0/1 knapsack problem

- Only one of each item is available

$x_i \in \{0, 1\}$

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq C$$

Input: n items with integer weights $\{w_1, w_2, \ldots, w_n\}$ and values $\{v_1, v_2, \ldots v_n\}$ and a knapsack capacity C

Output: the subset with maximum value where the sum of weights does not exceed C

BRUTE FORCE KNAPSACK(W, V, C)

```
1  bestSum = 0
2  bestSubset = []
3  for each subset of [1...n]
4        sum = 0
5        weight = 0
6        for each i in this subset
7              sum = sum + V[i]
8              weight = weight + W[i]
9        if weight ≤ C and sum > bestSum
10             bestSum = sum
11             bestSubset = current subset
```

$O(n\,2^n)$

$O(n)$

2^n

Greedy Strategies (counter examples)

- Most valuable first

$w = \{2, 2, 4\}$ $C = 4$

$v = \{3, 3, 5\}$

- Lightest first

$w = \{1, 2\}$ $C = 2$

$v = \{1, 2\}$

- Best value/weight ratio

$w = \{5, 1, 4\}$

$v = \{25, 6, 16\}$ $C = 5$

Dynamic Programming

- Define $M[i,j]$ as the optimal value for filling a capacity j knapsack with some subset of the first i items

$$M[i,j] = \max \begin{cases} M[i-1,j] \leftarrow \text{don't take item } i \\ M[i-1, j-w_i] + v_i \end{cases}$$

take item i

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	1	1	1	1	1	1
2	0	0	0	0	4	4	4	4	4	5	5
3	0	0	0	0	4	4	4	4	4	5	7
4	0	0	0	5	5	5	5	9	9	9	9

$$C = 10$$

$$V = \{1, 4, 3, 5\}$$

$$W = \{5, 4, 6, 3\}$$

```
TABULATION KNAPSACK(W, V, C)
1   M = empty n + 1 × C + 1 table
2   initialize the first row to all zeroes
3   for i = 1 to n
4       M[i, 0] = 0
5       for j = 1 to C
6           if W[i] > j
7               M[i, j] = M[i − 1, j]
8           else
9               a = M[i − 1, j]          ← don't take item i
10              b = M[i − 1, j − W[i]] + V[i]   ← take item i
11              M[i, j] = max(a, b)
12  return M[n, C]
```

Time Complexity: $\Theta(nC)$

Space Complexity: $\Theta(nC)$

```
MemoizedKnapsack(W, V, C, i, j)
1  if M[i, j] is empty
2      if W[i] > j
3          M[i, j] = MemoizedKnapsack(W, V, C, i − 1, j)
4      else
5          a = MemoizedKnapsack(W, V, C, i − 1, j)
6          b = MemoizedKnapsack(W, V, C, i − 1, j − W[i]) + V[i]
7          M[i, j] = max(a, b)
8  return M[i, j]
```

Space Complexity: $\Theta(nC)$

Time Complexity: $\Theta(nC)$

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0		0	0		1	1			1
2		0			4			4			5
3								4			7
4											9

$$C = 10$$

$$V = \{1, 4, 3, 5\}$$

$$W = \{5, 4, 6, 3\}$$

Complexity

$\Theta(nC)$

$n = 50$

$C = 31$

$\Theta(nC) \approx 1550$

what happens if n doubles?

$n = 100$ $\quad \Theta(nC) \approx 3100$

What happens if the length of C doubles?

$C = 1023$ $\quad \Theta(nC) \approx 50{,}000$

A child is climbing a flight of n stairs. The child can take either 1, 2, or 3 steps at a time. How many possible ways can the child climb the stairs?

$n=0$	1
$n=1$	1
$n=2$	2
$n=3$	4
$n=4$	7
$n=5$	13
$n=6$	24
$n=7$	44
$n=8$	81

$$S(n) = S(n-1) + S(n-2) + S(n-3)$$