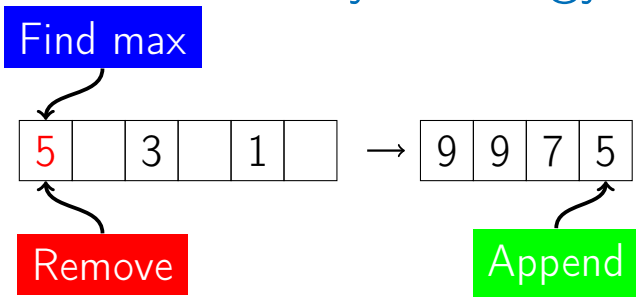


Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max



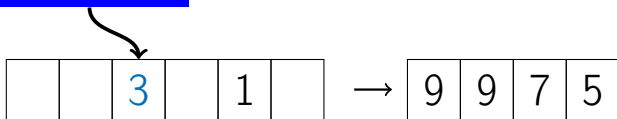
Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max

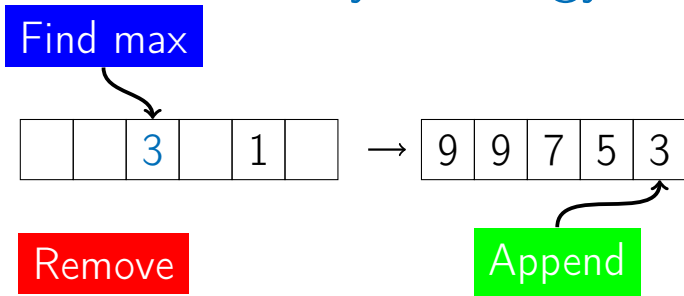


Remove

Append

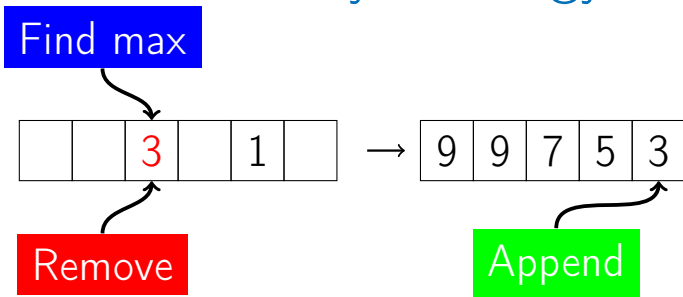
- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

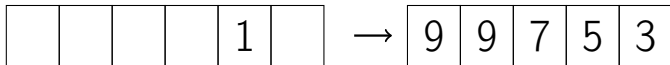
# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max



Remove

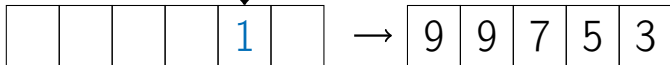
Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list



# Greedy Strategy

Find max

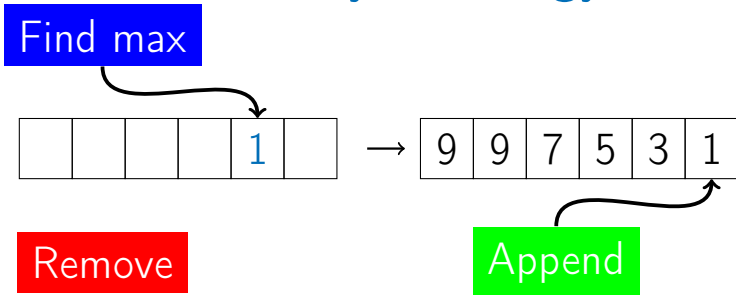


Remove

Append

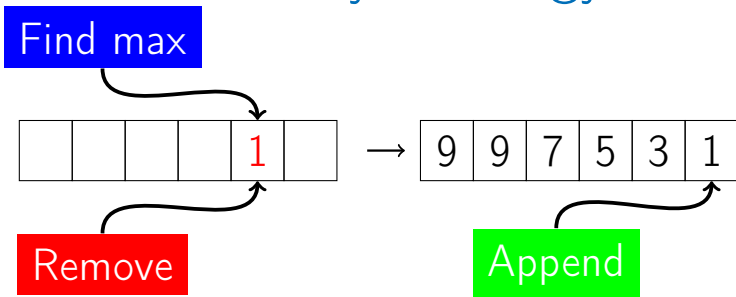
- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

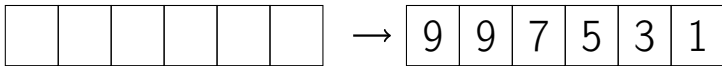
# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy

Find max

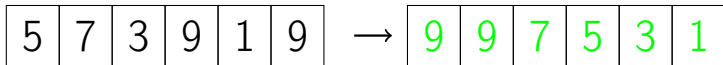


Remove

Append

- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Greedy Strategy



- Find **max** digit
- **Append** it to the number
- **Remove** it from the list of digits
- Repeat while there are digits in the list

# Outline

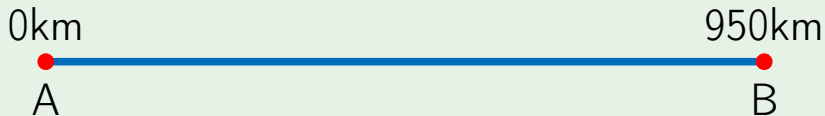
- 1 Largest Number
- 2 Car Fueling
- 3 Implementation and Analysis
- 4 Main Ingredients

## Car Fueling

Distance with full tank = 400km

## Car Fueling

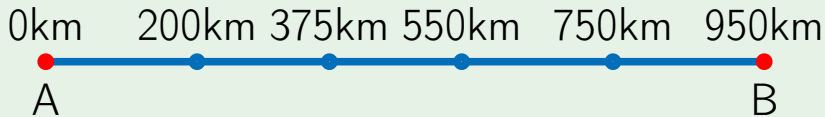
Distance with full tank = 400km





## Car Fueling

Distance with full tank = 400km



## Car Fueling

Distance with full tank = 400km



## Car Fueling

Distance with full tank = 400km



## Car Fueling

Distance with full tank = 400km



Minimum number of refills = 2

# Car Fueling

**Input:** A car which can travel at most  $L$  kilometers with full tank, a source point  $A$ , a destination point  $B$  and  $n$  gas stations at distances  $x_1 \leq x_2 \leq x_3 \leq \dots \leq x_n$  in kilometers from  $A$  along the path from  $A$  to  $B$ .

**Output:** The minimum number of refills to get from  $A$  to  $B$ , besides refill at  $A$ .

# Greedy Strategy

- Make some greedy choice
- Reduce to a smaller problem
- Iterate

# Greedy Choice

- Refill at the the closest gas station
- Refill at the farthest reachable gas station
- Go until there is no fuel

# Greedy Choice

- Refill at the the closest gas station
- Refill at the farthest reachable gas station
- Go until there is no fuel



# Greedy Algorithm

- Start at  $A$

# Greedy Algorithm

- Start at  $A$
- Refill at the farthest reachable gas station  $G$

# Greedy Algorithm

- Start at  $A$
- Refill at the farthest reachable gas station  $G$
- Make  $G$  the new  $A$

# Greedy Algorithm

- Start at  $A$
- Refill at the farthest reachable gas station  $G$
- Make  $G$  the new  $A$
- Get from new  $A$  to  $B$  with minimum number of refills

## Definition

**Subproblem** is a similar problem of smaller size.

# Subproblem

## Examples

■ `LargestNumber(3, 9, 5, 9, 7, 1) =`

# Subproblem

## Examples

- `LargestNumber(3, 9, 5, 9, 7, 1) =`  
“9” +

# Subproblem

## Examples

- $\text{LargestNumber}(3, 9, 5, 9, 7, 1) =$   
“9” +  $\text{LargestNumber}(3, 5, 9, 7, 1)$



# Subproblem

## Examples

- $\text{LargestNumber}(3, 9, 5, 9, 7, 1) =$   
“9” +  $\text{LargestNumber}(3, 5, 9, 7, 1)$
- Min number of refills from  $A$  to  $B =$

# Subproblem

## Examples

- $\text{LargestNumber}(3, 9, 5, 9, 7, 1) =$   
“9” +  $\text{LargestNumber}(3, 5, 9, 7, 1)$
- Min number of refills from  $A$  to  $B =$   
first refill at  $G +$

# Subproblem

## Examples

- $\text{LargestNumber}(3, 9, 5, 9, 7, 1) =$   
“9” +  $\text{LargestNumber}(3, 5, 9, 7, 1)$
- Min number of refills from  $A$  to  $B =$   
first refill at  $G$  + min number of refills  
from  $G$  to  $B$

# Safe Move

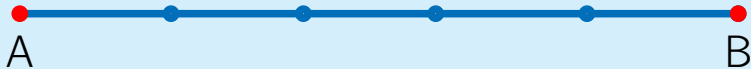
## Definition

A greedy choice is called **safe move** if there is an optimal solution consistent with this first move.

## Lemma

To refill at the farthest reachable gas station is a **safe move**.

# Proof



# Proof



# Proof



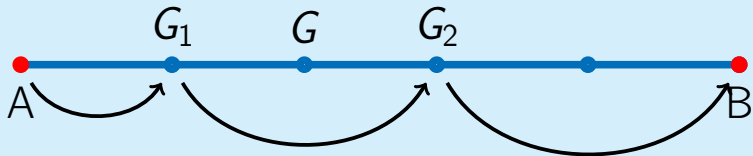


## Proof



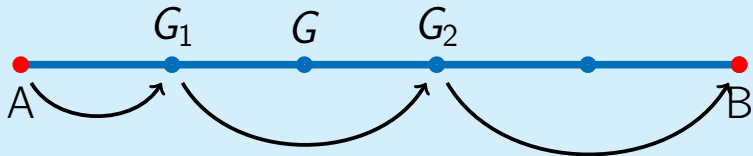
First case:  $G$  is closer than  $G_2$

## Proof



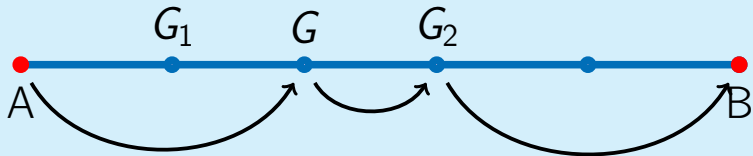
First case:  $G$  is closer than  $G_2$

## Proof



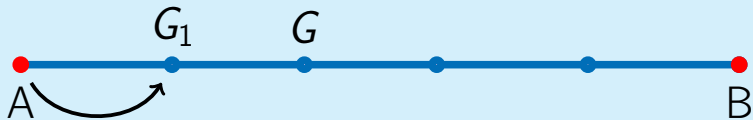
First case:  $G$  is closer than  $G_2$   
Refill at  $G$  instead of  $G_1$

## Proof



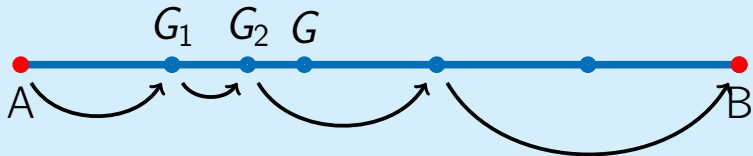
First case:  $G$  is closer than  $G_2$   
Refill at  $G$  instead of  $G_1$

## Proof



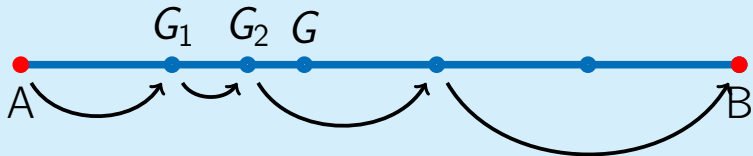
Second case:  $G_2$  is closer than  $G$

## Proof



Second case:  $G_2$  is closer than  $G$

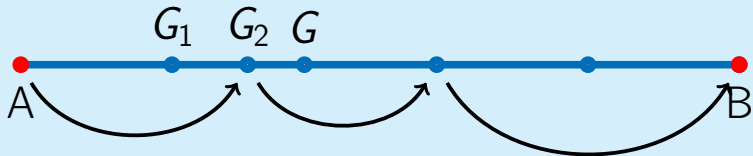
## Proof



Second case:  $G_2$  is closer than  $G$

Avoid refill at  $G_1$

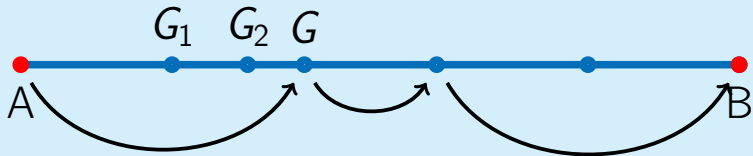
## Proof



Second case:  $G_2$  is closer than  $G$   
Avoid refill at  $G_1$



## Proof



Second case:  $G_2$  is closer than  $G$   
Avoid refill at  $G_1$

# Proof

- Route  $R$  with the minimum number of refills

## Proof

- Route  $R$  with the minimum number of refills
- $G_1$  — position of first refill in  $R$

# Proof

- Route  $R$  with the minimum number of refills
- $G_1$  — position of first refill in  $R$
- $G_2$  — next stop in  $R$  (refill or  $B$ )

## Proof

- Route  $R$  with the minimum number of refills
- $G_1$  — position of first refill in  $R$
- $G_2$  — next stop in  $R$  (refill or  $B$ )
- $G$  — farthest refill reachable from  $A$

## Proof

- Route  $R$  with the minimum number of refills
- $G_1$  — position of first refill in  $R$
- $G_2$  — next stop in  $R$  (refill or  $B$ )
- $G$  — farthest refill reachable from  $A$
- If  $G$  is closer than  $G_2$ , refill at  $G$  instead of  $G_1$

## Proof

- Route  $R$  with the minimum number of refills
- $G_1$  — position of first refill in  $R$
- $G_2$  — next stop in  $R$  (refill or  $B$ )
- $G$  — farthest refill reachable from  $A$
- If  $G$  is closer than  $G_2$ , refill at  $G$  instead of  $G_1$
- Otherwise, avoid refill at  $G_1$



# Outline

- 1 Largest Number
- 2 Car Fueling
- 3 Implementation and Analysis
- 4 Main Ingredients



$$A = x_0 \leq x_1 \leq x_2 \leq \dots \leq x_n \leq x_{n+1} = B$$

**MinRefills**( $x, n, L$ )

```
numRefills  $\leftarrow$  0, currentRefill  $\leftarrow$  0
while currentRefill  $\leq$   $n$ :
    lastRefill  $\leftarrow$  currentRefill
    while (currentRefill  $\leq$   $n$  and
            $x[\textit{currentRefill} + 1] - x[\textit{lastRefill}] \leq L$ ):
        currentRefill  $\leftarrow$  currentRefill + 1
    if currentRefill == lastRefill:
        return IMPOSSIBLE
    if currentRefill  $\leq$   $n$ :
        numRefills  $\leftarrow$  numRefills + 1
return numRefills
```

## Lemma

The running time of  $\text{MinRefills}(x, n, L)$  is  $O(n)$ .

## Lemma

The running time of  $\text{MinRefills}(x, n, L)$  is  $O(n)$ .

## Proof

- *currentRefill* changes from 0 to  $n + 1$ , one-by-one

## Lemma

The running time of  $\text{MinRefills}(x, n, L)$  is  $O(n)$ .

## Proof

- *currentRefill* changes from 0 to  $n + 1$ , one-by-one
- *numRefills* changes from 0 to at most  $n$ , one-by-one

## Lemma

The running time of  $\text{MinRefills}(x, n, L)$  is  $O(n)$ .

## Proof

- *currentRefill* changes from 0 to  $n + 1$ , one-by-one
- *numRefills* changes from 0 to at most  $n$ , one-by-one
- Thus,  $O(n)$  iterations



# Outline

- 1 Largest Number
- 2 Car Fueling
- 3 Implementation and Analysis
- 4 Main Ingredients

# Reduction to Subproblem

- Make a first move
- Then solve a problem of the same kind
- Smaller: fewer digits, fewer fuel stations
- This is called a “subproblem”

# Safe move

- A move is called **safe** if there is an optimal solution consistent with this first move



# Safe move

- A move is called **safe** if there is an optimal solution consistent with this first move
- Not all first moves are safe

# Safe move

- A move is called **safe** if there is an optimal solution consistent with this first move
- Not all first moves are safe
- Often greedy moves are not safe

# General Strategy

Problem

# General Strategy

Problem  $\xrightarrow{\text{greedy choice}}$

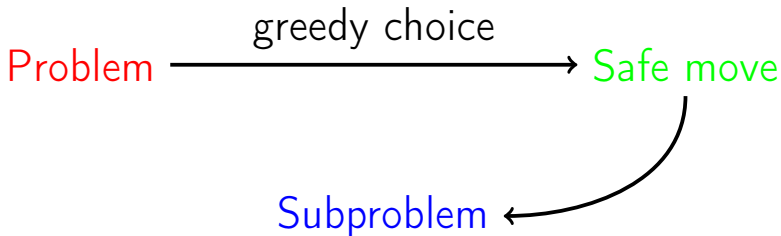
- Make a greedy choice

# General Strategy

Problem  $\xrightarrow{\text{greedy choice}}$  Safe move

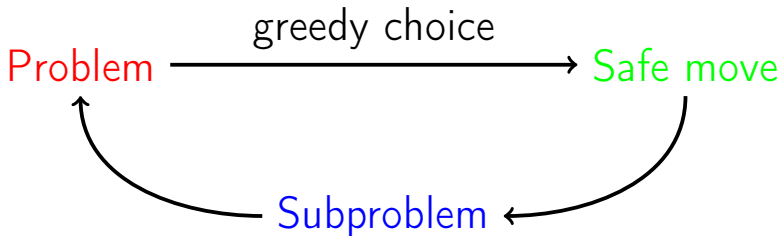
- Make a greedy choice
- Prove that it is a safe move

# General Strategy



- Make a greedy choice
- **Prove** that it is a **safe move**
- Reduce to a **subproblem**

# General Strategy



- Make a greedy choice
- **Prove** that it is a **safe move**
- Reduce to a **subproblem**
- Solve the **subproblem**