We are trying to get $c[n, w]$ based on the recursive relations below.

Input n items where i-th item has value v_i and weighs w_i

Output the maximum value of items that can be carried in a knapsack of capacity w

$$c[i, w] = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ c[i-1, w] & \text{if } w_i > w \\ \max\{v_i + c[i-1, w - w_i], c[i-1, w]\} & \text{if } i > 0 \text{ and } w \geq w_i \end{cases}$$

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	6	6	6	6	6
2	0	6	10	16	16	16
3	0	6	10	16	18	22

$$\begin{aligned} & c[2,1] = c[1,1] \text{because } w_2 > w \\ & c[2,2] = \max(v_2 + c[1,0], c[1,2]) && = 10 \\ & c[2,3] = \max(v_2 + c[1,1], c[1,3]) = \max(10+6, 6) && = 16 \\ & c[2,4] = \max(v_2 + c[1,2], c[1,4]) = \max(10+6, 6) && = 16 \\ & c[3,3] = \max(v_3 + c[2,0], c[2,3]) = \max(12, 16) && = 16 \end{aligned}$$

Fractional Kanpsack Problem

Input n items where i-th item has value v_i and weighs w_i (v_i and w_i are positive integers)

Output The maximum value of items that can be carried in a knapsack of capacity w

Greedy Algorithm: at each iteration, choose the item with the highest $\frac{v_i}{w_i}$ and continue when $W - w_i > 0$

Subproblems

- $F - KP(i, w)$ fractional kanpsack problem within w capacity for the first i items
- Goal: $F - KP(n, W)$

Optimal Substructure

Suppose OPT is an optimal solution to $F - KP(i, w)$, there are 2 cases:

1. Full/partial item i in OPT
   Remove w' of item i from OPT is an optimal solution of $F - KP(i-1, w-w')$
2. Item i is not in OPT
   OPT is an optimal solution of $F - KP(i-1, w)$

Proof

[Let j be the item with the maximum v_i/w_i. Then there exists an optimal solution in which you take as much of item j as possible.]

Suppose there exists an optimal solution in which you didn't take as much of item j as possible.

1. If the knapsack is not full, add some more of item j, and you have a higher value solution

2. There must exist some item $k \neq j$ with $\frac{v_k}{w_k} < \frac{v_j}{w_j}$ that is in the knapsack.
3. We can therefore take a piece of k with ϵ weight, out of the knapsack, and put a piece of j with ϵ weight in and increase the knapsack value.

Graph Terminology

$$\begin{aligned}\text{Graph } G = (V, E)& \\ V =& \text{ set of verticies (or node)} \\ E =& \text{ set of edges (or links)} \subseteq (V \times V)\end{aligned}$$

$$\begin{aligned}V &= \{1, 2, 3, 4, 5\} \\ E &= \{(1, 2), (1, 3), (1, 4), (2, 4), (2, 5), (4, 5)\}\end{aligned}$$

Undirected vs Directed

Undirected edge $(u, v) = (v, u)$; $\quad \forall v, (v, v) \notin E$ (No self loops)

Directed edge (u, v) goes from vertex u to v

Unweighted vs Weighted
Weighted: Graph associates weights with edges

Graph Degrees

The degree of a vertex u is th enumber of edges incident to u

Event verticies verticies with even degrees

Odd verticies verticies with odd degrees

In a directed graph

In-degree of u the number of edges entering u

Out-degree of u the number of edges leaving u

Handshaking Lemma:
If $G = (V, E)$ is an undirected grpah, then

$$\sum_{v \in V} degree(v) = 2|E|$$

Every undirected graph has an even number of odd verticies.

Representation

Adjacency Matrix $A = (a_{ij})$ where $a_{ij} = 1$ if $(i, j) \in E$

Adjacency List $Adj[u]$ is the list of verticies adjacent to u