Shortest Paths
using Greedy Method

Graphs: Shortest Paths

- Consider a graph G=<N,A> where N is the set of nodes and A is the set of directed edges. Each edge has nonnegative length. One of the node is designated as a source node.
- The problem is to determine the length of the shortest path from the source to each of the other nodes of the graph.
- This problem can be solved by a greedy algorithm often called "***Dijkstra's Algorithm***"

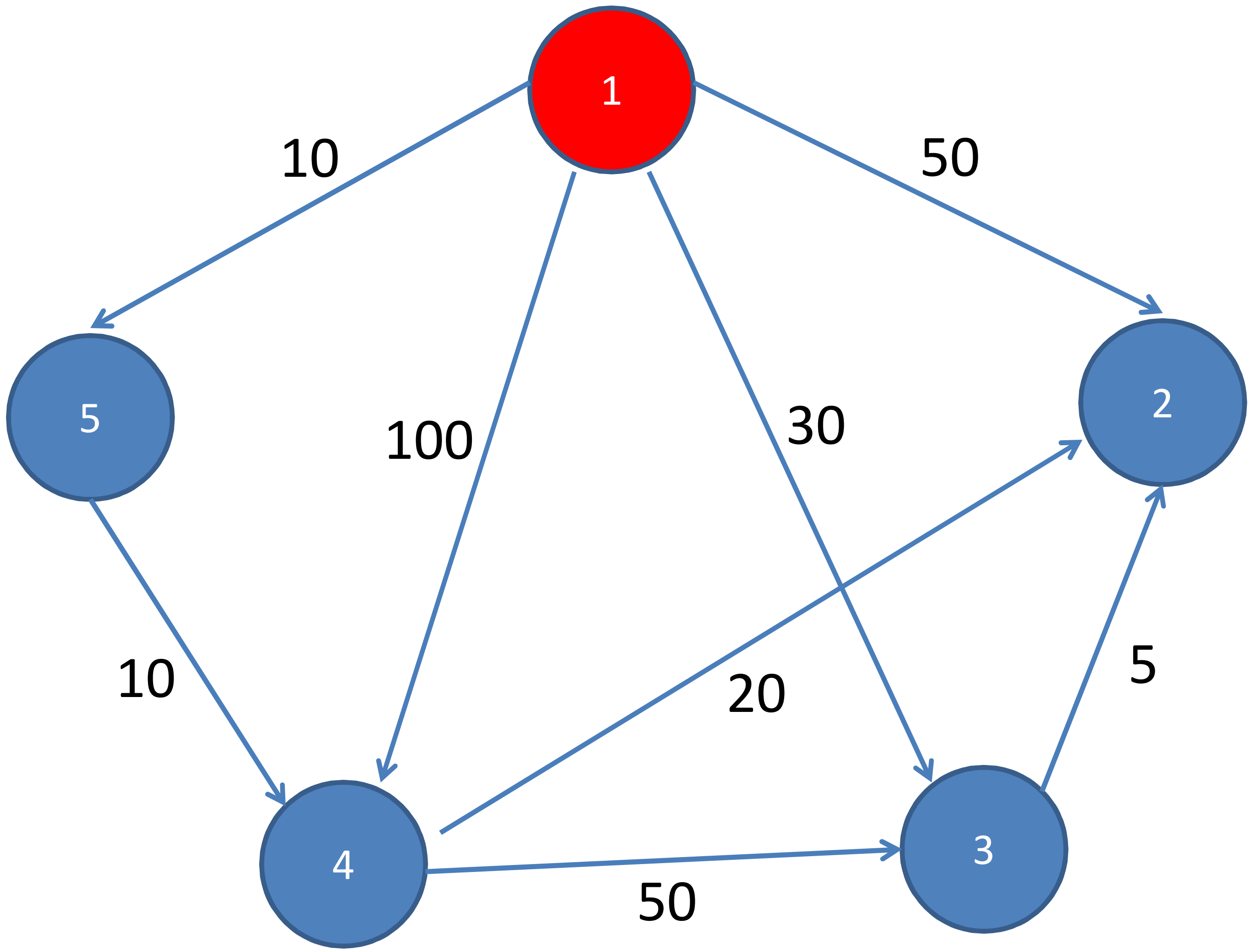
Dijkstra's Algorithm

- Assume that the nodes of G are numbered from 1 to n, so N={1,2,…,n}, where node 1 is source.

- Suppose that a matrix L gives the length of each directed edge;

  L[i,j]>=0, if the edge (i,j) belongs to A, and L[i,j]=infinity otherwise.

Dijkstra's Algorithm - Example



Dijkstra's Algorithm

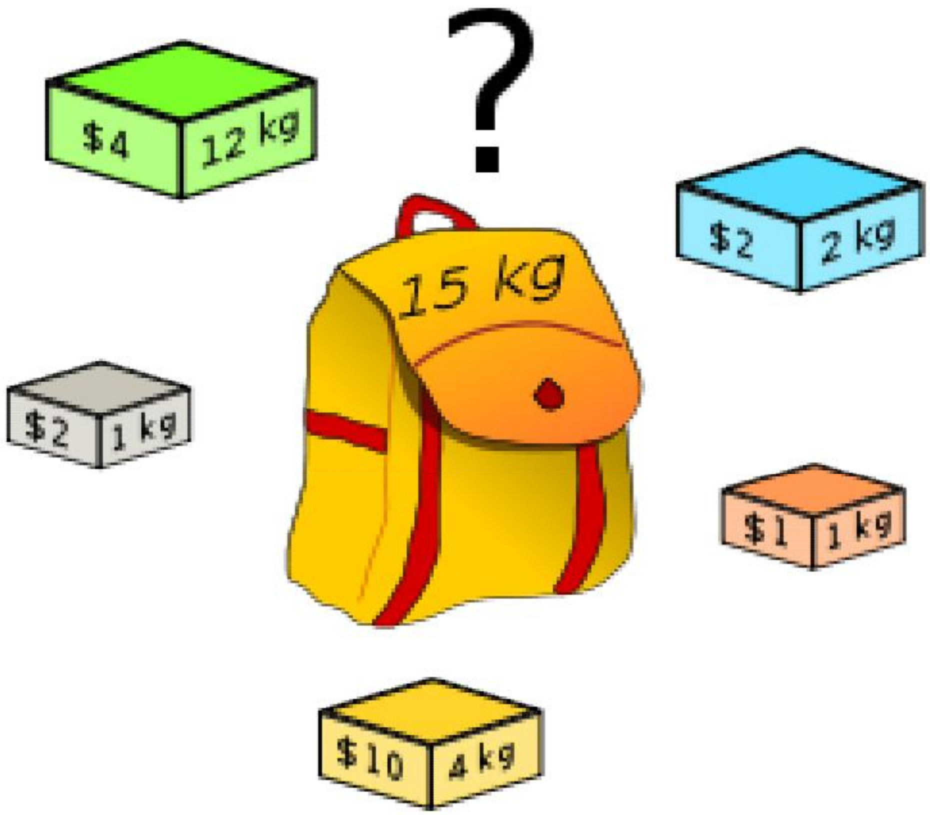
Step	V	C {2..n}	D[2..n]
Source Node : 1			2 3 4 5
Initialize	--	{2,3,4,5}	[50,30,100,**10**]
1	5 [10]	{2,3,4}	[50,30,**20**,10]
2	4 [20]	{2,3}	[40,**30**,20,10]
3	3 [30]	{2}	[**35**,30,20,10]

Step	V	C {2..n}	D[2..n]
Source Node : 5			1 2 3 4
Initialize	--	{1,2,3,4}	[∞, ∞, ∞,**10**]
1	4 [10]	{1,2,3}	[∞,**30**,60,10]
2	2 [30]	{1,3}	[∞,30,**60**,10]
3	3 [60]	{1}	[∞,30,60,10]

Fractional Knapsack Problem (1)

- We are given ***n*** objects and ***a*** knapsack.
- For ***i***=1,2..***n*** ; object ***i*** has a positive weight $\boldsymbol{w_i}$ and a positive value $\boldsymbol{v_i}$. The knapsack can carry a weight not exceeding ***W***.
- ***Our aim*** is *to fill the knapsack in a way that maximizes the value of the included objects, while respecting the capacity constraints.*
- In this simple version of a problem we assume that the objects can be broken into smaller pieces, so we may decide to carry only a fraction ***xi*** of object ***i*** *; where 0<= xi <= 1*

Knapsack Problem (1)



<u>Solution:</u> **if any number of each box is available, then three yellow boxes and three grey boxes.**

if only the shown boxes are available, then all but not the green box.

Knapsack Problem (1)

In symbols, the problem can be stated:

$$\text{Maximize} \sum_{i=1}^{n} v_i x_i \text{ subject to} \sum_{i=1}^{n} w_i x_i \leqslant W,$$

For this problem, an optimal solution is:

$$\sum_{i=1}^{n} w_i x_i = W$$

Knapsack Problem (1) - Example

- N=5 Objects and W=100

	1	2	3	4	5
W	**10**	**20**	**30**	**40**	**50**
V	**20**	**30**	**66**	**40**	**60**

1. *Select objects in order of decreasing value*, we choose

   Object 3 (v3=66,w3=30), Object 5 (v5=60,w5=50) and object 4 (w4=40/2=20,v4=20), **So, v3 + v5 + v4 = 66 + 60 + 20 = 146**

2. *Select objects in order of increasing weight*, we choose

   Object 1 (w1=10,v1=20), Object 2 (w2=20,v2=30), Object 3 (w3=30,v3=66), Object 4 (w4=40,v4=40), **So v1 + v2 + v3 + v4 = 20+30+66+40 = 156**

Knapsack Problem (1) - Example

- N=5 Objects and W=100

	1	2	3	4	5
W	**10**	**20**	**30**	**40**	**50**
V	**20**	**30**	**66**	**40**	**60**
Vi/Wi	**2.0**	**1.5**	**2.2**	**1.0**	**1.2**

3. *Select objects in order of decreasing vi/wi*, we choose

Object 3 (v3=66,w3=30), Object 1 (v1=20,w1=10), object 2 (v2=30,w2=20) and Object 5 (w5=50 X 4/5=40, v5=60 X 4/5=48)

So, v3 + v1 + v2 + v5 = 66 + 20 + 30 + 48 = 164

Conclusion: *If objects are selected in order of decreasing vi/wi, then algorithm knapsack finds an optimal solution.*

Knapsack Problem (1) - Example

Select	Xi					Value
	1	2	3	4	5	
Max Vi	**0**	**0**	**1**	**0.5**	**1**	**146**
Min Wi	**1**	**1**	**1**	**1**	**0**	**156**
Max Vi/Wi	**1**	**1**	**1**	**0**	**0.8**	**164**

It's also called *Fractional Knapsack*