# Greedy Algorithms
# Fractional Knapsack and Graphs

**(Class 27)**

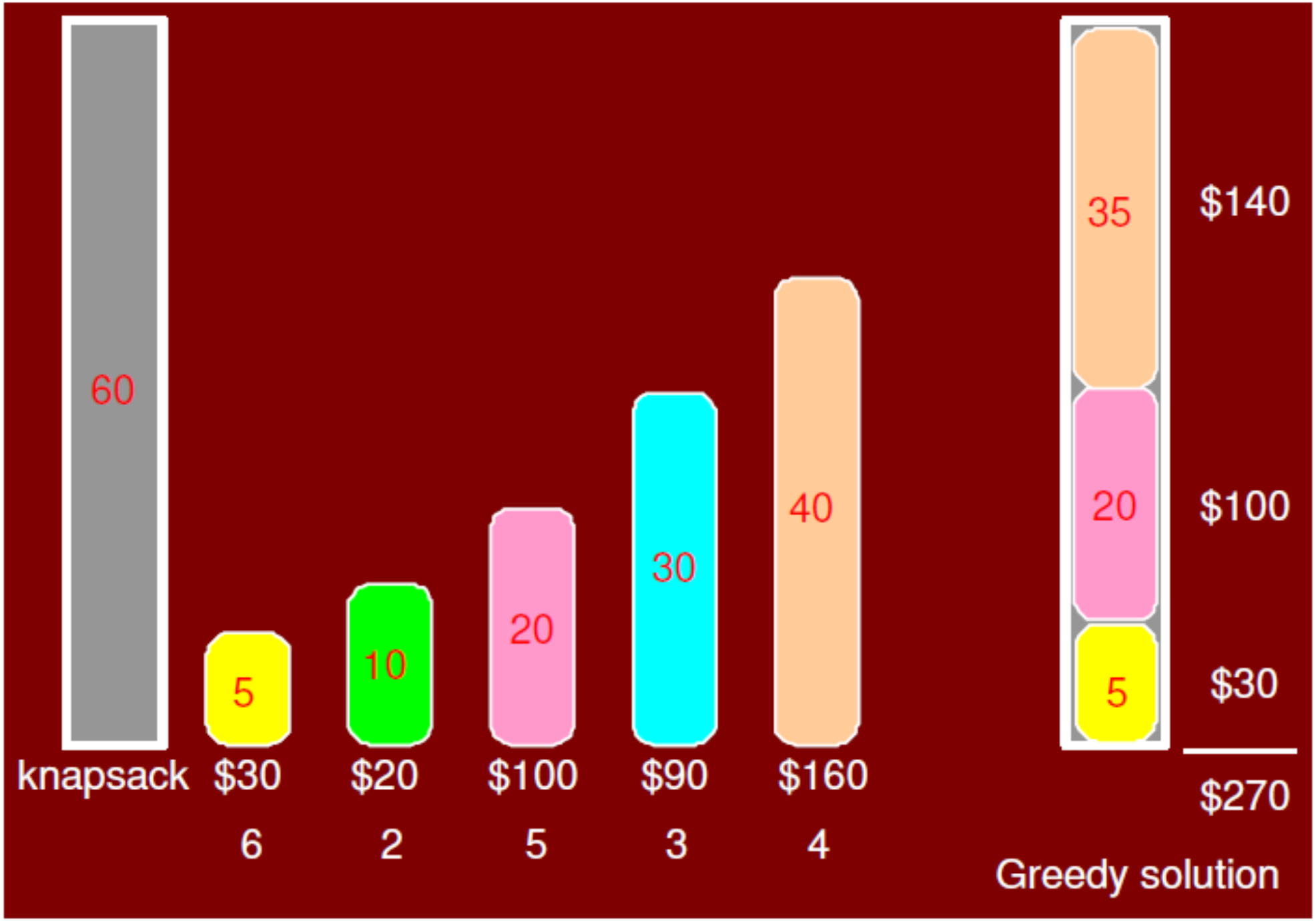
From Book's Page Number 547 (Chapter 20)

# Fractional Knapsack Problem From Book's Page Number 429

- Earlier we saw the 0-1 knapsack problem.
- A knapsack can only carry $W$ total weight.
- There are $n$ items; the $i^{th}$ item is worth $v_i$ and weighs $w_i$.
- Items can either be put in the knapsack or not.
- The goal was to maximize the value of items without exceeding the total weight limit of $W$.

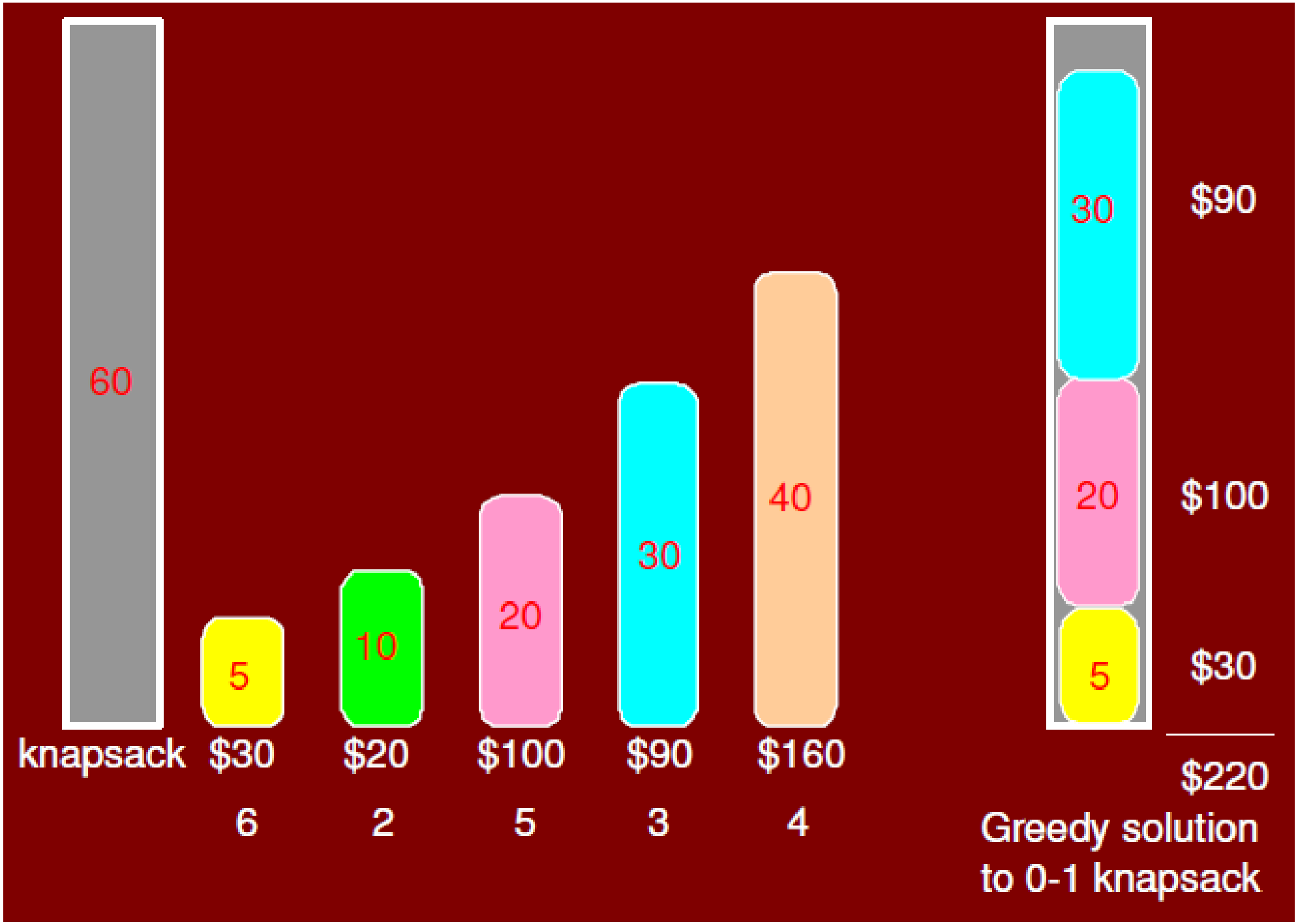
- In the fractional knapsack problem, the setup is exactly the same.
- But we are allowed to take fraction of an item for a fraction of the weight and fraction of value.
- The 0-1 knapsack problem is hard to solve.
- However, there is a simple and efficient greedy algorithm for the fractional knapsack problem.

- Let $p_i = v_i/w_i$ denote the *value per unit weight* ratio for item $i$.
- Sort the items in decreasing order of $p_i$.
- Add items in decreasing order of $p_i$.
- If the item fits, we take it all.
- At some point there is an item that does not fit in the remaining space.
- We take as much of this item as possible thus filling the knapsack completely.

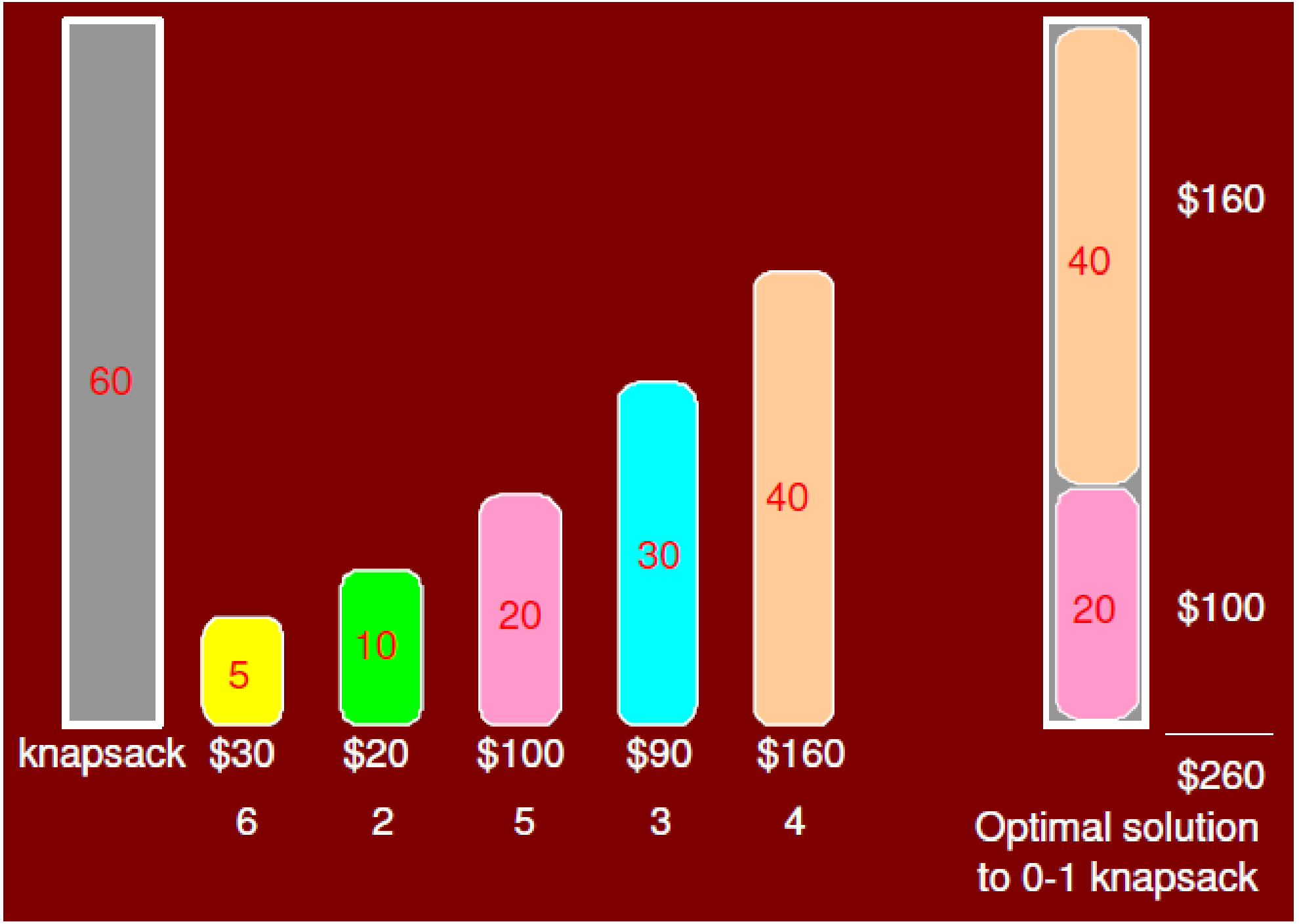

60
5
10
20
30
40
35
20
5
$140
$100
$30
$270
knapsack
$30
$20
$100
$90
$160
6
2
5
3
4
Greedy solution

- It is easy to see that the greedy algorithm is optimal for the fractional knapsack problem.
- Given a room with sacks of gold, silver and bronze, a person would probably take as much gold as possible.
- Then take as much silver as possible and finally as much bronze as possible.
- It would never benefit to take a little less gold so that one could replace it with an equal weight of bronze.

- We can also observe that the greedy algorithm is not optimal for the 0-1 knapsack problem.
- Consider the example shown in the figure.


60
5
10
20
30
40
30
20
5
$90
$100
$30
knapsack
$30
$20
$100
$90
$160
$220
6
2
5
3
4
Greedy solution
to 0-1 knapsack

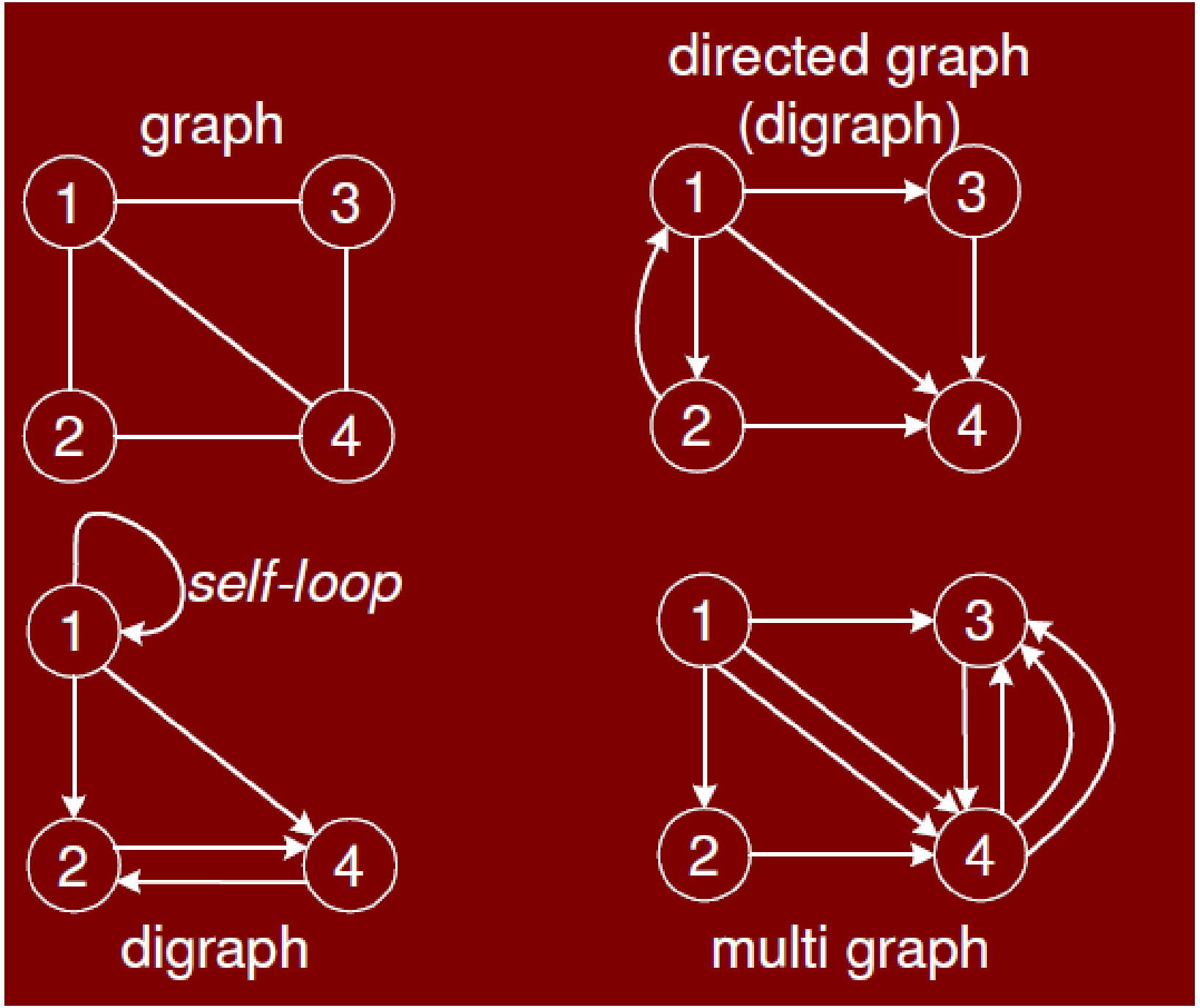
- If you were to sort the items by $p_i$.
- Then you would first take the items of weight 5, then 20, and then (since the item of weight 40 does not fit) you would settle for the item of weight 30, for a total value of \$30 + \$100 + \$90 = \$220.
- On the other hand, if you had been less greedy, and ignored the item of weight 5, then you could take the items of weights 20 and 40 for a total value of \$100+\$160 = \$260.


60
5
10
20
30
40
40
20
knapsack
$30
$20
$100
$90
$160
6
2
5
3
4
$160
$100
$260
Optimal solution
to 0-1 knapsack

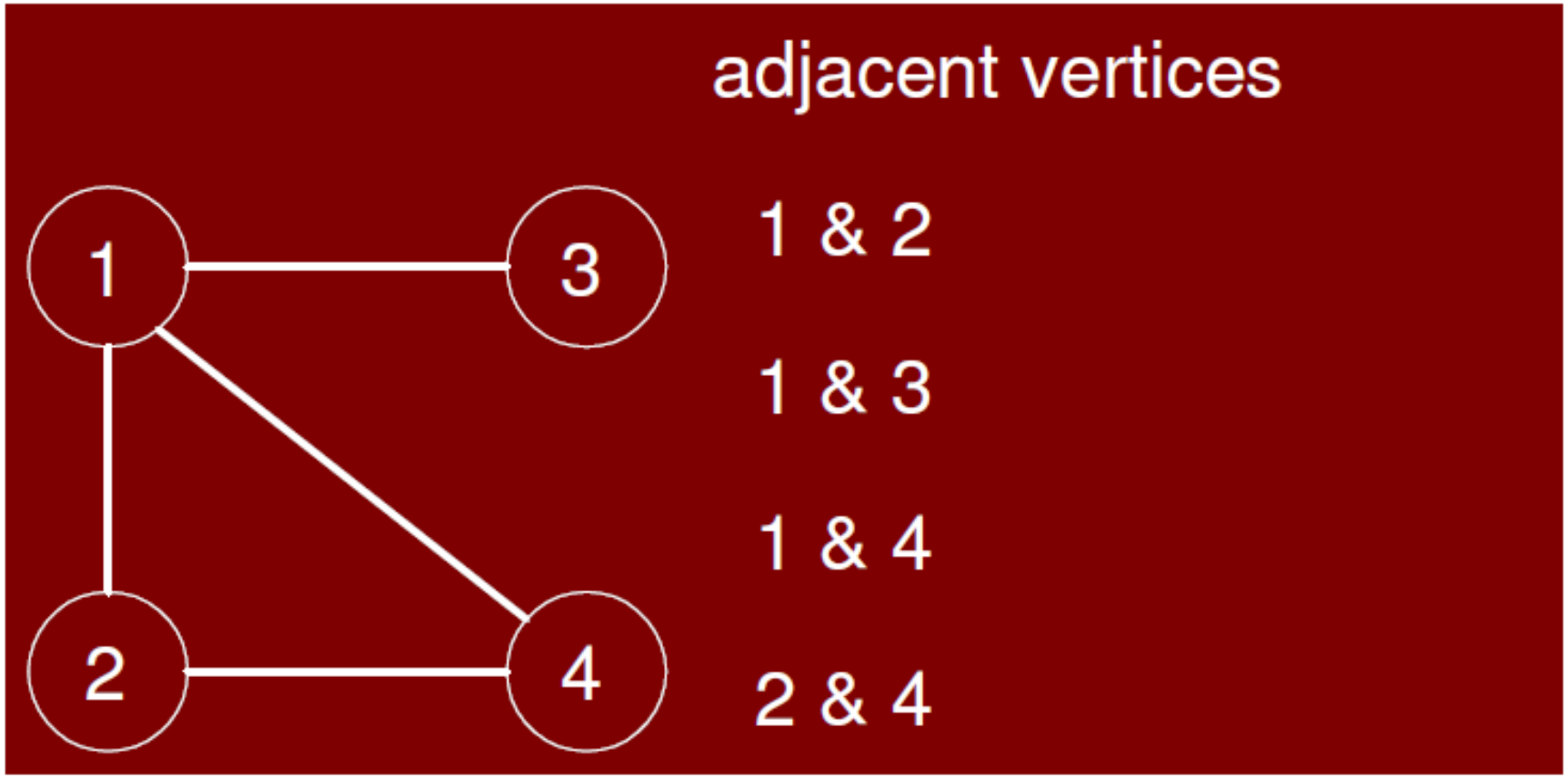
# Graphs From Book's Page Number 547

- We begin a major new topic: Graphs.
- Graphs are important discrete structures because they are a flexible mathematical model for many application problems.
- Any time there is a set of objects and there is some sort of "connection" or "relationship" or "interaction" between pairs of objects, a graph is a good way to model this.
- Examples of this can be found in computer and communication networks transportation networks, e.g., roads, VLSI, logic circuits surface meshes for shape description in computer-aided design, Convolutional Neural Networks (CNN), etc.

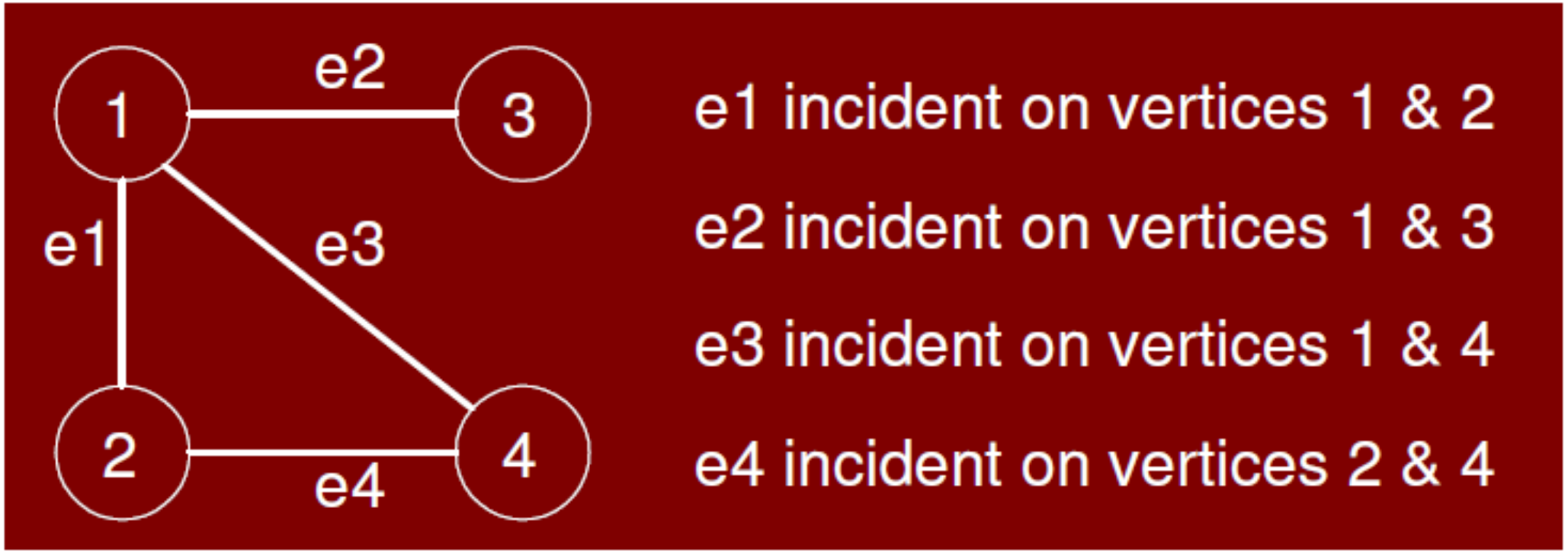
- A graph $G = (V, E)$ consists of a finite set of *vertices* $V$ (or nodes) and $E$, a binary relation on $V$ called *edges*.
- $E$ is a set of pairs from $V$.
- If a pair is *ordered*, we have a *directed* graph.
- For *unordered* pair, we have an *undirected* graph.


graph
directed graph
(digraph)
1
3
2
4
1
3
2
4
self-loop
1
2
4
digraph
1
3
2
4
multi graph

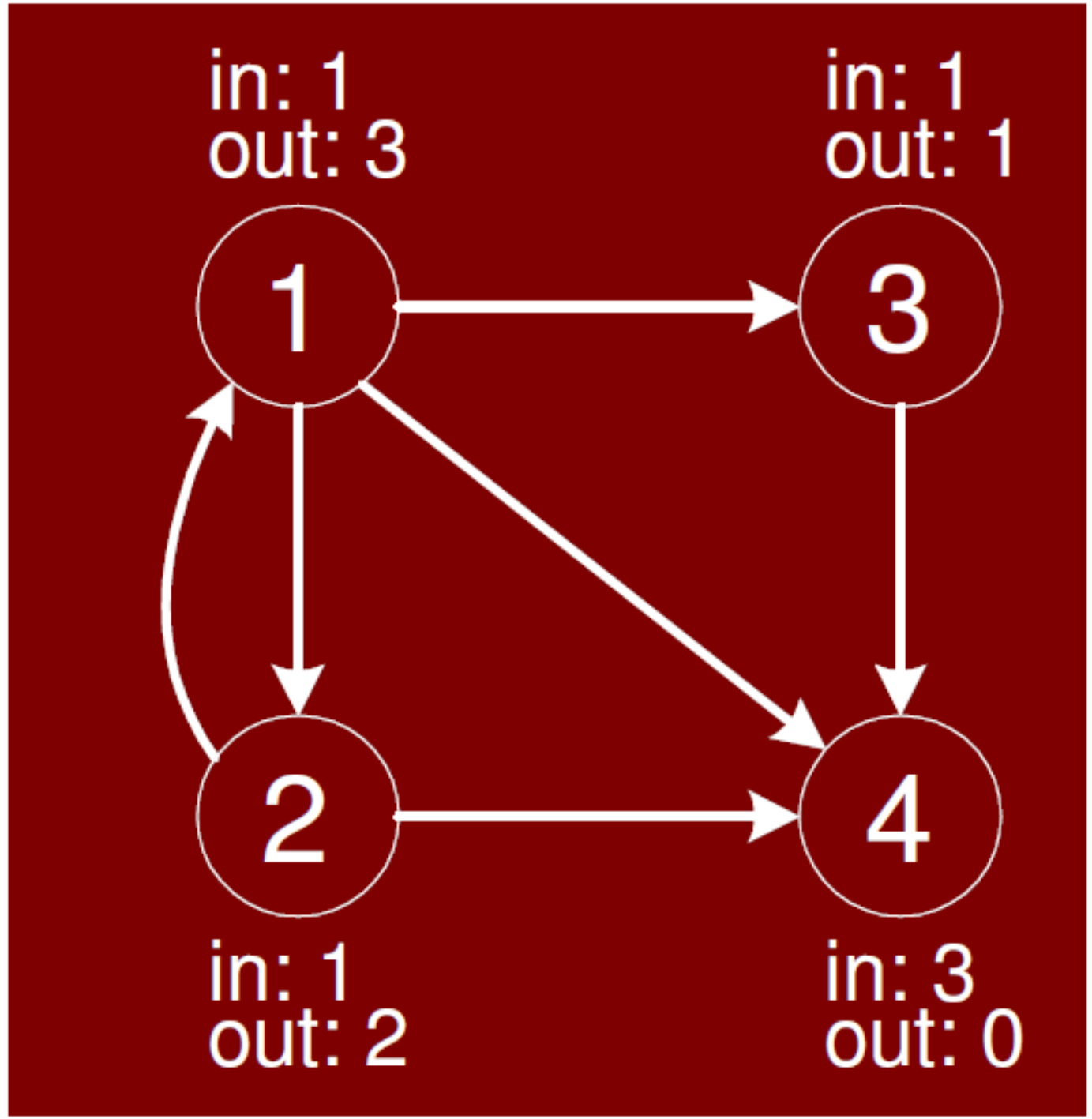
- A vertex $w$ is adjacent to vertex $v$ if there is an edge from $v$ to $w$.



- In an undirected graph, we say that an edge is *incident* on a vertex if the vertex is an endpoint of the edge.



- In a directed graph, the number of edges coming out of a vertex is called the *out-degree* of that vertex.
- Number of edges coming in is the *in-degree*.
- In an undirected graph, we just talk of degree of a vertex.
- It is the number of edges incident on the vertex.


in: 1
out: 3
in: 1
out: 1
1
3
2
4
in: 1
out: 2
in: 3
out: 0