Local Search

Subset Sum and Backtracking

Outline

- Example problem: Subset Sum
  - Optimization vs decision
  - Solution space
- Local search
  - Partial solution
  - Backtracking
  - Stuck at local optimum

Outline

- Example problem: Subset Sum
  - Optimization vs decision
  - Solution space

- Local search
  - Partial solution
  - Backtracking
  - Stuck at local optimum

Learning objectives:
You are able to
- explain the concepts of a solution space of a problem and a partial solution to a problem
- design a local search / backtracking algorithm for the subset sum problem
- explain why a local search algorithm can get stuck at a local optimum

Subset Sum

Setting:
You find a pile of coins at home. You want to get rid of them, because who has coins?

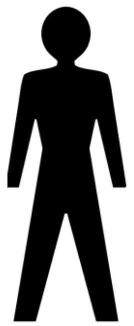
Coin system:
Each coin has an integer value.

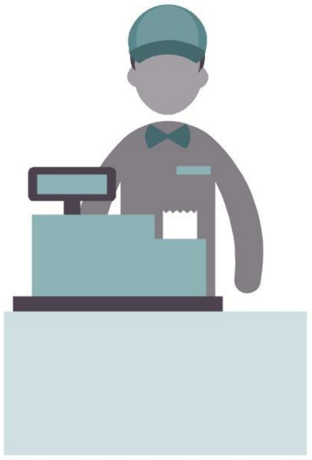
Subset Sum

Setting:
You find a pile of coins at home. You want to get rid of them, because who has coins?

Coin system:
Each coin has an integer value.

Subset Sum

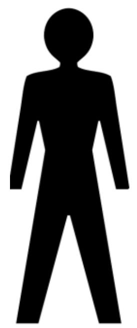


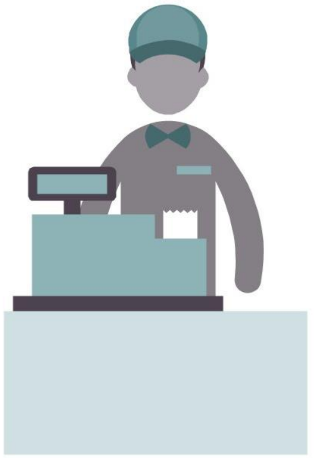


Setting:
You find a pile of coins at home. You want to get rid of them, because who has coins?

Coin system:
Each coin has an integer value.

Subset Sum





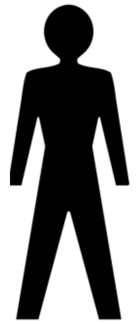
Setting:
You find a pile of coins at home. You want to get rid of them, because who has coins?

Coin system:
Each coin has an integer value.

You want to pay for your purchases with the coins.

You don’t want change!

Subset Sum



Input:
A (multi-)set $S \subset \mathbb{N}$

Coin system:
Each coin has an integer value.

You want to pay for your purchases with the coins.

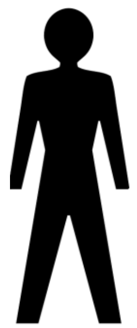
You don't want change!

Subset Sum

Input:
A (multi-)set $S \subset \mathbb{N}$

An element can appear multiple times.

Coin system:
Each coin has an integer value.



You want to pay for your purchases with the coins.

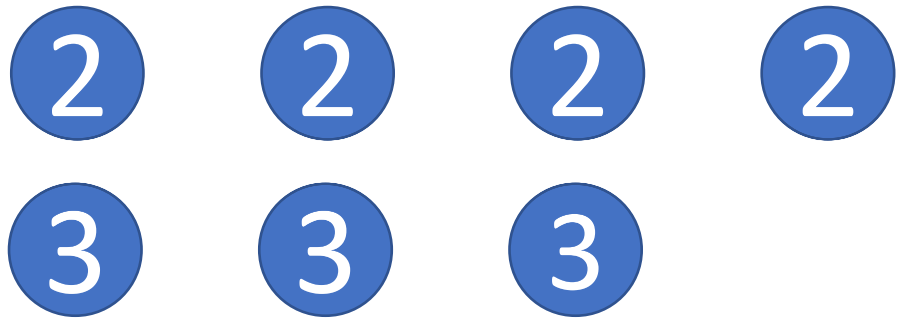
You don't want change!

Subset Sum – an Example

Spoiler alert:
The goal of the lecture is to introduce some terminology and to show that the backtrack search can be very slow.

Subset Sum – an Example

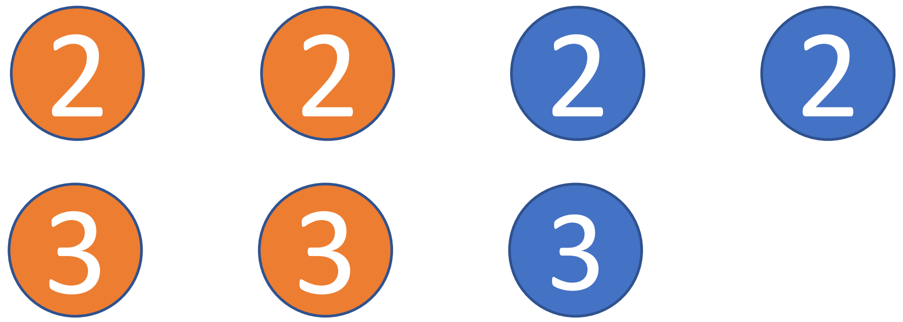
Input: multiset $\{2,2,2,2,3,3,3\}$



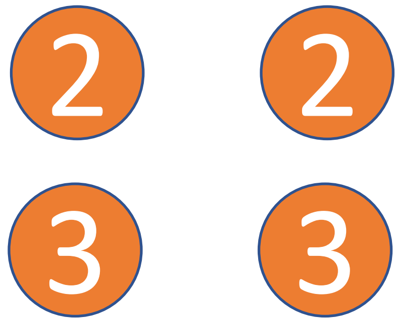
Decision problem:
Does a subset add up to 10?

Subset Sum – an Example

Input: multiset $\{2,2,2,2,3,3,3\}$



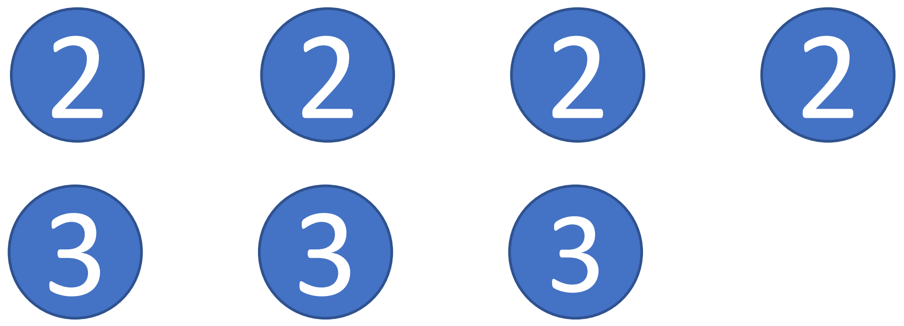
Answer: Yes!



Decision problem:
Does a subset add up to 10?

Subset Sum – an Example

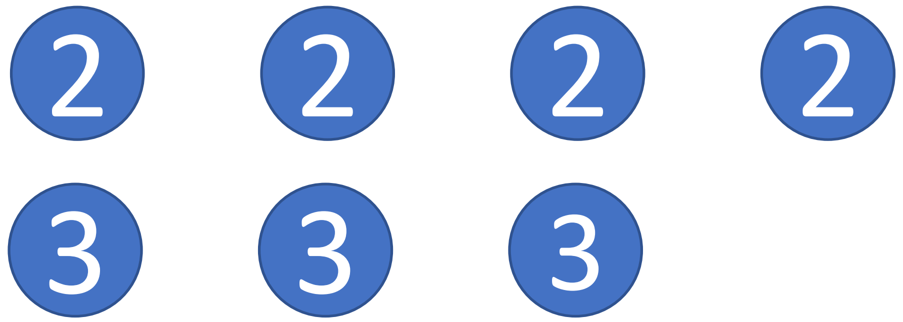
Input: multiset $\{2,2,2,2,3,3,3\}$



Optimization problem:
The largest (or smallest) set of coins that adds up to 8?

Subset Sum – an Example

Input: multiset $\{2,2,2,2,3,3,3\}$

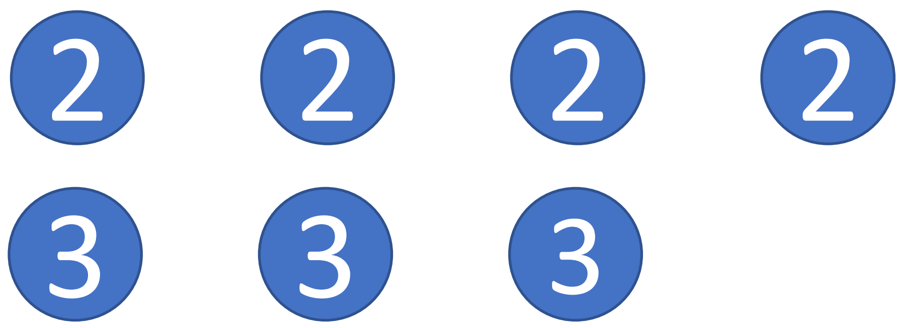


Optimization problem:
The largest (or smallest) set of coins that adds up to 8?

Two possibilities:

Subset Sum – an Example

Input: multiset $\{2,2,2,2,3,3,3\}$



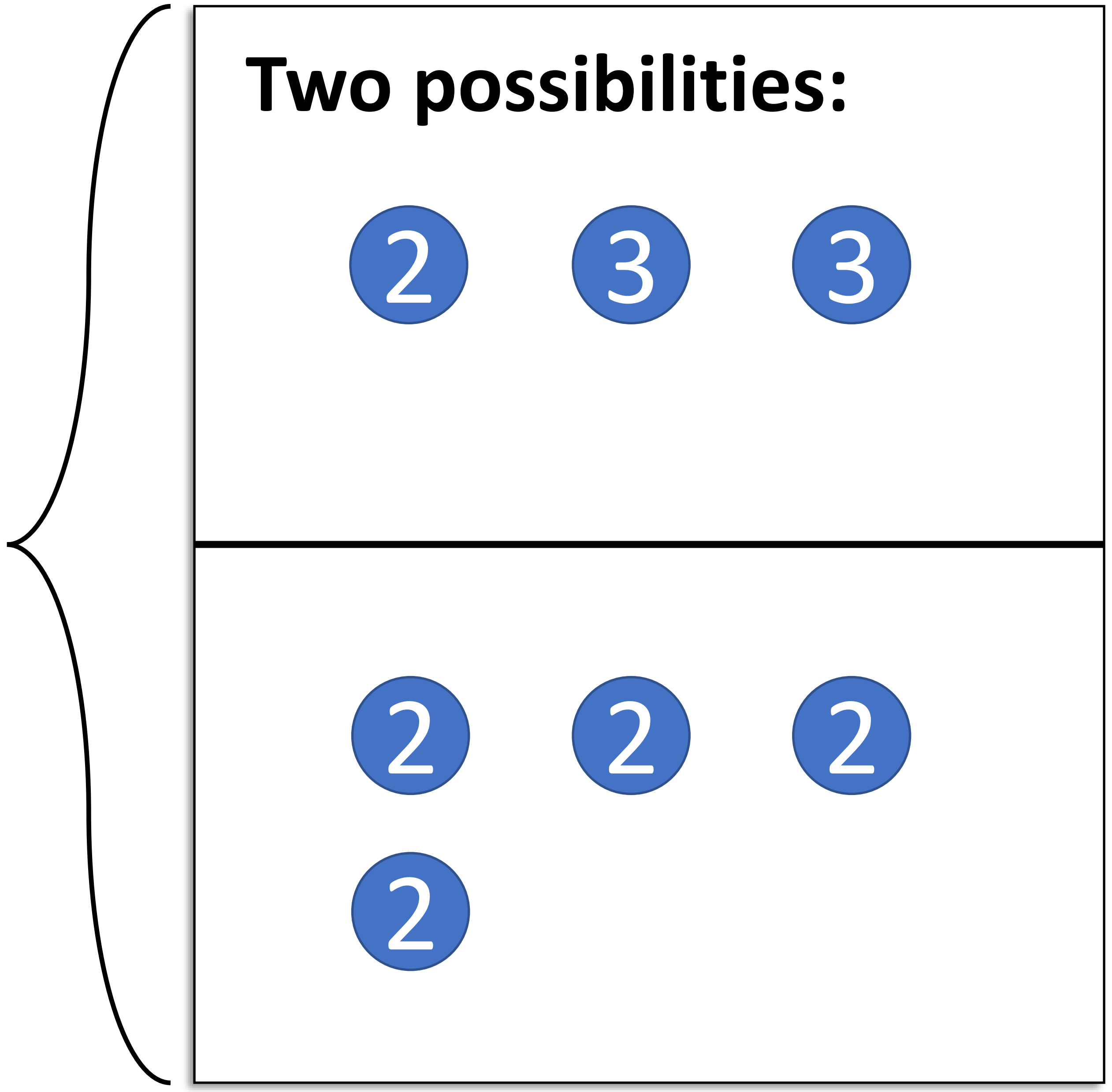
Optimization problem:
The largest (or smallest) set of coins that adds up to 8?

Two possibilities:

Subset Sum – an Example

The solution space:
The set of all possible feasible solutions

Sets that sum up to 8



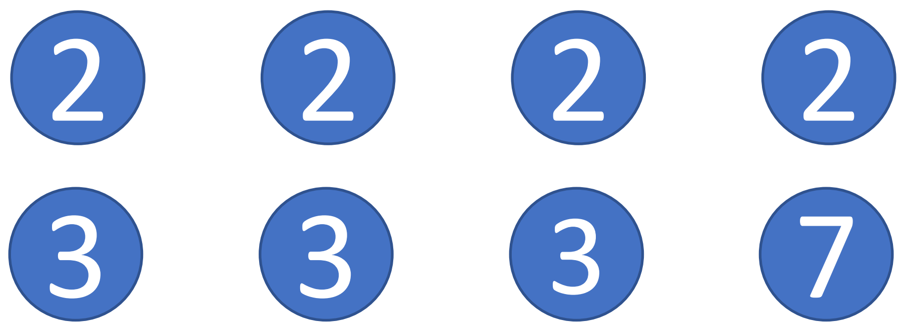
Outline

- Example problem: Subset Sum
  - Optimization vs decision
  - Solution space
- Local search
  - Partial solution
  - Backtracking
  - Stuck at local optimum

Subset Sum – Backtracking

Algorithm idea:
Iteratively choose smallest coin possible.

Input: multiset $\{2,2,2,2,3,3,3,7\}$



Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

2 2 2

3 3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

2

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

2 2

3 3 3 7

Algorithm idea:
Iteratively choose smallest coin possible.

2 2

Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

2

3 3 3 7

Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2

Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

3 3 3 7

Algorithm idea:
Iteratively choose
smallest coin possible.

2 2 2 2

Optimization problem:
The largest set of coins
that adds up to 9?

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

3 3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2 2

A partial solution:
Sum is 8. Now what?

Subset Sum – Backtracking

Input: multiset {2,2,2,2,3,3,3,7}

3 3 7

Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2 2

3

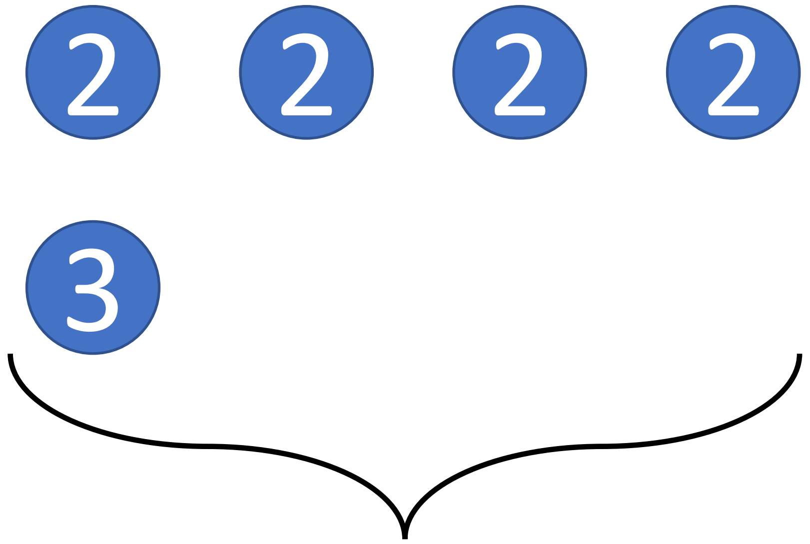
Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.



Sum is 11

Subset Sum – Backtracking

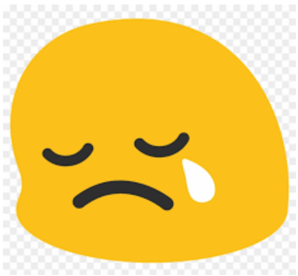
Input: multiset {2,2,2,2,3,3,3,7}

3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

$\underbrace{2 \quad 2 \quad 2 \quad 2 \quad 3}_{\text{Sum is 11}}$



Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

3 3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2 2

A partial solution:
Sum is 8. Now what?

Subset Sum – Backtracking

Input: multiset {2,2,2,2,3,3,3,7}

3 3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2 ~~2~~

Backtrack!

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

2

3 3 3 7

Optimization problem:
The largest set of coins that adds up to 9?

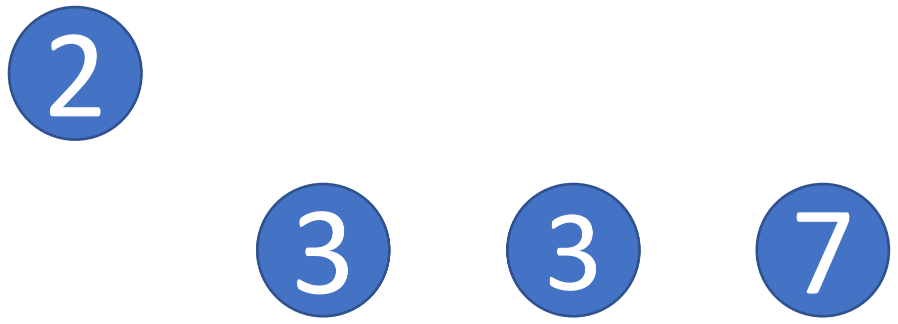
Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2

Backtrack!

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$



Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.

Backtrack!

Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$

2

3 3 7

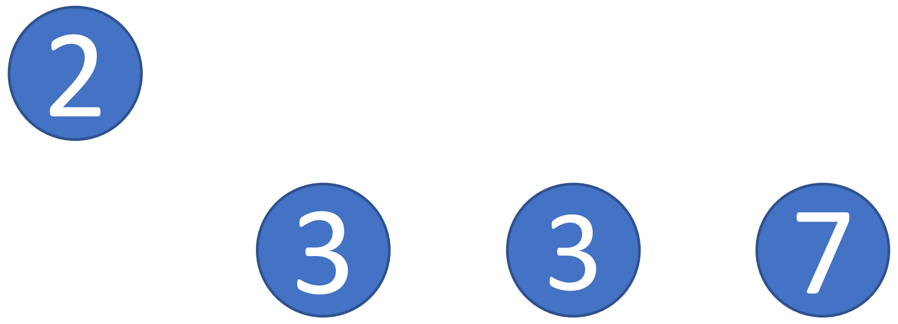
Algorithm idea:
Iteratively choose smallest coin possible.

2 2 2 3

Optimization problem:
The largest set of coins that adds up to 9?

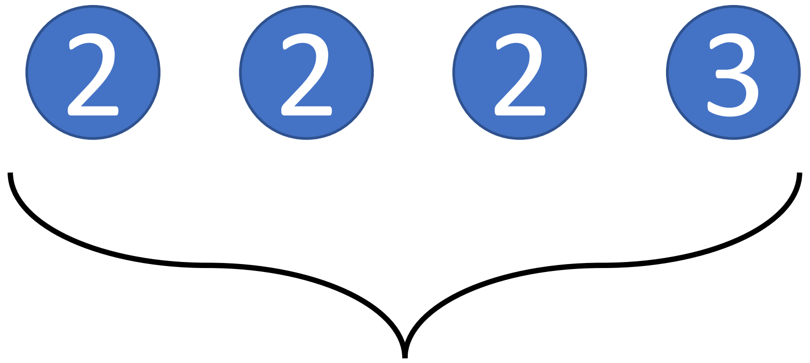
Subset Sum – Backtracking

Input: multiset $\{2,2,2,2,3,3,3,7\}$



Optimization problem:
The largest set of coins that adds up to 9?

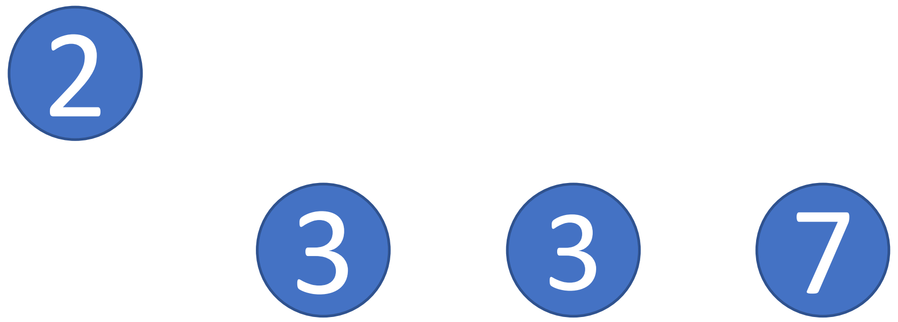
Algorithm idea:
Iteratively choose smallest coin possible.



Sum is 9.
This happens to even be optimal.

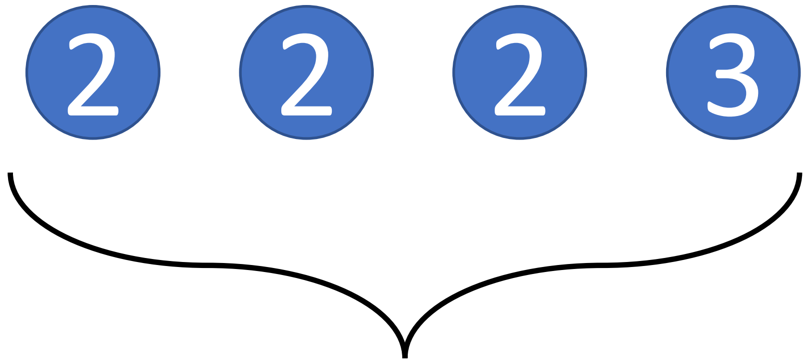
Subset Sum – Backtracking

Input: multiset {2,2,2,2,3,3,3,7}



Optimization problem:
The largest set of coins that adds up to 9?

Algorithm idea:
Iteratively choose smallest coin possible.



Sum is 9.
This happens to even be optimal.

Subset Sum – Backtracking

Algorithm idea:
Iteratively choose smallest coin possible.

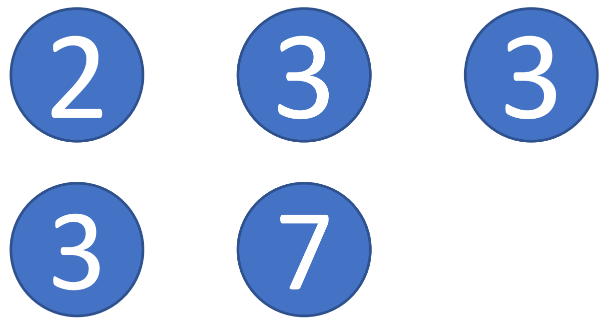
Choosing the currently "best looking" alternative is called *greedy.*

Outline

- Example problem: Subset Sum
  - Optimization vs decision
  - Solution space
- Local search
  - Partial solution
  - Backtracking
  - Stuck at local optimum

Subset Sum – Local Optimum

Input: $\{2,3,3,3,7\}$



Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Local Optimum

Input: $\{2,3,3,3,7\}$

2 3 3

3 7

The local search:

2

7

Optimization problem:
The largest set of coins that adds up to 9?

Subset Sum – Local Optimum

Input: $\{2,3,3,3,7\}$

2 3 3
3 7

Optimization problem:
The largest set of coins that adds up to 9?

The local search:

2
7

Optimum:

3 3
3

Subset Sum – Local Optimum

Input: $\{2,3,3,3,7\}$

2 3 3
3 7

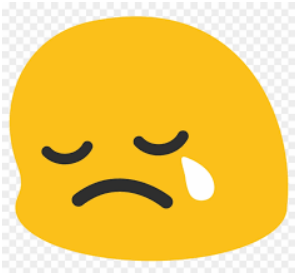
Optimization problem:
The largest set of coins that adds up to 9?

The local search:

2
7

Optimum:

3 3
3



Subset Sum – Local Optimum

Input: $\{2,3,3,3,7\}$

2 3 3
3 7

Optimization problem:
The largest set of coins that adds up to 9?

The local search:

2
7

Local optimum

Optimum:

3 3
3

Subset Sum – Brute Force

Backtracking:

1. Also backtrack from feasible solutions.
2. Eventually, go through all possibilities

maxSize $= 0$

SubsetSum($\boldsymbol{X}, \boldsymbol{T}, \boldsymbol{s}$)

If($T \; = 0$)

  maxSize $:=$ max$\{s,$ maxSize$\}$

  return

If($T < 0$ or $X = \emptyset$)

  return

ForEach(element $x \in X$)

  SubsetSum($\boldsymbol{X}$ $\setminus x, T - x, s + 1$**)**

Subset Sum – Brute Force

maxSize $= 0$
SubsetSum($\boldsymbol{X, T, s}$)
If($T\ = 0$)
    maxSize $:=$ max$\{s,$ maxSize$\}$
    return

If($T < 0$ or $X = \emptyset$)
    return

ForEach(element $x \in X$)
    SubsetSum($\boldsymbol{X} \setminus x, T - x, s + 1$)

Correctness:
Goes through all possible feasible solutions.

Runtime:
Go through all subsets: $O(2^n)$

Subset Sum – Brute Force

maxSize $= 0$
SubsetSum($\boldsymbol{X}, \boldsymbol{T}, \boldsymbol{s}$)
If($T \ = 0$)
    maxSize $:=$ max$\{s,$ maxSize$\}$
    return

If($T < 0$ or $X = \emptyset$)
    return

ForEach(element $x \in X$)
    SubsetSum($\boldsymbol{X} \setminus x, T - x, s + 1$)

Correctness:
Goes through all possible feasible solutions.

Runtime:
Go through all subsets: $O(2^n)$

Number of coins

Subset Sum – Brute Force

maxSize $= 0$
SubsetSum($\boldsymbol{X}, \boldsymbol{T}, \boldsymbol{s}$)
If($T \; = 0$)
    maxSize $:=$ max$\{s, \text{maxSize}\}$
    return

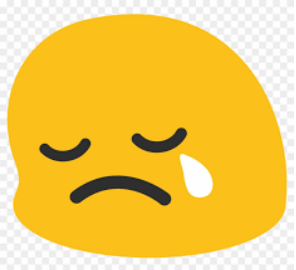
If($T < 0$ or $X = \emptyset$)
    return

ForEach(element $x \in X$)
    SubsetSum($\boldsymbol{X} \setminus x, T - x, s + 1$)

Correctness:
Goes through all possible feasible solutions.

Runtime:
Go through all subsets: $O(2^n)$

Number of coins



Remark about runtime

Runtime:
Go through all subsets: $O(2^n)$

The problem is hard, i.e., it is believed that we cannot do polynomial time in general.

This hardness is very subtle. Check Book chapter 3.8 for details. There is a DP algorithm with complexity nT,where T is the target sum. But T can be exponential.

Remark about runtime

Runtime:
Go through all subsets: $O(2^n)$

The problem is hard, i.e., it is believed that we cannot do polynomial time in general.

In the next lecture, we will see an example where a greedy local search works really well.

Wrap-up

Local search:

- Search space
- Partial solution

Step by step augment the partial solution.

Backtracking:
An algorithm for the subset sum problem

Brute force:
Very slow runtime but analysis trivial.