Greedy Algorithms

Brute-Force, Heuristic Algorithms

SoftUni Foundation

SoftUni Team

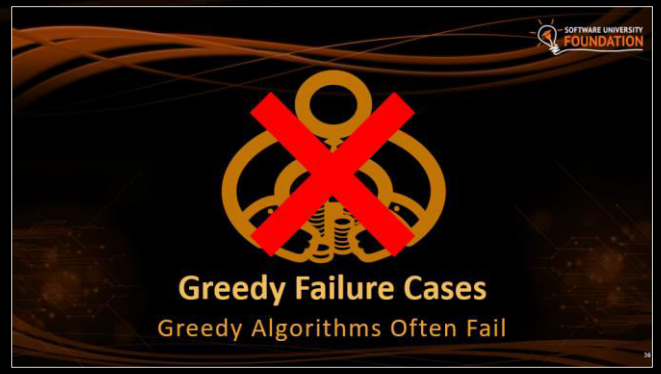
Technical Trainers

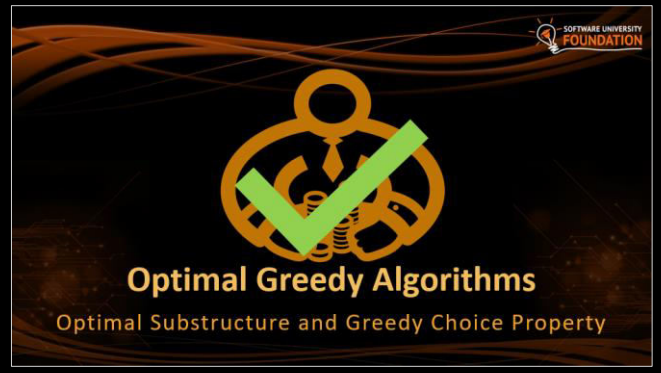
Software University

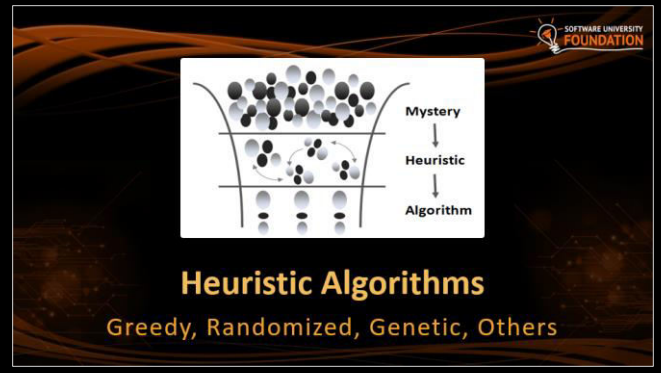
http://softuni.bg

Table of Contents

SoftUni Foundation







Have a Question?

SoftUni Foundation

sli.do

#DsAlgo

SoftUni Foundation

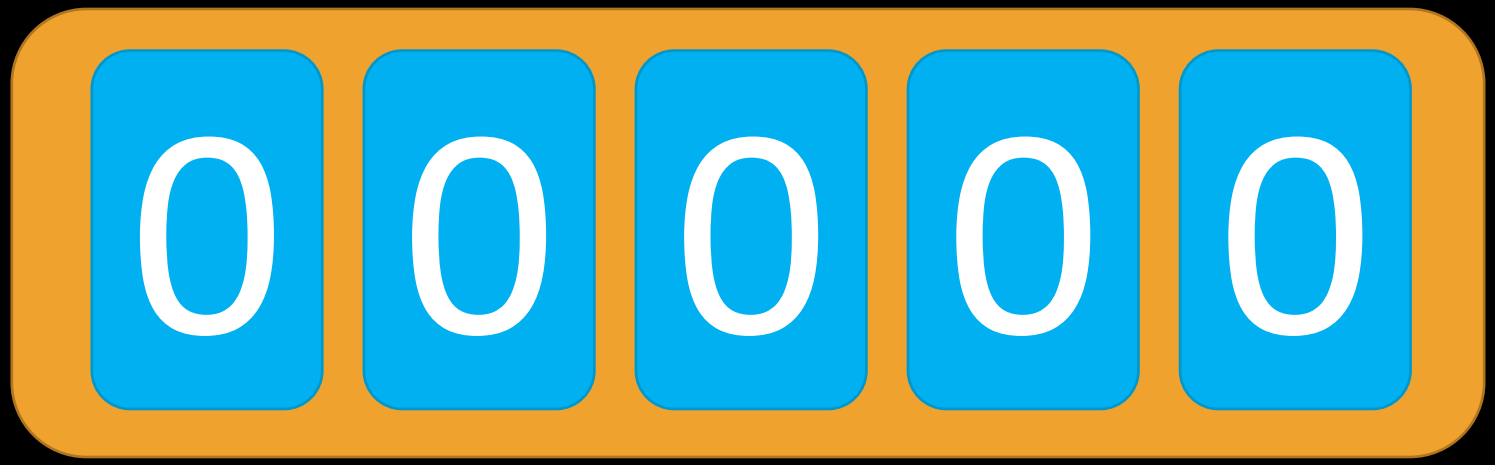
Brute-Force Algorithms

Trying All Solutions

Brute-Force Algorithms

SoftUni Foundation

- Trying all possible combinations
- Picking the best solution
- Usually slow and inefficient



Brute-Force Algorithms

SoftUni Foundation

Brute-Force Algorithms

SoftUni Foundation

Brute-Force Algorithms

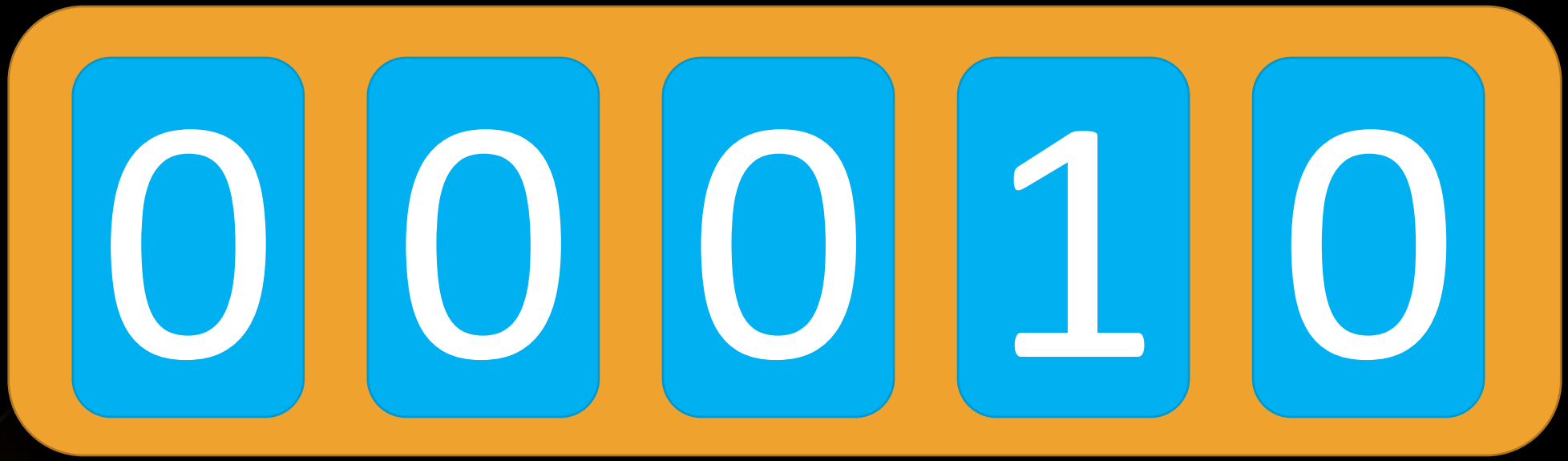
SoftUni Foundation

Brute-Force Algorithms

SoftUni Foundation

Brute-Force Algorithms

SoftUni Foundation



Brute-Force Algorithms

SoftUni Foundation

Brute-Force Algorithms

SoftUni Foundation

Brute-Force Algorithms

SoftUni Foundation

10 x 10 x 10 x 10 x 10 = 100,000 combinations

SoftUni Foundation

Greedy Algorithms

Picking Locally Optimal Solution

Optimization Problems

SoftUni Foundation

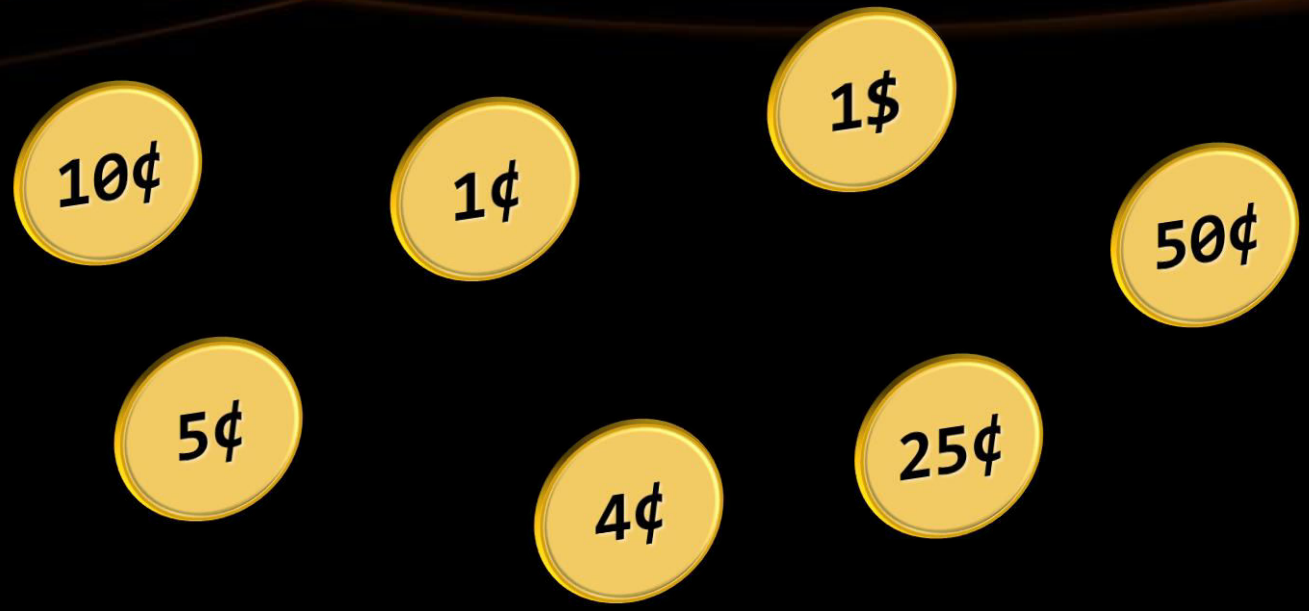
- In computer science, an optimization problem is the problem of finding the best solution from all feasible solutions
  - Finding the optimal among many candidates
- Examples of optimization problems:
  - Find the shortest path from Sofia to Varna on the map
  - Find the maximum increasing subsequence in integer sequence
  - Traveling salesman problem (minimal Hamilton cycle)
    - Find the shortest route that visits each city and returns to the origin city

Greedy Algorithms

SoftUni Foundation

- Greedy algorithms are a category of algorithms
  - For solving optimization problems
  - Usually more efficient than the other algorithms
    - But can produce non-optimal (incorrect) result
- Greedy algorithms pick the best local solution
  - Pick the optimum from their current position & point of view
- Greedy algorithms assume always choosing a local optimum leads to the global optimum
  - This is sometimes correct, but sometimes it is not

SoftUni Foundation



Sum of Coins

Problem

Greedy Algorithms: Sum of Coins

SoftUni Foundation

- Consider the US currency coins:
  - { 0.01, 0.02, 0.05, 0.10}
- Problem "Sum of Coins":
  - Gather a sum of money, using the least possible number of coins
- Greedy algorithm for "Sum of Coins":
  - Take the largest coin while possible
  - Then take the second largest
  - Etc.

Sum of Coins Visualization

SoftUni Foundation

Target: 18

Actual: 0

Sum of Coins Visualization

SoftUni Foundation

Target: 18

Actual: 10

Sum of Coins Visualization

SoftUni Foundation

Target: 18

Actual: 15

Sum of Coins Visualization

SoftUni Foundation

Target: 18

Actual: 17

Sum of Coins Visualization

SoftUni Foundation

Target: 18

Actual: 18

Sum of Coins – Solution

SoftUni Foundation

```
int finalSum = 18;
int currentSum = 0;
int[] coins = { 10, 10, 5, 5, 2, 2, 1, 1 };
Queue<int> resultCoins = new Queue<int>();

// Next Slide

Console.WriteLine("Sum not found");
```

Sum of Coins – Solution(2)

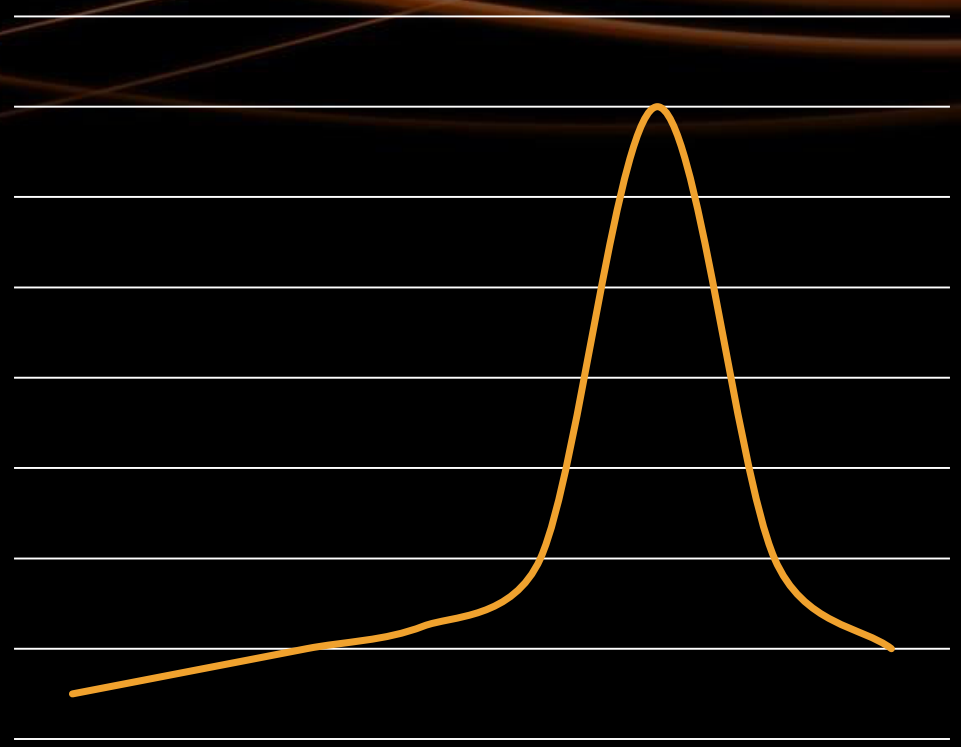
SoftUni Foundation

```
for (int i = 0; i < coins.Length; i++)
{
  if (currentSum + coins[i] > finalSum) continue;

  currentSum += coins[i];
  resultCoins.Enqueue(coins[i]);

  if (currentSum == finalSum)
    // Sum Found
}
```

SoftUni Foundation

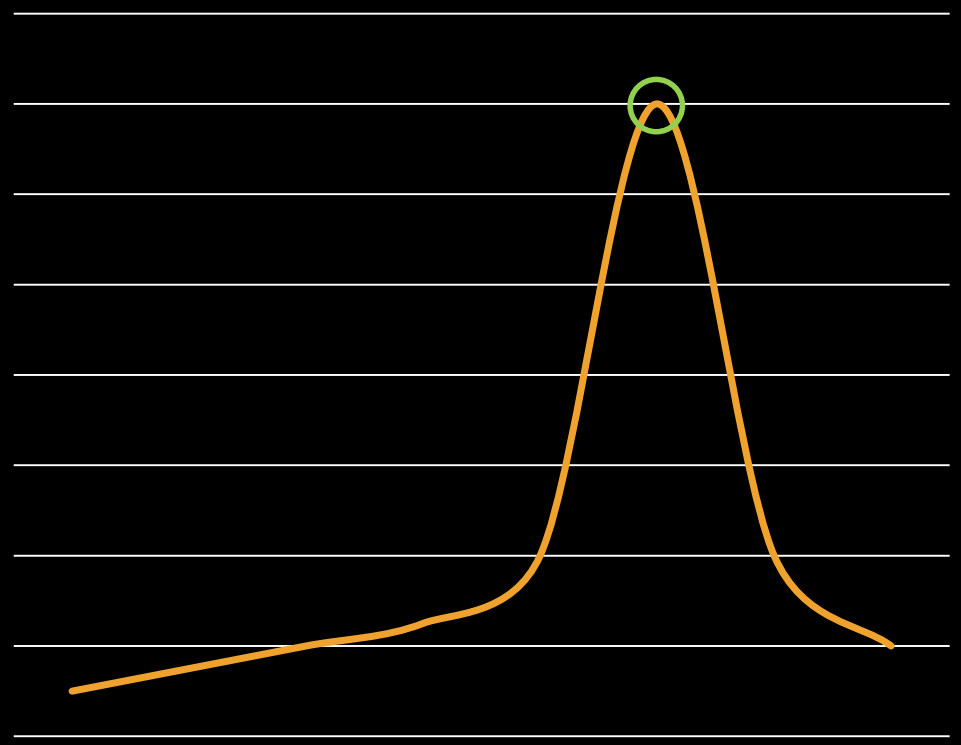


Peak Point

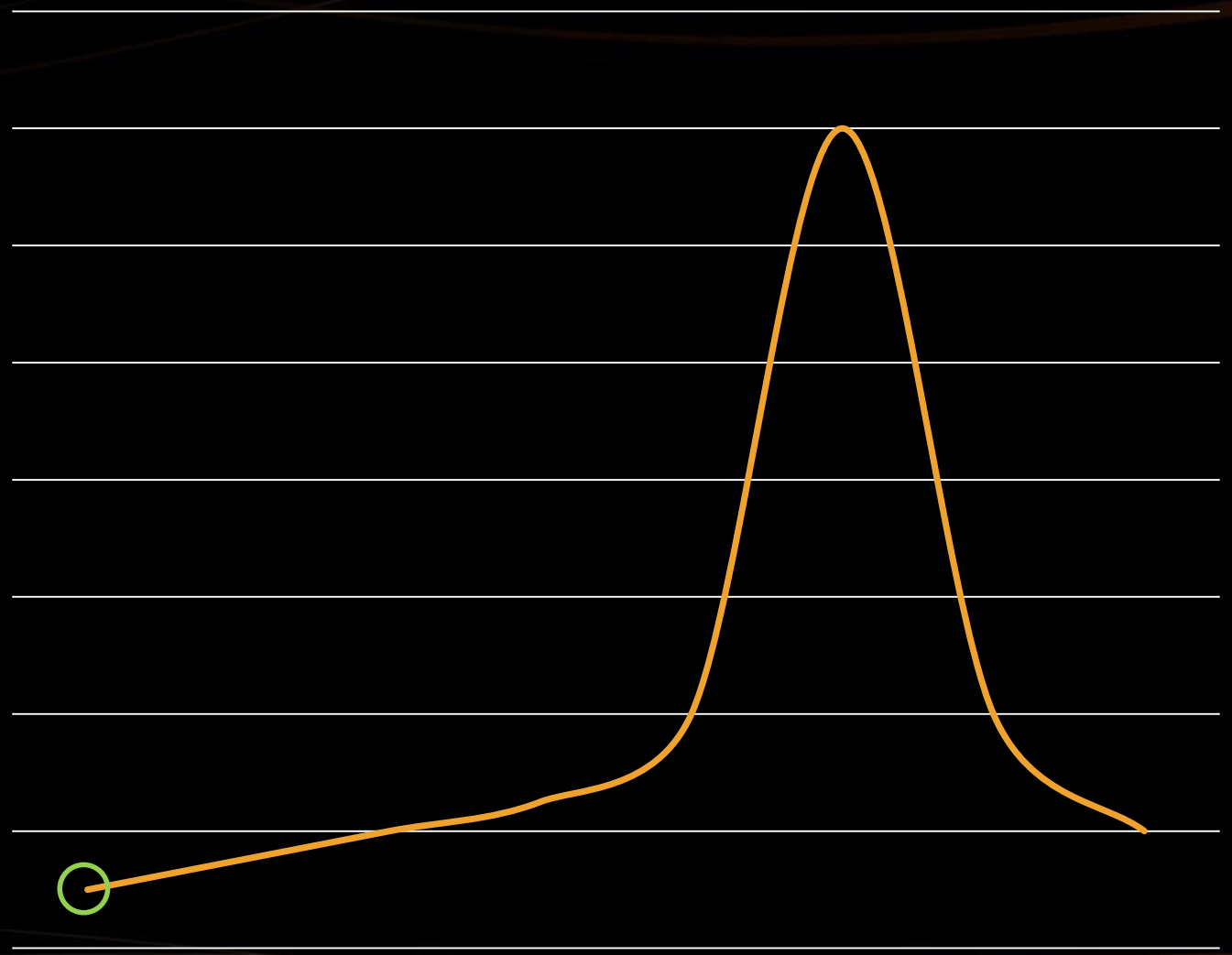
Problem

Greedy Algorithms: Peak Point

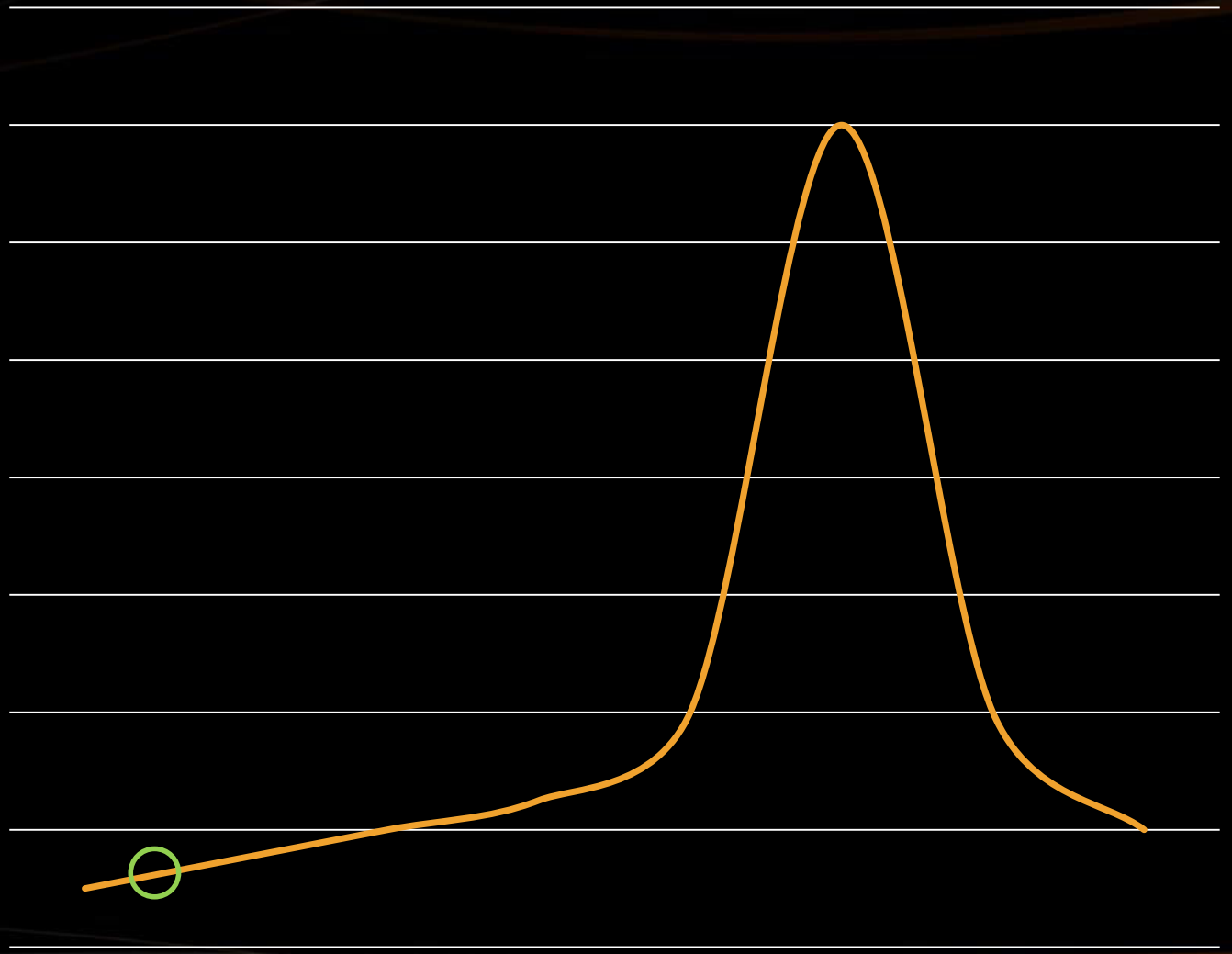
- Find the peak point on a curve



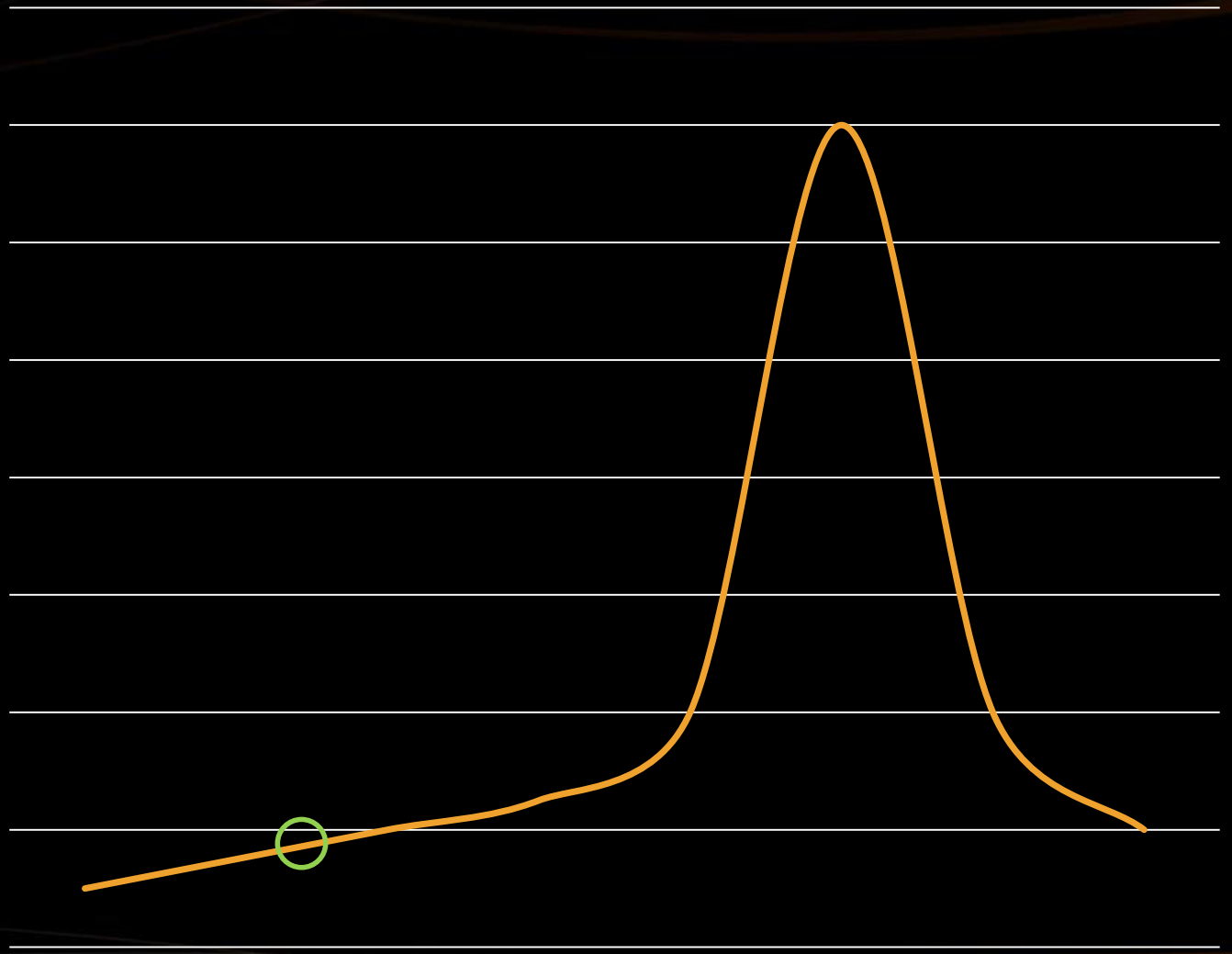
Peak Point Visualization



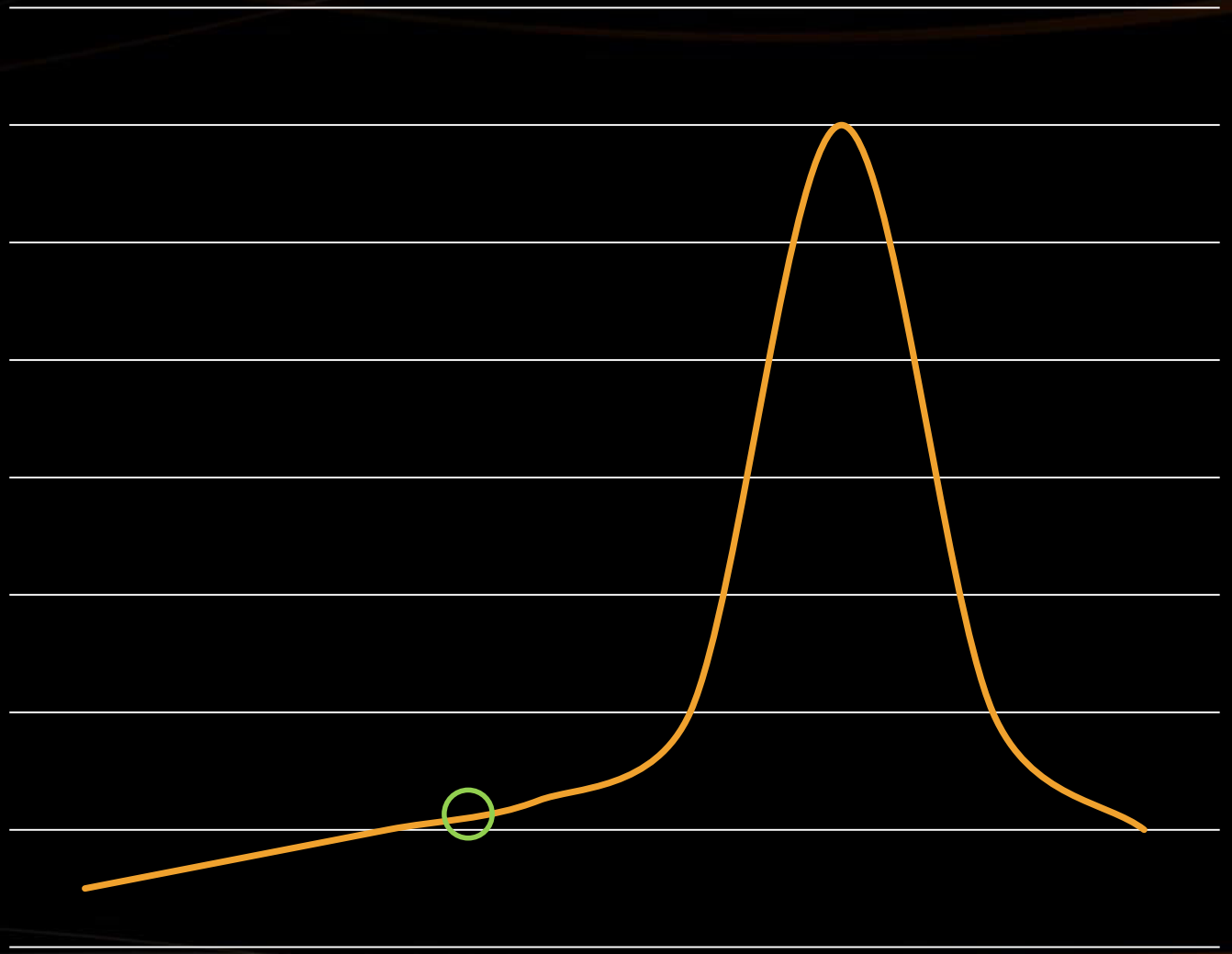
Peak Point Visualization



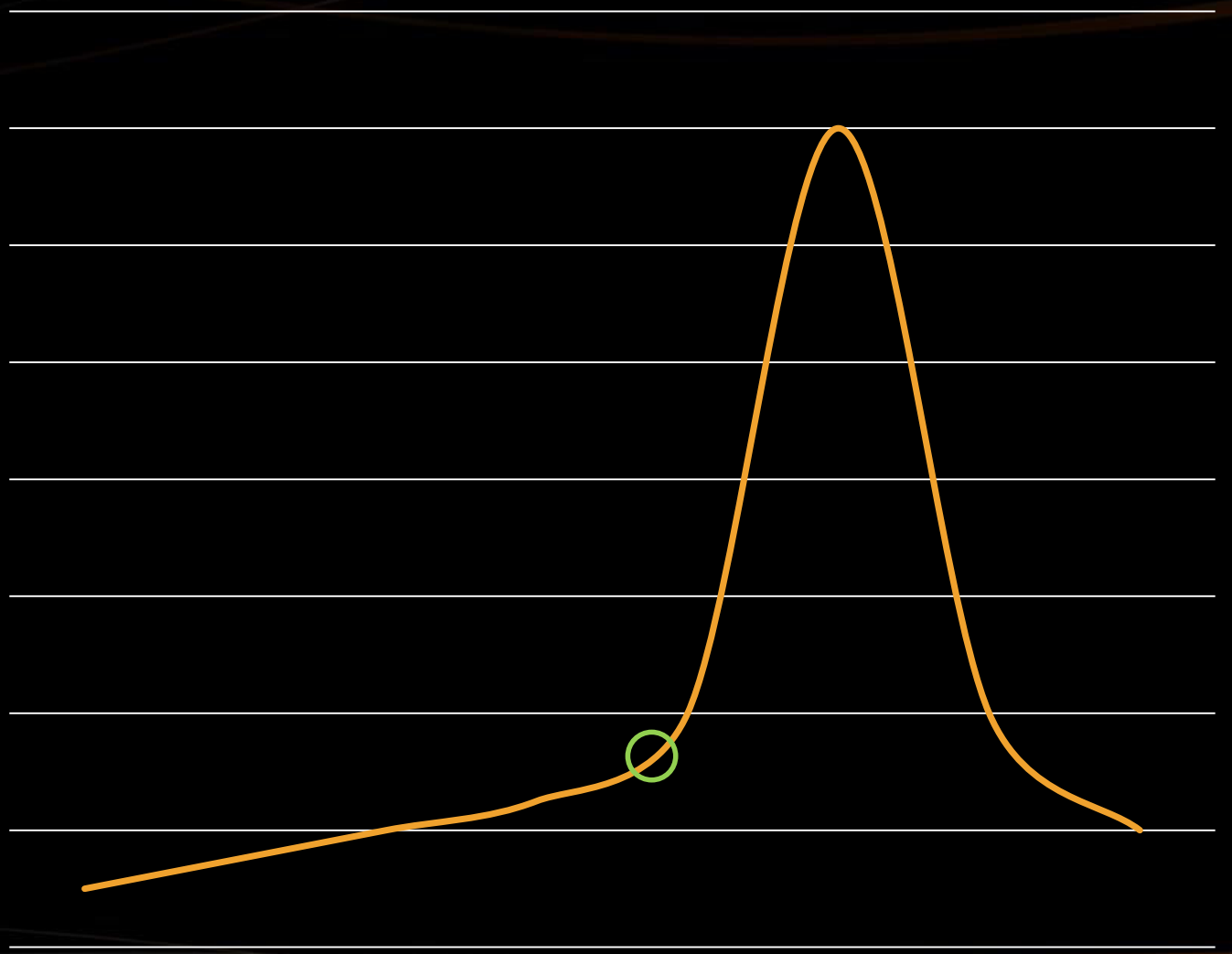
Peak Point Visualization



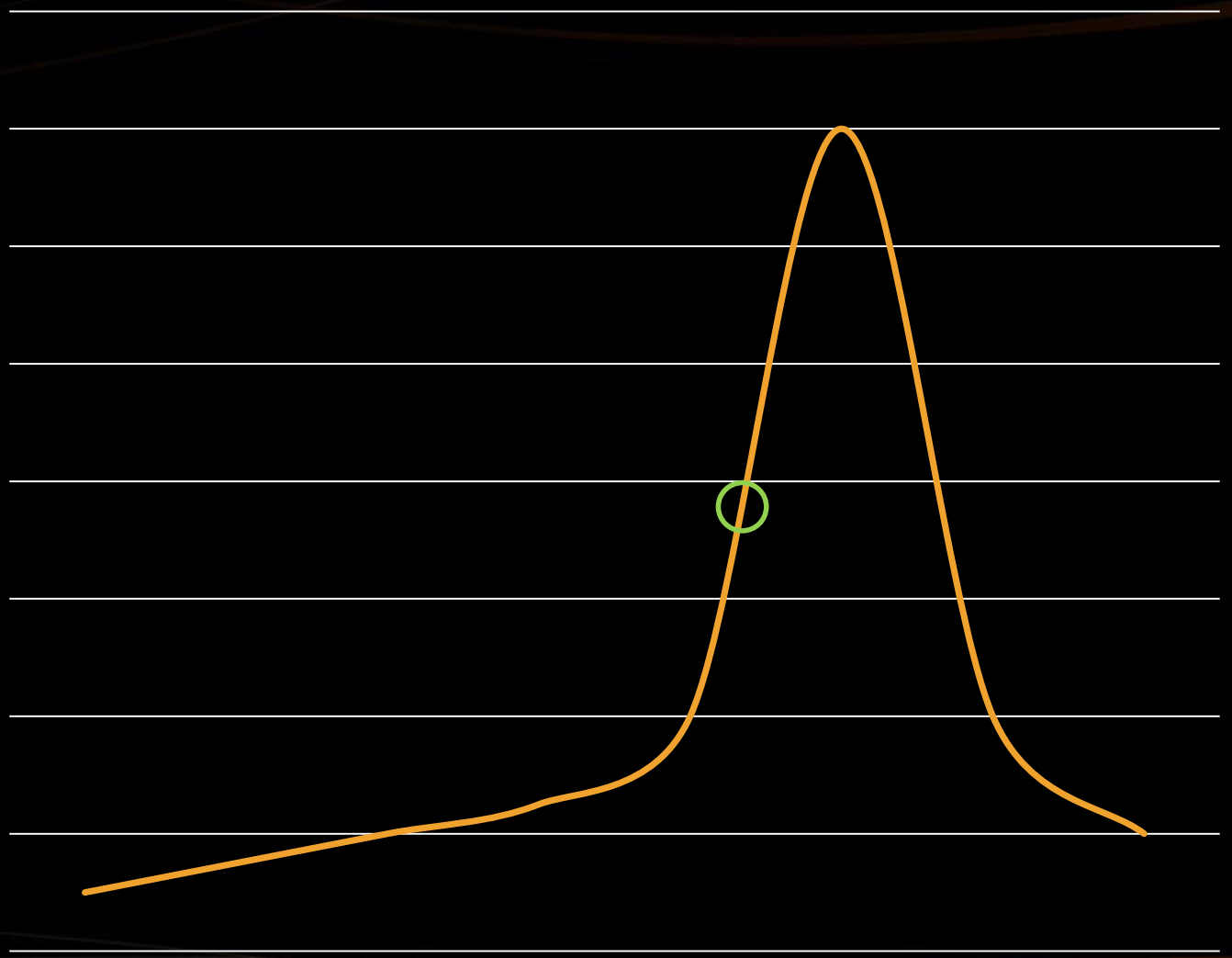
Peak Point Visualization



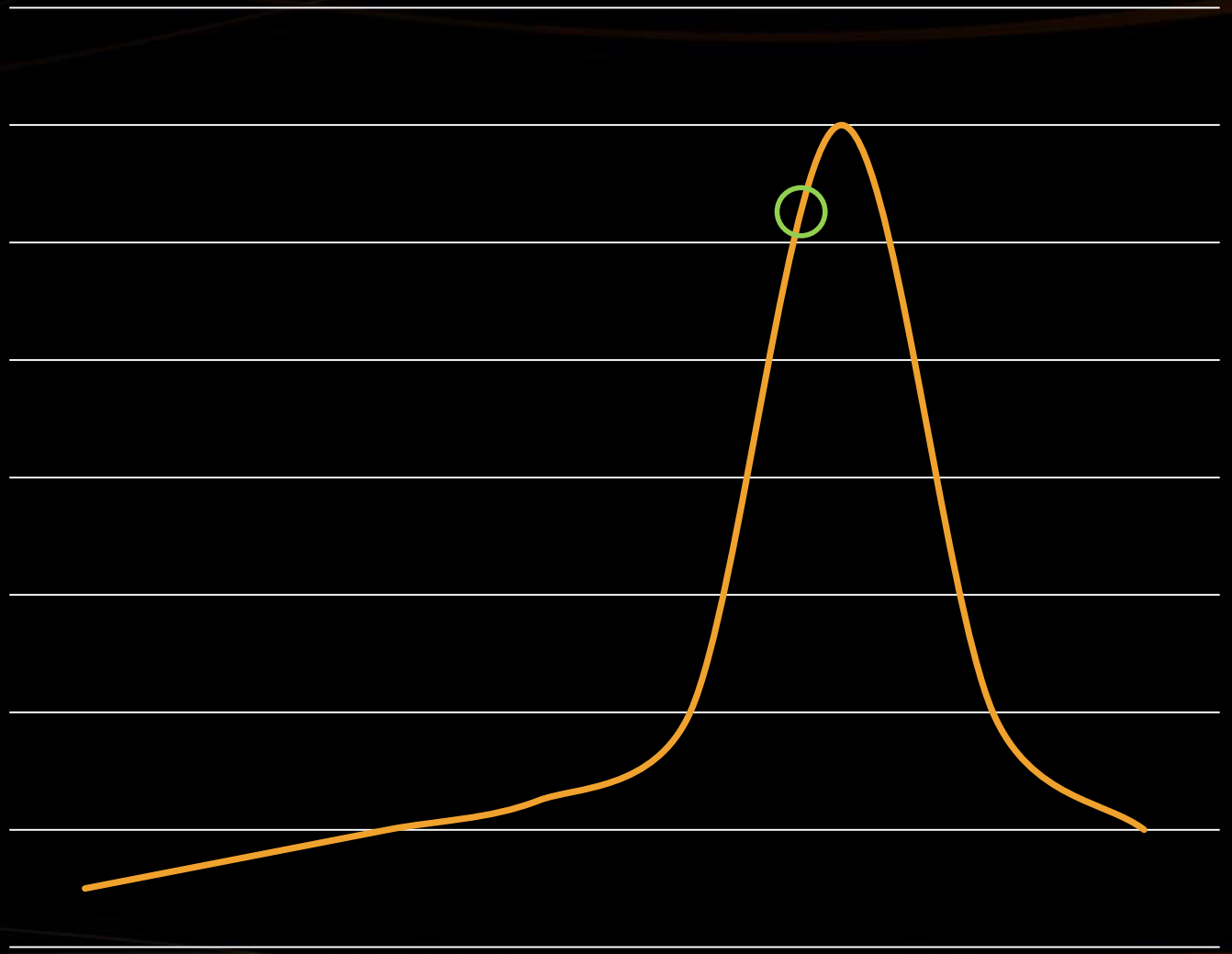
Peak Point Visualization



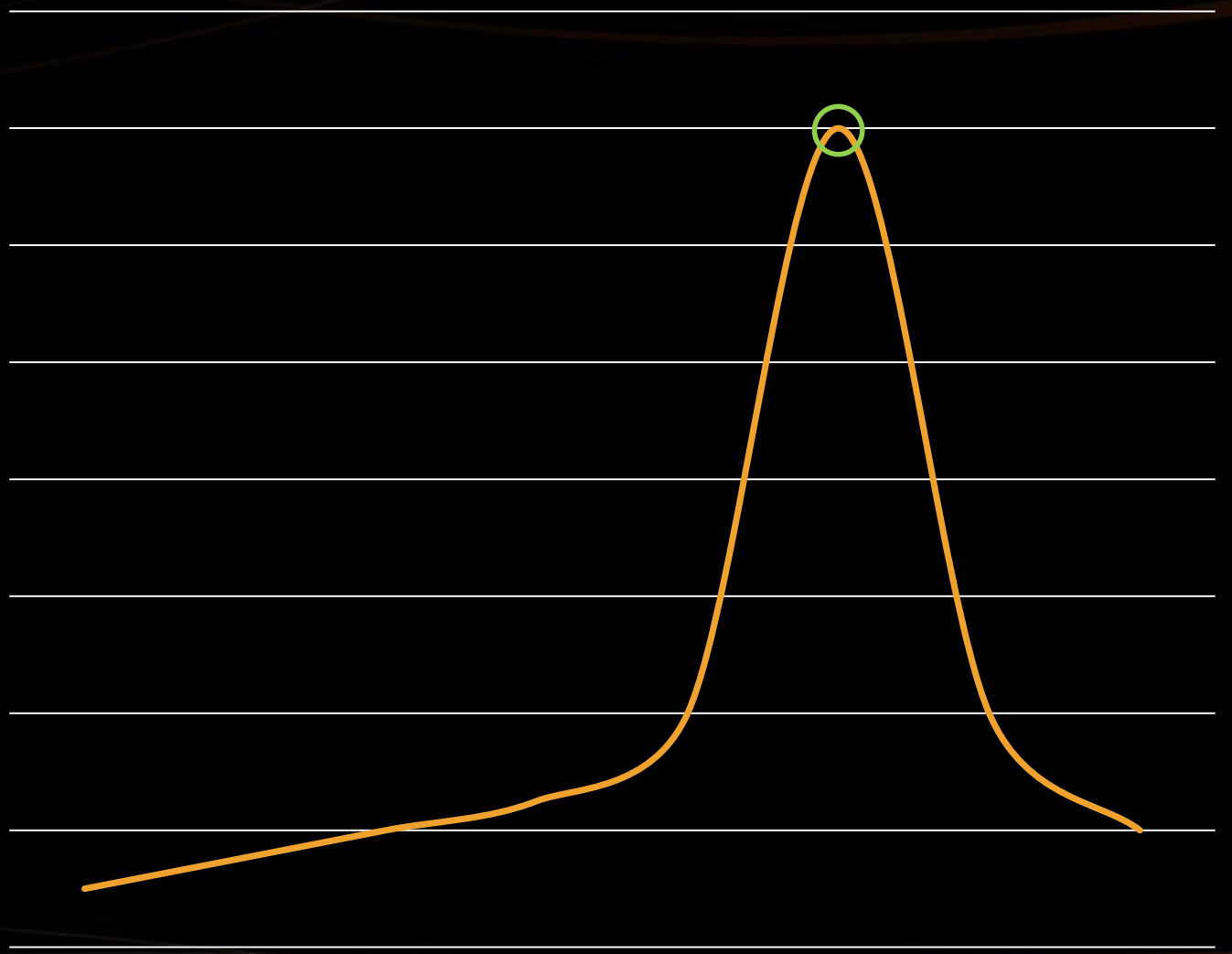
Peak Point Visualization



Peak Point Visualization



Peak Point Visualization



SoftUni Foundation

Greedy Algorithm for Sum of Coins

Live Coding Exercise (Lab)

Greedy Failure Cases

Greedy Algorithms Often Fail

Sum of Coins Failure

SoftUni Foundation

Target: 18

Actual: 0

Sum of Coins Failure

Target: 18

Actual: 10

Sum of Coins Failure

SoftUni Foundation

Target: 18

Actual: 15

Sum of Coins Failure

SoftUni Foundation

Target: 18

Actual: 16

Sum of Coins Failure

SoftUni Foundation

Target: 18

Actual: 17

Sum of Coins Failure

SoftUni Foundation

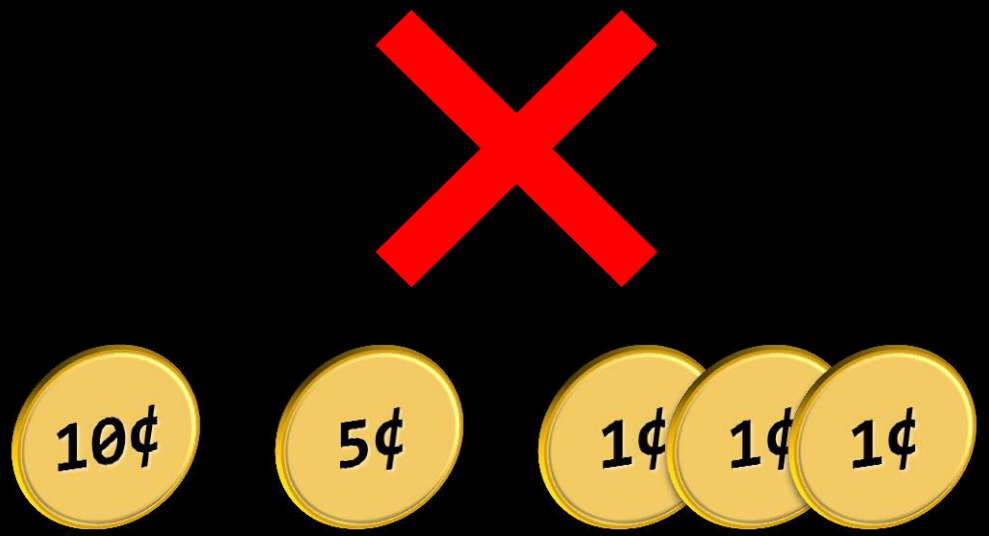
Target: 18

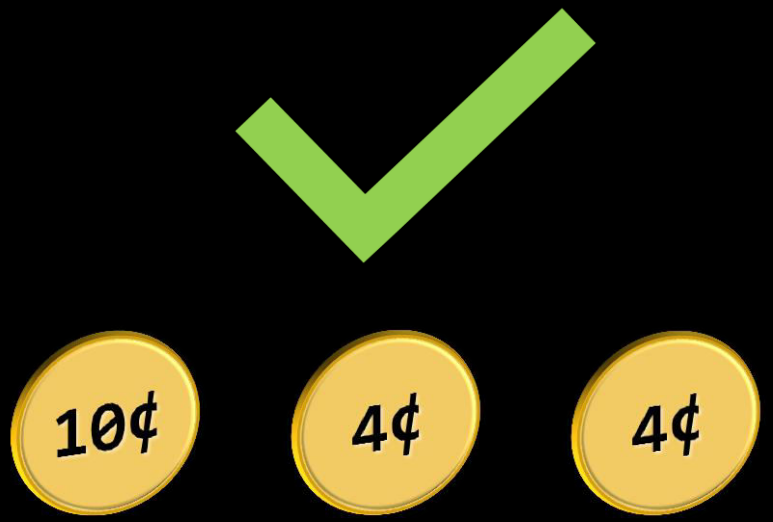
Actual: 18

Sum of Coins Failure

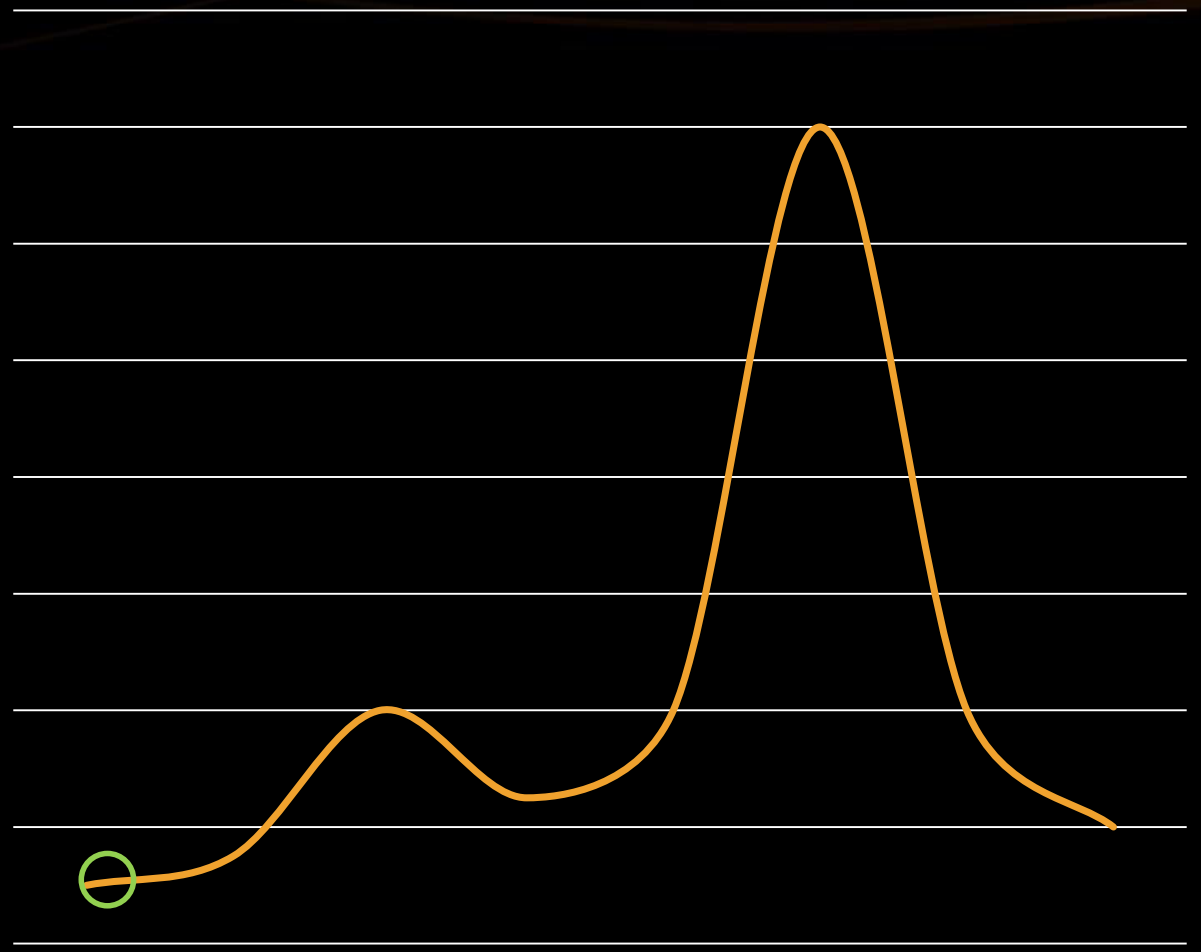
SoftUni Foundation

Target: 18

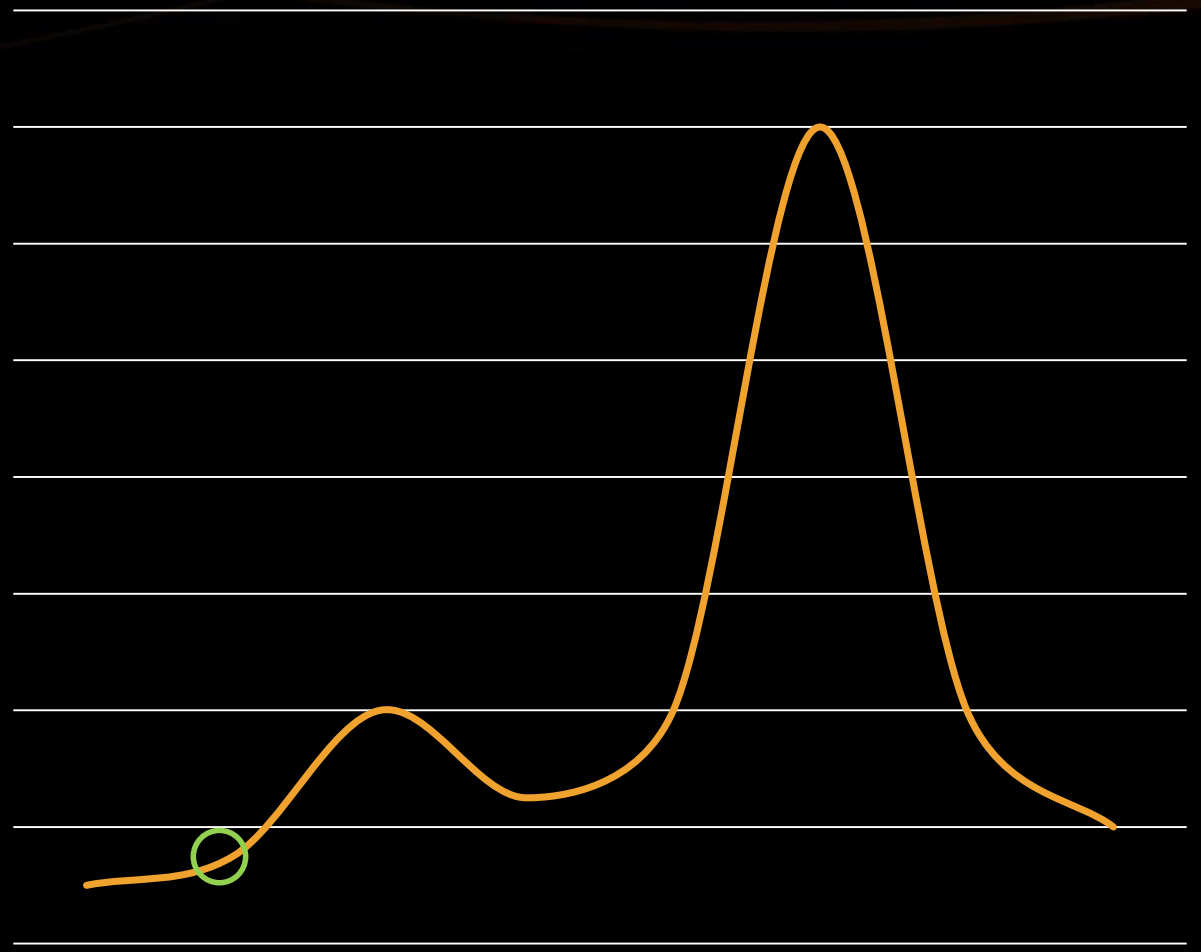




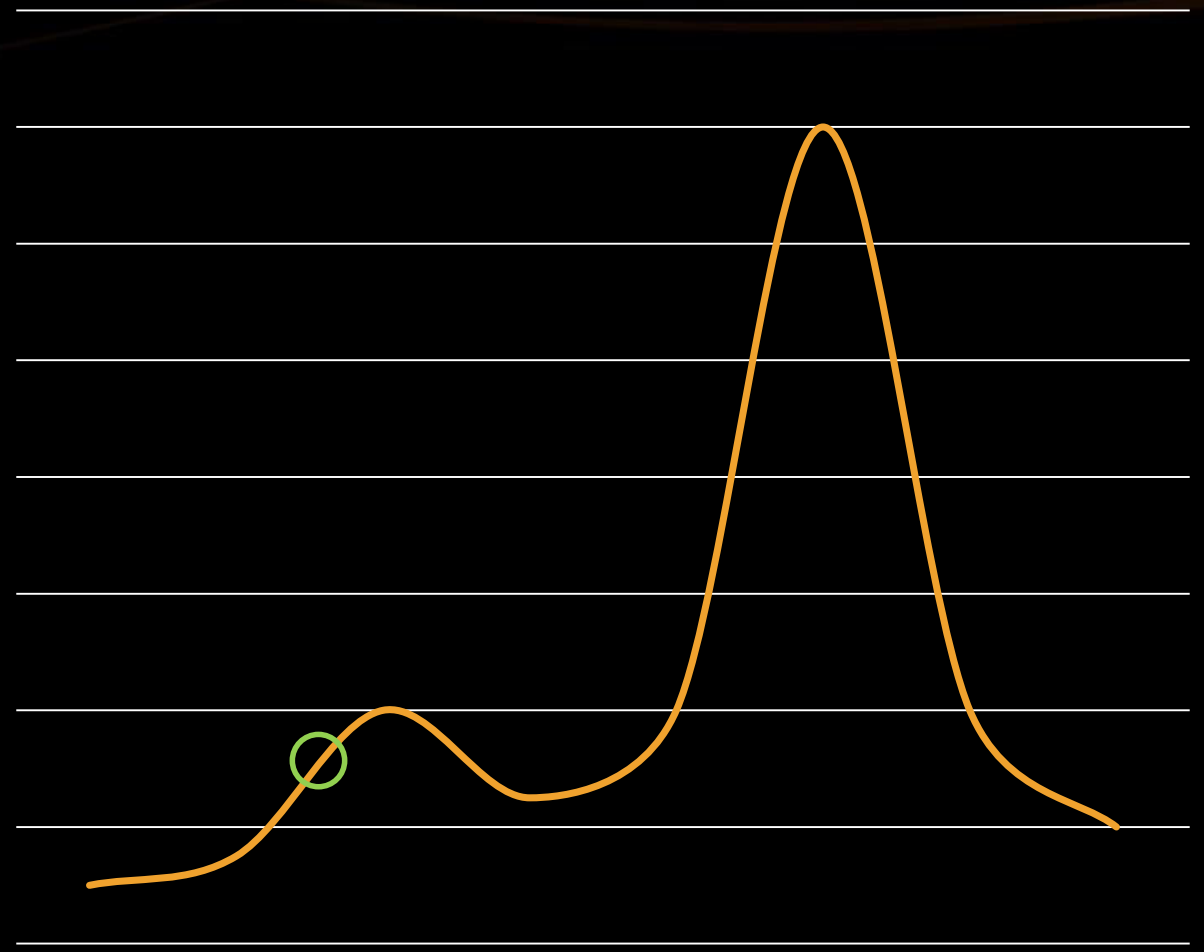
Peak Point Failure



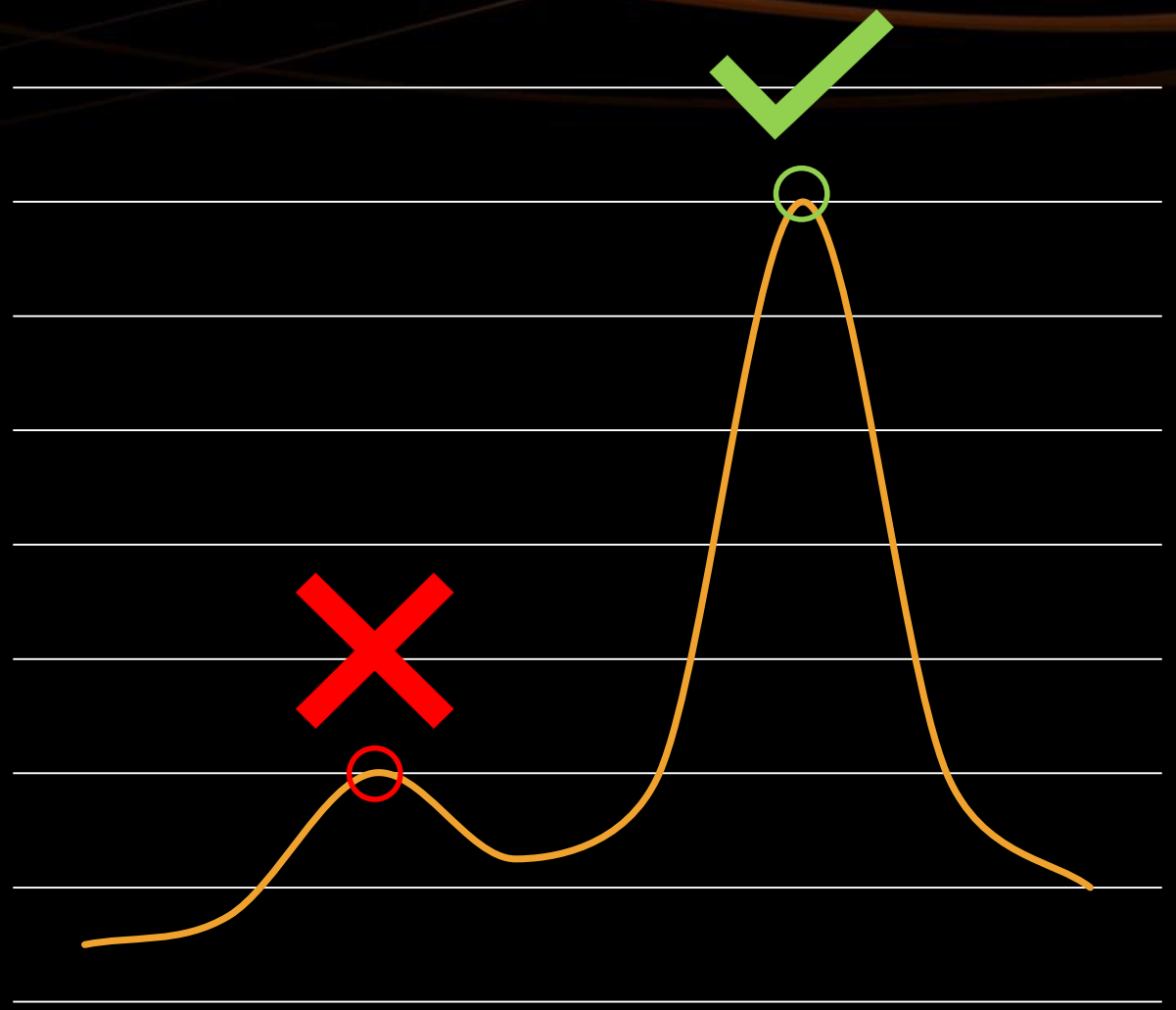
Peak Point Failure



Peak Point Failure

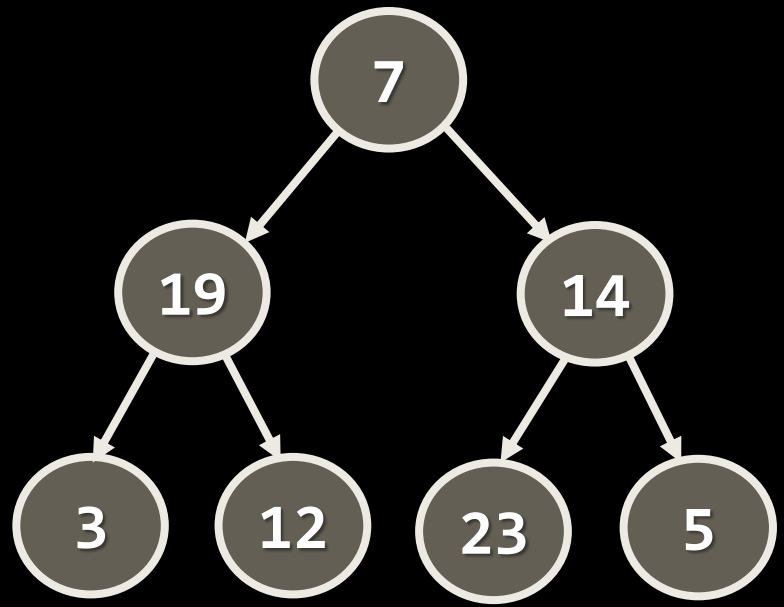


Peak Point Failure



Max Sum Path in Binary Tree

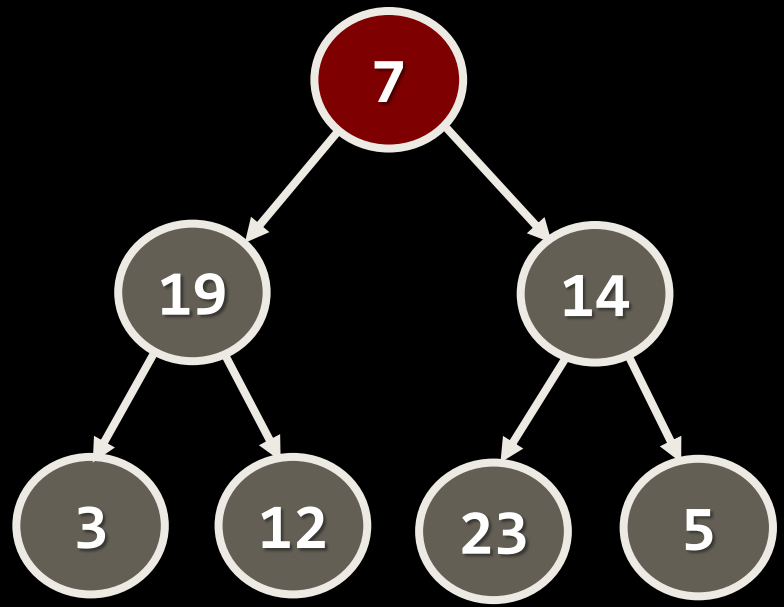
- Largest sum path in a tree (starting at root)
  - Greedy strategy: choose largest next node



Max Sum in Binary Tree

SoftUni Foundation

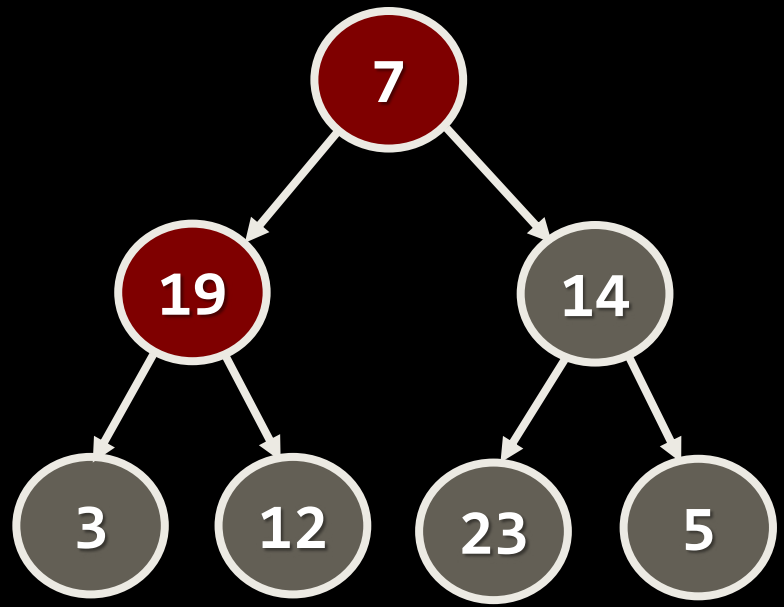
Sum: 7



Max Sum in Binary Tree

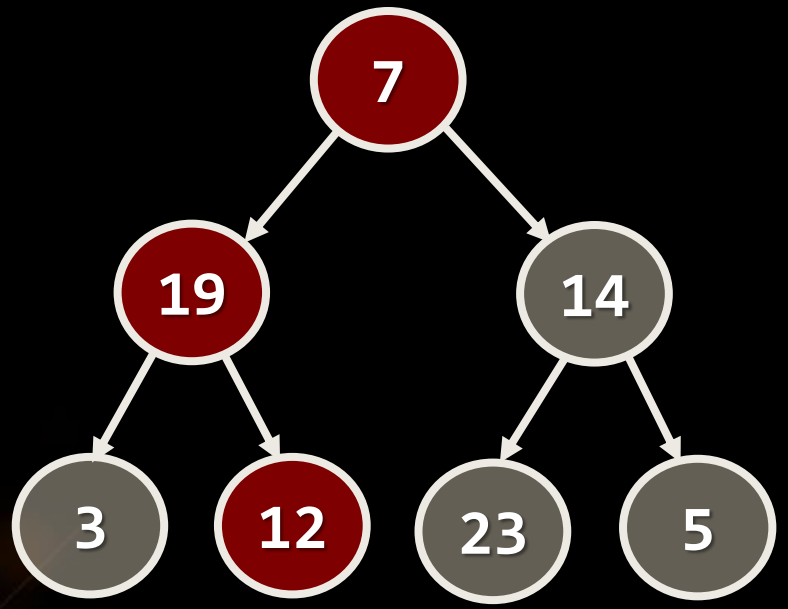
SoftUni Foundation

Sum: 26

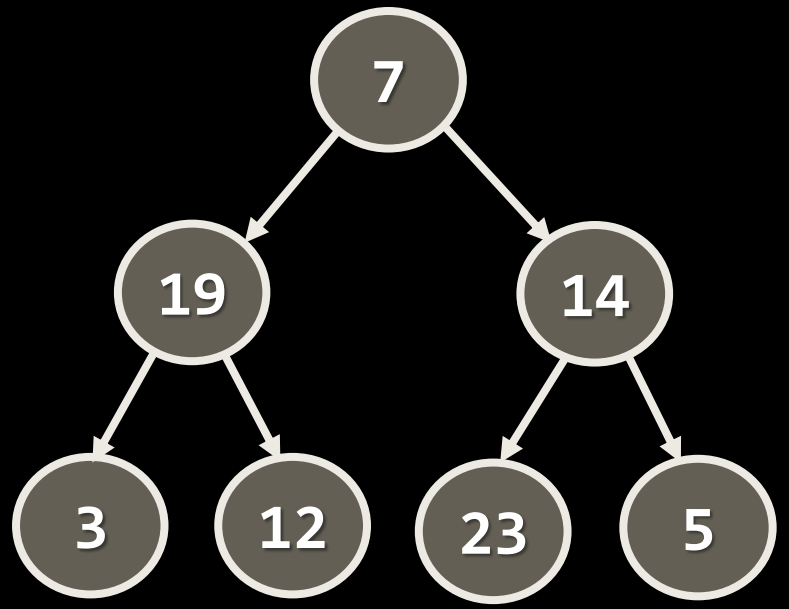


Max Sum in Binary Tree

Sum: 38
Greedy

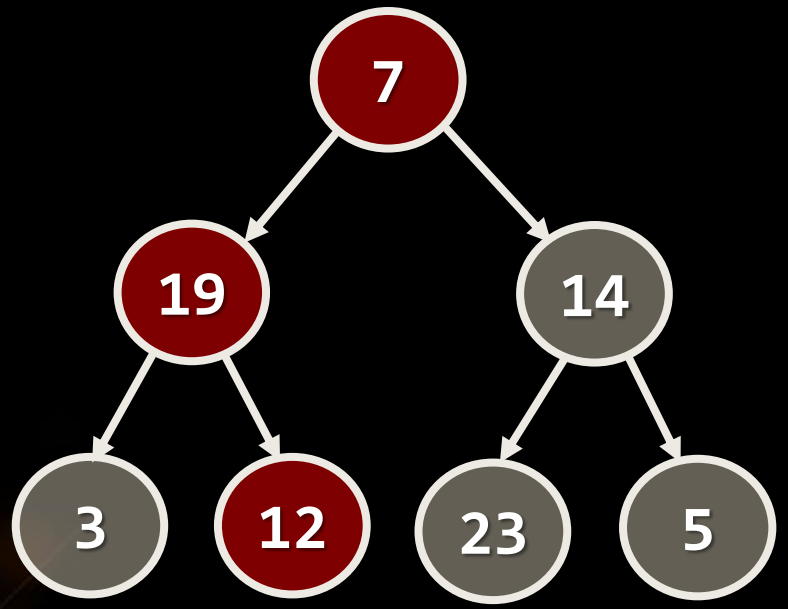


Sum: 0
Actual

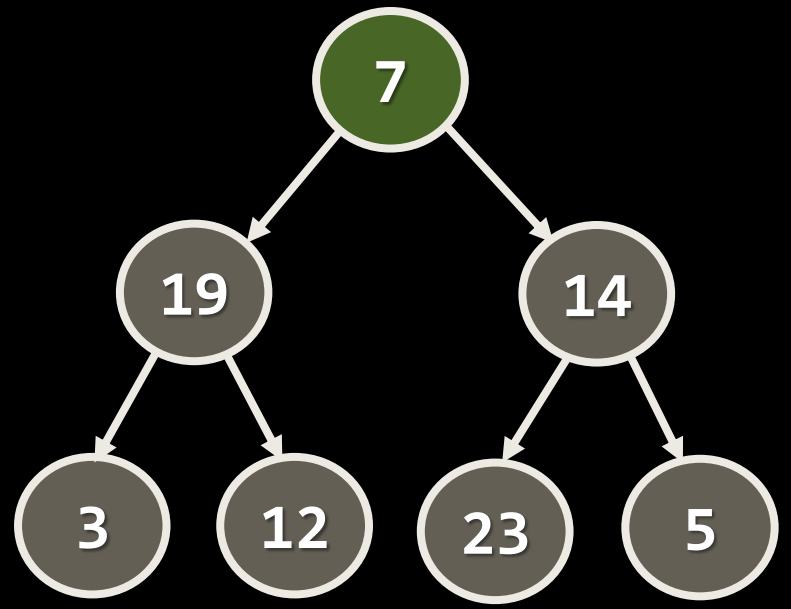


Max Sum in Binary Tree

Sum: 38
Greedy



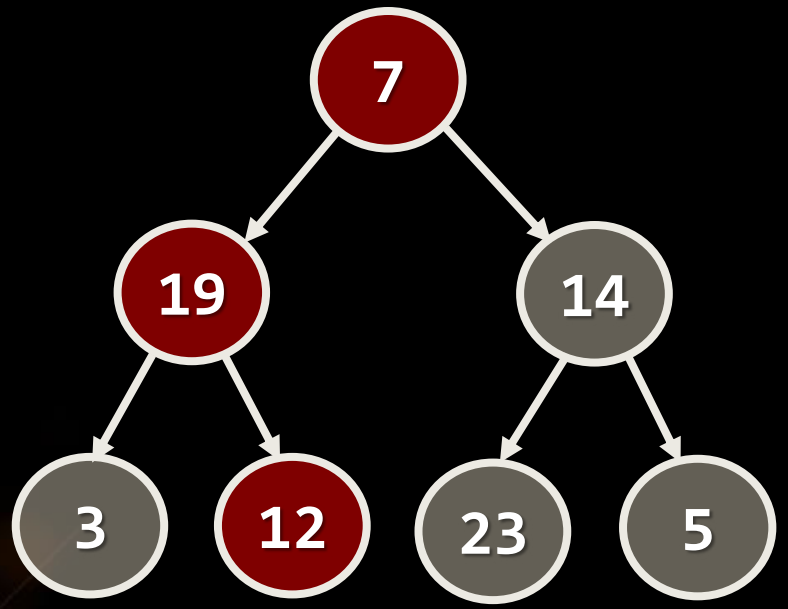
Sum: 7
Actual



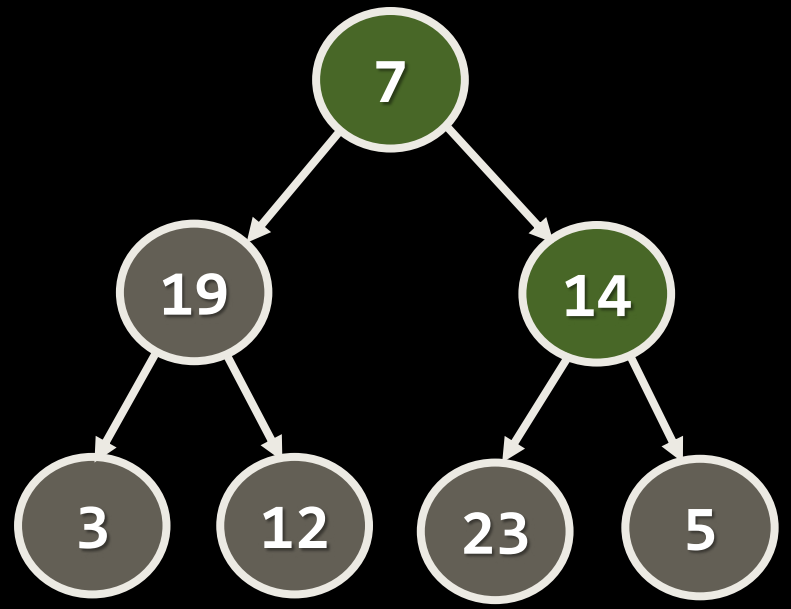
Max Sum in Binary Tree

SoftUni Foundation

Sum: 38
Greedy

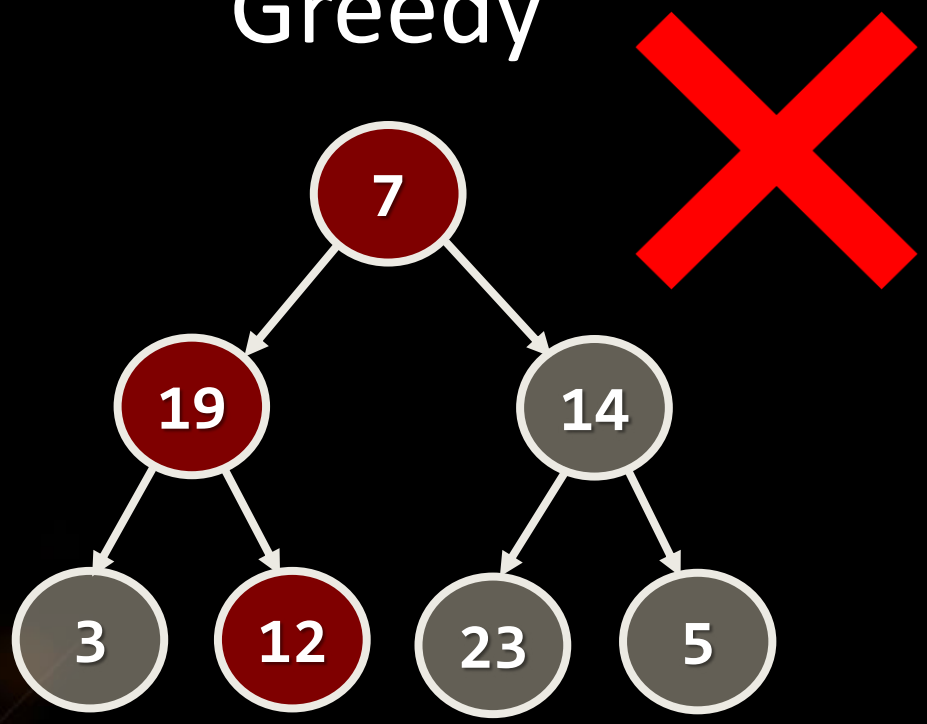


Sum: 21
Actual

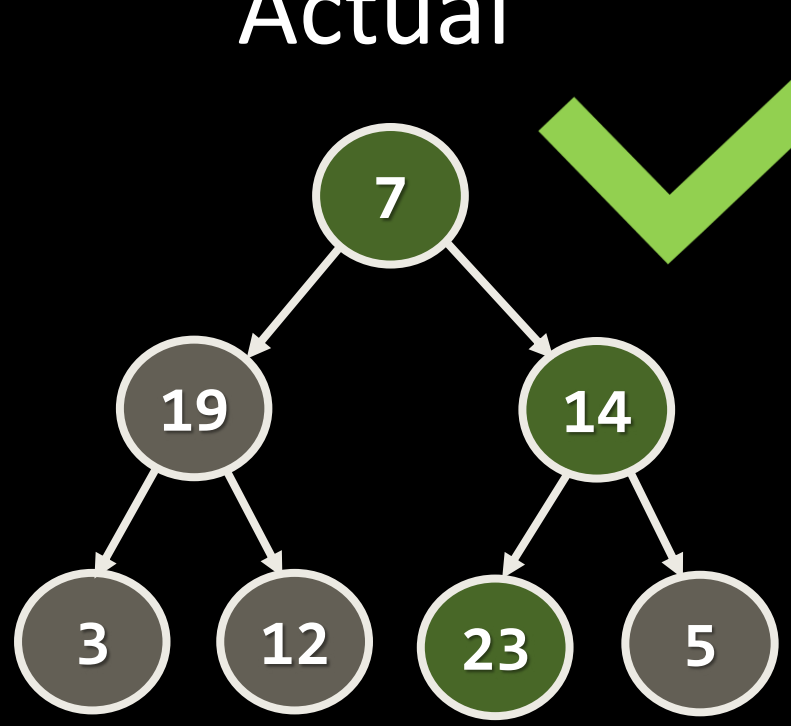


Max Sum in Binary Tree

Sum: 38
Greedy



Sum: 44
Actual



SoftUni Foundation

Optimal Greedy Algorithms

Optimal Substructure and Greedy Choice Property

Optimal Greedy Algorithms

SoftUni Foundation

- Suitable problems for greedy algorithms have these properties:
  - Greedy choice property
  - Optimal substructure
- Any problem having the above properties is guaranteed to have an optimal greedy solution

Greedy Choice Property

SoftUni Foundation

- Greedy choice property
  - A global optimal solution can be obtained by greedily selecting a locally optimal choice
  - Sub-problems that arise are solved by consequent greedy choices
    - Enforced by optimal substructure

Optimal Substructure Property

SoftUni Foundation

- Optimal substructure property
  - After each greedy choice the problem remains an optimization problem of the same form as the original problem
  - An optimal global solution contains the optimal solutions of all its sub-problems

Greedy Algorithms: Example

- The "Max Coins" game
  - You are given a set of coins
  - You play against another player, alternating turns
  - Per each turn, you can take up to three coins
  - Your goal is to have as many coins as possible at the end

Max Coins – Greedy Algorithm

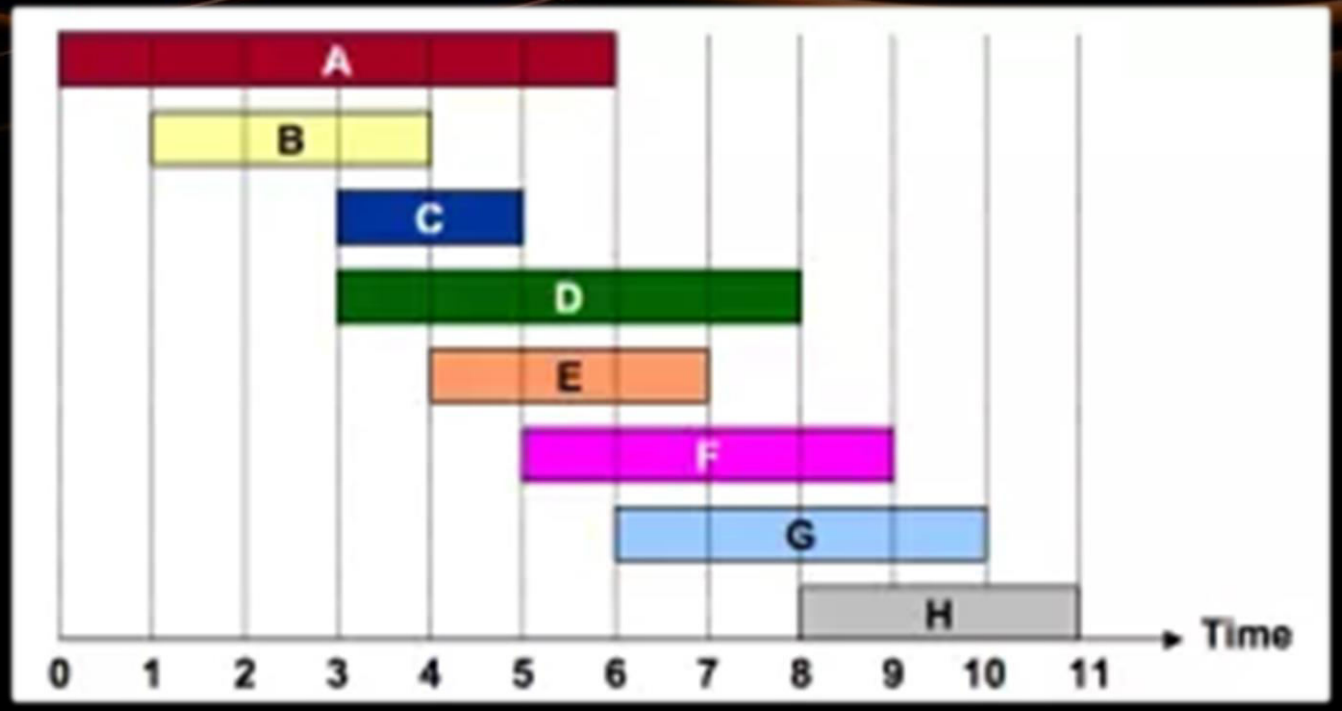
SoftUni Foundation

- A simple greedy strategy exists for the "Max Coins" game

```
At each turn take the maximum number of coins
```

- Always choose the local maximum (at each step)
  - You don't consider what the other player does
  - You don't consider your actions' consequences
- The greedy algorithm works optimally here
  - It takes as many coins as possible

SoftUni Foundation



Activity Selection Problem

The Greedy Algorithm and Why It Is Optimal

Activity Selection Problem

SoftUni Foundation

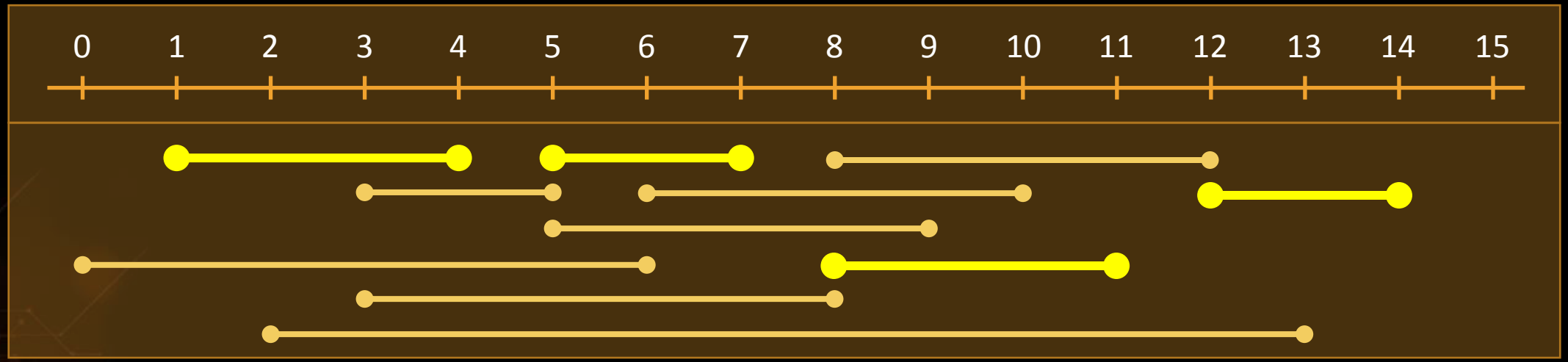
- The Activity Selection Problem
  (a.k.a. Conference Scheduling Problem)
  - Given a set of activities $S = \{a_1, a_2, \ldots, a_n\}$
    - Each having a start & finish time: $a_i = \{s_i, f_i\}$
  - Activities are "compatible" if they don't overlap
    - i.e. their intervals do not intersect
  - What is the maximum-size subset of compatible activities?
    - i.e. which is the largest list of compatible activities that can be scheduled?

Activity Selection Problem – Example

SoftUni Foundation

- Activity selection problem example:

Index	1	2	3	4	5	6	7	8	9	10	11
Start (s_i)	1	3	0	5	3	5	6	8	8	2	12
Finish (f_i)	4	5	6	7	8	9	10	11	12	13	14



- Several optimal solutions: $\{a_1, a_4, a_8, a_{11}\}$, $\{a_2, a_4, a_9, a_{11}\}$, ...

Activity Selection Problem: Greedy

SoftUni Foundation

- A greedy algorithm for the Activity Selection Problem:

```
While (S in not empty)
  Select from S an activity A with the earliest finish
  Remove A + all activities from S conflicting with A
```

- Greedy characteristics of the above algorithm
  - Taking the earliest finish activity gives more time for all others
  - We pick an activity preserving the "maximum remaining time"

Activity Selection Problem: Implementation

SoftUni Foundation

- Single scan (linear) implementation of the greedy algorithm for the Activity Selection Problem:

```
Sort the activities from S by their finish time
last = first activity from S
for (a in S)
  if (a.Start >= last.Finish)
    // a does not conflict with last
    print a
    last = a
```

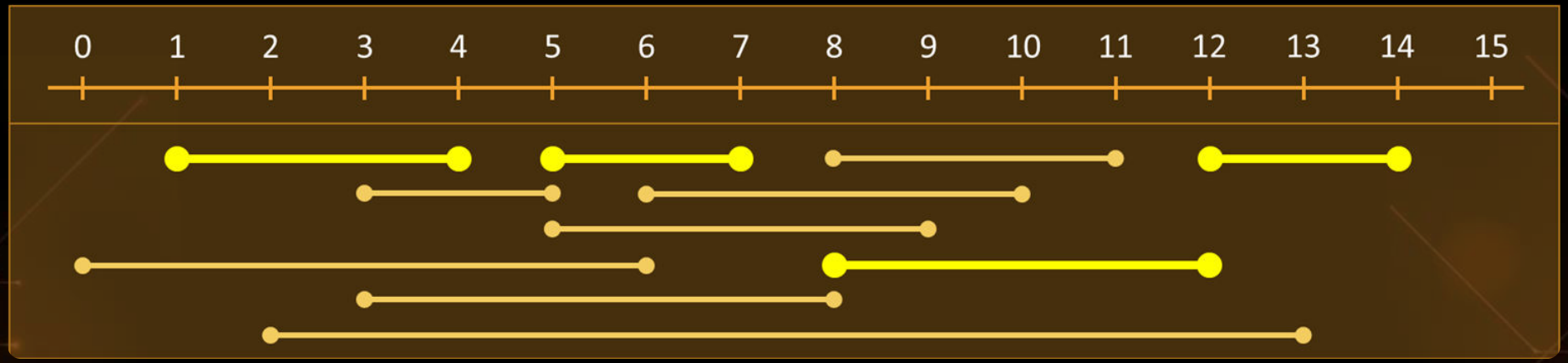
Proving Greedy Optimality

SoftUni Foundation

- To prove the discussed greedy is optimal we:
  - Need to prove the problem has a greedy choice property
  - Need to prove the problem has optimal substructure
- Greedy properties are satisfied:
  - A solution starting with a greedy choice exists: the earliest finish
  - Optimal substructure exists:
    - If we remove the activity A with the earliest finish time activity
    - We reduce the problem to the same problem of smaller size (without activities in S, which intersect the activity A)

SoftUni Foundation

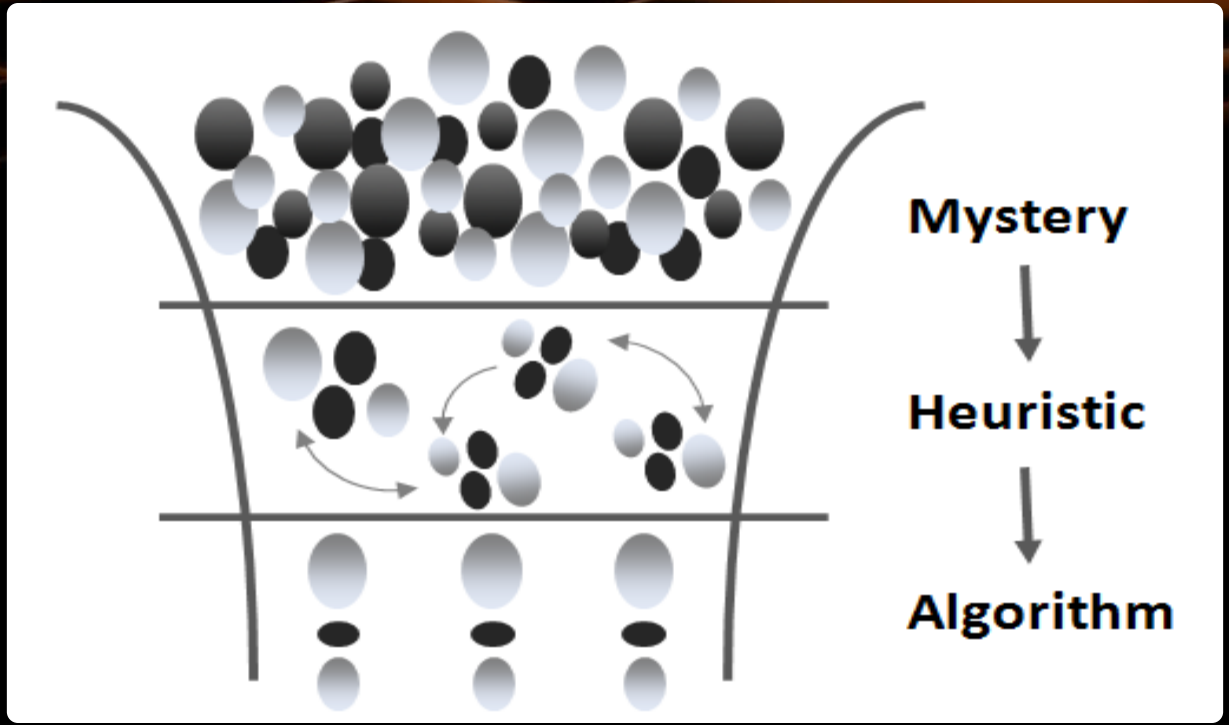
Index	1	2	3	4	5	6	7	8	9	10	11
Start (s_i)	1	3	0	5	3	5	6	8	8	2	12
Finish (f_i)	4	5	6	7	8	9	10	11	12	13	14



Activity Selection Problem

Live Demo

SoftUni Foundation



Heuristic Algorithms

Greedy, Genetic, Branch and Bound

Heuristic Algorithms

SoftUni Foundation

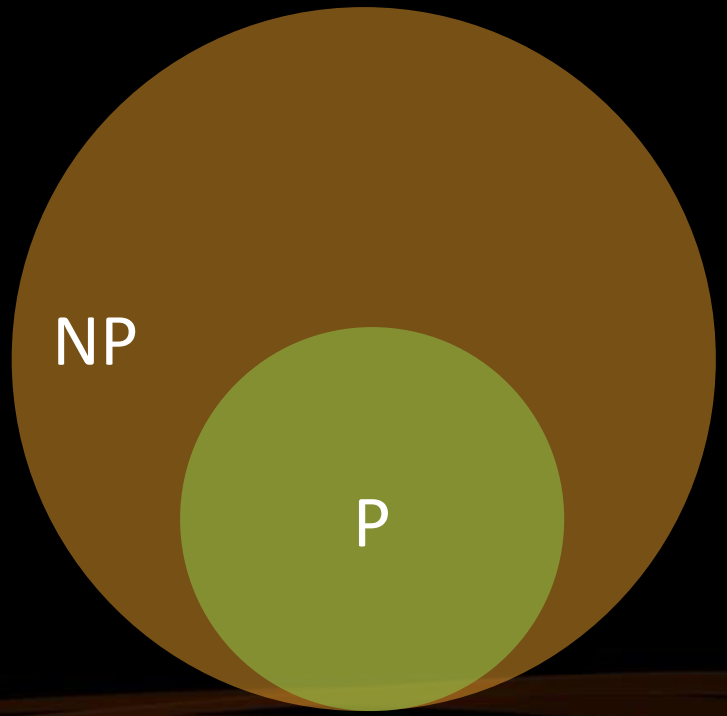
- Some problems are hard to be solved (no fast algorithm exists)
- Heuristic algorithms solve hard problems approximately
  - They find a good solution, but cannot guarantee it is optimal
- Examples of heuristic algorithms:
  - Greedy algorithms – make a greedy choice at each step
  - Genetic algorithms – based on inheritance, mutation, selection, …
  - Branch and bounds – optimized backtracking with cut-offs
  - Others…

P vs NP

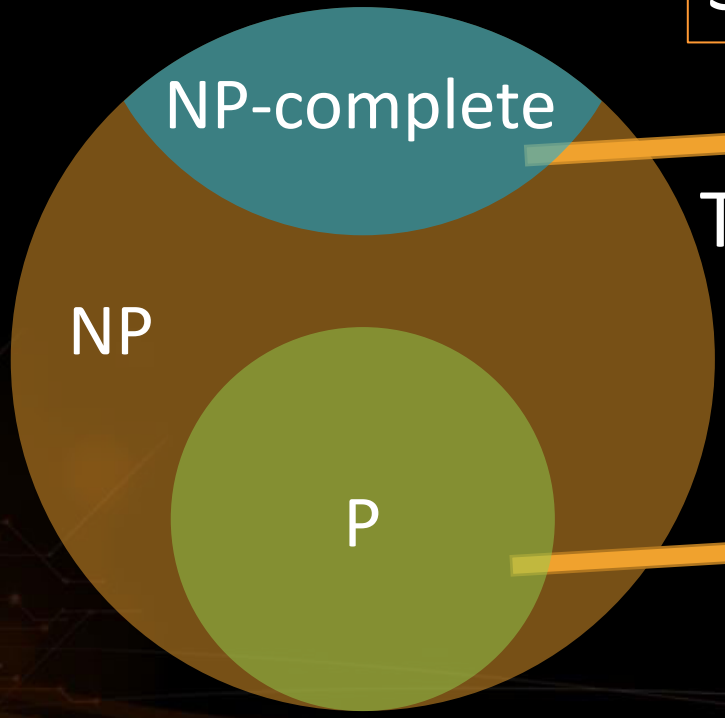
- P - Category in which the algorithms are solvable in polynomial time
- Polynomial -> $5N^5 + 3N^4 + N$ is $O(N^5)$
- Not a polynomial -> 2^N

P vs NP

- NP - Category in which the algorithms are verifiable in polynomial time.
- Decision problems
- Example:
  - Does a route shorter than L exist?
  - Yes/No



P vs NP



Subset sum

Knapsack
(Can a value >= V be achieved?)

Set-Cover problem

The hardest in NP

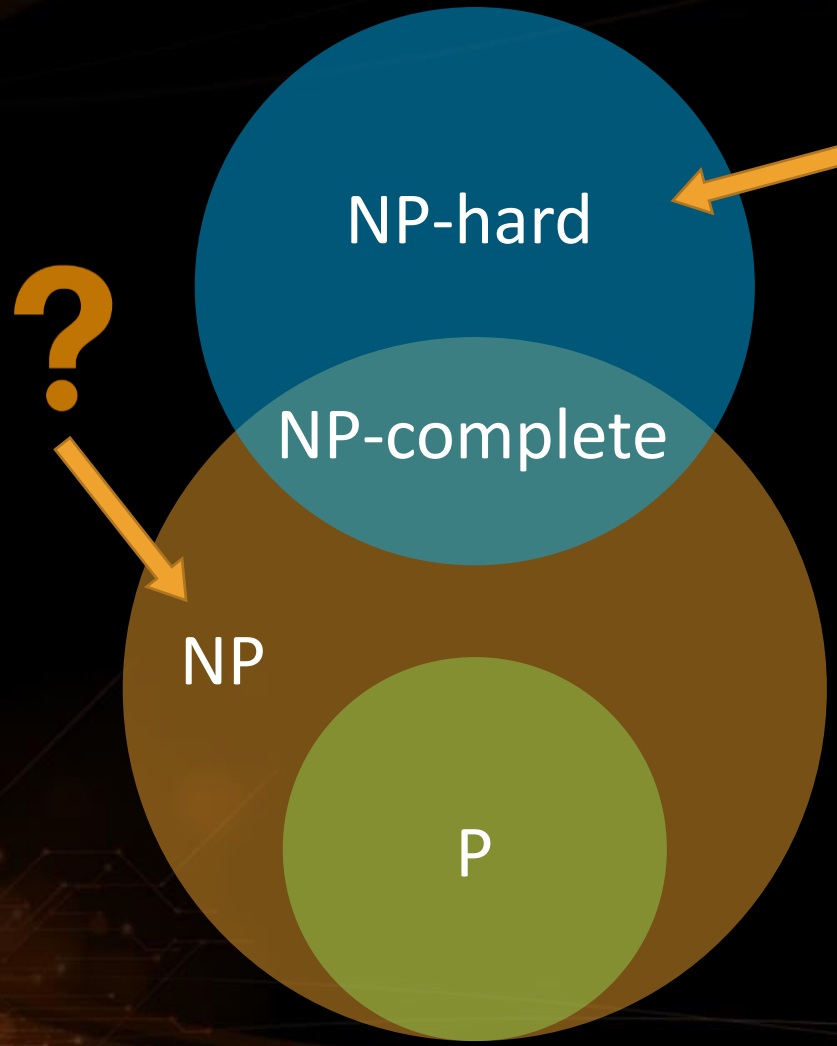
Finding shortest route

Sorting

Look-ups in hash tables

Traversing lists

P vs NP



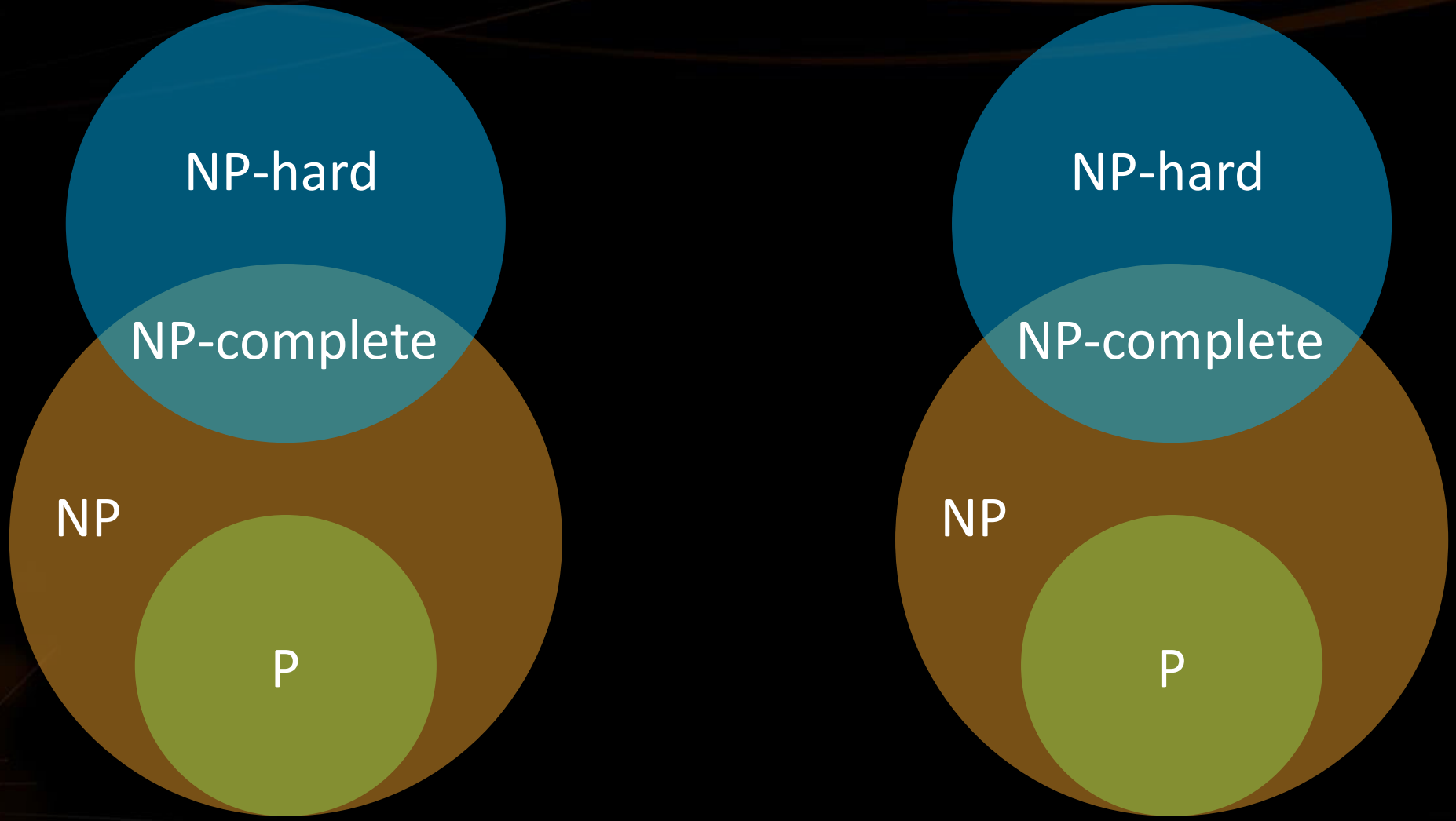
At least as hard
as NP-complete

Knapsack (optimization)
(maximize V)

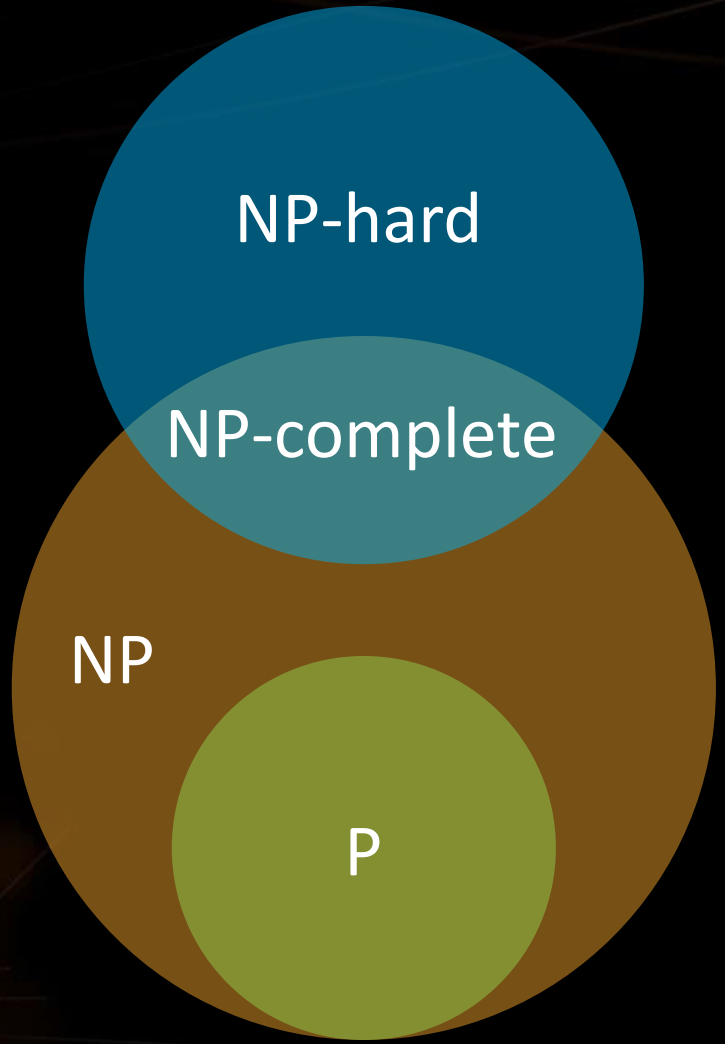
Halting problem

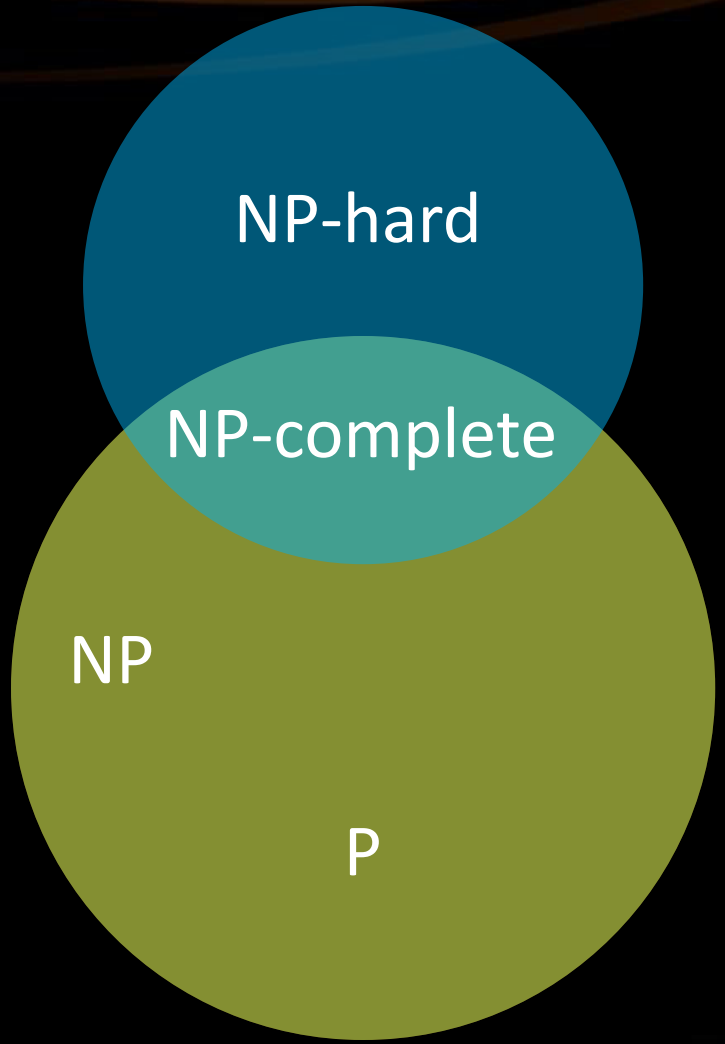
Traveling salesman

P vs NP

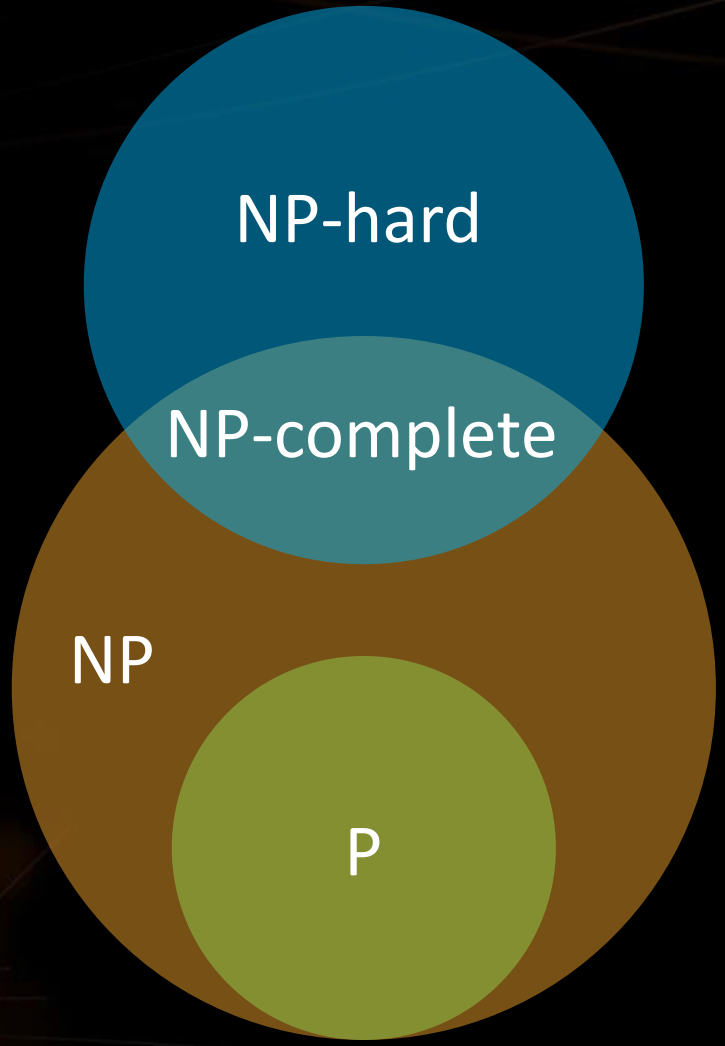


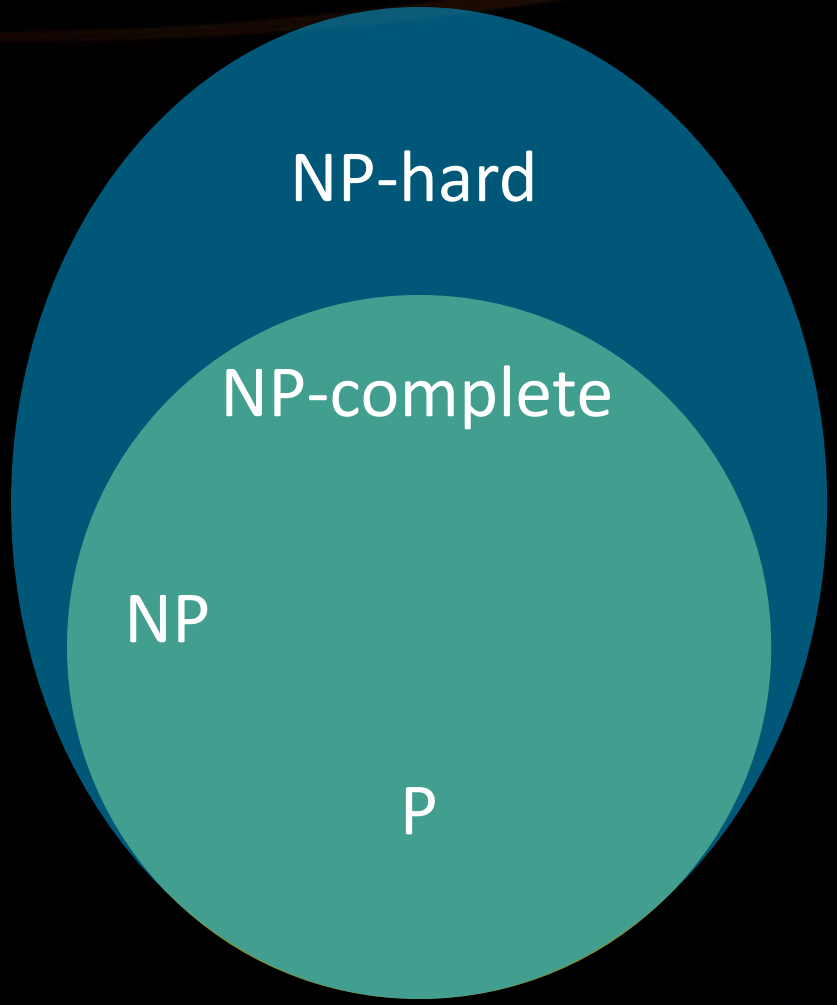
P vs NP





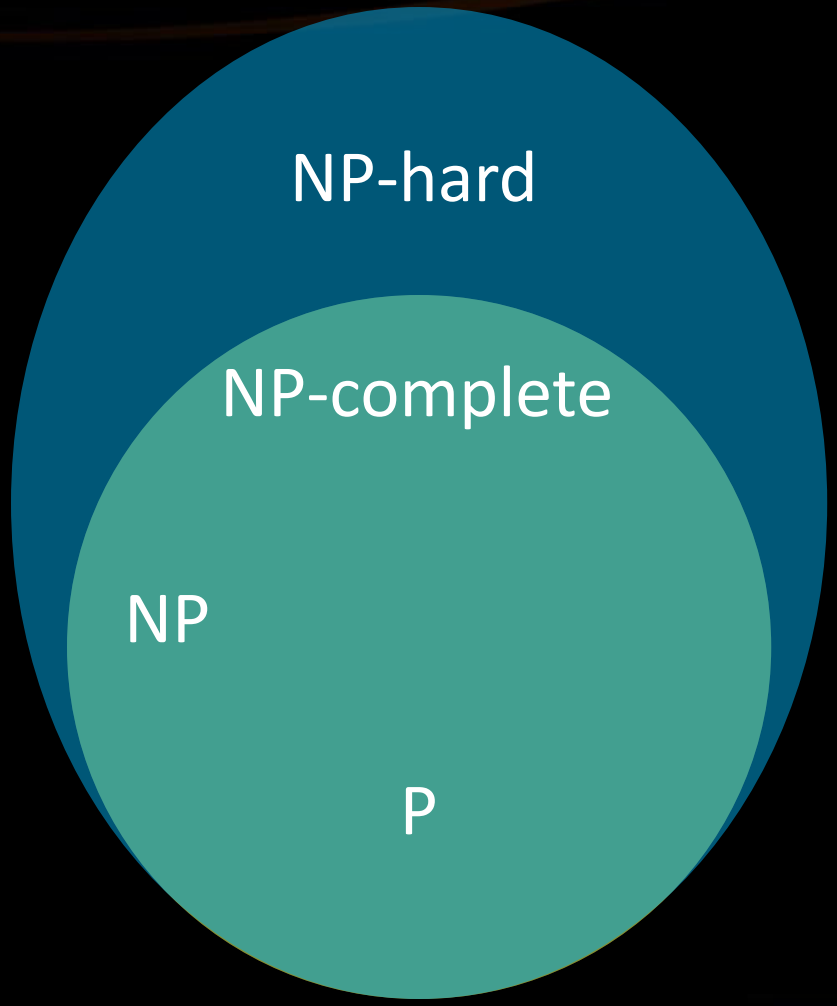
P vs NP



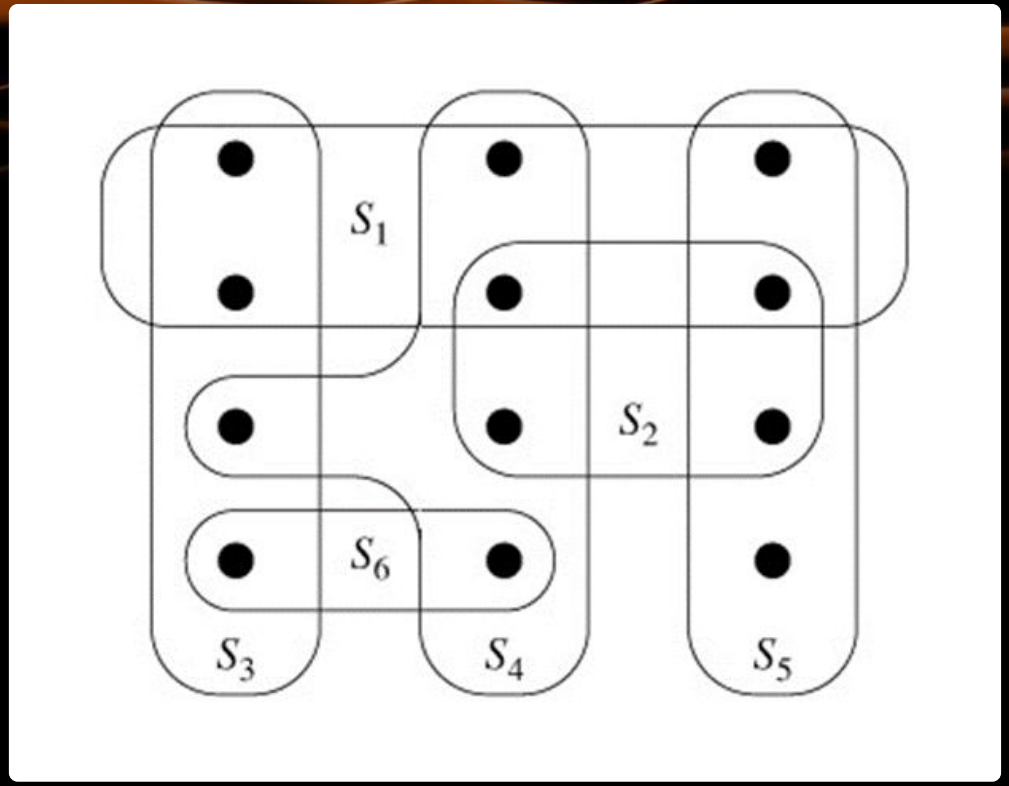


P vs NP

- Millennium Problem
  - Prize of $1,000,000
  - Prove either P=NP or P!=NP
  - Currently > 100 attempts



SoftUni Foundation



The Set Cover Problem

Using Greedy for Approximation

Set Cover Problem

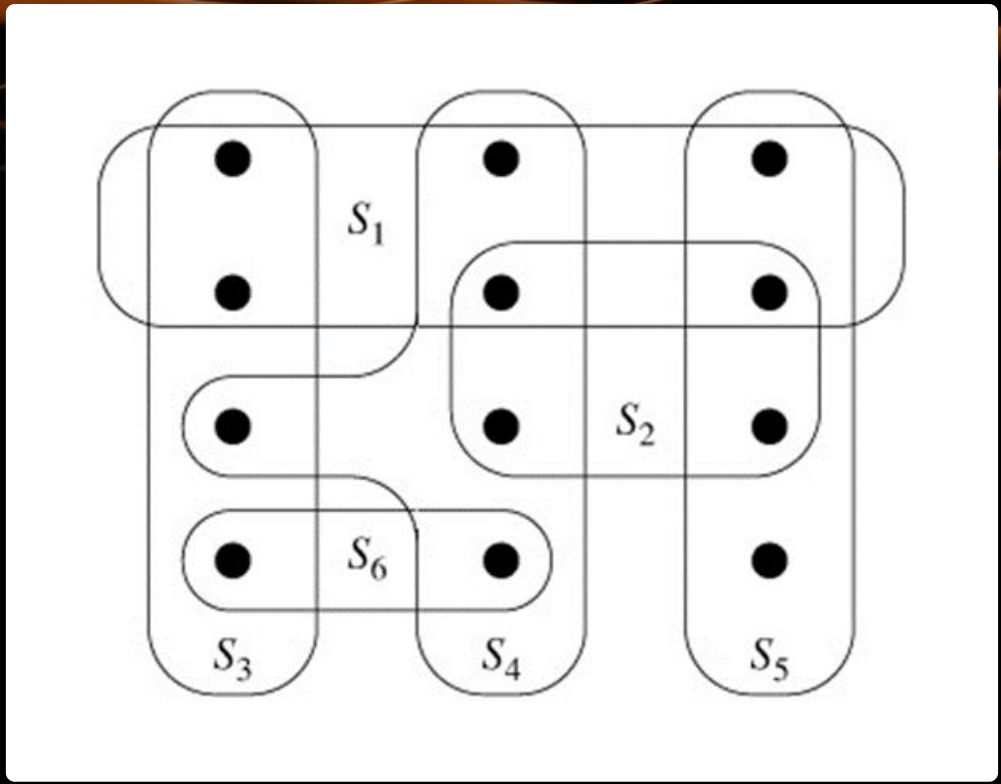
SoftUni Foundation

- The Set Cover Problem (SCP) is such a problem
- SCP formulation:
  - Given a set U={1,2,…,m} called "the Universe"
  - And a set S={{…}, {…}, {…}, …} of n sets whose union = U
  - Find the smallest subset of S, the union of which = U (if it exists)
- Example of SCP:
  - U = {1, 2, 3, 4, 5, 6}
  - S = { {1, 3}, {1}, {2, 4}, {2, 5}, {2, 6}, {2, 3, 6}, {4, 6}, {4, 5, 6}, {6} }

Set Cover Problem

SoftUni Foundation

- The SCP turns out very complex!
  - The optimal solution is NP-complete
  - Infeasible for real-world calculations (unless P = NP)
- Good solutions can be achieved through a greedy approach:
  - At each step pick the set containing the largest number of uncovered elements, e. g.
    - U = {1, 2, 3, 4, 5, 6}
    - S = { {1, 3}, {1}, {2, 4}, {2, 5}, {2, 6}, {2, 3, 6}, {4, 6}, {4, 5, 6}, {6} }



The Set Cover Problem

Live Coding Exercise (Lab)

References & Further Reading

SoftUni Foundation

- Recommended further reading on greedy algorithms:
  - Lecture @ Boston University (Shang-Hua Teng)
    - www.cs.bu.edu/~steng/teaching/Fall2003/lectures/lecture7.ppt
  - Lecture @ University of Pennsylvania
    - www.cis.upenn.edu/~matuszek/cit594-2005/Lectures/36-greedy.ppt
  - Book on Algorithms @ Berkeley University
    - www.cs.berkeley.edu/~vazirani/algorithms/chap5.pdf
  - Book "Programming = ++ Algorithms;" – Chapter 9
    - www.programirane.org

Summary

- Greedy algorithms make local optimal decision at each step and expect to achieve the global optimal solution
- Greedy algorithms give optimal solution for problems holding:
  - Greedy choice & optimal substructure
- Heuristic algorithms give approximation
  - Not always the optimal solution

Greedy Algorithms

SoftUni Foundation

Questions?

https://softuni.bg/opencourses/algorithms

License

SoftUni Foundation

- This course (slides, examples, labs, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Fundamentals of Computer Programming with C#" book by Svetlin Nakov & Co. under CC-BY-SA license
  - "Data Structures and Algorithms" course by Telerik Academy under CC-BY-NC-SA license

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM