Knapsack Problem

- Hiker must decide which items to include in his/her trip
- He must choose between
  - ‘p’ - number of objects, where
  - ‘w_i’ – weight of i^{th} Object
  - ‘u_i’ – utility of i^{th} object

Objective is to : Maximize u_i
Subject to : Weight Limitation
Maximum W_i allowed should be given
Contd…

- We show knapsack problem as longest path problem on an acyclic network and then transform into a shortest path problem.
- We assign one layer corresponding to each item $i$ and one layer for source ‘$s$’ and sink ‘$t$’.
- The layer corresponding to an item $I$ has $w+1$ nodes $i^0, i^1, i^2, i^3, i^4 \ldots, i^w$.
  - where $i^k$ signifies that item $i$ has consumed ‘$k$’ units of the knapsack capacity.
The node $i^k$ has two outgoing arcs
1. Do not include $i+1$ in the knapsack
2. Include item $i+1$ in the knapsack
We choose second alternative only if
\[ k + w_{i+1} \leq W. \]
We assign arc corresponding to the first decision
($i^k$, $(i+k)^k$) with zero utility and,
The arc corresponding to the second decision
(provided that $k + w_{i+1} \leq W$)
\[ [ik, (i+1)^{k+w_{i+1}}] \] with utility $u_{i+1}$. 

Contd…
Contd…

- Every feasible solution of the knapsack problem defines a directed path from node ‘s’ to node ‘t’ where both feasible solution and path have same utility.
- In other words every path from node s to node t defines a feasible solution to the knapsack problem with the same utility.
- From eg: The path s-1^0^\text{-}2^2^\text{-}3^5^\text{-}4^5\text{-}t implies the solution in which we choose item 2 and 3 in the knapsack and exclude items 1 and 4.
Thus we can find the maximum utility selection of items by finding a maximum utility path, i.e. longest path in the network.

Longest path problem is then converted into shortest path problem by defining arc costs equal to the negative of the arc utilities.