Algorithms Design and Analysis [ETCS-301]

Dr. A K Yadav
Amity School of Engineering and Technology
(affiliated to GGSIPU, Delhi)
akyadav1@amity.edu
akyadav@akyadav.in
www.akyadav.in
+91 9911375598

September 17, 2019

0-1 Knapsack Problem I

- Set of n items from 1 to n are given
- w_i is the weight of the i^{th} item
- v_i is the value of the i^{th} item
- We can pick any item as a whole, fraction of weight is not allowed
- Total weight we pick can not be more than W but value of the sum of picked items must be maximum
- So 0-1 Knapsack Problem is an optimization problem stated as:

$$\text{Maximize } \sum_{i=1}^{n} v_i x_i$$

$$\text{under the constraints } \sum_{i=1}^{n} x_i w_i \leq W, \text{ and } x_i \in \{0, 1\}$$

0-1 Knapsack Problem II

- In bounded Knapsack Problem we can pick upto c_i copies of x_i item:

  Maximize $\sum_{i=1}^{n} v_i x_i$

  under the constraints $\sum_{i=1}^{n} x_i w_i \leq W$, and $x_i \in \{0 \dots c_i\}$

- In unbounded Knapsack Problem we can pick any number of x_i item including none:

  Maximize $\sum_{i=1}^{n} v_i x_i$

  under the constraints $\sum_{i=1}^{n} x_i w_i \leq W$, and $x_i \geq 0$

Compute the maximum Bag Value I

- Total number of items is n
- $v[i]$ is the value of the i^{th} item whose weight is $w[i]$.
- Let Table $m[0 \ldots n, 0 \ldots W]$ is used to store the value of the bag for n items and W maximum capacity.
- $m[i, j]$ is the value of the bag from i item and j capacity.

Compute the maximum Bag Value II

Bottom-up Approach:

OPTIMAL-0-1KP(w,v,n,W)

1. Let $m[0 \ldots n, 0 \ldots W]$
2. for $i = 0$ to n
3. for $j = 0$ to W
4. if $i = 0$ or $j = 0$ //no item or nil bag capacity
5. $m[i, j] = 0$
6. else if $w[i] \leq j$// i^{th} item can be accommodated in bag
7. if $(m[i-1, j-w[i]] + v[i]) > m[i-1, j]$
8. $m[i, j] = m[i-1, j-w[i]] + v[i]$//pick the item
9. else
10. $m[i, j] = m[i-1, j]$//don't pick the item
11. return m

Compute the maximum Bag Value III

Top-down Approach:

Mamoized-0-1KP(w,v,n,W)

1. Let $m[0 \ldots n, 0 \ldots W]$
2. for $i = 0$ to n
3. for $j = 0$ to W
4. $m[i, j] = 0$
5. return $knapsack(m, w, v, n, W)$

$knapsack(m, w, v, i, j)$

1. if $m[i, j] > 0$
2. return $m[i, j]$
3. if $i = 0$ or $j = 0$ //no item or nil bag capacity
4. return 0
5. if $w[i] < j$

Compute the maximum Bag Value IV

6. return $max(knapsack(m, w, v, i-1, j),$ $knapsack(m, w, v, i-1, j-w[i]) + v[i])$
7. else
8. return $knapsack(m, w, v, i-1, j)$
9. return m

Complexity of the Knapsack is $O(nW)$

Thank you

Please send your feedback or any queries to akyadav1@amity.edu, akyadav@akyadav.in or contact me on +91 9911375598