The knapsack problem

- Given
  - n objects numbered from 1 to n. Object i has a positive weight w_i and a positive value v_i
  - a knapsack that can carry a weight not exceeding W
- Problem
  - Fill the knapsack in a way that maximize the value of the included objects, while respecting the capacity constraints
  - In this version, we assume that the objects can be broken into small pieces

Formal description

- Given W; w_i, v_i
- Find an array x_i, $1<=i<=n$, $0<=x_i<=1$, to
  - Maximize $\sum_{i=1}^{n} x_i v_i$
  - And be subject to $\sum_{i=1}^{n} x_i w_i \leq W$

A greedy algorithm

```
Knapsack(w[], v[], W)
{
   for (i=1; i<=n; i++)
     x[i] =0;
   weight = 0;

    while (weight < W) {
        i = select the best remaining object;
        if (weight + w[i] < W)
          x[i] = 1;
        else
          x[i] = (W-weight)/w[i];
    }
    return x;
}
```

The key is which object to select

Selection methods

1. Choose the most valuable remaining object
2. Choose the lightest remaining object
3. Choose the object with the highest value per weight unit.

n=5, W=100

w	10	20	30	40	50	
v	20	30	66	40	60	
v/w	2.0	1.5	2.2	1.0	1.2	
Method 1			1	3	2	146
Method 2	1	2	3	4		156
Method 3	2	3	1		4	164

Optimality of method 3

- If the objects are selected in order of decreasing v_i / w_i then the algorithm knapsack finds an optimal solution
  - The algorithm generates a solution either including all objects or fill the knapsack full