

# Greedy Method

(最美的臉型與五官未必能組合成最美的臉孔)

There exist some problems (usually, optimization problems) whose solutions consist of  $n$  components  $(x_1, x_2, \dots, x_n)$ .

Solutions of these problems each involve  $n$  decisions, each for determining one component, i.e.,

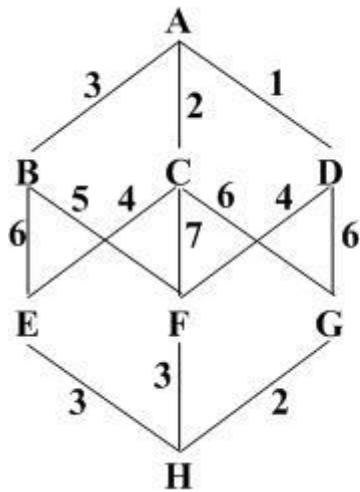
For  $i \leftarrow 1$  to  $n$

determine  $x_i$  (find locally optimal  $x_i$ ).

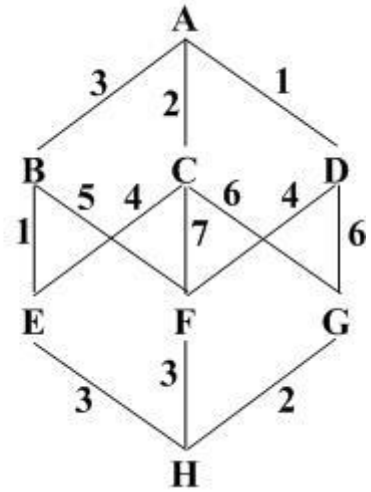
Since  $x_i$  is locally optimal,  $(x_1, x_2, \dots, x_n)$  is not necessarily globally optimal.

The greedy method is often used in designing approximation algorithms.

**Ex. Find a shortest path from  $A$  to  $H$  by the greedy method.**



$x_1=AD, x_2=DF, x_3=FH$   
**optimal solution: A-D-F-H**



$x_1=AD, x_2=DF, x_3=FH$   
**optimal solution: A-B-E-H**

### **Ex. The Knapsack Problem.**

**We are given  $n$  objects and a knapsack. Object  $i$  has a weight  $w_i > 0$  and the knapsack has a capacity  $M$ .**

**If a fraction  $x_i$  ( $0 \leq x_i \leq 1$ ) of object  $i$  is placed into the knapsack, then a profit of  $p_i x_i > 0$  is earned.**

**The objective is to obtain a filling of the knapsack that maximizes the total profit earned.**

**Mathematically, the problem may be formulated as follows.**

$$\text{Max} \quad \sum_{i=1}^n p_i x_i \quad (1)$$

$$\text{s.t.} \quad \sum_{i=1}^n w_i x_i \leq M \quad (2)$$

$$0 \leq x_i \leq 1 \quad (3)$$

**A feasible solution is any set  $(x_1, x_2, \dots, x_n)$  satisfying (2) and (3). An optimal solution is a feasible solution that maximizes (1).**

**Greedy method 1 : largest profit first.**

**Objects are selected in nonincreasing order of  $p_i$ ,**

**i. e.,  $p_{s1} \geq p_{s2} \geq \dots \geq p_{sn}$ .**

**Greedy method 2 : smallest weight first.**

**Objects are selected in nondecreasing order of  $w_i$ ,**

**i. e.,  $w_{s1} \leq w_{s2} \leq \dots \leq w_{sn}$ .**

**Greedy method 3 : maximal profit per unit of  
capacity first.**

**Objects are selected in nonincreasing order of  $p_i/w_i$ ,**

**i.e.,  $p_{s1}/w_{s1} \geq p_{s2}/w_{s2} \geq \dots \geq p_{sn}/w_{sn}$ .**

**For example, consider the following problem instance.**

$$\begin{aligned}
 \text{Max} \quad & 25x_1 + 24x_2 + 15x_3 \\
 \text{s.t.} \quad & 18x_1 + 15x_2 + 10x_3 \leq 20 \\
 & 0 \leq x_1, x_2, x_3 \leq 1
 \end{aligned}$$

	$s_1$	$s_2$	$s_3$	$x_1$	$x_2$	$x_3$	total profit
<b>Method 1</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>1</b>	<b>2/15</b>	<b>0</b>	<b>28.2</b>
<b>Method 2</b>	<b>3</b>	<b>2</b>	<b>1</b>	<b>0</b>	<b>2/3</b>	<b>1</b>	<b>31</b>
<b>Method 3</b>	<b>2</b>	<b>3</b>	<b>1</b>	<b>0</b>	<b>1</b>	<b>1/2</b>	<b>31.5</b>

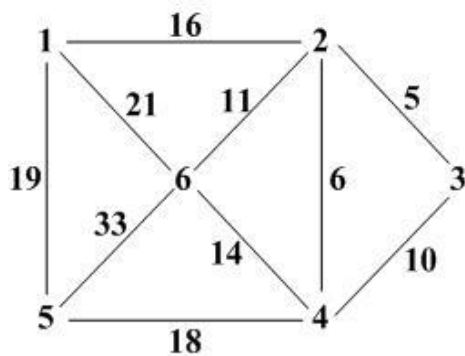
**The time complexities of the above three greedy methods are all  $O(n \log n)$ .**

**Theorem. Greedy method 3 always generates an optimal solution to the knapsack problem.**

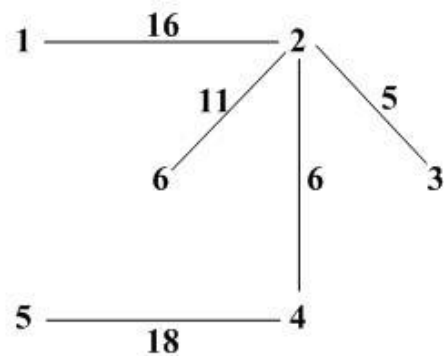
**Proof. Refer to *Computer Algorithms*, by Horowitz, Sahni, and Rajasekaran, p. 201.**

### Ex. The Minimum Spanning Tree Problem.

Let  $G = (V, E)$  be a weighted connected undirected graph. A spanning subgraph  $G' = (V, E')$  ( $E' \subseteq E$ ) of  $G$  is a *spanning tree* of  $G$  iff  $G'$  is a tree.  $G'$  is called a *minimum spanning tree* of  $G$  if it has the smallest total weight of  $E'$  among all spanning trees of  $G$ .



$G = (V, E)$



the minimum spanning  
tree of  $G$

## **Greedy method 1. Prim's algorithm.**

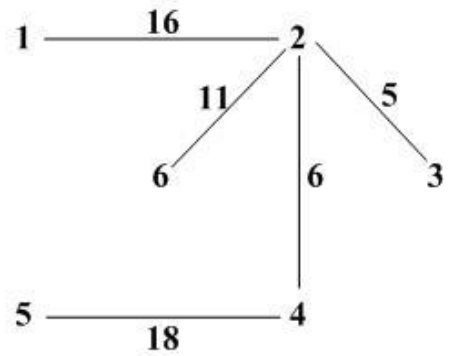
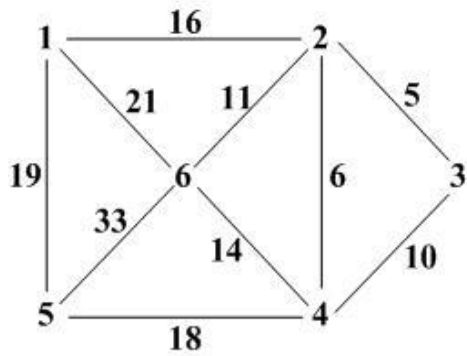
**Step 1. Arbitrarily choose a vertex from  $V$ .**

**Step 2. Select the edge with minimum weight among all edges that connect the chosen vertices with those unchosen ones.**

**Step 3. Repeat step 2 until a spanning tree is formed.**



For example, if we start at vertex 1, then the edges are selected in the sequence of (1, 2), (2, 3), (2, 4), (2, 6), (4, 5).



**Theorem.** The spanning tree produced by Prim's algorithm is minimum.

**Proof.** Refer to *Graph Algorithms*, by Even, p. 24.

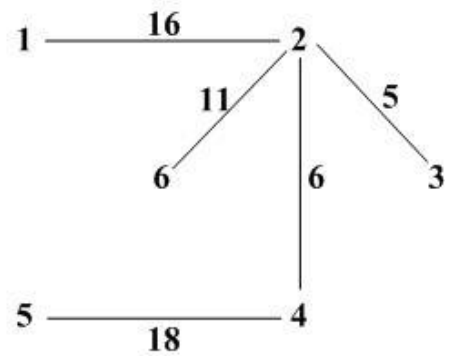
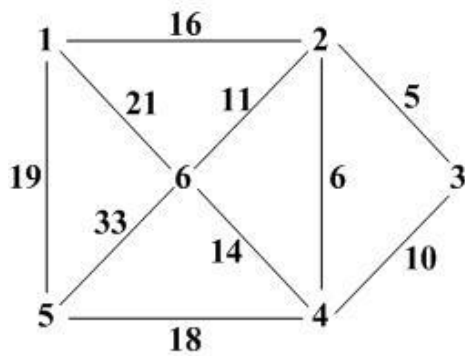
**Prim's algorithm takes  $O(|V|^2)$  time.**

**Greedy method 2. Kruskal's algorithm.**

**Step 1. Sort edges in nondecreasing order of weights.**

**Step 2. Select feasible edges (not form a cycle), one at a time, from the sorted list of edges.**

**For example, for the same example, feasible edges are selected in the sequence of (2, 3), (2, 4), (2, 6), (1, 2), (4, 5).**

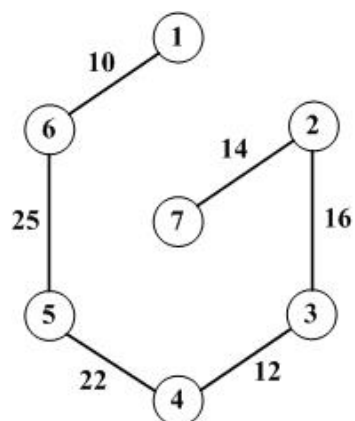
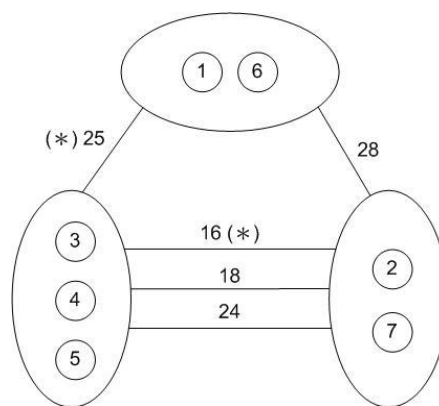
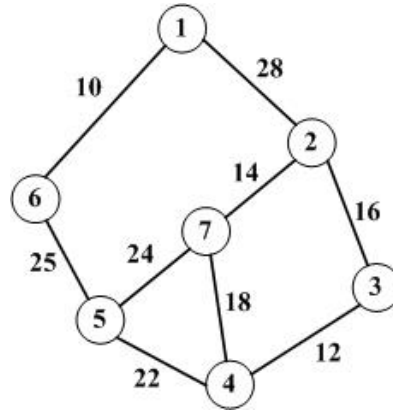


**Theorem.** The spanning tree produced by Kruskal's algorithm is minimum.

**Proof.** Refer to *Computer Algorithms*, by Horowitz, Sahni, and Rajasekaran, p. 225.

**Kruskal's algorithm takes  $O(|E| \cdot \log |E|)$  time.**

### Greedy method 3. Sollin's algorithm

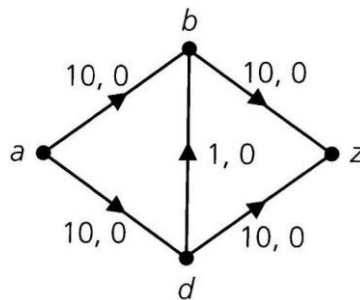


**Sollin's algorithm can be implemented in  $O(|E| \cdot \log \log |V|)$  time.**

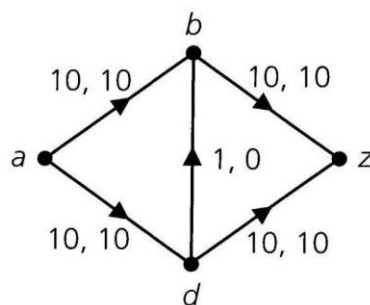
**Refer to : A. C. Yao, "An  $O(|E| \cdot \log \log |V|)$  algorithm for finding minimum spanning trees," *Info. Proc. Lett.*, vol. 4, no. 1, pp. 21-23, 1975.**

**An example of inferior greedy algorithms.**

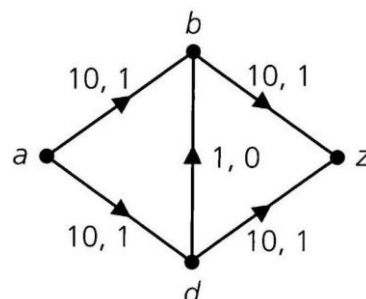
**Ford & Fulkerson's maximum-flow minimum-cut algorithm:**



**$(a, b, z)$  and  $(a, d, z)$  are selected as augmenting paths.**



**$(a, d, b, z)$  and  $(a, b, d, z)$  are selected, in the sequence, as augmenting paths.**



**The worst-case time complexity is exponential.**