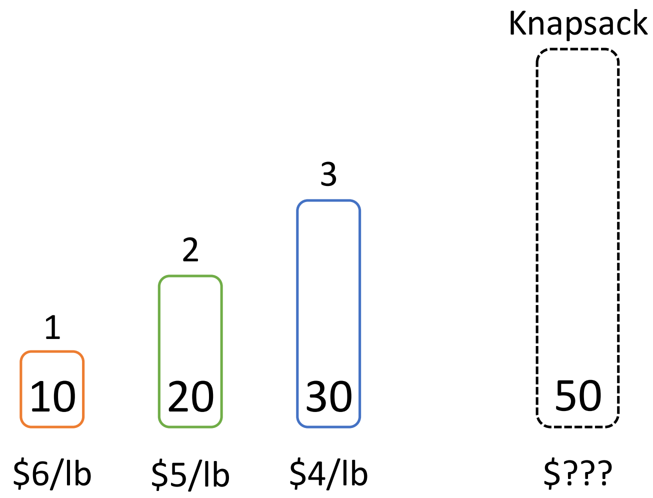
CO202 – Software Engineering – Algorithms

Greedy Algorithms - Solutions

Exercise 1: Implement Fractional Knapsack

```
FRACTIONAL-KNAPSACK(v,w,K)
```

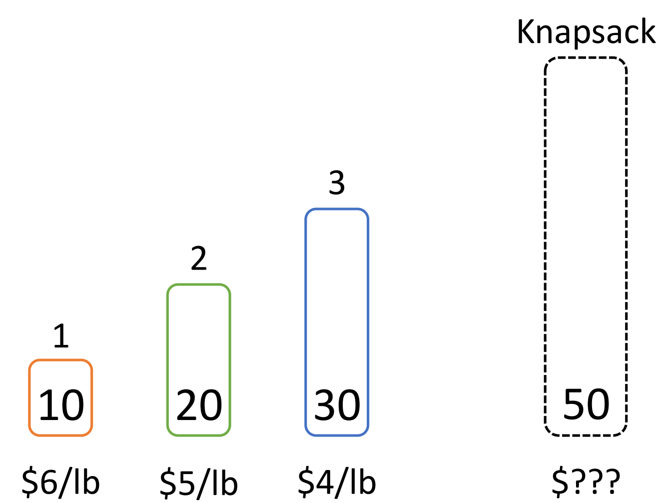
i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4



Exercise 1: Implement Fractional Knapsack

```
FRACTIONAL-KNAPSACK(v,w,K)
 n = v.length
 load = 0
 value = 0
 i = 1
 while load < K and i ≤ n
       f = min((K-load)/w[i],1)
       load = load + f * w[i]
       value = value + f * v[i]
       i = i+1
return load and value
```

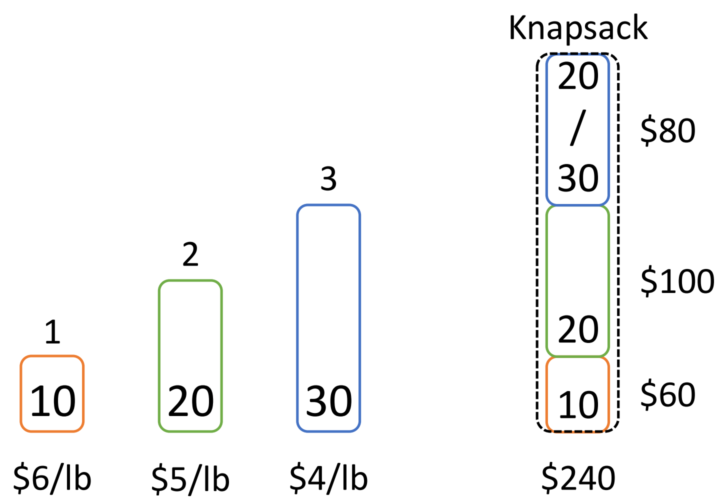
i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4



Exercise 1: Implement Fractional Knapsack

```
FRACTIONAL-KNAPSACK(v,w,K)
    n = v.length
    load = 0
    value = 0
    i = 1
    while load < K and i ≤ n
        f = min((K-load)/w[i],1)
        load = load + f * w[i]
        value = value + f * v[i]
        i = i+1
    return load and value
```

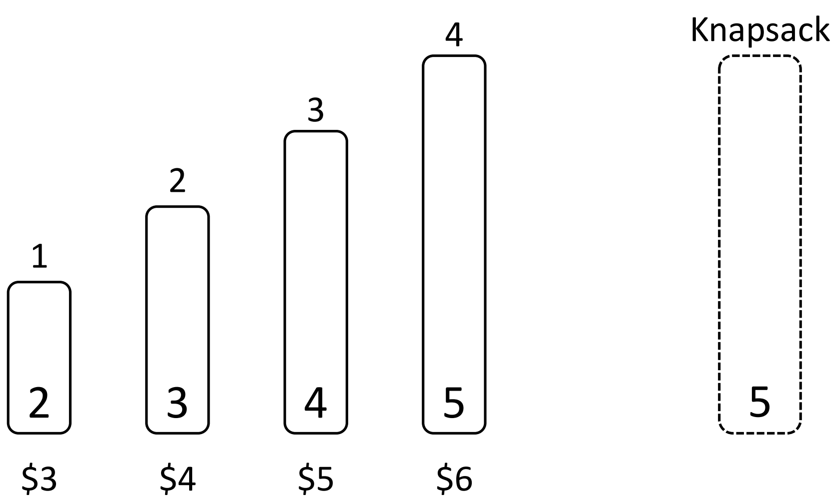
i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i/w_i	6	5	4



Time to sort $O(n \lg n)$, and then $O(n)$

Exercise 2: Solving 0-1 Knapsack

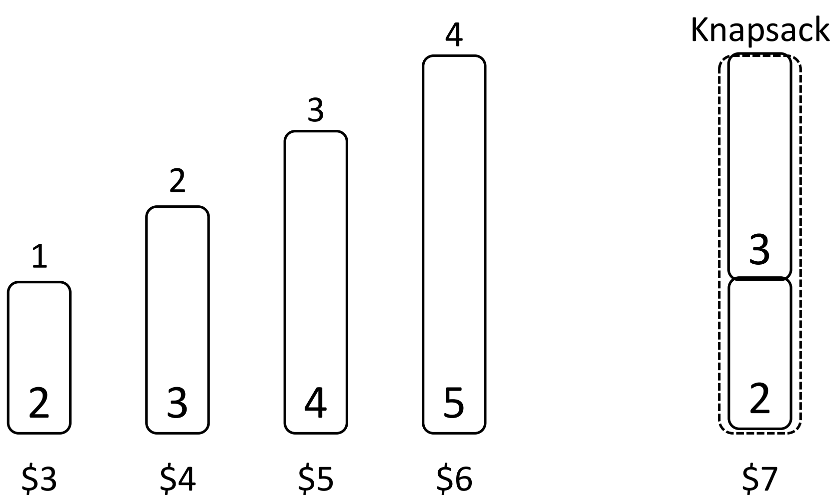
```
for i = 1 to n
    c[i,0] = 0
    for j = 1 to K
        if w[i] ≤ j
            c[i,j] = max(v[i] + c[i-1,j-w[i]], c[i-1,j])
        else
            c[i,j] = c[i-1,j]
```



i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1						
2						
3						
4						

Exercise 2: Solving 0-1 Knapsack

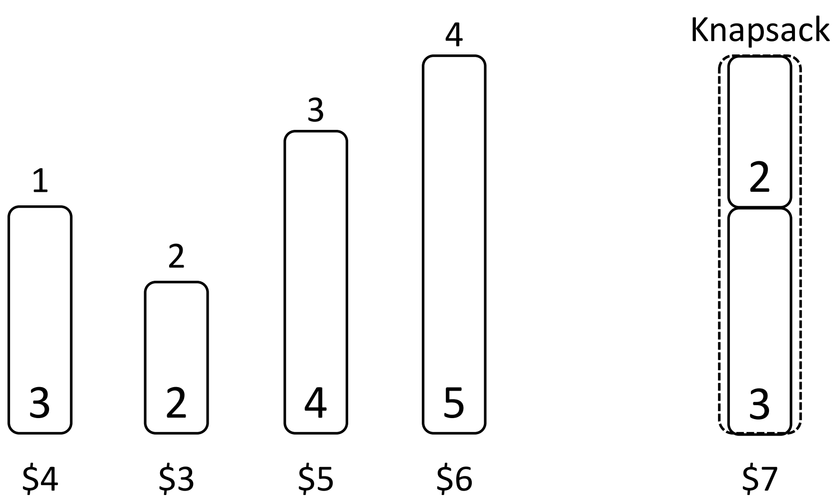
```
for i = 1 to n
    c[i,0] = 0
    for j = 1 to K
        if w[i] ≤ j
            c[i,j] = max(v[i] + c[i-1,j-w[i]], c[i-1,j])
        else
            c[i,j] = c[i-1,j]
```



i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Exercise 2: Solving 0-1 Knapsack

```
for i = 1 to n
    c[i,0] = 0
    for j = 1 to K
        if w[i] ≤ j
            c[i,j] = max(v[i] + c[i-1,j-w[i]], c[i-1,j])
        else
            c[i,j] = c[i-1,j]
```



i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	4	4	4
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

Exercise 3: Coin Change Problem

Prove that a greedy strategy of picking the highest valued coin which is less or equal than the remaining amount is not guaranteed to produce optimal results.

Exercise 3: Coin Change Problem

Prove that a greedy strategy of picking the highest valued coin which is less or equal than the remaining amount is not guaranteed to produce optimal results.

Counter example: Assuming the amount of money to be 6 and the coin set 1,3,4. The greedy strategy would yield the coin sequence 4,1,1 while the optimal solution with least number of coins is 3,3.

Exercise 4: Planning a Party

Invite as many people as possible from a set of n people, such that

1. Every person invited should know at least five other people that are invited
2. Every person invited should not know at least five other people that are invited

Hint: Maximizing the number of invitees is the same as minimizing the number of people that are not invited.

Exercise 4: Planning a Party

Invite as many people as possible from a set of n people, such that

1. Every person invited should know at least five other people that are invited
2. Every person invited should not know at least five other people that are invited

- Label the people that could be invited $1, \ldots n$
- For a subset S of $\{1, 2, \ldots, n\}$, k_i is the number of people in S that the i-th person knows, and d_i is the number of people in S the i-th person does not know.
- Algorithm:
  - Let $P = \{1, 2, \ldots, n\}$ be the set of potential invitees
  - While there is $i \in P$ such that $k_i < 5$ or $d_i < 5$: $P = P - \{i\}$
  - Return P