# Knapsack Problem

The input to the problem are

(a) $n$ items of costs $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$ respectively

(b) a weight $W$

For a subset $S \subseteq \{1, \dots, n\}$ we denote by $w(S) = \sum_{j \in S} w_j$ and $v(S) = \sum_{j \in S} v_j$

The goal of the Knapsack Problem is

GOAL: Identify a set $S \subseteq \{1, \dots, n\}$ such that $w(S) \leq W$ and

$$v(S) = \max \{ v(S') \mid w(S') \leq W \}$$

i.e., $v(S)$ is maximized.

We know that:

**Theorem:** The Knapsack problem is NP-hard.

Therefore a polynomial time algorithm may not be possible. But we have a pseudo-polynomial algorithm. We saw the following.

**Claim 0:** There is an algorithm, KS-algo, which given an instance of the Knapsack problem returns an optimal solution in time $O(n^2 v^*)$ where $n$ is the no. of items and $v^*$ is the maximum cost among all items.

We will denote by $S^*$ the optimal solution and $OPT = v(S^*)$.

Note that KS-algo runs in time $O(n^2 v^*)$, this is not polynomial in input size because the input size of $v^*$ is only $(\log v^*)$.

Let us note down some easy properties satisfied by OPT.

Claim 1) Let $S \subseteq \{1, \ldots, n\}$ be an arbitrary set. If $w(S) \leq W$ then $v(S) \leq OPT$

The claim follows from the fact that OPT is the maximum value possible.

The following assumption can be made on the input items of a knapsack instance.

Assumption: $\forall i \leq n$, $w_i \leq W$

We can throw away all items which does not satisfy above assumption. Why? Because they are never going to be part of the output. The above assumption gives us the following.

Claim 2) $\forall i \leq n$ $v_i \leq OPT$

Proof: Assume not. Then $v_i > OPT$. A contradiction by Claim 1.

Since Knapsack problem is NP-complete, we hope to get atleast a "good" approximation. We will give an approximation algorithm which outputs a set $S$ which satisfy the following condition

$$v(S) \geq (1-\epsilon) OPT$$

Def (polynomial time approx scheme (PTAS)): For every fixed $\epsilon > 0$, we have a polynomial time algorithm which gives the above approximation guarantee.

Def (fully poly time approx scheme (FPTAS)): For every $\epsilon > 0$, we have an algorithm which runs in time polynomial in the input size and $(1/\epsilon)$ along with giving the above approximation guarantee.

Note: The Knapsack Problem is a maximization problem. In a symmetric minimization problem we expect approximation guarantees as follows.

$$v(S) \leq (1+\epsilon)\,OPT$$

PTAS and FPTAS are also defined for these minimization problem.

## FPTAS for Knapsack

The fundamental idea of the approximation algorithm is to use the most significant bits of the input costs. That is, if $v_1, \ldots, v_n$ are the input costs, then $\left\lceil \frac{v_1}{b} \right\rceil, \ldots, \left\lceil \frac{v_n}{b} \right\rceil$ for a number $b$ is a good approximation of the costs. It uses less number of bits, but only the least significant bit information has been lost.

The following claim will be very useful

**Claim 3:** $\forall v, b > 1$

$$v \leq \left\lceil \frac{v}{b} \right\rceil b \leq v + b$$

We now give the approximate Knapsack algo.

Approx KS $(v_1, \ldots, v_n, w_1, \ldots, w_n, W, \epsilon)$

{

1) Let $b = \frac{\epsilon v^*}{n}$, where $v^* = \max\{v_1, \ldots, v_n\}$

2) Call KS-algo with input costs $\left\lceil \frac{v_1}{b} \right\rceil, \ldots, \left\lceil \frac{v_n}{b} \right\rceil$

   ie.. $S =$ KS-algo $\left( \left\lceil \frac{v_1}{b} \right\rceil, \ldots, \left\lceil \frac{v_n}{b} \right\rceil, w_1, \ldots, w_n, W \right)$

3) Return $S$

}

We now show that the above algorithm is an FPTAS and $v(S) \geq (1-\epsilon)\,OPT$.

## Running time of Approx-KS

Approx-KS calls KS-algo with $\lceil \frac{v_1}{b} \rceil, \ldots, \lceil \frac{v_n}{b} \rceil$.
This takes time $O\left(n^2 \left\lceil \frac{v^*}{b} \right\rceil\right)$. Substituting $b = \frac{\epsilon v^*}{n}$, we have a running time $O\left(\frac{n^3}{\epsilon}\right)$.

**Claim 4: Approx-KS runs in time $O\left(\frac{n^3}{\epsilon}\right)$**

## Approximation guarantee of Approx-KS

Let $S^*$ be the optimal set. Therefore

① $\sum_{j \in S} v_j \leq \sum_{j \in S^*} v_j$ (claim 1)

But why was $S$ returned by KS-algo in line 2 of our approximation algorithm?
- Because it is the optimal set when the costs are $\lceil \frac{v_1}{b} \rceil, \ldots, \lceil \frac{v_n}{b} \rceil$. In fact $S$ is better than $S^*$ for these costs.

Hence

$$\sum_{j \in S^*} \left\lceil \frac{v_j}{b} \right\rceil \leq \sum_{j \in S} \left\lceil \frac{v_j}{b} \right\rceil \quad \text{(Claim 1)}$$

Multiplying by $b$ on both sides we have

② $\sum_{j \in S^*} \left\lceil \frac{v_j}{b} \right\rceil b \leq \sum_{j \in S} \left\lceil \frac{v_j}{b} \right\rceil b$

From Claim 3, we have

③ $\sum_{j \in S} \left\lceil \frac{v_j}{b} \right\rceil b \leq \sum_{j \in S} v_j + |S| b \leq \sum_{j \in S} v_j + nb$

From ①, ② & ③ we have

$$OPT = \sum_{j \in S^*} v_j \leq \sum_{j \in S} v_j + nb$$

That is

**Claim 5: $OPT \leq v(S) + nb$**

From Claim 2 we know that $v^* \leq OPT$

Since $b = \frac{\epsilon v^*}{n}$, we have $b \leq \frac{\epsilon OPT}{n}$

We can rewrite Claim 5 as

$$OPT \leq v(S) + n\left(\frac{\epsilon OPT}{n}\right)$$

or $v(S) \geq (1-\epsilon) OPT$

This gives the approximation guarantee.