Pseudo-Polynomial

Review

Pseudo-Polynomial Time

- Polynomial: polynomial in the **length of the input** (#bits for the input)
- Pseudo-polynomial: polynomial in the **numeric value**

- The time complexity of 0-1 knapsack problem is $\Theta(nW)$
  - n: number of objects
  - W: knapsack's capacity (non-negative integer)
  - polynomial in the numeric value
    = pseudo-polynomial in input size
    = exponential in the length of the input

https://youtu.be/9oI7fg-MIpE

Time Complexity Definition

- Time complexity is in measure the time an algorithm takes to run as a function of
  - ✅ the length of the input in bits
  - ✅ the value of the input

```
function(n)
  for i = 1 to n
    print i
```

$$O(n) = O(2^{\text{bits in } n}) = O(2^m)$$

- `n` is a value

$n = 4 = \underbrace{100}_{3}$

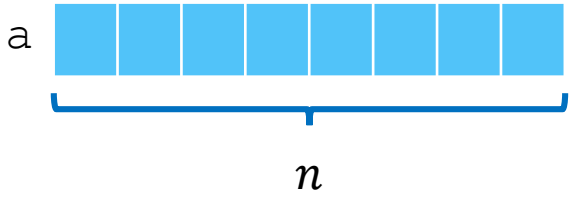
$n = 8 = \underbrace{1000}_{4}$

$n = 16 = \underbrace{10000}_{5}$

```
function(a)
  for i = 1 to n
    print i
```

$O(n)$

- `a` is an array



`a = [0, 1, 1]` (3)

`a = [0, 1, 0, 1]` (4)

`a = [0, 1, 0, 1, 1]` (5)

Time Complexity Definition

- Time complexity is in measure the time an algorithm takes to run as a function of
  - ✔ the length of the input in bits
  - ✔ the value of the input

```
function(n)
  for i = 1 to n
    print i
```

$O(n)$
$= O(2^{\text{bits in } n})$
$= O(2^m)$

- `n` is a value

```
function(a)
  for i = 1 to n
    print i
```

$O(n)$

- `a` is an array

- The time complexity of 0-1 knapsack problem is $\Theta(nW) = \Theta(n2^{\text{bits in } W}) = O(n2^m)$
  - n: number of objects
  - W: knapsack's capacity (non-negative integer)

= exponential in the length of the input
= polynomial in the numeric value
= pseudo-polynomial in input size