Design and Analysis of Algorithms

02-03
Greedy Algorithms

Fractional Knapsack

Imran Ihsan
Assistant Professor, Department of Computer Science
Air University, Islamabad, Pakistan
www.imranihsan.com

Long Hike

Long Hike

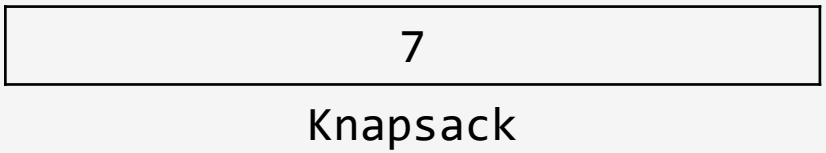
Long Hike

Fractional Knapsack

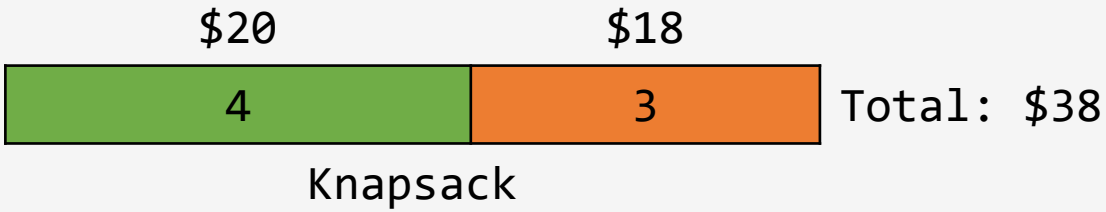
Input: Weights $w_1, \ldots, w_n$ and values $v_1, \ldots, v_n$ of n items; capacity W .

Output: The maximum total value of fractions of items that fit into a bag of capacity W .

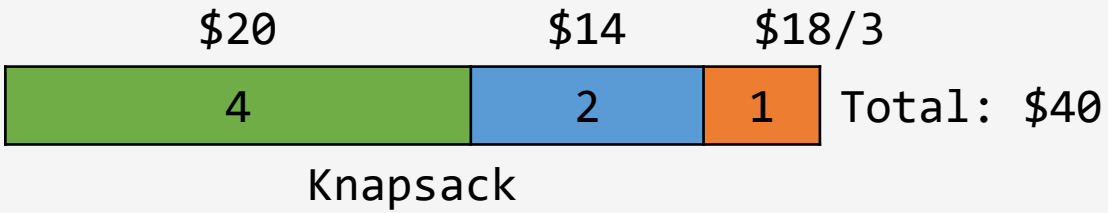
Example



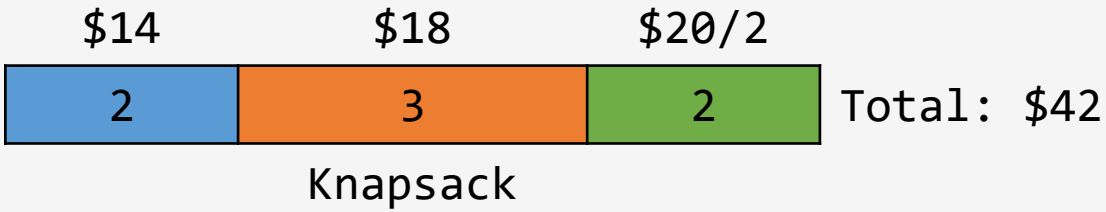
Example



Example



Example



Quiz

You are given a knapsack of capacity 7 kg and 3 items. First item has value $20 and weight 4 kg, second item has value $18 and weight 3 kg, third item has value $14 and weight 2 kg. What is the maximum total value of the fractions of items that fit into the knapsack in this case?

A. 38

B. 40

C. 42

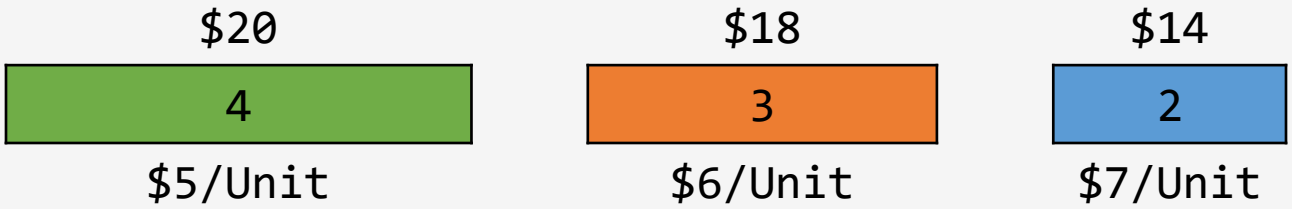
D. 43

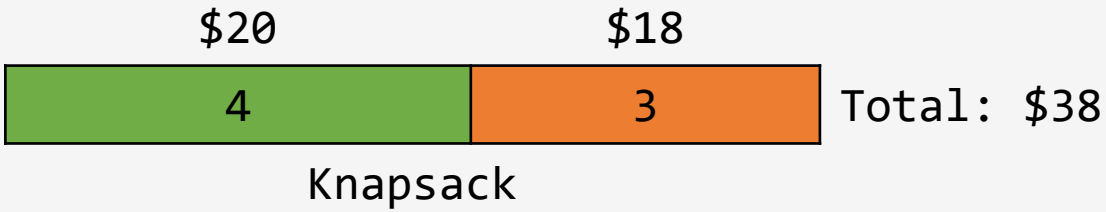
Safe Move

Lemma

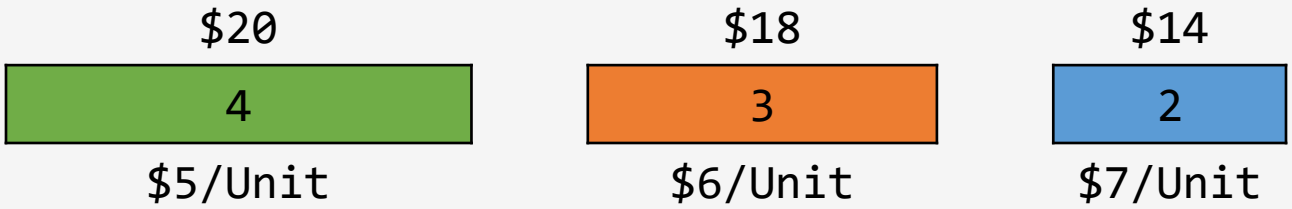
> There exists an optimal solution that uses as much as possible of an item with the maximal value per unit of weight

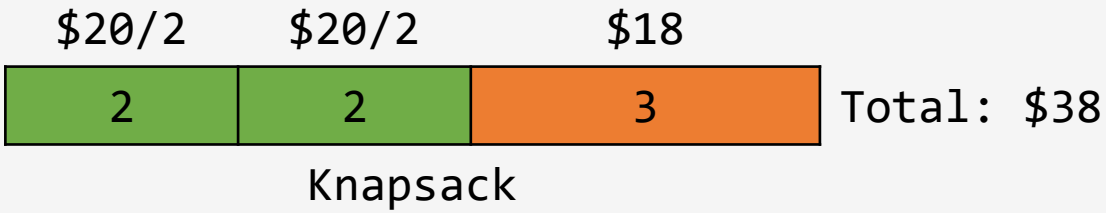
Example



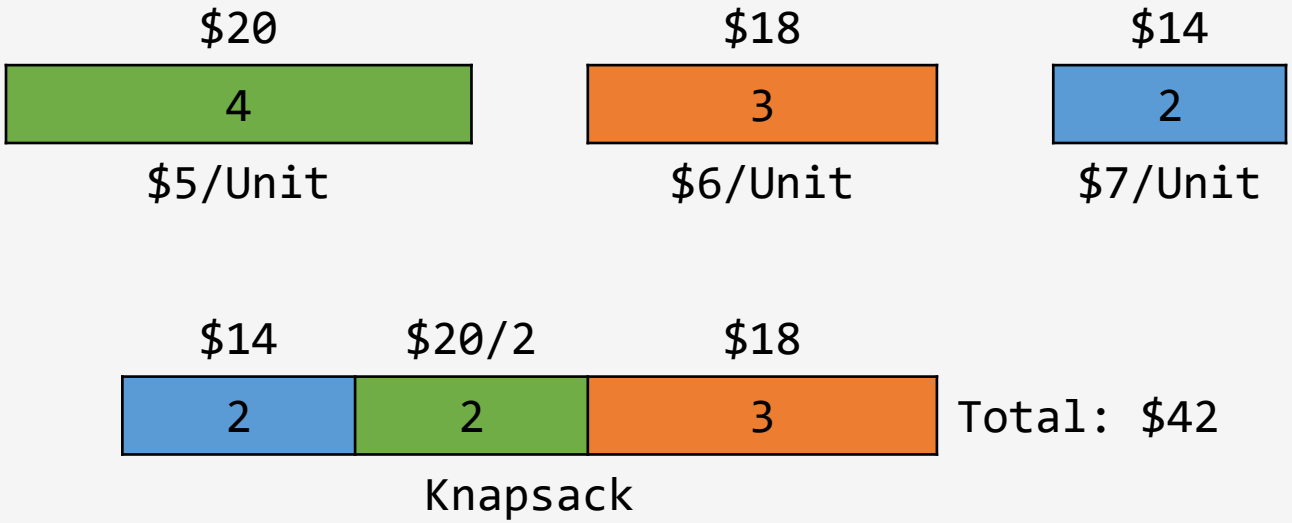


Example


$20
4
$5/Unit
$18
3
$6/Unit
$14
2
$7/Unit


$20/2
$20/2
$18
2
2
3
Total: $38
Knapsack

Example



Greedy Algorithm

While knapsack is not full

Choose item i with maximum v_i/w_i

If item fits into knapsack, take all of it

Otherwise take so much as to fill the knapsack

Return total value and amounts taken

Pseudocode

```
Knapsack(W , w1, v1, . . . , wn, vn)
    A ← [0, 0, . . . , 0], V ← 0
    repeat n times:
        if W = 0:
            return (V , A)
        select i with wi > 0 and max(vi/wi)
        a ← min(wi, W )
        V ← V + a vi/wi
        wi ← wi –a, A[i] ← A[i]+a, W ← W –a
    return (V , A)
```

Running Time

Lemma

The running time of Knapsack is $O(n^2)$.

Proof

Select best item on each step is $O(n)$

Main loop is executed n times

Overall, $O(n^2)$

Optimization

It is possible to improve asymptotic!

First, sort items by decreasing `v/w`

Assume $v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_n/w_n$

```
Knapsack(W , w1, v1, . . . , wn, vn)
      A ← [0, 0, . . . , 0], V ← 0
      for i from 1 to n:
        if W = 0:
            return (V , A)
        a ← min(wi, W )
        V ← V + a vi/wi
        wi ← wi –a, A[i] ← A[i]+a, W ← W –a
      return (V , A)
```

Asymptotic

Now each iteration is O(1)

Knapsack after sorting is O(n)

Sort + Knapsack is O(n log n)