Design and Analysis of Algorithms

L18: Knapsack Problem

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

- Text book 2: Sec: 4.3
- R1: Introduction to Algorithms
    - Cormen et al.

Example: Knapsack Problem

- A flower street vendor procures the flowers from KR Market and sells these during the day. The quantity of flowers available are limited as below.

Lilies: 8Kg
Profit: ₹240

Roses: 10Kg
Profit: ₹250

Jasmine: 6Kg
Profit: ₹120

Daisies: 6Kg
Profit: ₹210

- The vendor has a carrying bag with a capacity of 20kg,

Example: Knapsack Problem

- A flower street vendor procures the flowers from KR Market and sells these during the day. The quantity of flowers available are limited along with respective profits are as below.
  - Roses: 10kg with a total profit of ₹ 250
  - Lilies: 8kg with a total profit of ₹ 240
  - Daisies: 6kg with a total profit of ₹ 210.
  - Jasmine: 6Kg with a total profit of ₹ 120
- The vendor has a carrying bag with a capacity of 20kg, would like to maximize the profit for the day. The vendor can buy any quantity (from 0kg to its max limit as given above) for any flower.
- Q: Which quantity of each flower vendor should buy?

Flower Buying: Approach 1

- *Flowers: quantity/total profit*
  - *Roses 10Kg /Rs 250, Lilies: 8Kg / Rs 240*
  - *Daisies 6Kg / Rs 210, Jasmine: 6Kg / Rs 120*
- Equal quantity of each flower:
  - Buy same quantity of each variety of flower i.e. buy 20/4=5 kg of Rose, Daisies and Lilies and Jasmine
- The profit earned for the day is
  - Roses: 5*250/10 = ₹ 125
  - Lilies: 5*240/8 = ₹ 150
  - Daisies: 5*210/6 = ₹ 175
  - Jasmine: 5*120/6= ₹ 100
- Net profit: ₹ 125+150+175+100 = ₹ 550

Flower Buying: Approach 2

- *Flowers: quantity/total profit*
  - *Roses 10Kg /Rs 250, Lilies: 8Kg / Rs 240*
  - *Daisies 6Kg / Rs 210, Jasmine: 6Kg / Rs 120*
- Buy in equal proportions of their availability
  - Roses: 20*10/30 = 20/3Kg, Lilies: 20*8/30=16/3 Kg
  - Daisies: 20*6/30 = 4Kg, Jasmine 20*6/30 = 4Kgs
- The profit earned for the day is
  - Roses: (20/3)*250/10 = ₹ 500/3=Rs 166.6
  - Lilies: (16/3)*240/8 = ₹ 160
  - Daisies: 4*210/6 = ₹ 140
  - Jasmine: 4*120/6= ₹ 80
- Net profit: ₹ 166.67+160+140+80 = ₹ 546.67

Flower Buying: Approach 3

- *Flowers: quantity/total profit*
  - *Roses 10Kg /Rs 250, Lilies: 8Kg / Rs 240*
  - *Daisies 6Kg / Rs 210, Jasmine: 6Kg / Rs 120*
- Buy as per max profit known (greedy approach 1)
  - Roses: 10Kg, Lilies: 8Kg, Daisies: 2Kg, Jasmine: 0Kg
- The profit earned for the day is
  - Roses: 10*250/10 = ₹ 250
  - Lilies: 8*240/8 = ₹ 240
  - Daisies: 2*210/6 = ₹ 70
  - Jasmine: 0*120/6= ₹ 0
- Net profit: ₹ 250+240+70+0 = ₹ 560

Flower Buying: Approach 4

- *Flowers: quantity/total profit*
  - *Roses 10Kg /Rs 250, Lilies: 8Kg / Rs 240*
  - *Daisies 6Kg / Rs 210, Jasmine: 6Kg / Rs 120*
- Buy as per capacity from min (greedy approach 2)
  - Jasmine: 6Kg, Daisies: 6Kg, Lilies: 8Kg, Roses: 0Kg
- The profit earned for the day is
  - Jasmine: 6*120/6= ₹ 120
  - Daisies: 6*210/6 = ₹ 210
  - Lilies: 8*240/8 = ₹ 240
  - Roses: 0*250/10 = ₹ 0
- Net profit: ₹ 0+240+210+120 = ₹ 570

Flower Buying: Approach 5

- *Flowers: quantity/total profit*
  - *Roses 10Kg /Rs 250, Lilies: 8Kg / Rs 240*
  - *Daisies 6Kg / Rs 210, Jasmine: 6Kg / Rs 120*
- Greedy approach 3: get max profit per kg of flowers
  - Profits per Kg: R: Rs 25, L: Rs 30, D: 35, J: 20
  - Daisies: 6Kg, Lilies: 8Kg, Roses: 6Kg, Jasmine: 0Kg
- The profit earned for the day is
  - Roses: 6*250/10 = ₹ 150
  - Lilies: 8*240/8 = ₹ 240
  - Daisies: 6*210/6 = ₹ 210
  - Jasmine: 0*120/6= ₹ 0
- Net profit: ₹ 150+240+210+0 = ₹ 600

Flower Buying

- Profit comparisons:
  - Approach 1 (equal quantity): Rs 550/-
  - Approach 2 (in equal ratios): Rs 546.67
  - Approach 3 (Greedy: Max highest profit): Rs 560/-
  - Approach 4 (Greedy: Smallest capacities): Rs 570/-
  - Approach 5 (Greedy: max profit per kg): Rs 600/-
- Does the Greedy approach always works?
  - Yes (for fractional knapsack)
  - No (for 0-1 knapsack)
    - 0-1 knapsack: can not buy partial quantities
- Can there be multiple optimal solutions?
  - Consider that both Roses, Lilies have profit of Rs 25/Kg

Example 2: Suitcase Packing

- You are travelling by air and airline has limit of 15Kg on the check in bag.
- You have many number of items to carry with you.
- How do you decide which items to pack and which ones to leave behind.

Overview: Knapsack Problem

- Knapsack problem (fractional):
  - Given $\underline{n}$ objects, and a knapsack (bag) with a capacity $\underline{m}$, fill the knapsack to maximize the value as follows
    - Each object i has weight w_i (+ve number)
    - Each object i has +ve profit p_i (+ve number)
    - If a fraction x_i ($0 \le x_i \le 1$) of the object i is placed in the knapsack, the profit $p_i x_i$ is earned.
  - **Objective**: Obtain a filling of the knapsack that maximizes the total profit earned. Mathematically

$$\text{Maximize} \quad \sum_{1 \le i \le n} p_i x_i$$

$$\text{Subject to} \quad \sum_{1 \le i \le n} w_i x_i \le m \qquad \text{and } 0 \le x_i \le 1, \; 1 \le i \le n$$

Knapsack Problem

- Example (Book II): consider $n=3, m=20$, $(p_1,p_2,p_3)=(25,24,15)$, $(w_1,w_2,w_3)=(18,15,10)$
- Feasible solutions are

S#	Greedy Approach	x_1, x_2, x_3	$\Sigma w_i x_i$	$\Sigma p_i x_i$
1	Random	1/2, 1/3, 1/4	16.5	24.25
2	Highest profit	1, 2/15, 0	20	28.2
3	Lowest weight	0, 2/3, 1	20	31
4	Highest p_i/w_i	0, 1, 1/2	20	31.5

Knapsack Problem: Approach

- A problem that fits the subset paradigm
- Select x_i for each object i.
  - w_i's (18,15,10), p_i's=(25,24,15)
- Simple strategies: solutions with weight sum m.
- A1: Consider objects with highest profit
  - when an object doesn't fit, take its fraction
  - All objects are in full except possibly the last

  x_1=1, x_2=2/15, x_3=0; p_1=25, p_2=24*2/5, P=28.5
- A2: Consider objects with lowest weights
  - Fill knapsack as slowly as possible

  x_3=1, x_2=2/3, x_1=0; p_2=24*2/3, p_3=15; P=31
- A3: non-increasing p_i/w_i i.e. p_2/w_2, p_3/w_3, p_1/w_1

  x_2=1, x_3=1/2, x_1=0; p_2=24, p_3=15*1/2; P=31.5

Knapsack problem

- Greedy approach: Optimal solution
- Pick the object with highest profit per unit of weight
- Keep picking the objects in this order till an object can't be taken in full
  - Take a fraction of this object to fill the capacity
- Technique for proving optimality
  - Compare greedy approach with any optimal solution
  - Find first x_i at which two solutions differ
  - Make the x_i in optimal solution equal to that in greedy solution without any loss in total value
  - Repeated use of this transformation shows Greedy solutions is optimal as well

Knapsack Problem

- Lemma 1:
  - – In case the sum of all quantities is $\leq m$, then
    $x_i = 1, \; 1 \leq i \leq n$ is an optimal solution.
  - – So, let us consider that sum of weights exceed m.
    - All weight can't be included in full.
    - Some weights may not be included at alll
- Lemma 2:
  - All optimal solutions will fill the knapsack exactly.
  - Note:
    - Increase the quantity of some object i by a fractional amount till the total weight becomes exactly m
    - Decrease amount of other objects accordingly
- Analysis: Does it fit the subset paradigm?
  - Yes: we are selecting a subset of objects.

Algorithm: Knapsack Problem

```
Void GreedyKnapsack(float m, int n) {
//p[1:n] and w[1:n] contain the profits and weights
//The indices are ordered as per following criteria
// p[i]/w[i]≥p[i+1]/w[i+1] , 1≤i<n.
// m is knapsack size, and x[1:n] is the solution vector
    initialize x[i] to 0.0
    float U=m
    for i=1 to n
        if w[i] >U
            break
        x[i]=1.0
        U=U-w[xi]
    if i≤n
        x[i] = U/w[i]
```

Theorem: Knapsack Problem

Theorem:

If $p_1/w_1 \geq p_2/w_2 \geq ... \geq p_n/w_n$, then `GreedyKnapsack` generates an optimal solution to the given instance of the knapsack problem.

Metholodology to be used for proof:

- Compare the greedy solution with any optimal solution.
- If the two solutions differ, then first x_i at which they differ.
- Then show that x_i in the optimal solution equal to that in the greedy solution without any loss in total value.
- Repeated use of this transformation shows that greedy solution is optimal

Proof: Greedy Approach is Optimal

- Let $x=(x_1, \ldots, x_n)$ be the solution generated by GreedyKnapsack., i.e. $\Sigma w_i x_i=m$
- If all the x_i equal one, the solution is optimal.
- Let j be the least index such that $x_j \neq 1$.
- From the algorithm, we know that
  $0 \leq x_j < 1$, and $x_i=1$ for $1 \leq i < j$, and $x_i=0$ for $j < i \leq n$.
- Let $y=(y_1, \ldots, y_n)$ be the optimal solution, Thus
  $\Sigma w_i y_i=m$
- Let k be the smallest index such that $y_k \neq x_k$
- Since two solutions differ, such k must exist. Since all x_k prior to x_j's are 1, clearly $y_k < x_k$, otherwise $\Sigma w_i y_i > m$
- Further, $k \leq j$, otherwise $\Sigma w_i y_i > m$ since $\Sigma w_i x_i=m$
  - Proof ahead

Proof: Greedy Approach is Optimal…

x_1	x_2	…	x_{j-1}	$\mathbf{x_j}$	x_{j+1}	…				x_n
1	1	1	1	$\mathbf{x_j}$	0	0	0			0

Solution by Greedy Approach

First index where x_j is not 0

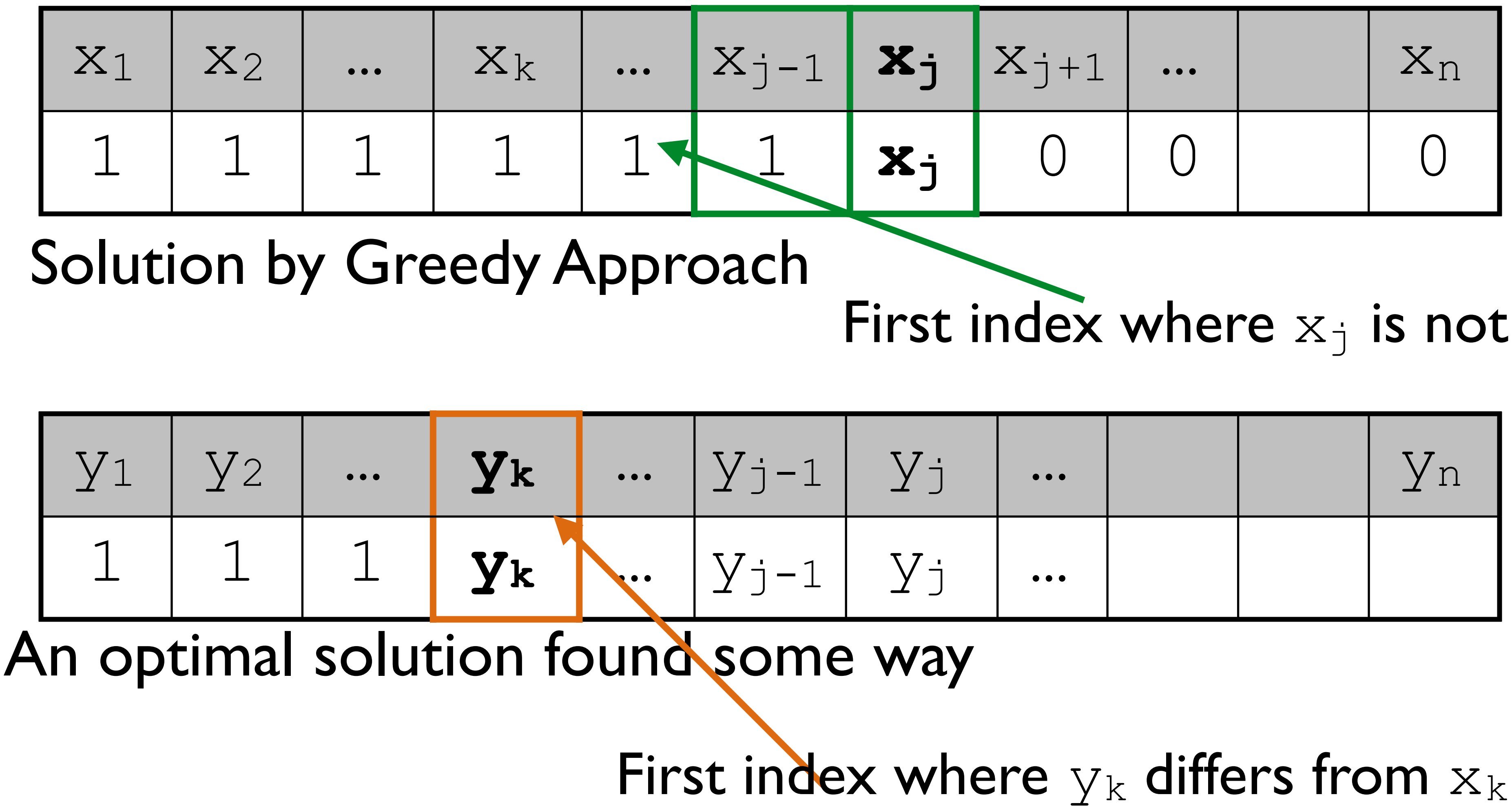
y_1	y_2	…	$\mathbf{y_k}$	…	y_j	…				x_n
1	1	1	$\mathbf{y_k}$	…	y_j	y_{j+1}	…			y_n

An optimal solution found some way

First index where y_k differs from x_k

Proof: Greedy Approach is Optimal…

case `1:` $k<j$, $x_k=1$, hence $y_k<x_k$



x_1	x_2	…	x_k	…	x_{j-1}	$\mathbf{x_j}$	x_{j+1}	…		x_n
1	1	1	1	1	1	$\mathbf{x_j}$	0	0		0

Solution by Greedy Approach

First index where x_j is not 0

y_1	y_2	…	$\mathbf{y_k}$	…	y_{j-1}	y_j	…			y_n
1	1	1	$\mathbf{y_k}$	…	y_{j-1}	y_j	…			

An optimal solution found some way

First index where y_k differs from x_k

Proof: Greedy Approach is Optimal…

case 2 : $k=j$

if $y_k \nleq x_k$, then $\Sigma w_i y_i > m$, because $\Sigma w_i x_i = m$

x_1	x_2	…	…	…	x_{j-1}	$\mathbf{x_j}$	x_{j+1}	…		x_n
1	1	1	1	1	1	$\mathbf{x_j}$	0	0		0

Solution by Greedy Approach

First index where x_j is not

y_1	y_2	…	…	…	…	$\mathbf{y_k}$	…			y_n
1	1	1	…	…	1	$\mathbf{y_k}$	…			

An optimal solution found some way

First index where y_k differs from x_k

Proof: Greedy Approach is Optimal…

case 3 : $k>j$, This is not possible since $\Sigma w_i y_i > m$

x_1	x_2	…	…	…	x_{j-1}	$\mathbf{x_j}$	x_{j+1}	…		x_n
1	1	1	1	1	1	$\mathbf{x_j}$	0	0		0

Solution by Greedy Approach

First index where x_j is not 0

y_1	y_2	…	…	…	…	…	…	$\mathbf{y_k}$		y_n
1	1	1	…	…	1	1	…	$\mathbf{y_k}$		

An optimal solution found some way

First index where y_k differs from x_k

Proof: Greedy Approach is Optimal…

- Summary of proof: $y_k < x_k$, there exists 3 possibilities

  i. $k<j$: since $x_k=1$, and $y_k \neq x_k$, and so $y_k<x_k$

  ii. $k=j$: since $\Sigma w_i x_i=m$, and $y_i=x_i$ for $1 \leq i<j$, then either $y_k< x_k$ or $\Sigma w_i y_i>m$

  iii. $k>j$: then $\Sigma w_i y_i>m$, which is not possible

- To show that $x=(x_1, \ldots, x_n)$ is optimal solution.
  - Increase y_k to x_k, and
  - Decrease as many of (y_{k+1}, …, y_n) as necessary so that total capacity is still m.
  - This gives a new solution $z=(z_1, \ldots, z_n)$ such that

    $z_i=x_i, 1 \leq i \leq k$; (note $z_k=x_k$), and

    $\Sigma_{k<i \leq n} w_i(y_i-z_i) = w_k(z_k-y_k)$… … … … (1)

Proof: Greedy Approach is Optimal…

- Thus, we have (increased y_k and decreased remaining y_i's)

$$\sum_{1 \le i \le n} p_i z_i = \sum_{1 \le i \le n} (p_i y_i) + (z_k - y_k) w_k \frac{p_k}{w_k} - \sum_{k < i \le n} (y_i - z_i) w_i \frac{p_i}{w_i}$$

$$\ge \sum_{1 \le i \le n} (p_i y_i) + \left[(z_k - y_k) w_k - \sum_{k < i \le n} (y_i - z_i) w_i \right] \frac{p_k}{w_k}$$

since $p_k/w_k \ge p_{k+1}/w_{k+1} \ge \ldots \ge p_n/w_n$

$$= \sum_{1 \le i \le n} (p_i y_i)$$ since $\Sigma_{k<i\le n}\, w_i(y_i - z_i) = w_k(z_k - y_k)$

- Thus, if $\Sigma p_i z_i > \Sigma p_i y_i$, then y could not have been optimal solution.
- If $\Sigma p_i z_i = \Sigma p_i y_i$, then either $z=x$ and x is optimal, or $z \ne x$.
- If $z \ne x$, then repeat the process to show that y is not optimal or transform y to x and hence x is optimal.

Summary

- Greedy approach (fractional) knapsack
- Pick the object with highest profit per unit of weight
- Keep picking the objects in this order till an object can’t be taken in full
  - Take a fraction of this object to fill the capacity
- The greedy approach for fractional Knapsack gives optimal solution.