In general, the knapsack problem is as follows. Given some set of items worth some amount each, stuff your knapsack with as much value as possible. The precise formulation depends on your application. We will look at two of them.

The fractional knapsack problem is the following. Given $W \in \mathbb{N}$ and a set of items $I \subseteq \mathbb{R}^+ \times \mathbb{N}$, where $i = (w, v)$ has weight w and value v, determine the most valuable set $I^* \subseteq [0,1]^I$ subject to the constraint $\sum_{i \in I} I^*_i \cdot i.w \leq W$.

(I^*: what percentage of each item that we take)

How do we decide what items to take?

Just take as much as you can of the highest value density items.

```
FKS(W, I)
    Let ρ : I → ℝ be ρ(i) = i.v / i.w
    Sort I in descending ρ order
    Let K be an empty knapsack
    Let w' = 0
    While w' < W and K ≠ I
        Put as much of the first element i = (w,v) of I into K as possible
        If we put all of i into K, w' += i.w
        If not w' = W
    Return K
```

The runtime here is $O(I \log I)$.

The 0-1 Knapsack Problem is the following. Given $W \in \mathbb{N}$ and a set of items $I \subseteq \mathbb{Z}^+ \times \mathbb{N}$, where $i = (w, v)$ has weight w and value v, determine the most valuable set $I^* \subseteq I$ subject to the constraint $\sum_{i \in I^*} i.w \leq W$.

How do we decide which items to take?
What is the recursive formula for the maximum value?
Give the algorithm.
What is the runtime?

(see rod cutting)

The trick is there is a first item we must take, which leaves the rest of the subproblem to solve optimally.

$$V(W, I) = \begin{cases} -\infty & W < 0 \\ 0 & W = 0 \text{ or } I = \emptyset \\ \max_{i \in I} \left(V(W - i.w, I \setminus \{i\}) + i.v \right) & \text{otherwise} \end{cases}$$

This works, but we can build a better solution by just considering if the next item is in I^* or not. similarly to if vertex i is on the SP or not with Floyd-Warshall.

$$V(W, I) = \begin{cases} -\infty & W < 0 \\ 0 & W = 0 \text{ or } I = \emptyset \\ \max \left(V(W, I \setminus \{i\}), V(W, I \setminus \{i\}) + i.v \right) & \text{otherwise and } i \in I \end{cases}$$

If we fix an ordering of I (usually the order it comes as), we get a simple algorithm. Since $|I| < \infty$, there are a finite number of possible weight combos, so we can assume that they are integers, which makes the problem less messy. W is also an integer.

```
01KS(W, I)
    Let V: Z_{≥0} × Z_{≥0} → R be a map (probably backed with a plain array)
                                (out of bounds calls return -∞)
    Let V(w, 0) = 0  for all w

    For w = 1 to W
        For i = 1 to |I|
            V(w, i) = max(V(w, i-1), V(w - I_i.w, i-1) + I_i.v)
                      ↑ we don't include item i      ↑ we do include item i

    Return V(W, |I|)
```

The runtime is $O(WI)$.