CS350: Data Structures

Dijkstra's Shortest Path Alg.

James Moscola
Department of Physical Sciences
York College of Pennsylvania

Shortest Path Algorithms

- **Several different shortest path algorithms exist**
  - **Dijkstra's Algorithm** - suitable for finding shortest path in a graph when all edge weights are positive values
  - Bellman-Ford Algorithm - slower than Dijkstra's algorithm, but can be used when negative edge weights exist
  - Floyd-Warshall Algorithm - finds the length of the shortest path for all pairs of vertices in a graph; can be used with both positive and negative edge weights

Uses for Dijkstra's Algorithm

- **Finding the shortest route between two locations for a GPS systems**
- **Determining the optimal path for packets through a network -- that is, the path with the shortest delay**
- **Determining the fewest number of moves in which one might solve a Rubik's Cube**
- **Many other uses ...**

Dijkstra's Algorithm

- **Algorithm to find the shortest weighted path from some source vertex s in a graph to all other vertices in the graph (i.e. one-to-all shortest path problem)**
  - The node s is the start vertex, from which to shortest path to all other nodes in the graph will be determined

- **Algorithm works as follows:**
  - Starts by assigning some initial distance value for each node in the graph
    - Distance from node s to itself is 0
    - Distances from node s to all other nodes in graph are initialized to INFINITY
  - Operates in steps, where at each step the algorithm improves the distance values for nodes in the graph
  - At each step the shortest distance from node s to another node in the graph is determined

Detailed Description of Dijkstra's Algorithm

- **Let the node at which we are starting be called the start node. Let the distance of node Y be the distance from the start node to Y. Dijkstra's algorithm will assign some initial distance values and will try to improve them step by step.**

  (1) Assign to every node a tentative distance value: set it to zero for our start node and to infinity for all other nodes.

  (2) Mark all nodes except the start node as unvisited. Set the start node as current. Create a set of the unvisited nodes called the unvisited set consisting of all the nodes except the start node.

  (3) For the start node, consider all of its unvisited adjacent nodes and calculate their tentative distances. For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B (through A) will be 6+2=8. If this distance is less than the previously recorded distance, then overwrite that distance. Even though a neighbor has been examined, it is not marked as visited at this time, and it remains in the unvisited set.

  (4) When we are done considering all of the neighbors of the current node, mark the current node as visited and remove it from the unvisited set. A visited node will never be checked again; its distance recorded now is final and minimal.

  (5) If the unvisited set is empty, then stop. The algorithm has finished.

  (6) Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step 3.

Detailed description from Wikipedia

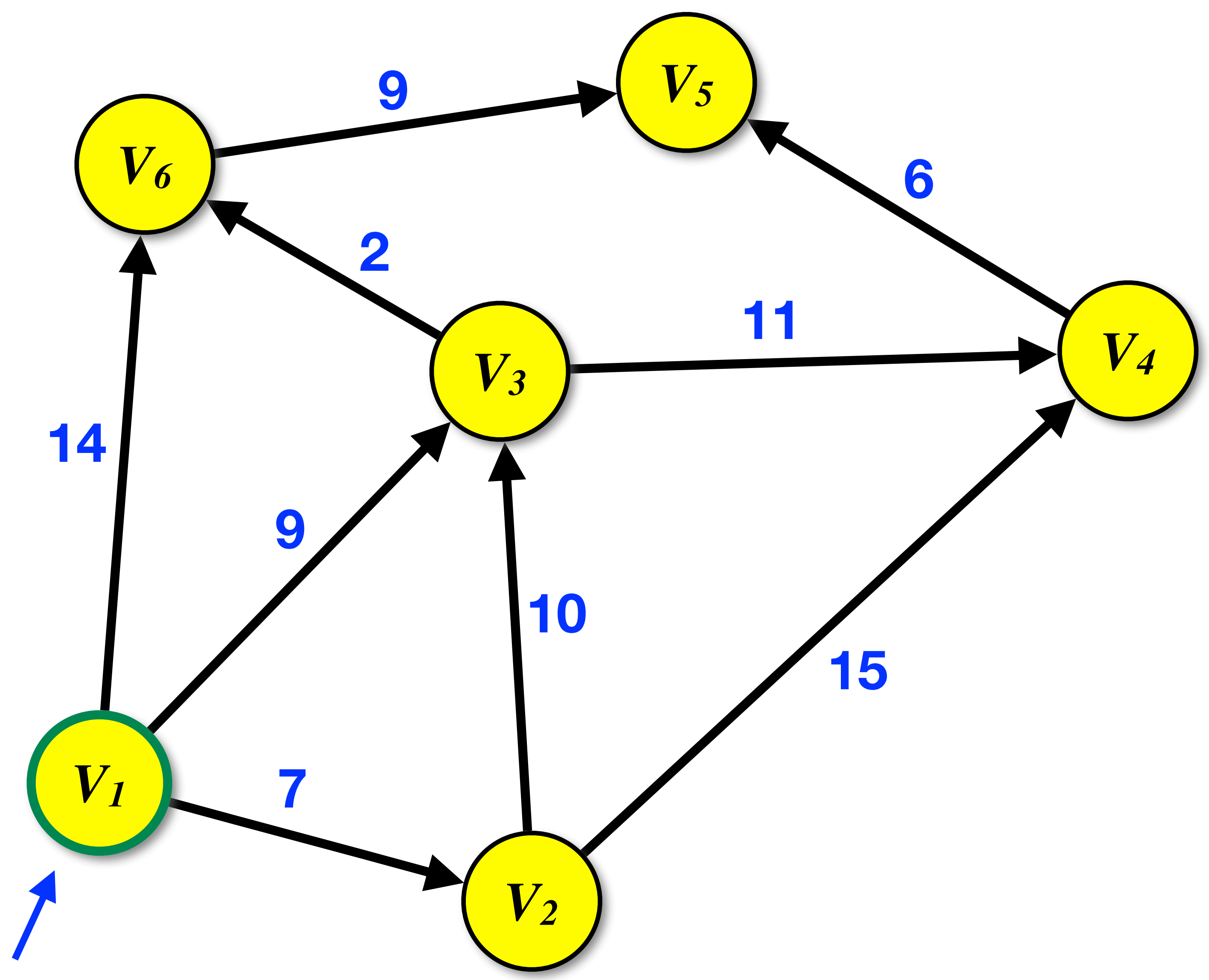
Implementation Ideas

- **Note that the last step in the description states the following:**

> (1) Set the unvisited node marked with the smallest tentative distance as the next "current node" and go back to step 3.

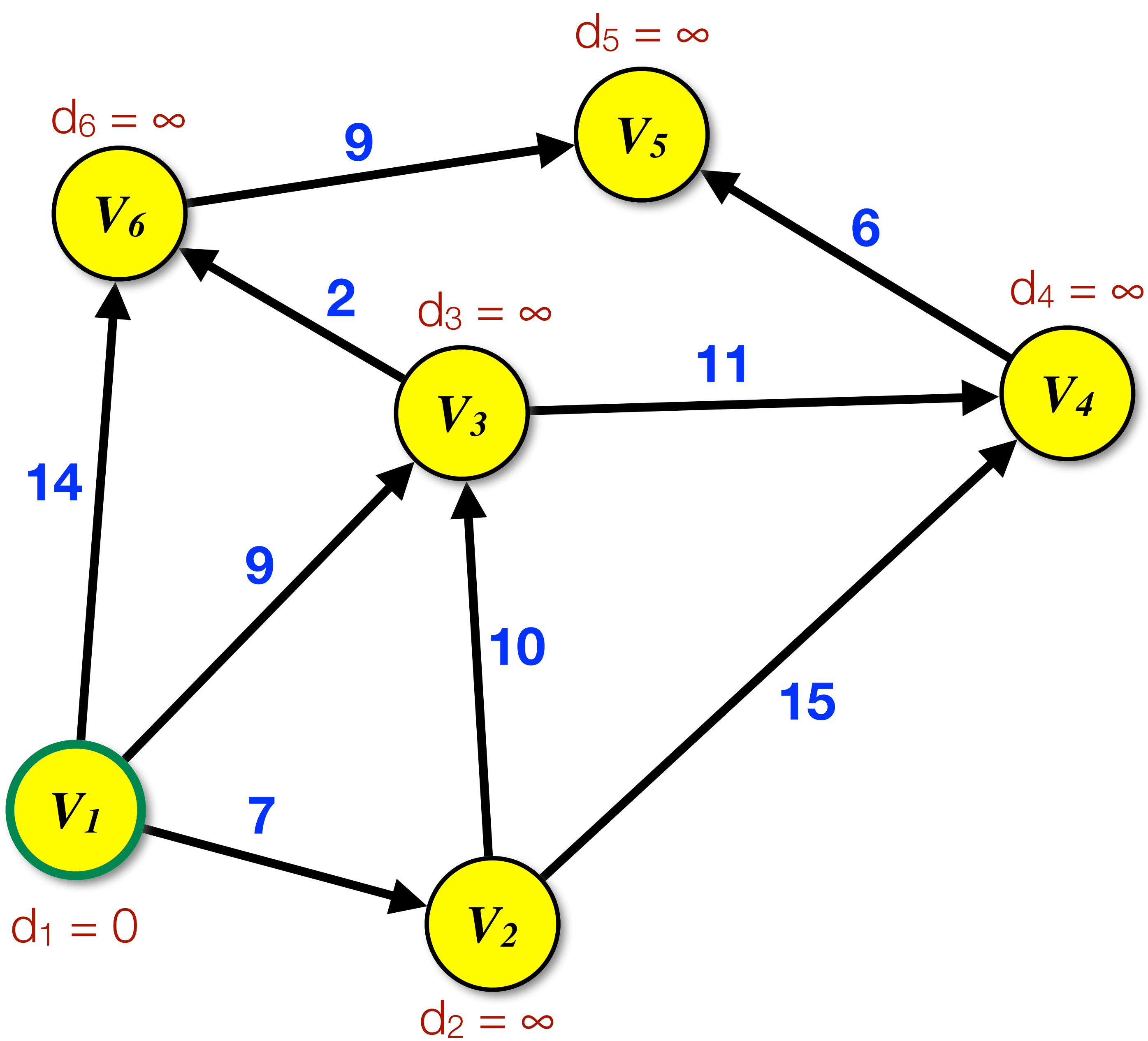
- **A simple way to keep track of the node with the smallest tentative distance is to maintain a minHeap as a priority queue; the element at the root of the minHeap is the node with the smallest tentative distance**

Example of Dijkstra's Algorithm



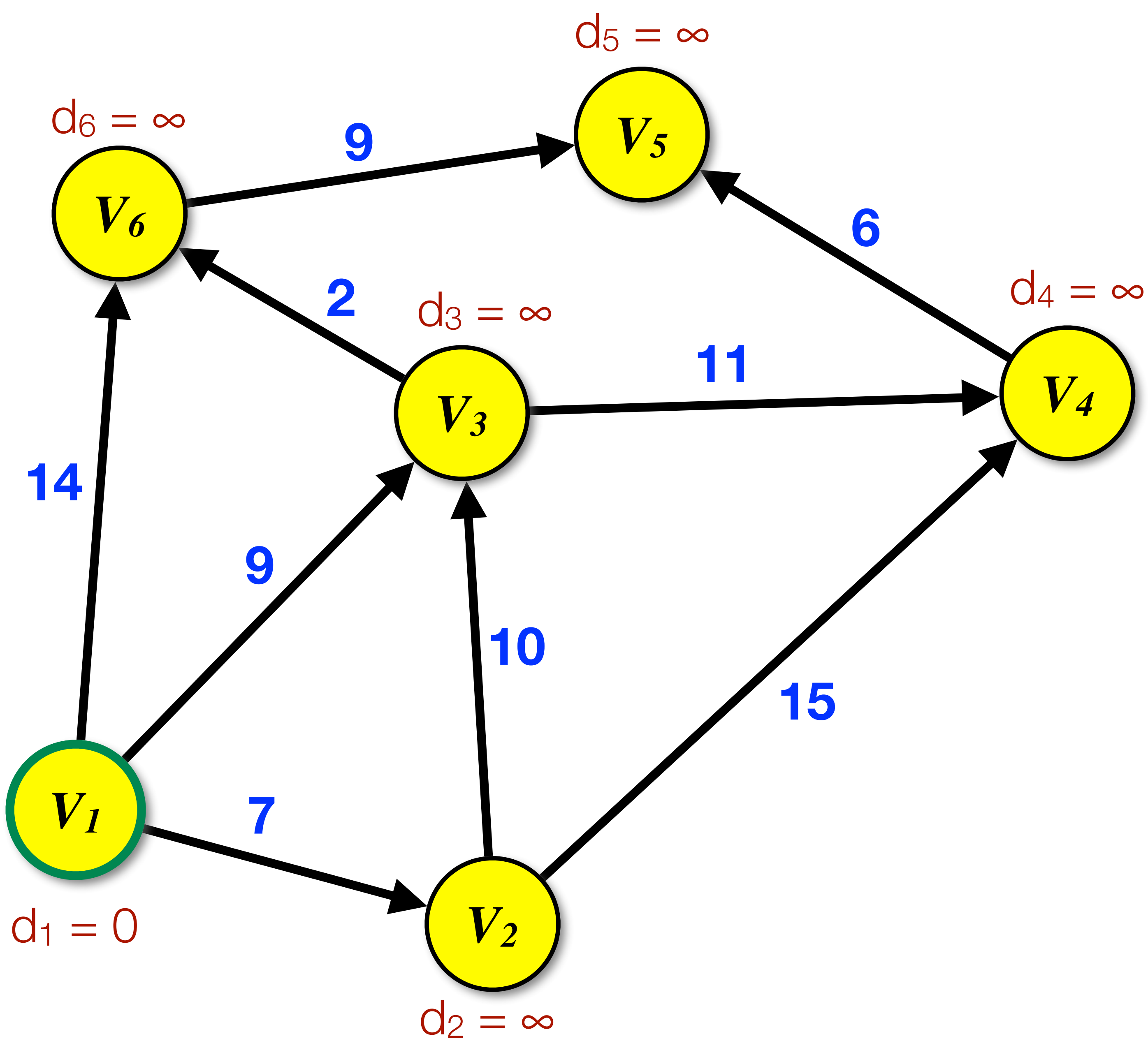
Select V_1 as the start node

Example of Dijkstra's Algorithm

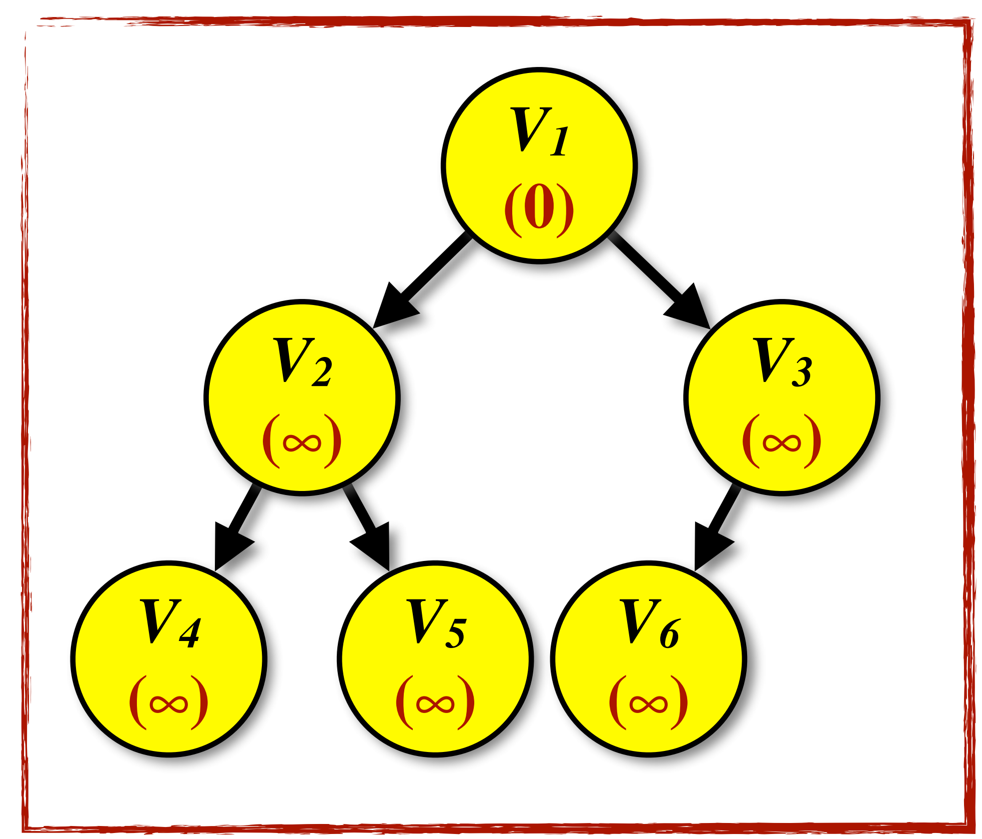


Initialize the distances from V_1 to all other nodes in the graph

Example of Dijkstra’s Algorithm

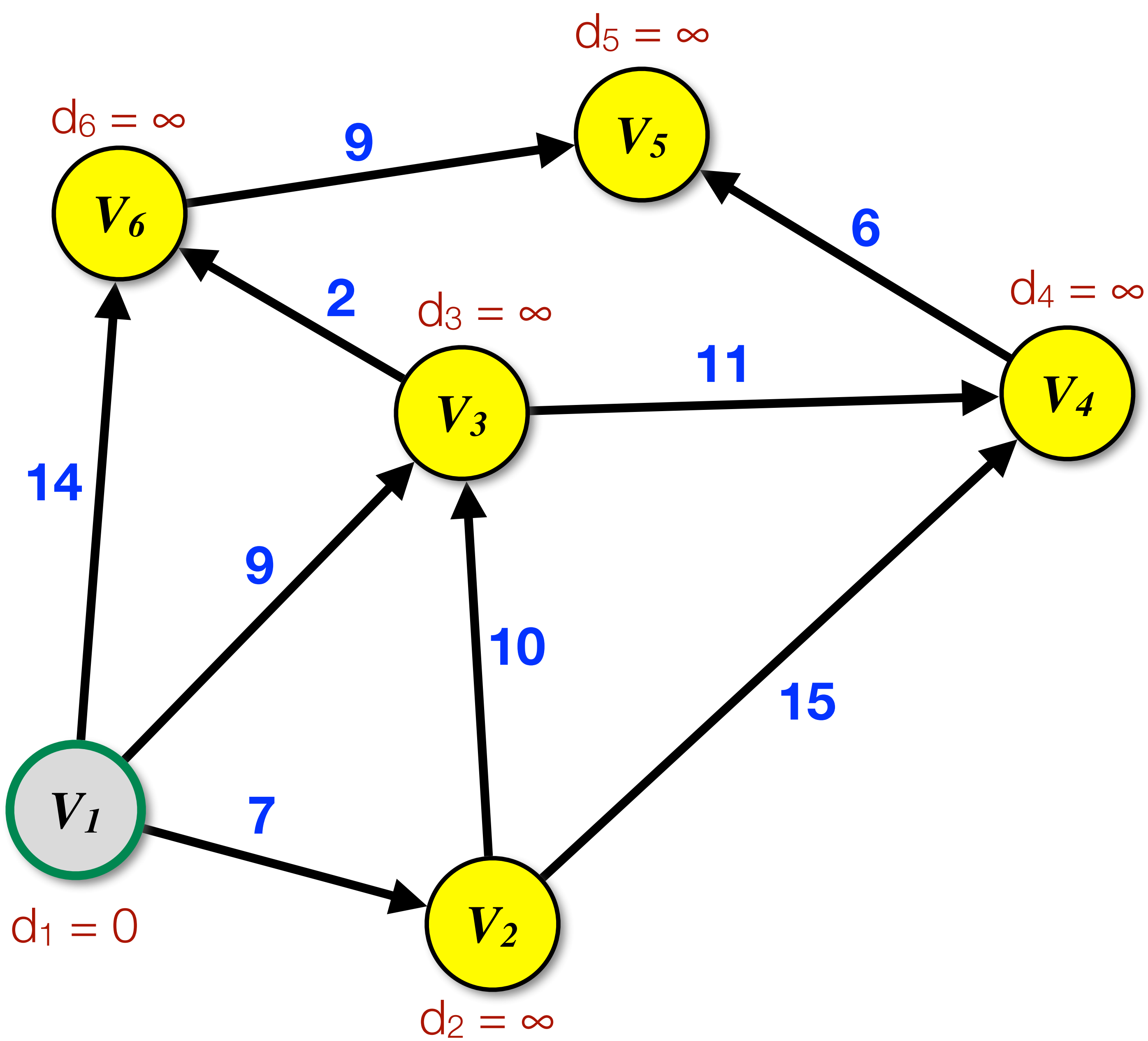


- Insert all vertices into a minHeap to keep track of minimum distances
- Sort the minHeap based on the distance from the source node

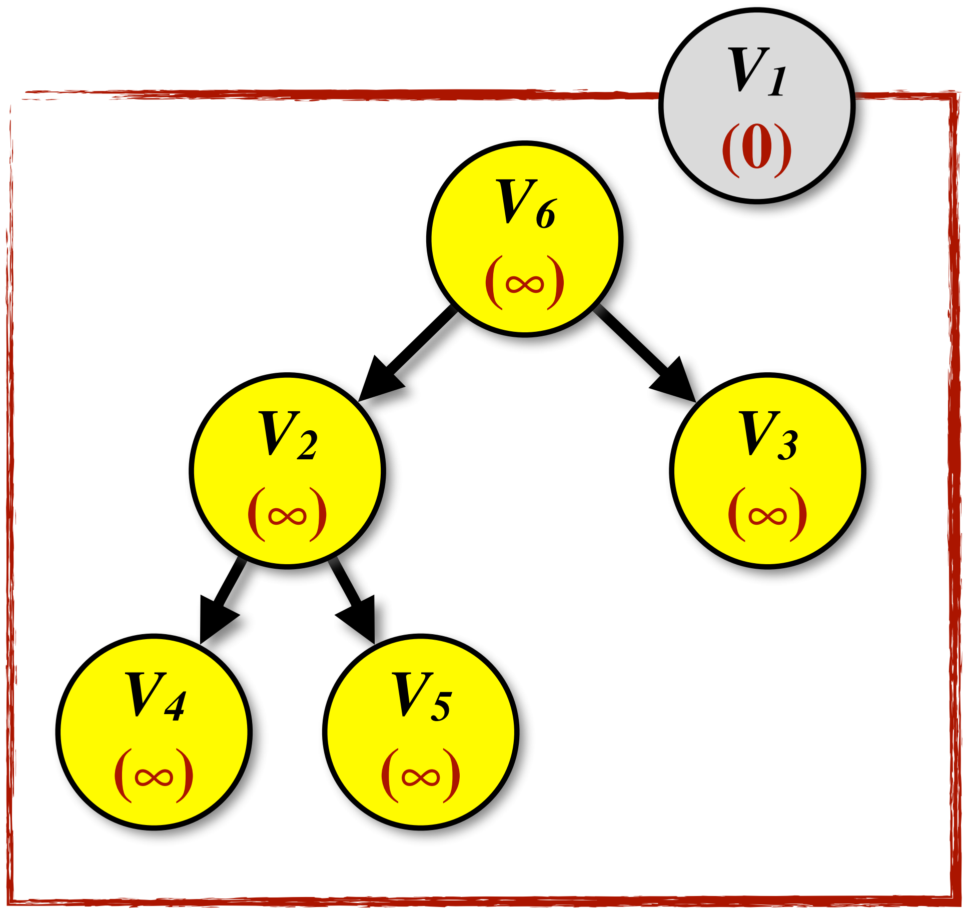


minHeap

Example of Dijkstra's Algorithm

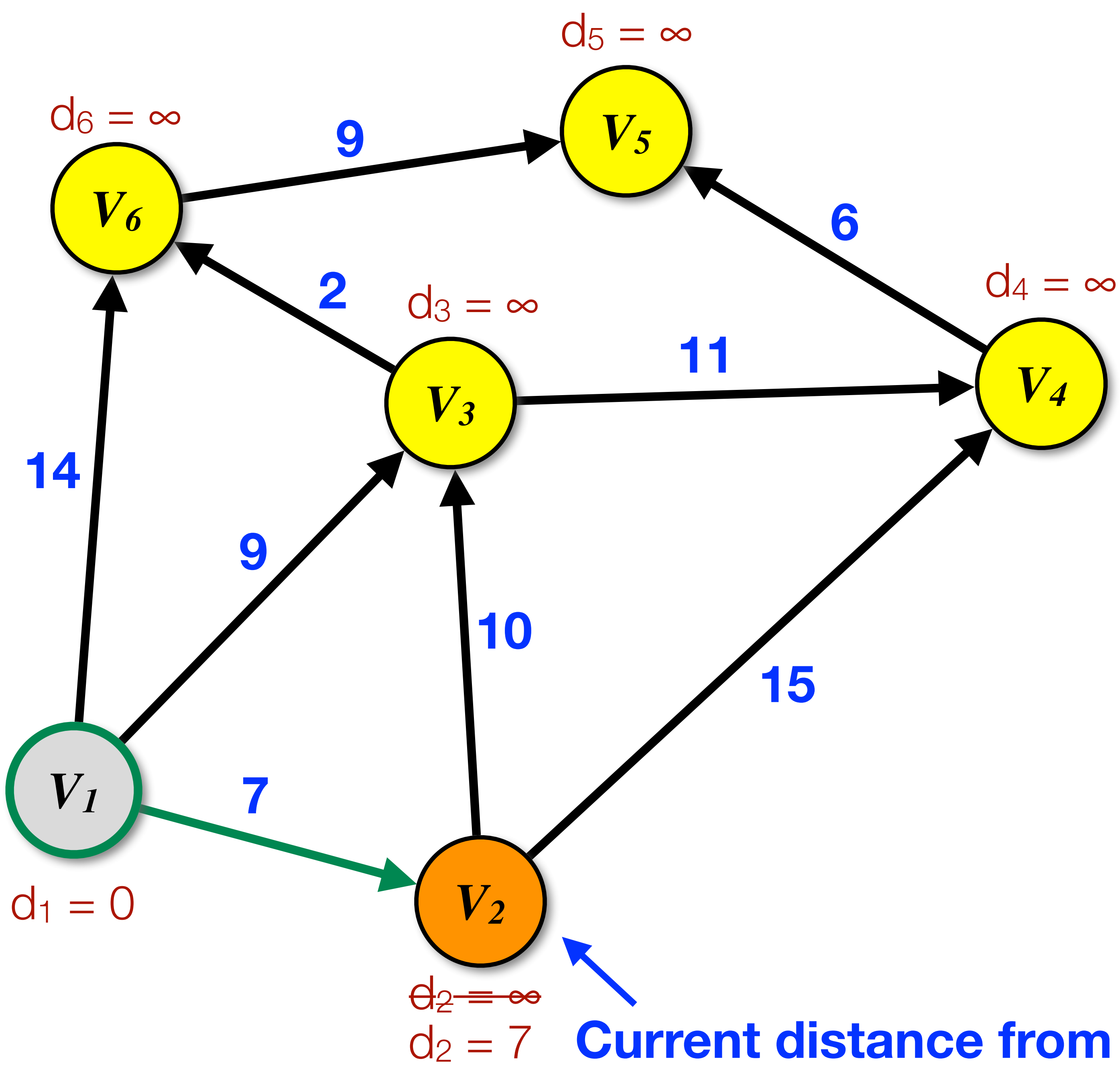


- **After initialization, pull the minimum node from the heap and begin processing by visiting the node's neighbors and updating their distances**



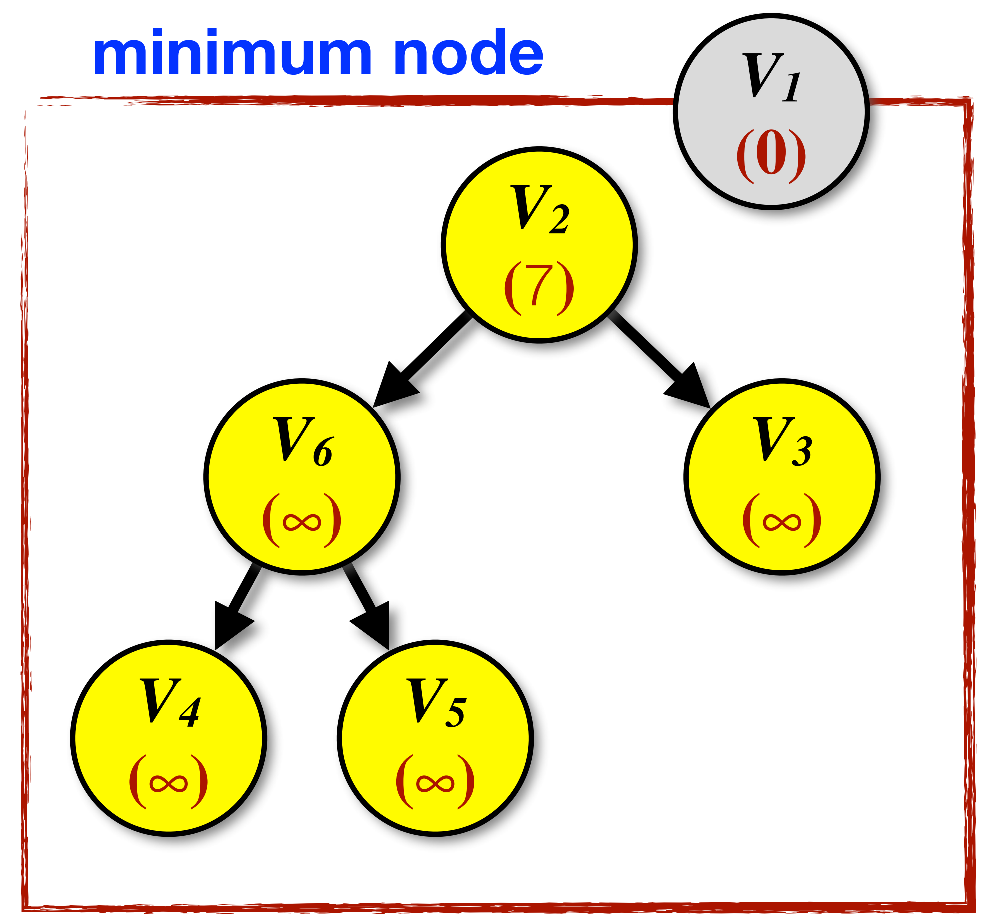
minHeap

Example of Dijkstra's Algorithm



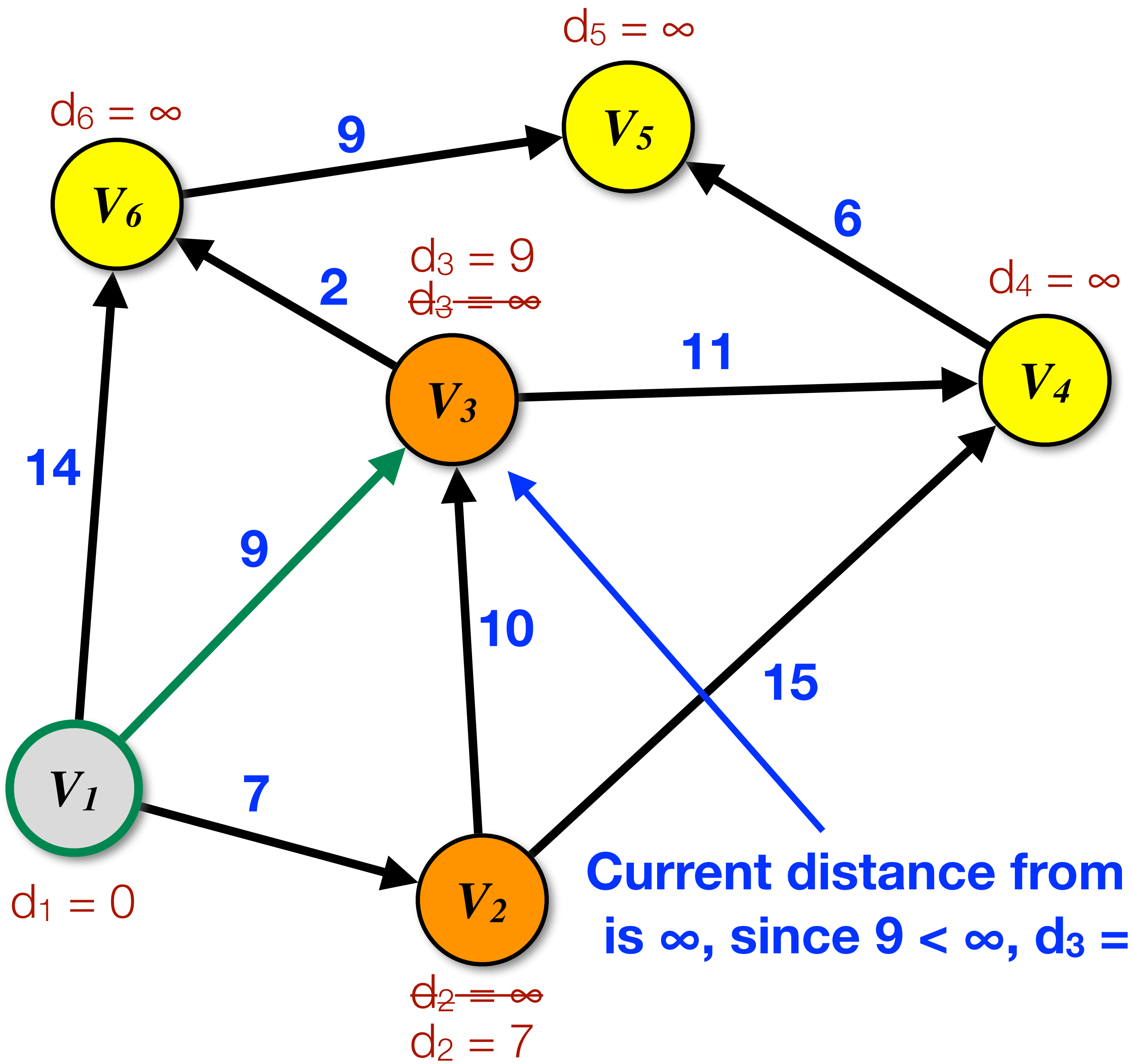
Current distance from V_1 is ∞, since 7 < ∞, d_2 = 7

- **When the distance of a node is updated, change the distance value in the minHeap and adjust the heap; in this case, V_2 becomes the new minimum node**



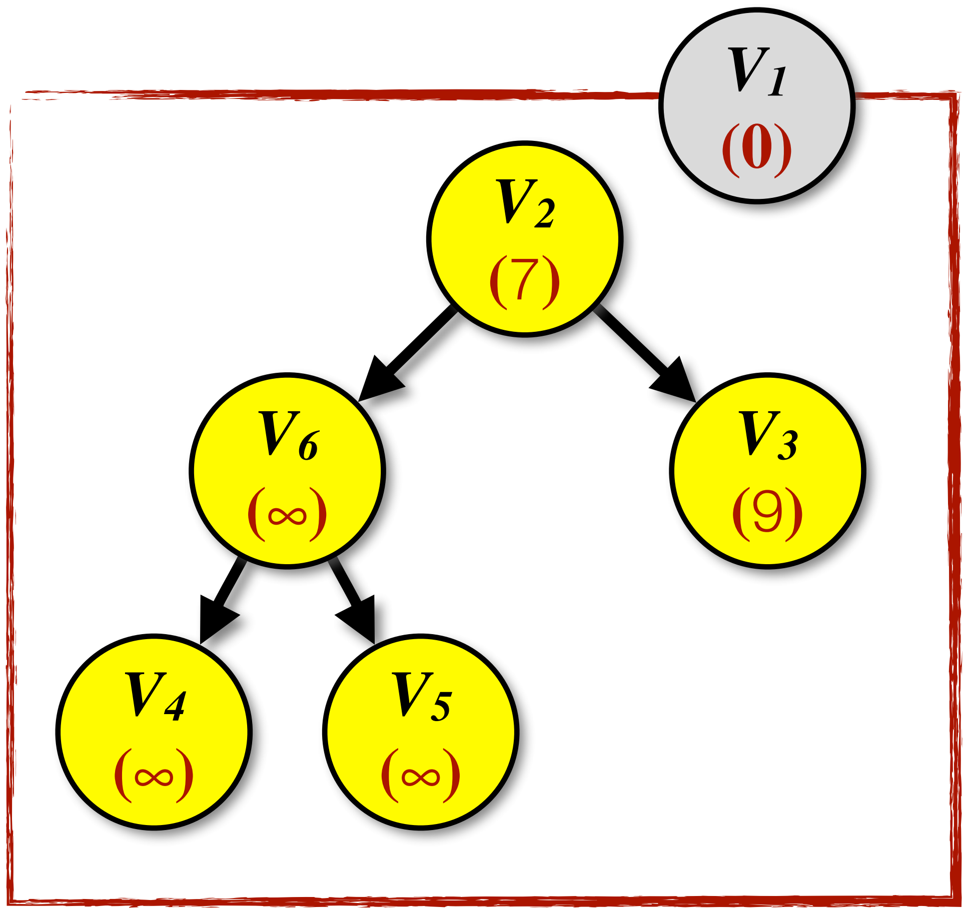
minHeap

Example of Dijkstra's Algorithm



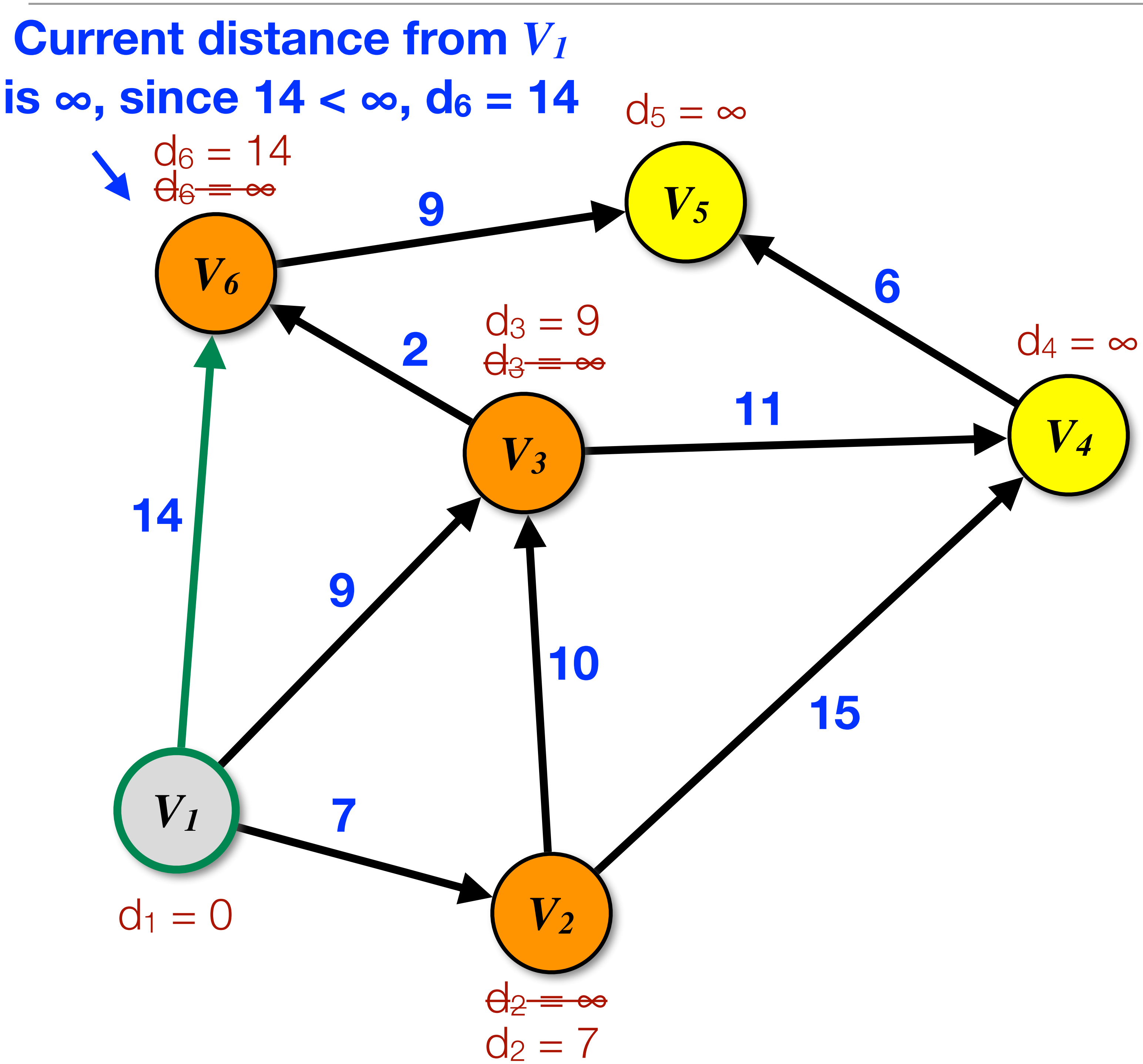
Current distance from V_1 is ∞, since 9 < ∞, d_3 = 9

- **When the distance of a node is updated, change the distance value in the minHeap and adjust the heap**

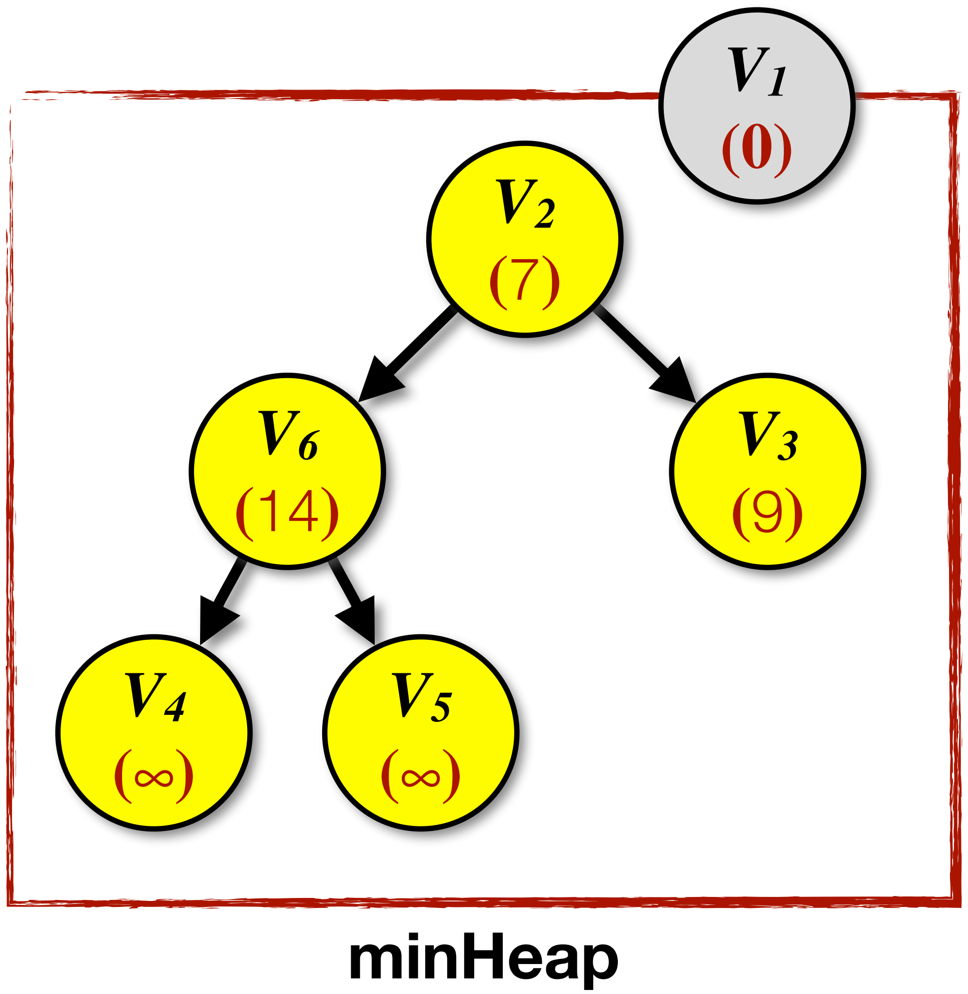


minHeap

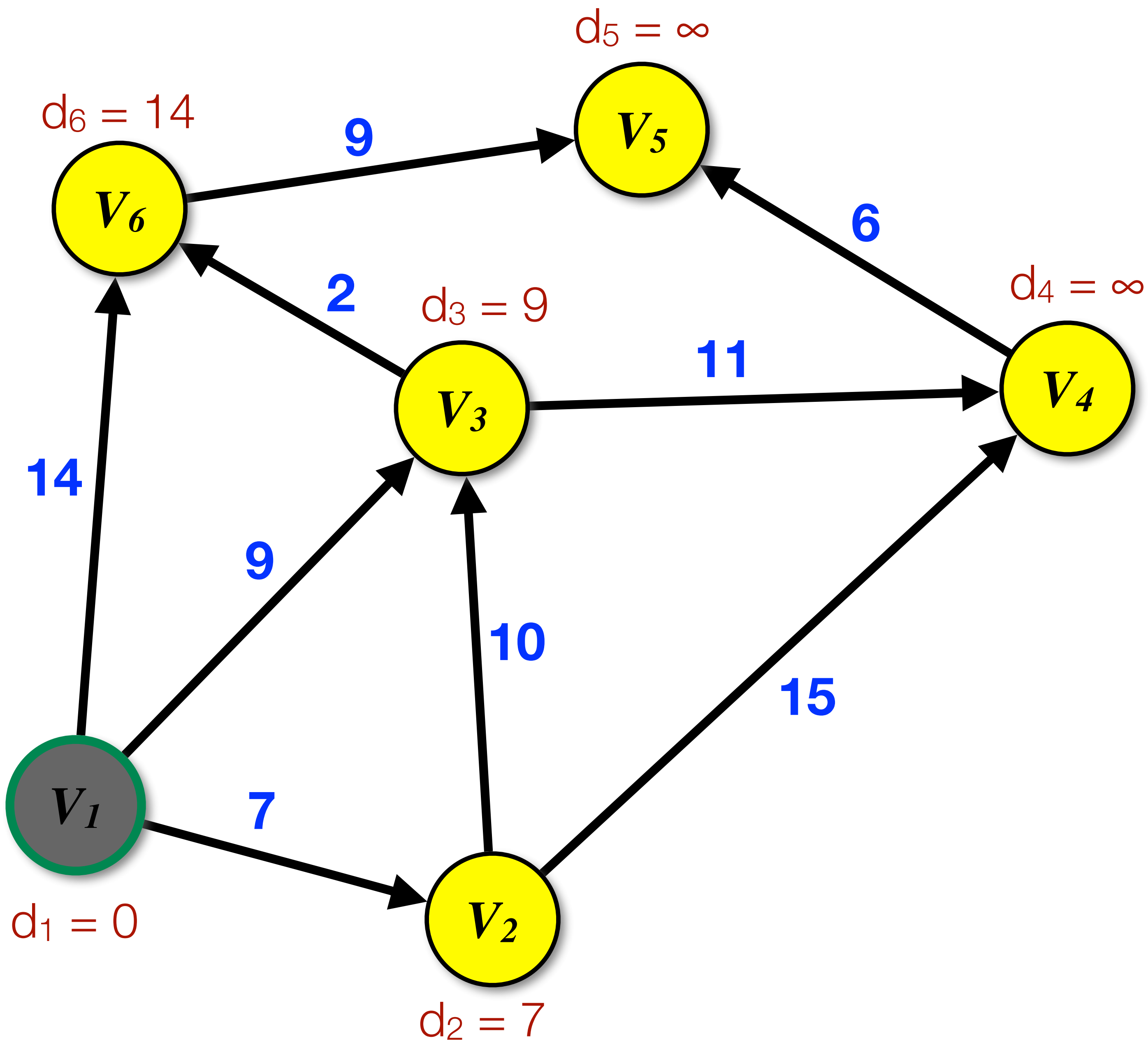
Example of Dijkstra's Algorithm



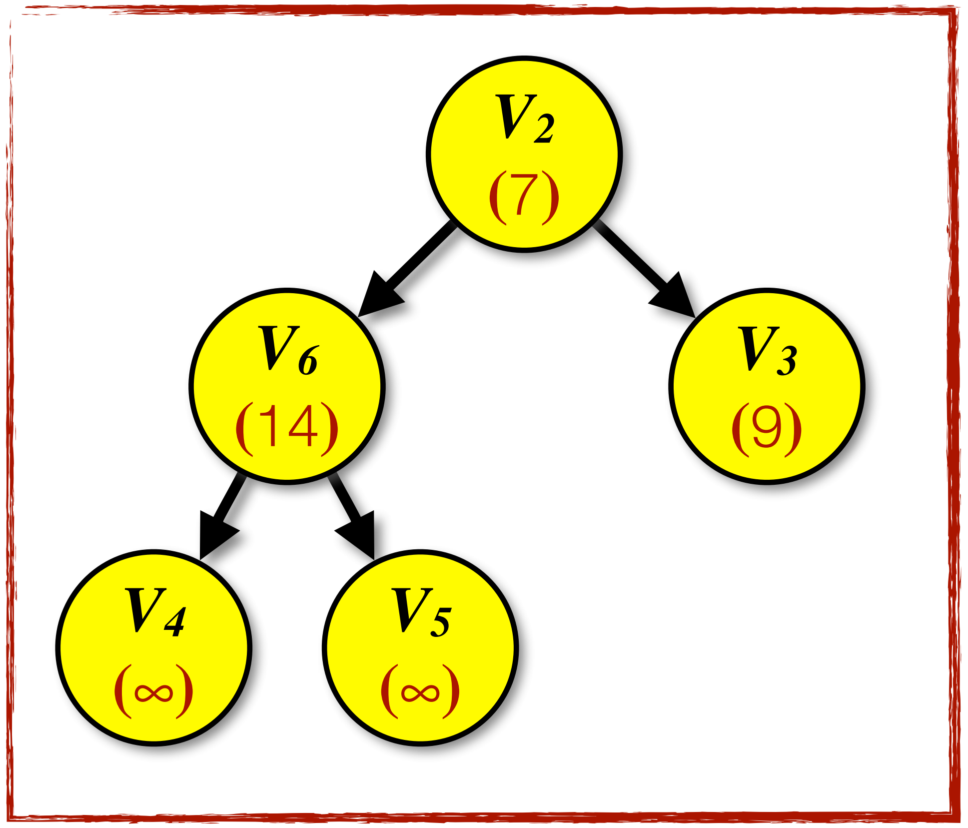
- When the distance of a node is updated, change the distance value in the minHeap and adjust the heap



Example of Dijkstra’s Algorithm

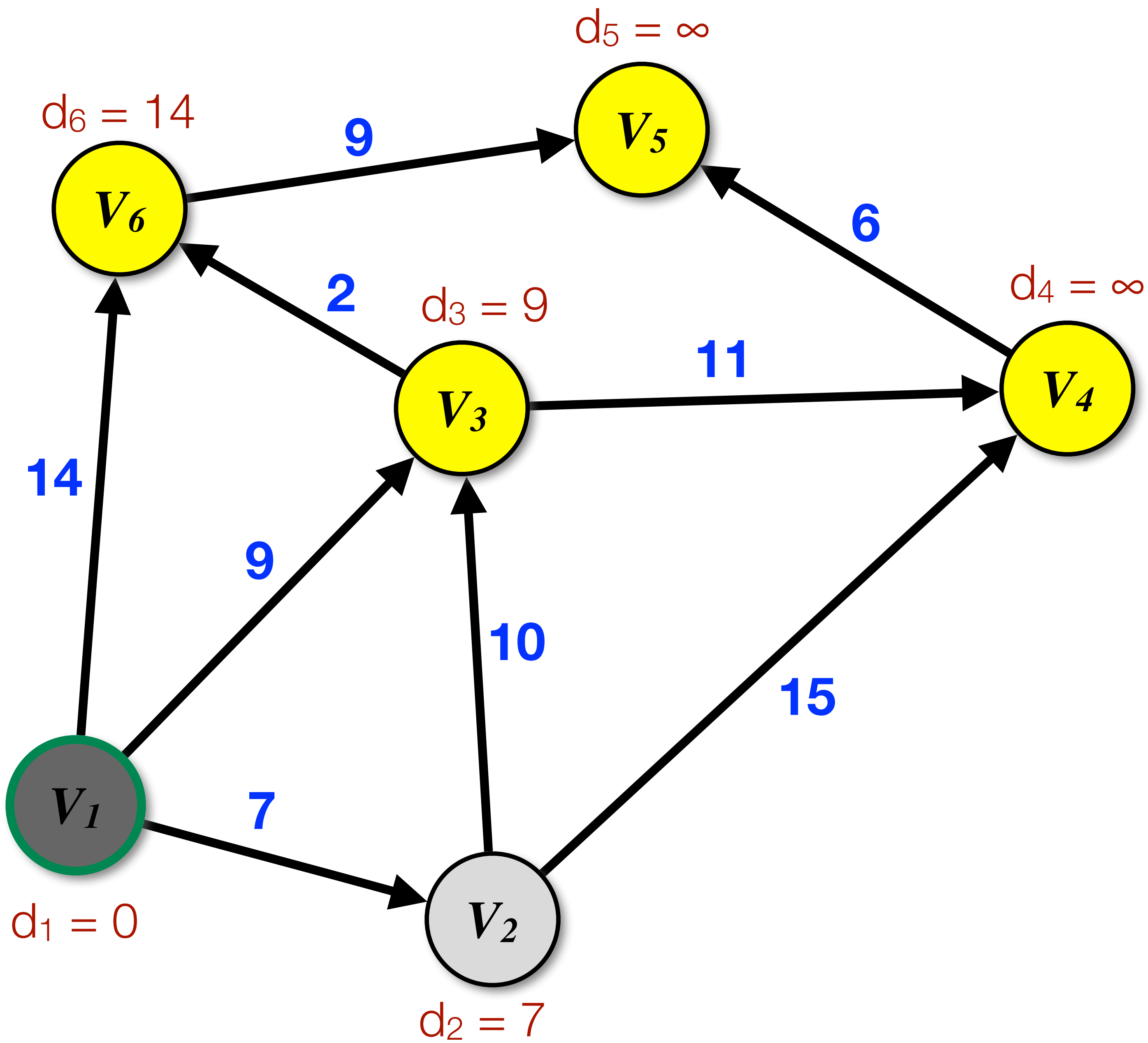


- **Done processing node *V_1***

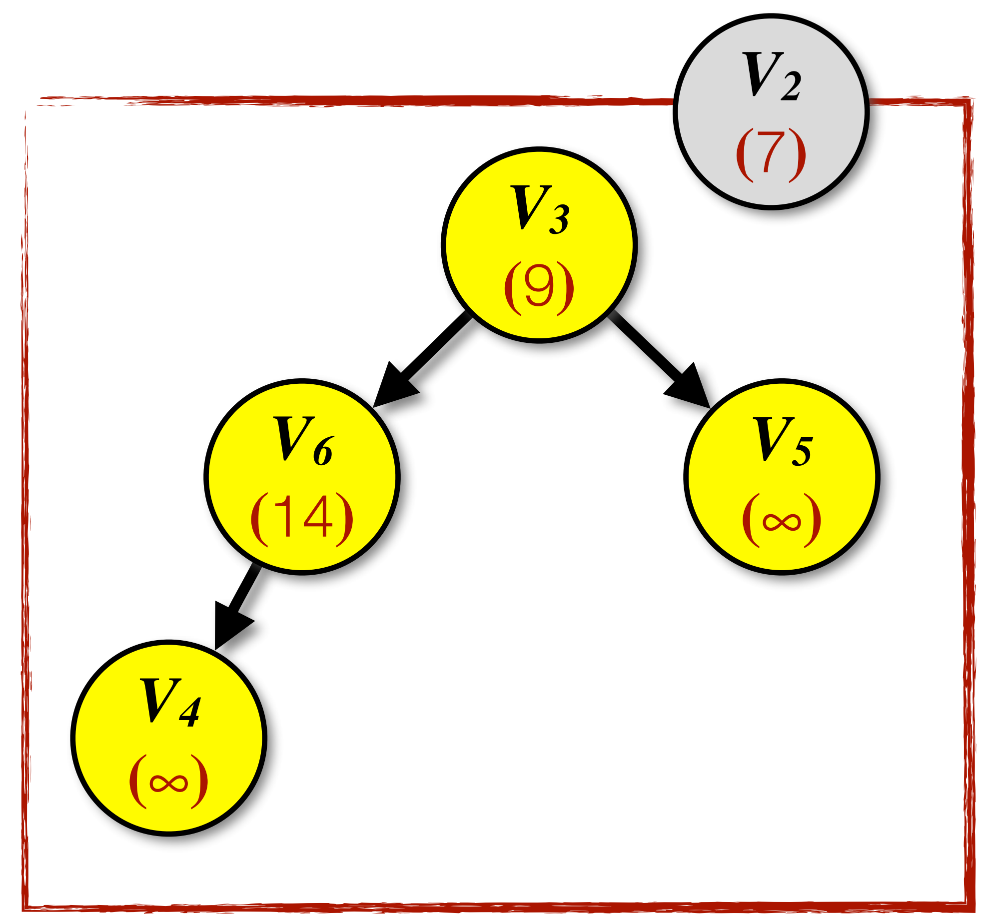


minHeap

Example of Dijkstra's Algorithm

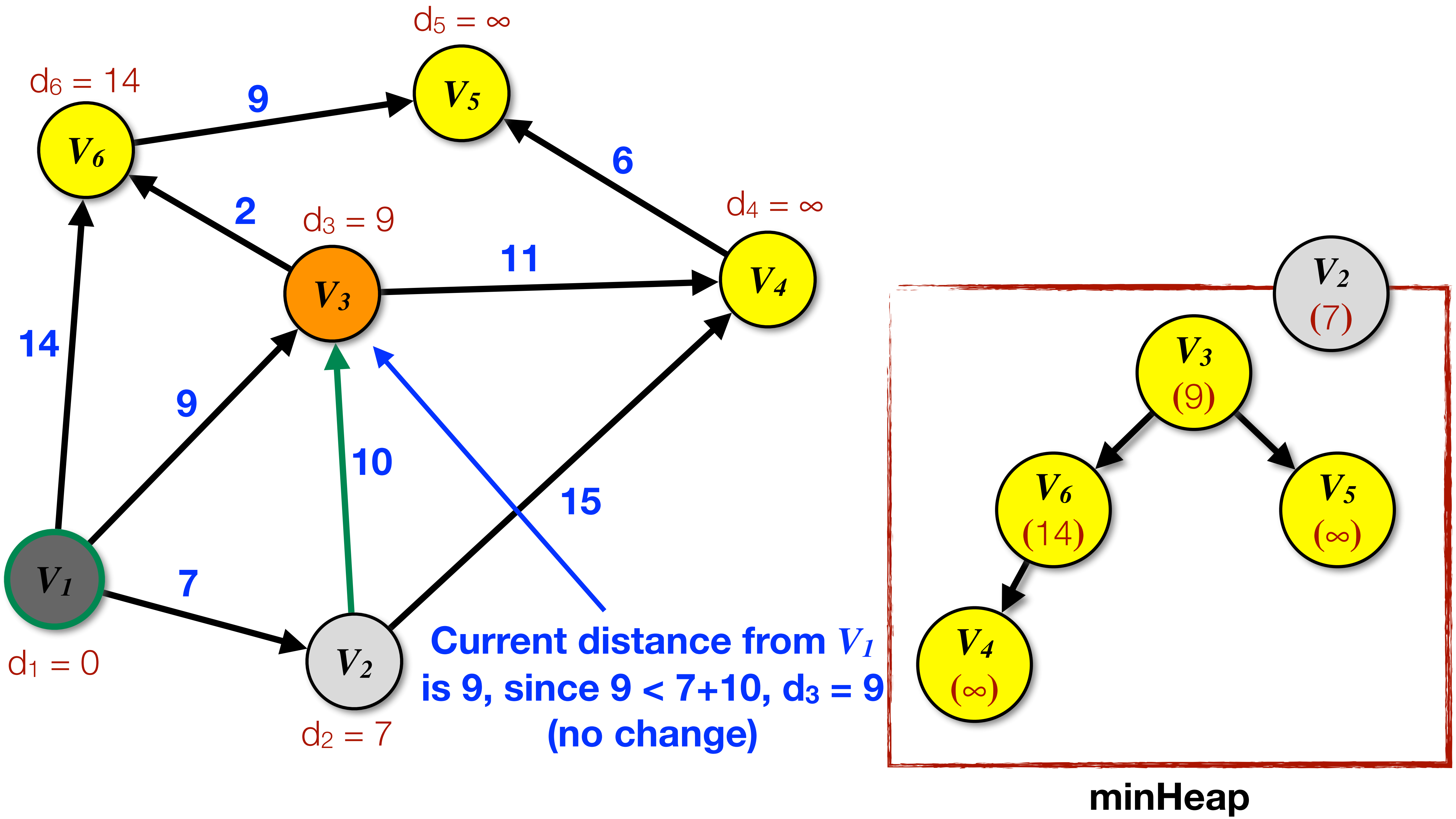


- **Pull the next minimum node from the minHeap and process it**

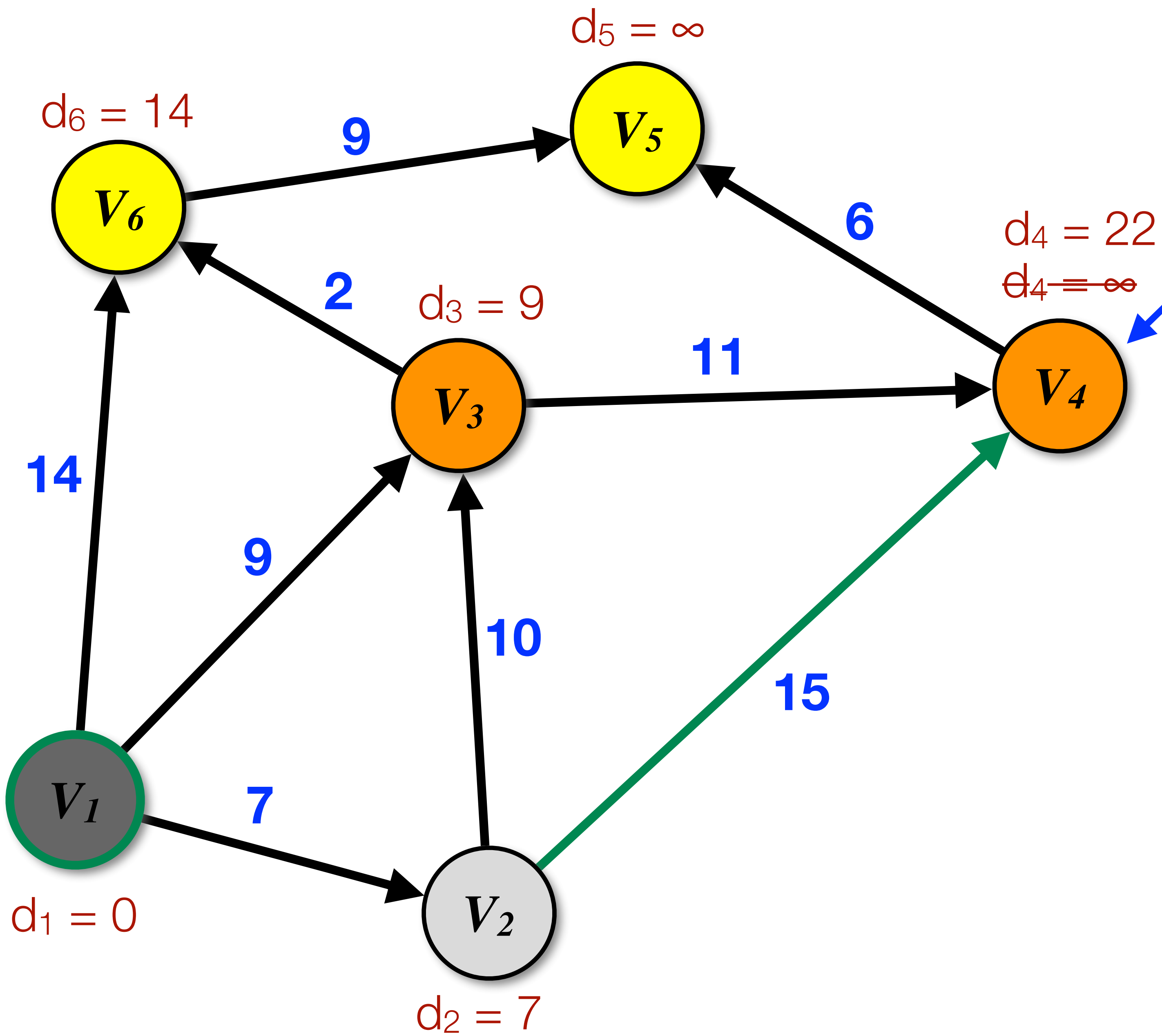


minHeap

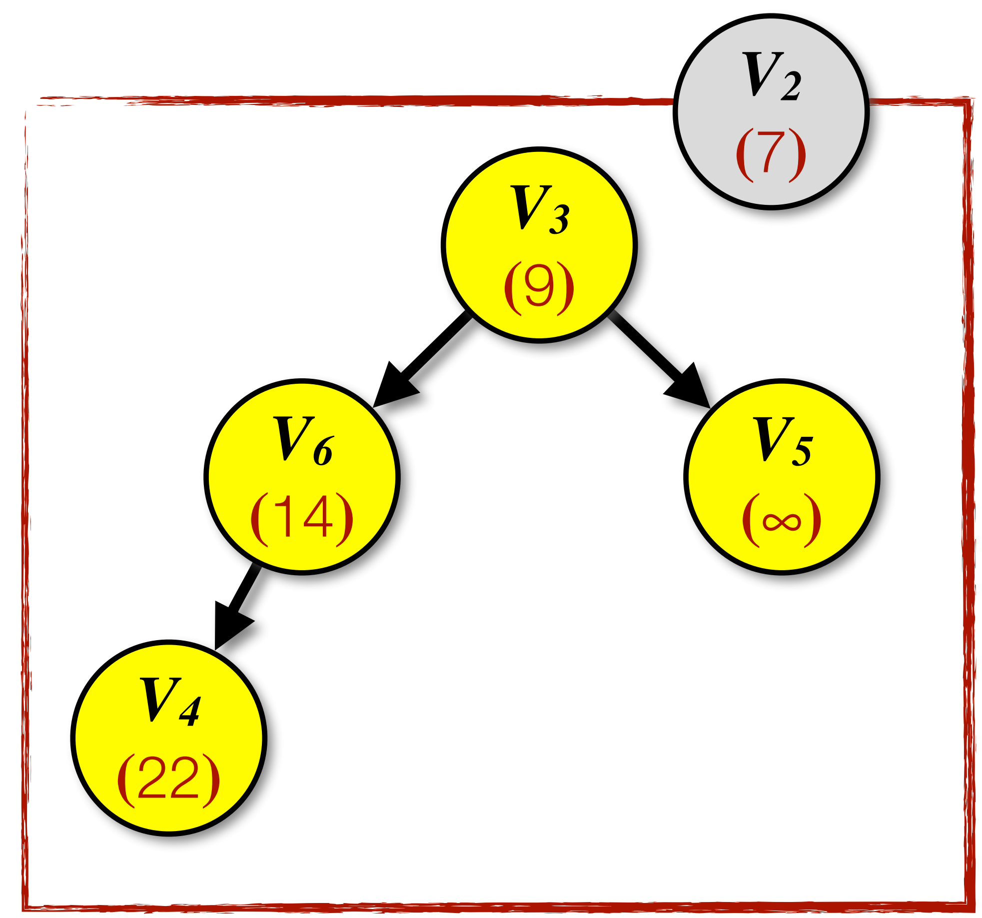
Example of Dijkstra's Algorithm



Example of Dijkstra's Algorithm

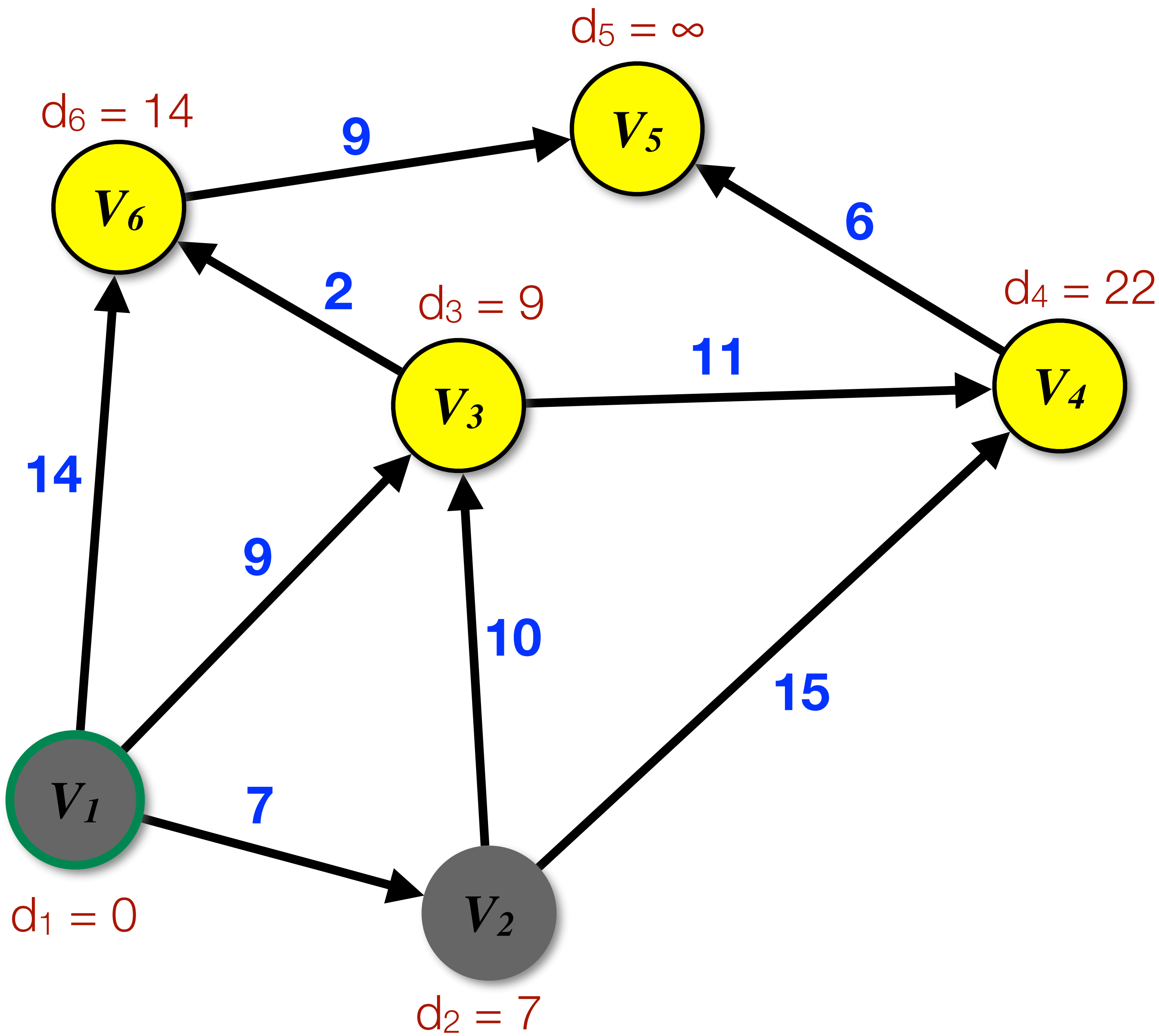


Current distance from V_1 is ∞, since $7+15 < \infty$, $d_4 = 22$

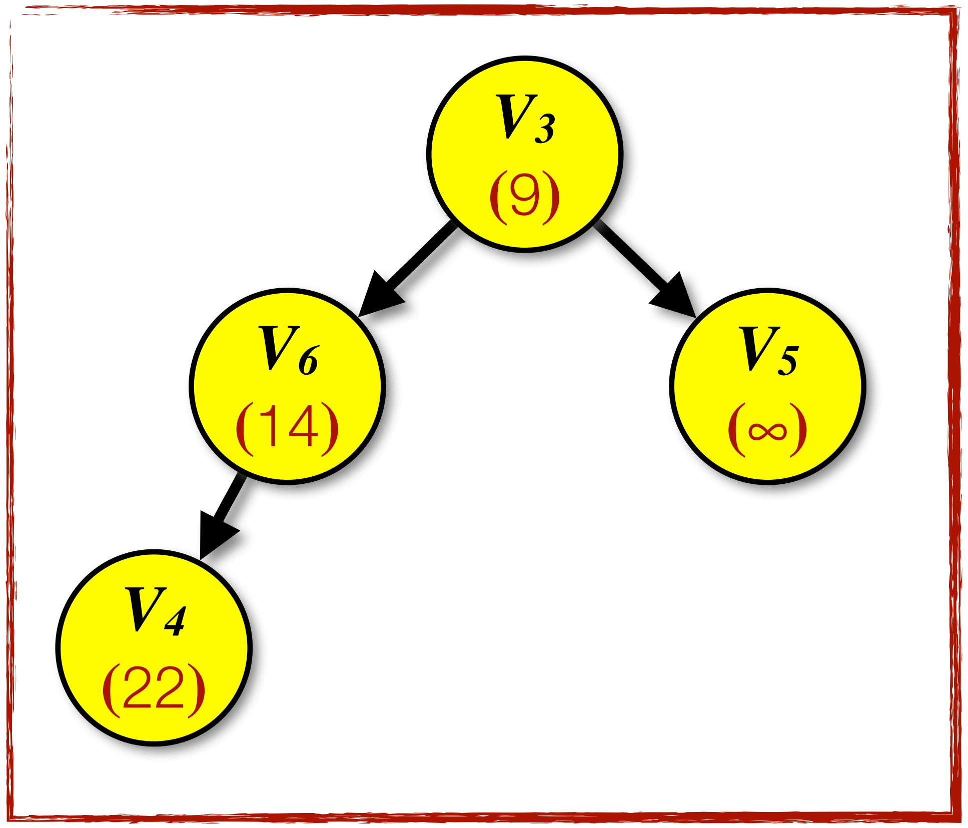


minHeap

Example of Dijkstra's Algorithm

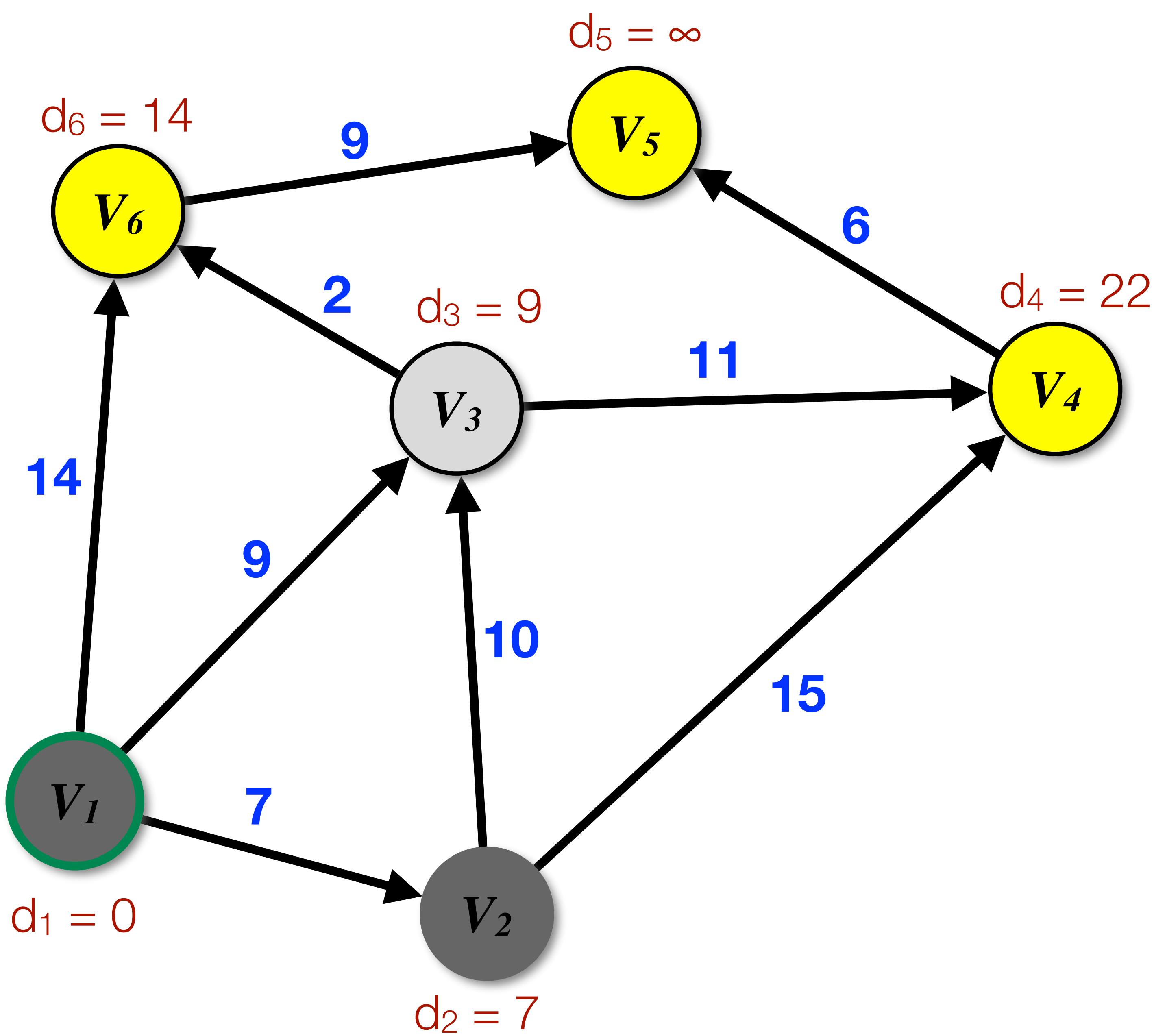


- **Done processing node V_2**

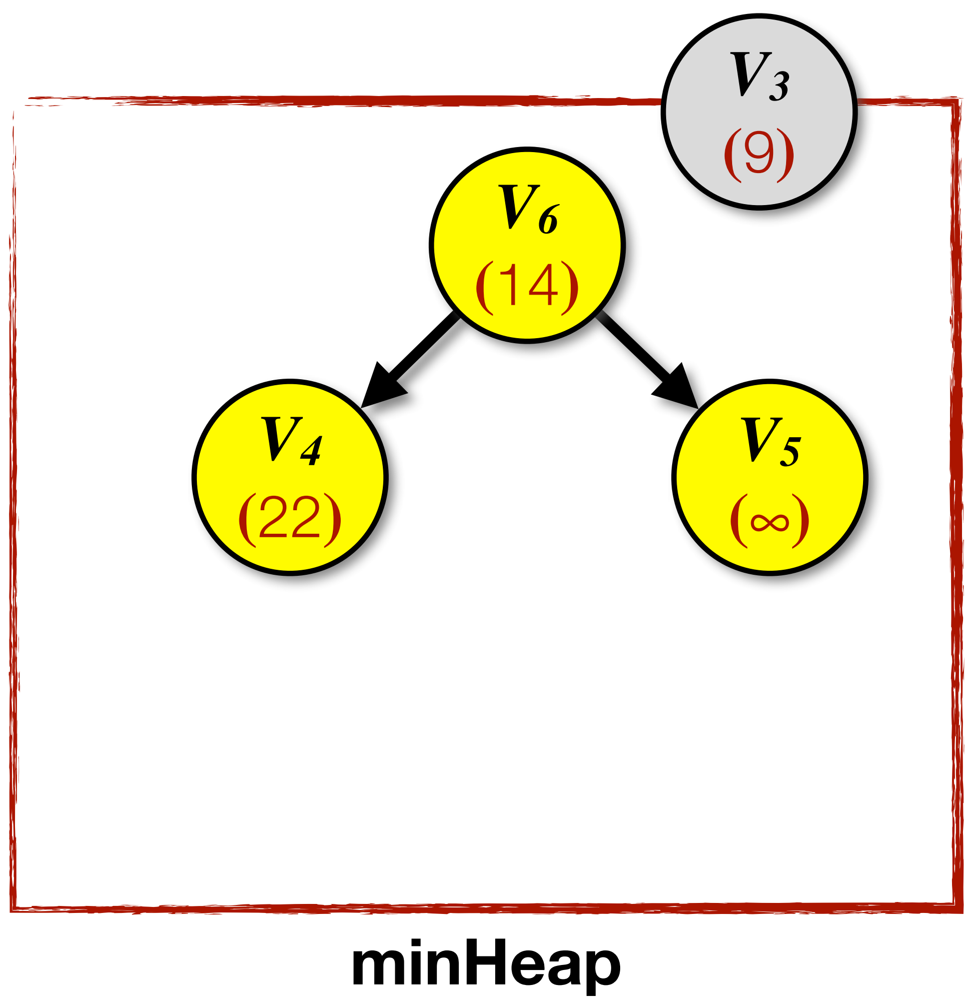


minHeap

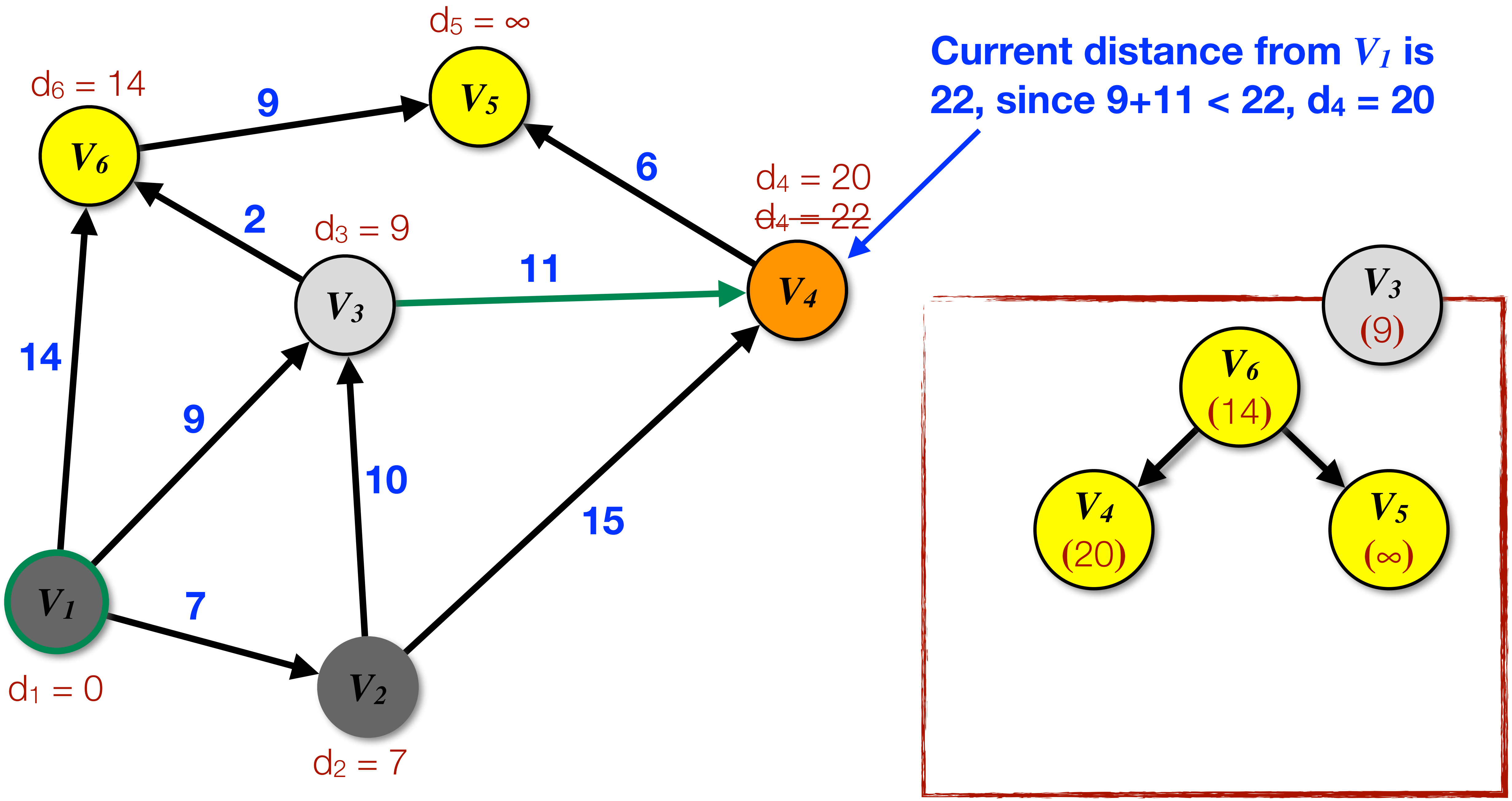
Example of Dijkstra's Algorithm



- **Pull the next minimum node from the minHeap and process it**



Example of Dijkstra's Algorithm

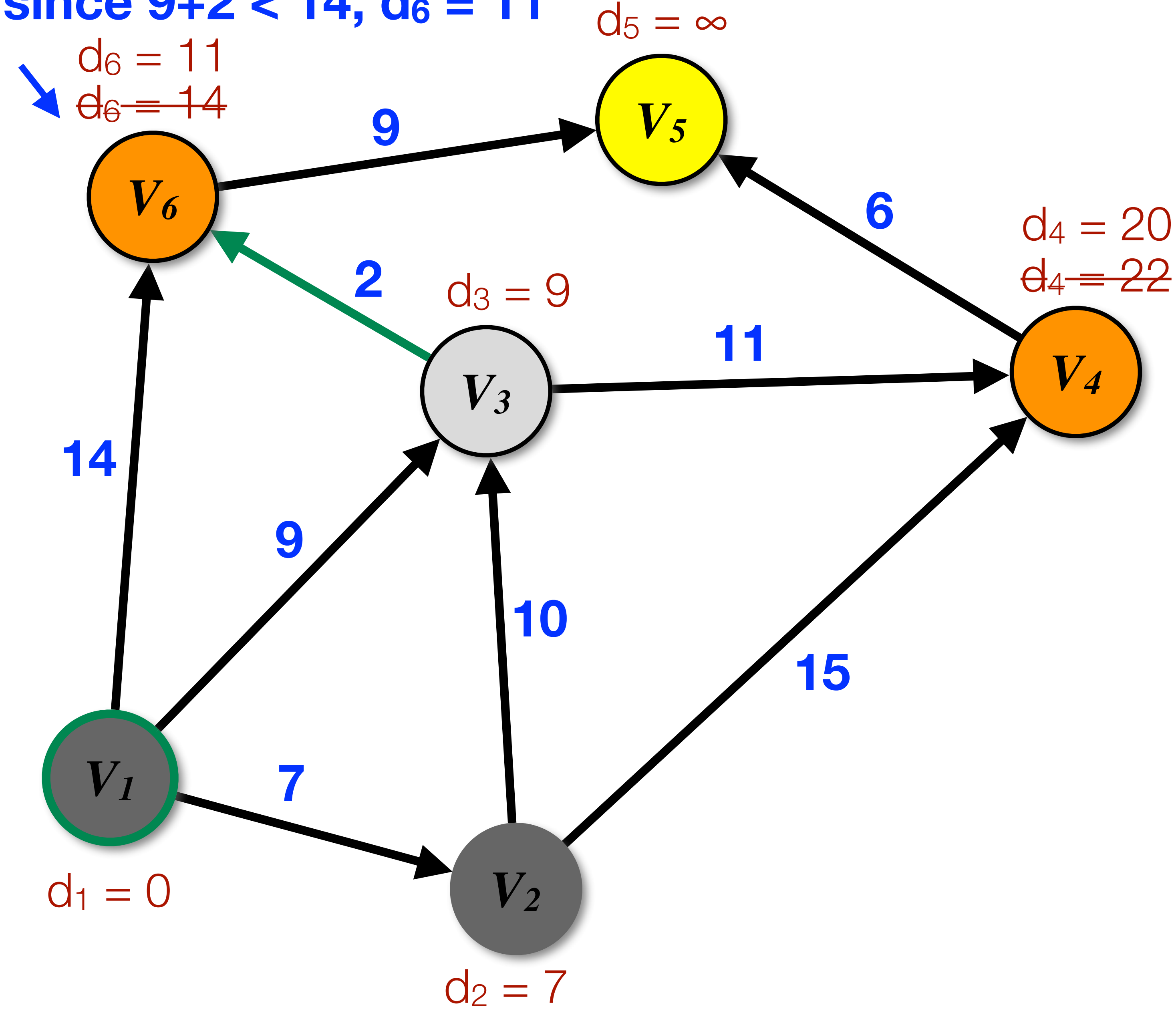


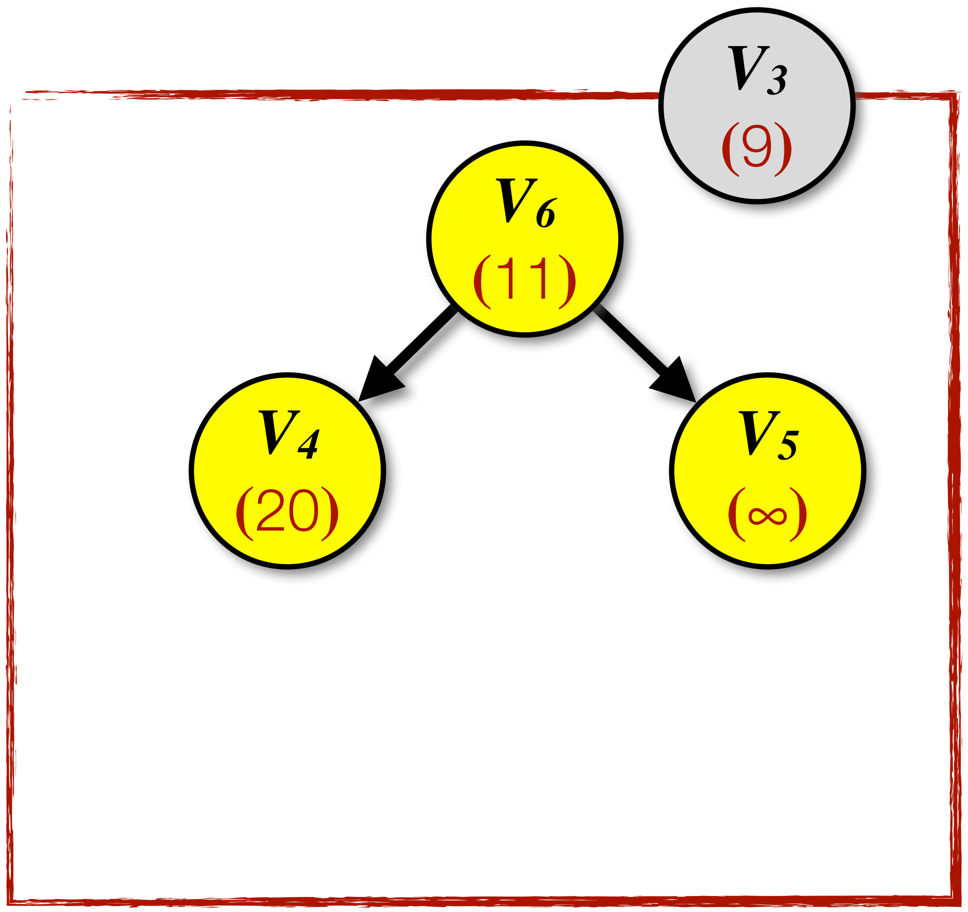
Current distance from V_1 is 22, since 9+11 < 22, d_4 = 20

minHeap

Example of Dijkstra's Algorithm

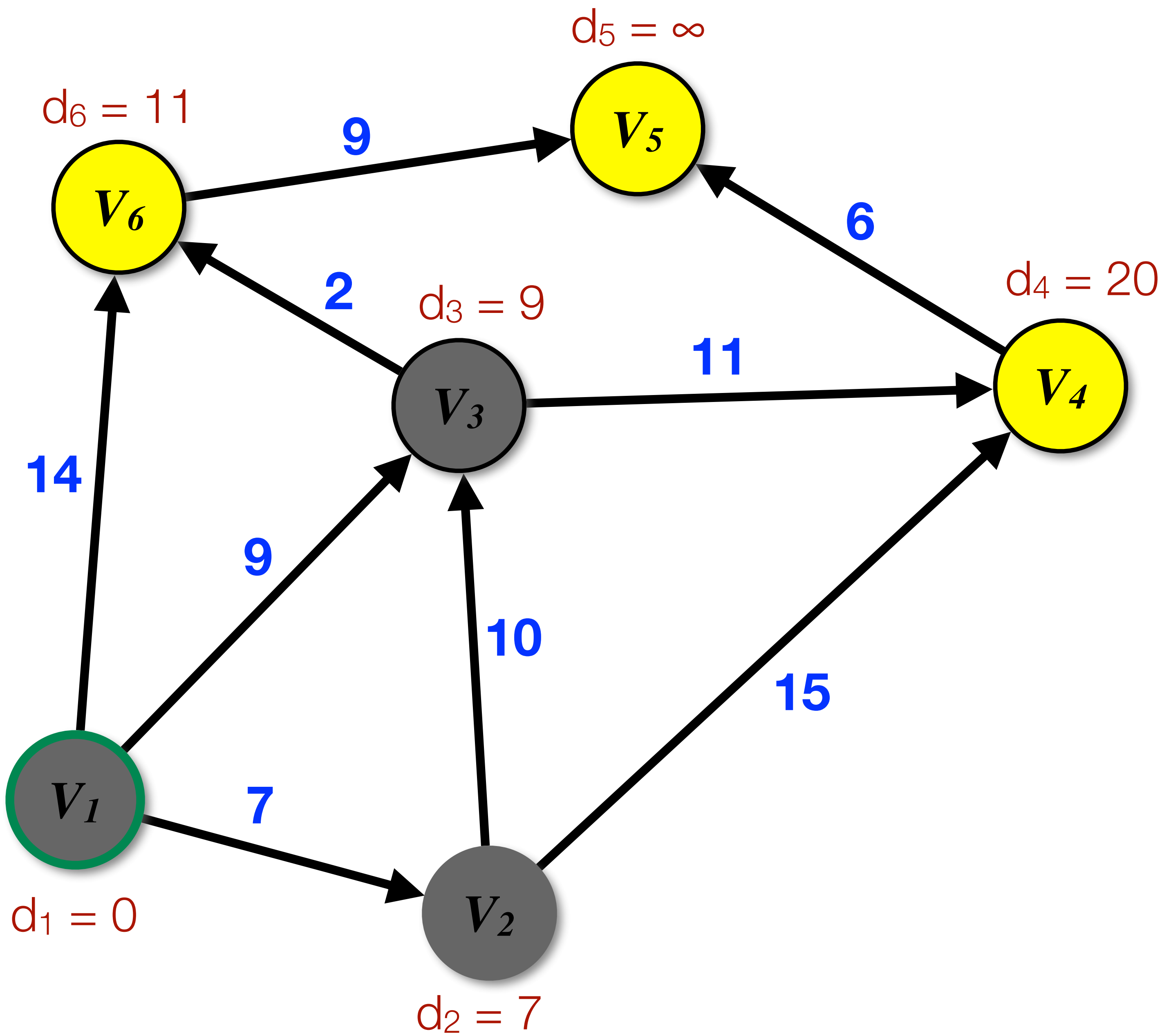
Current distance from V_1 is 14, since 9+2 < 14, d_6 = 11



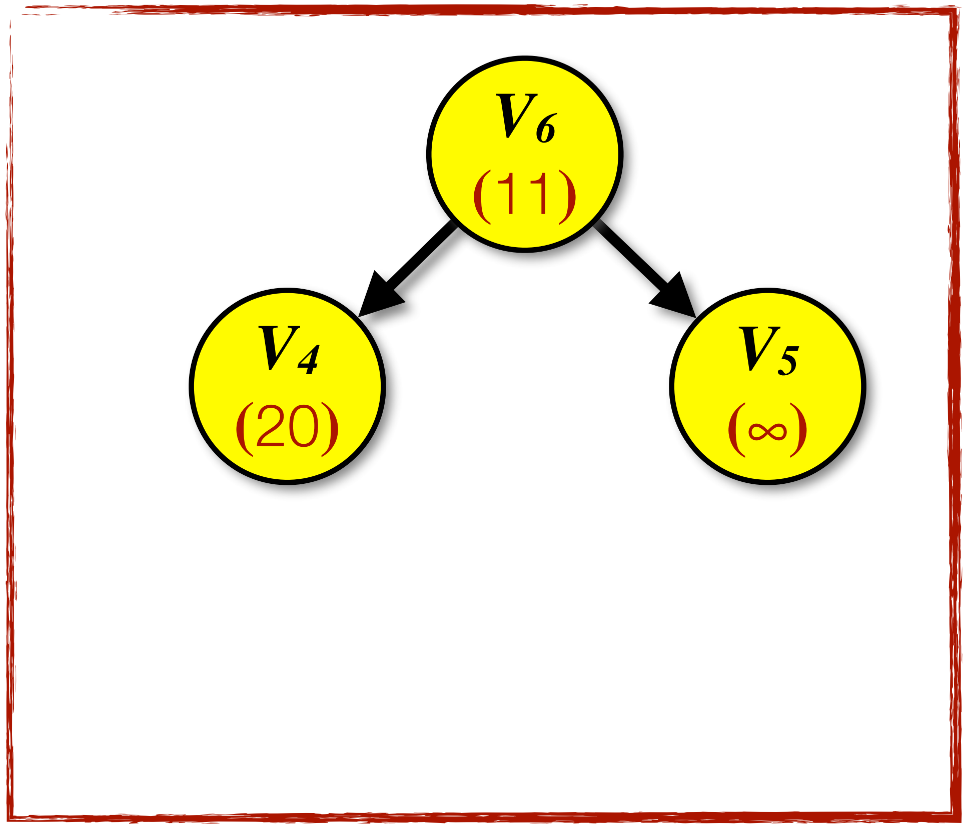


minHeap

Example of Dijkstra's Algorithm

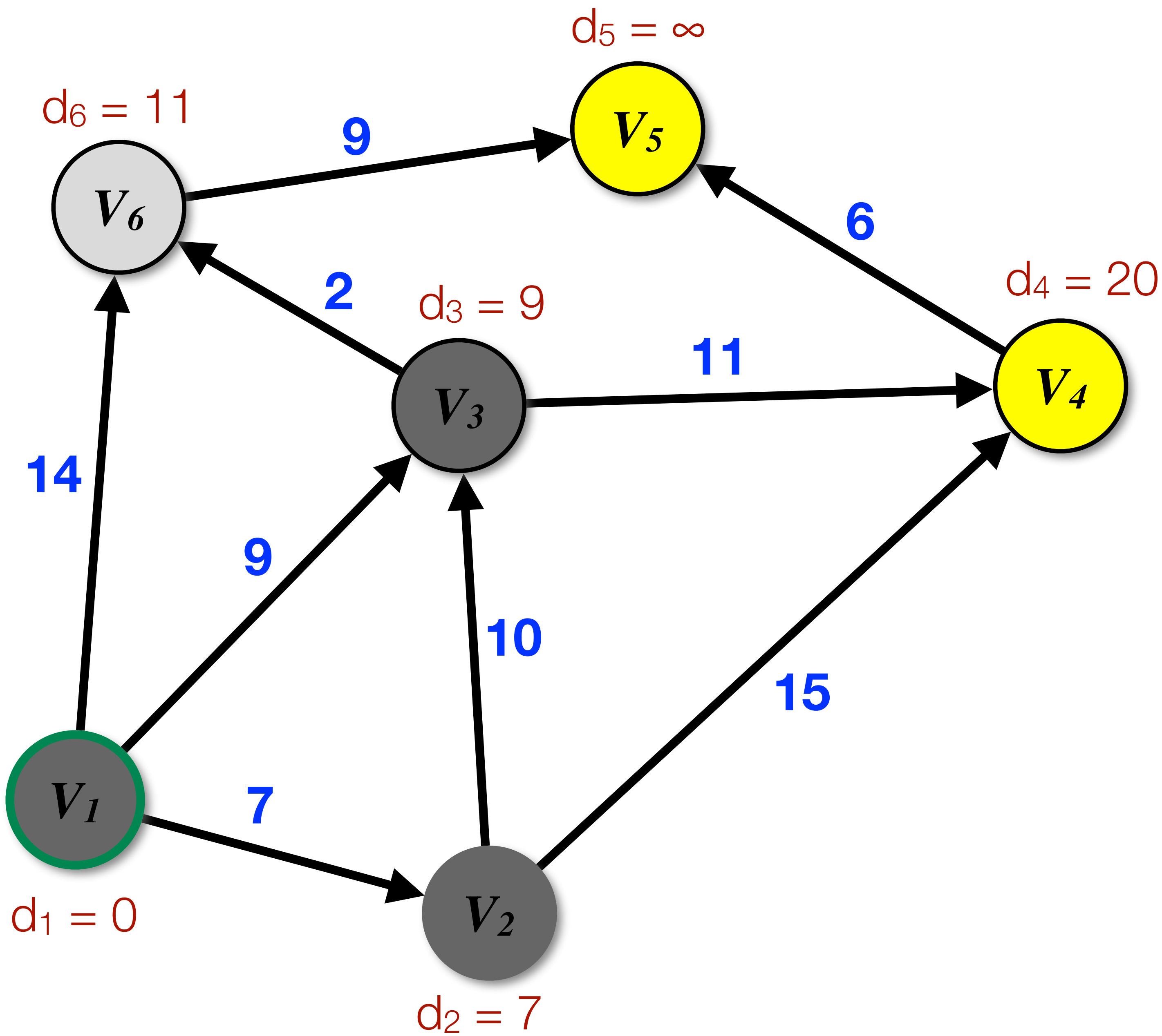


- **Done processing node V_3**

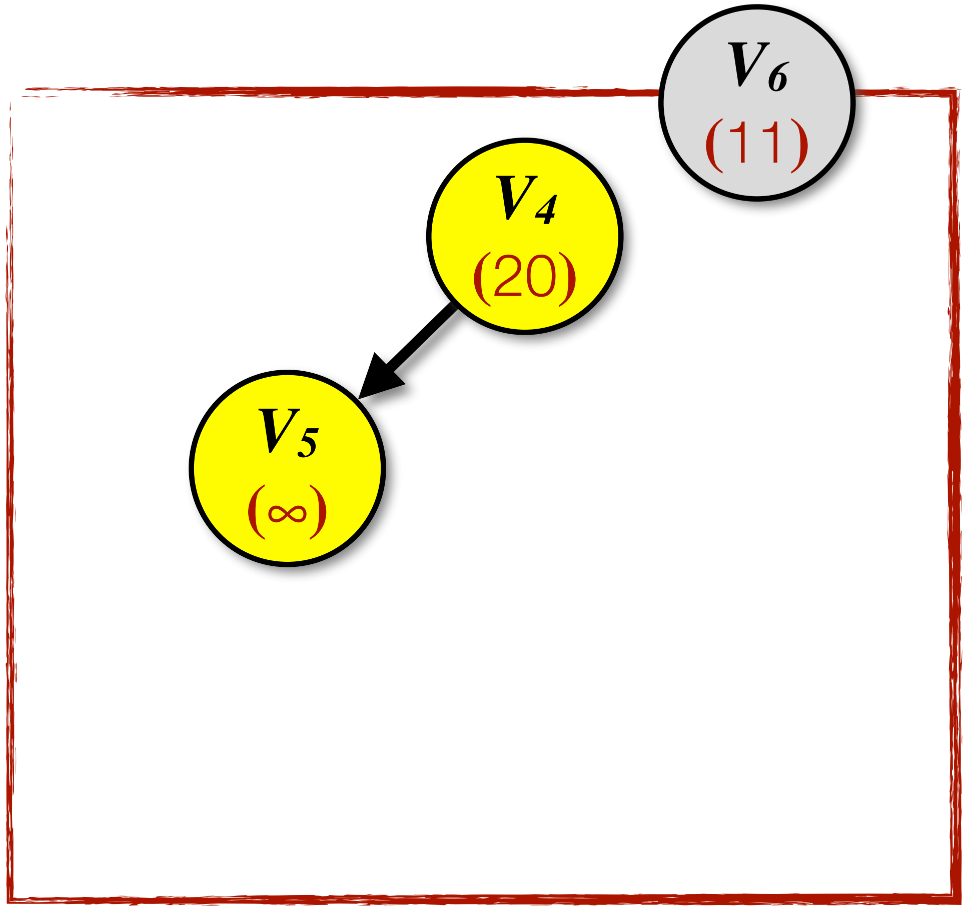


minHeap

Example of Dijkstra's Algorithm

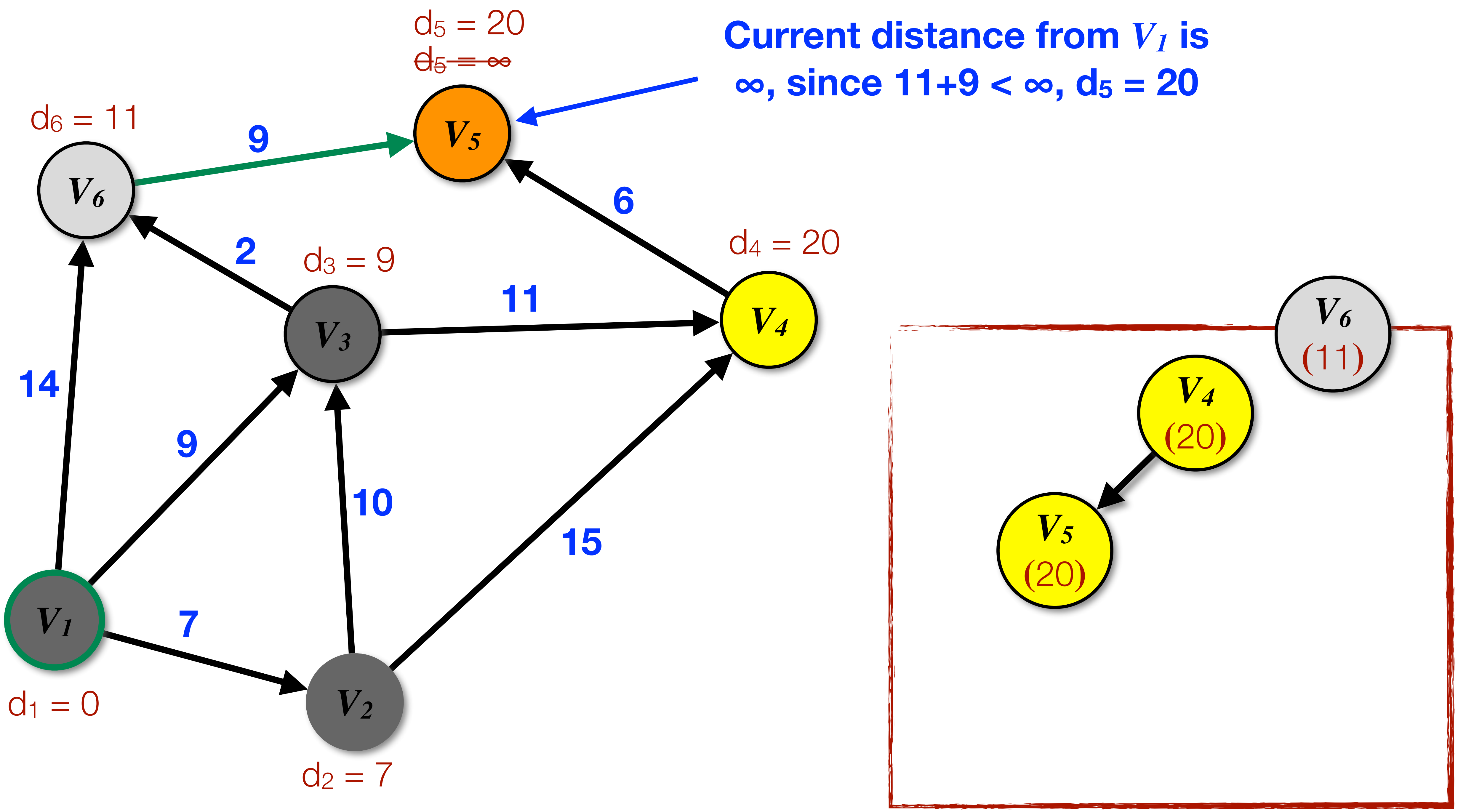


- **Pull the next minimum node from the minHeap and process it**



minHeap

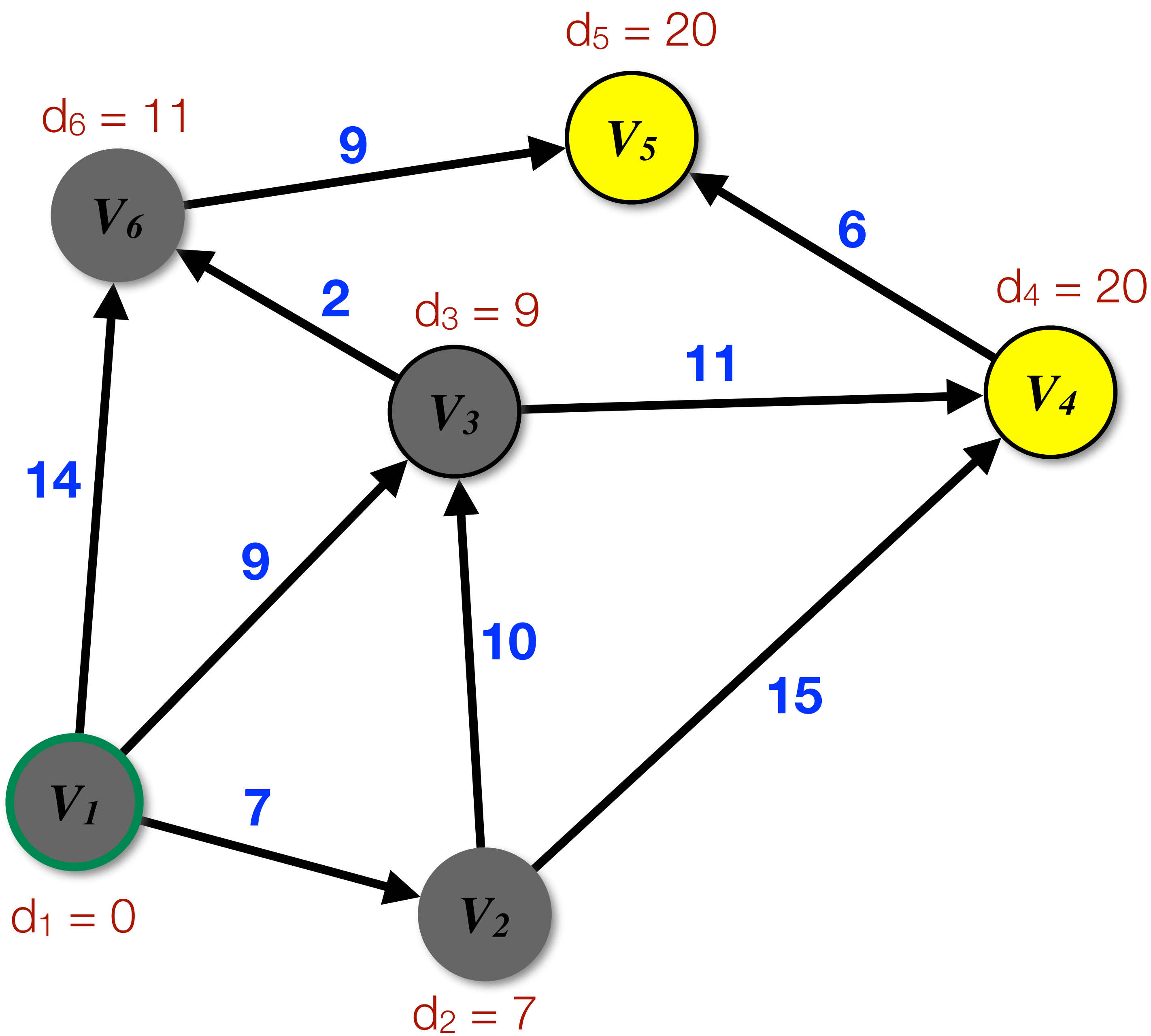
Example of Dijkstra's Algorithm



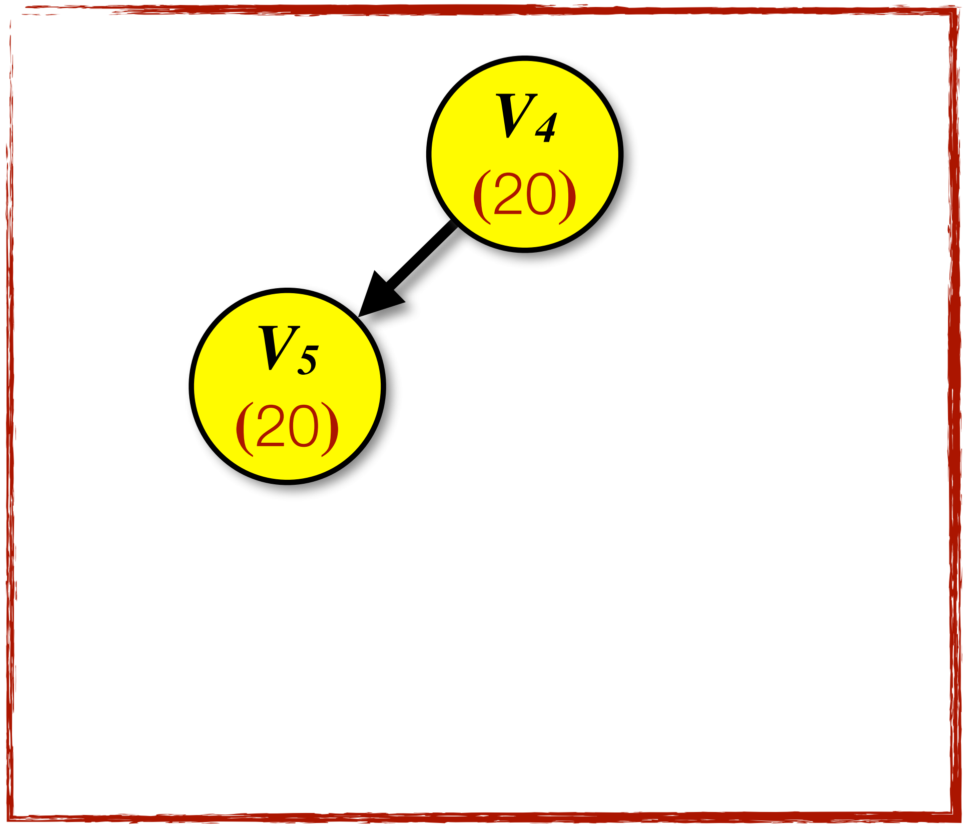
Current distance from V_1 is ∞, since 11+9 < ∞, d_5 = 20

minHeap

Example of Dijkstra's Algorithm

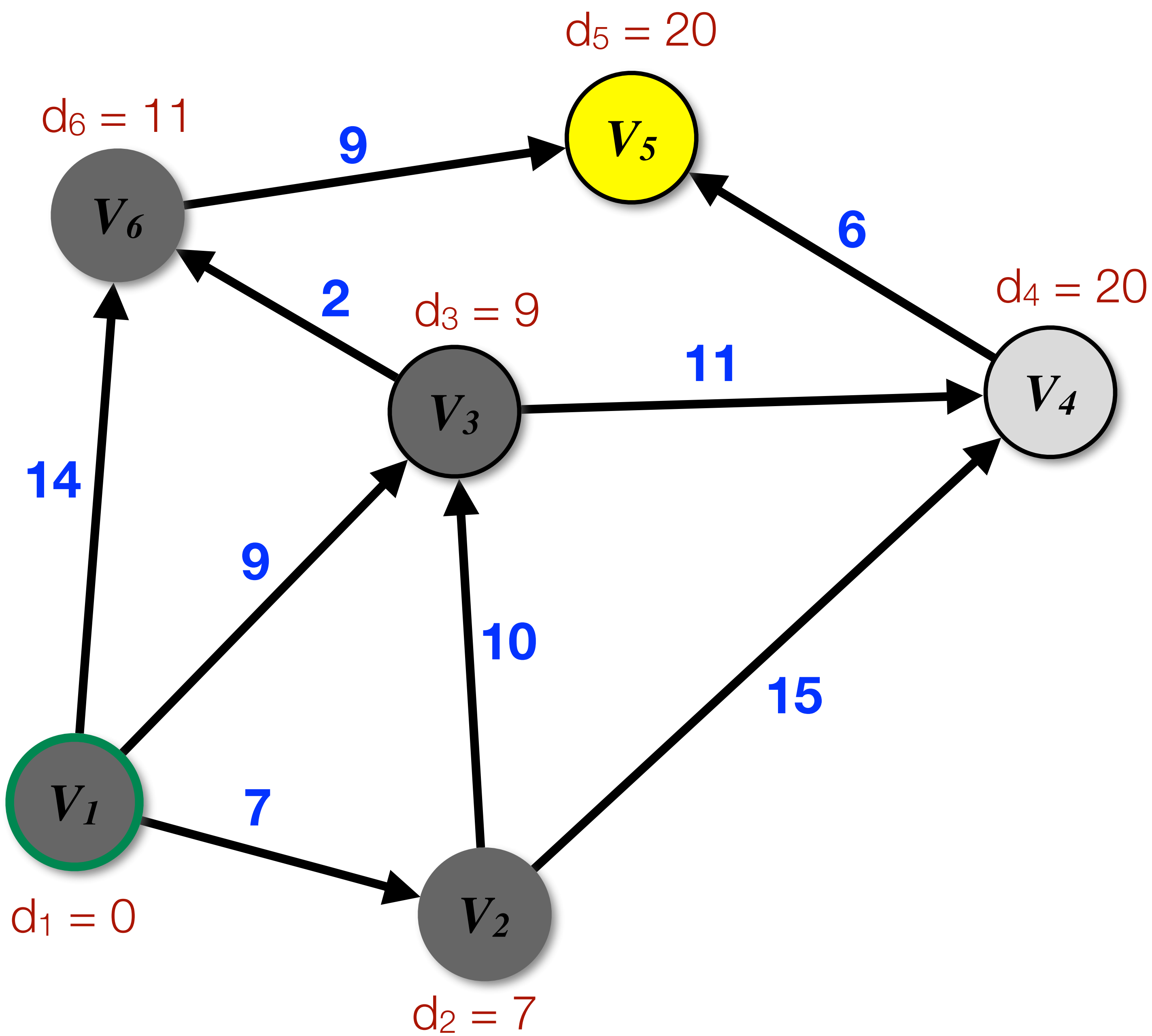


- Done processing node V_6

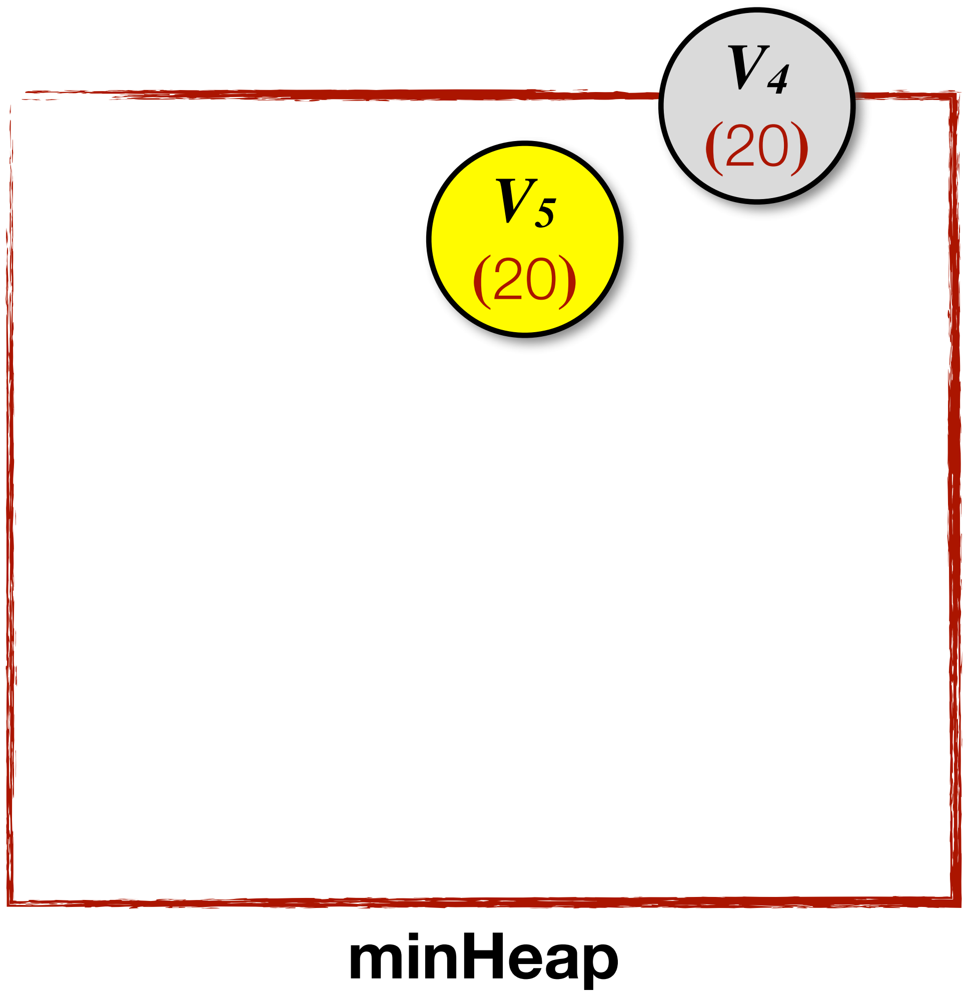


minHeap

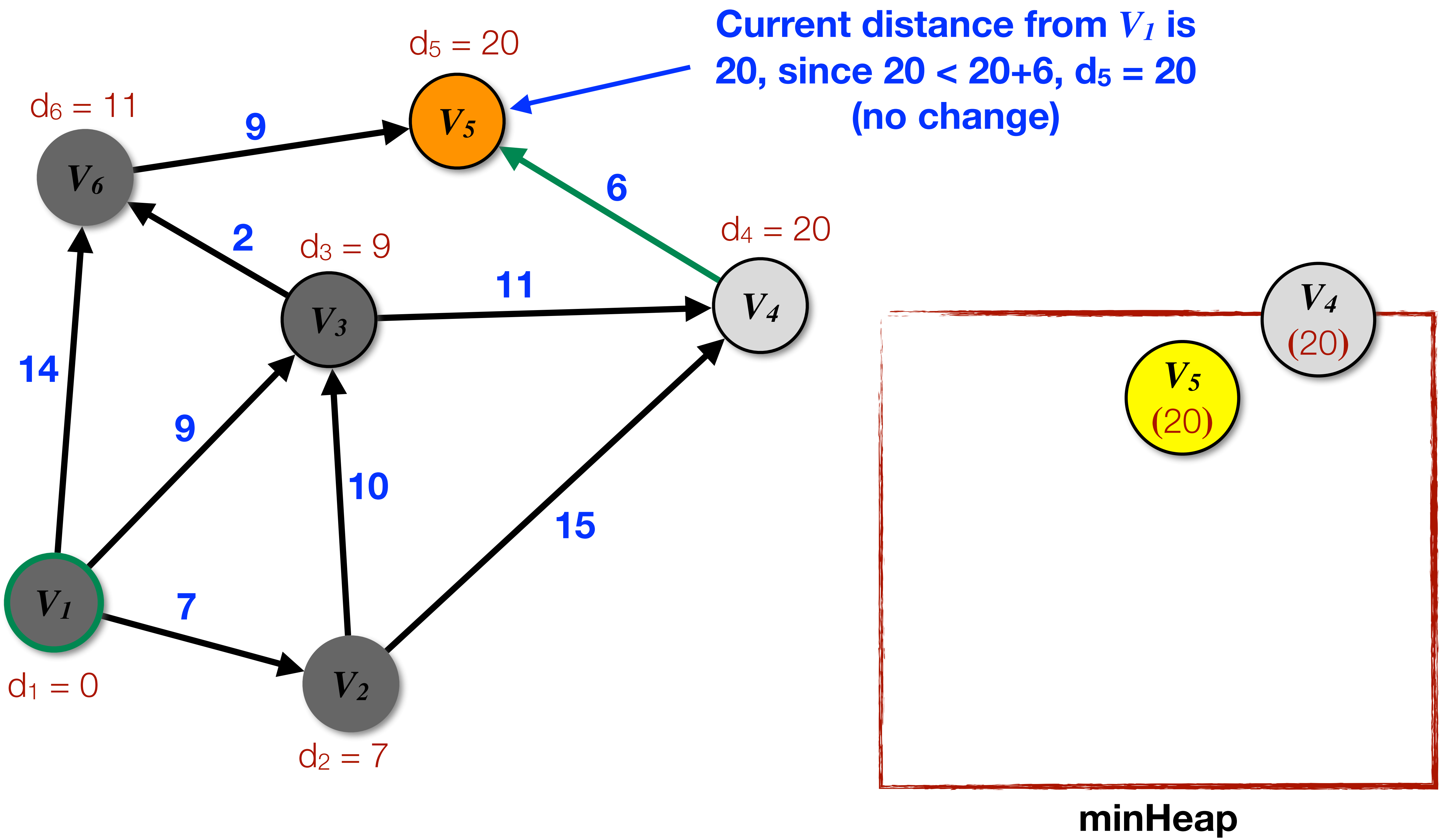
Example of Dijkstra's Algorithm



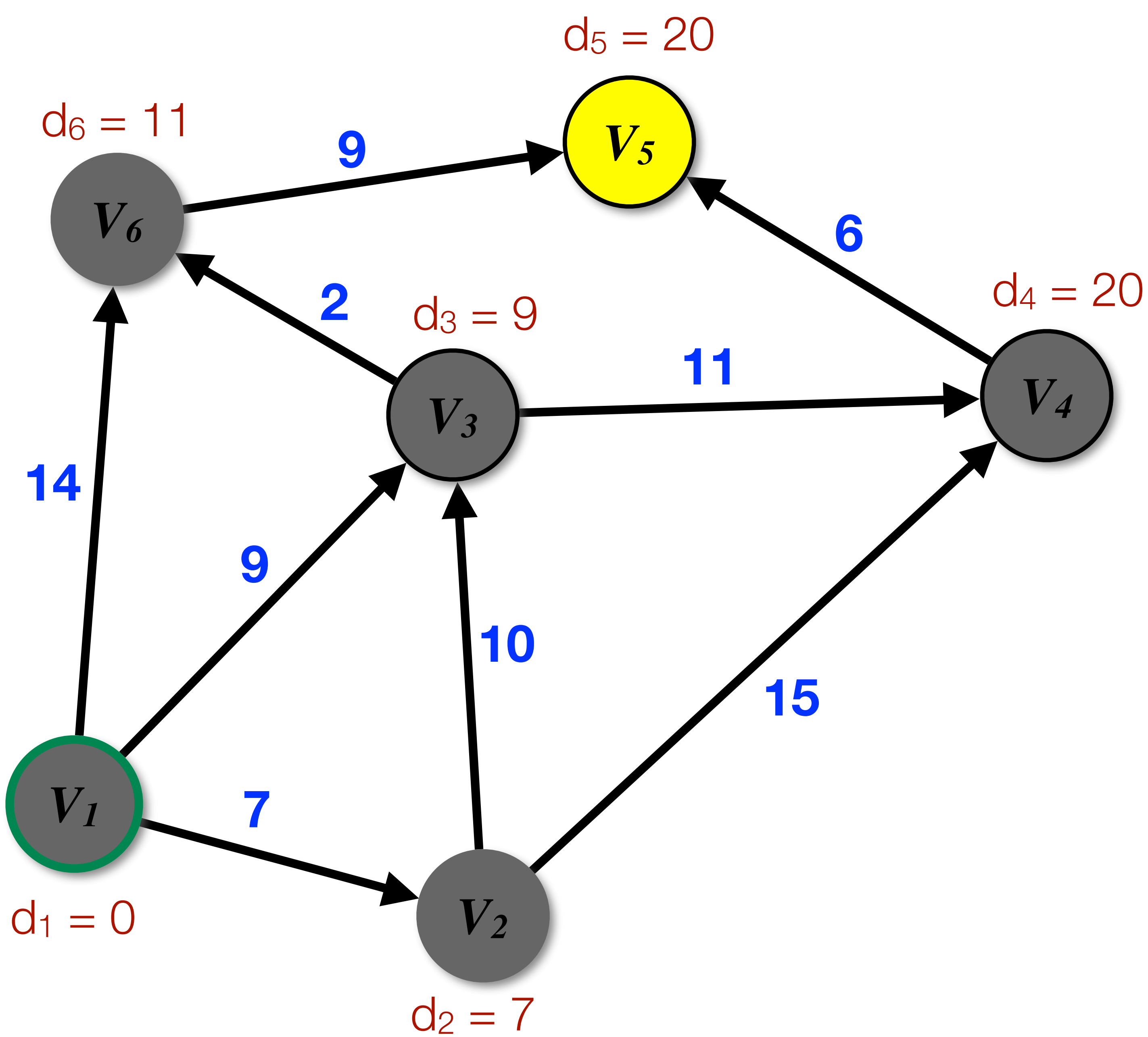
- **Pull the next minimum node from the minHeap and process it**



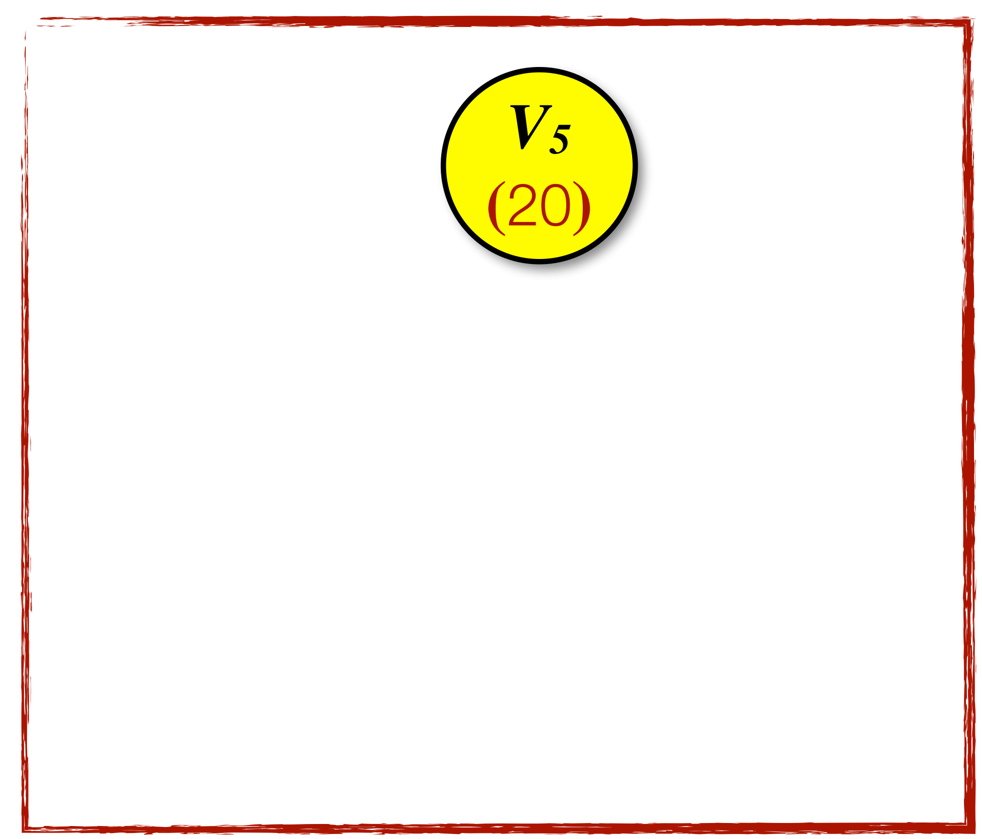
Example of Dijkstra's Algorithm



Example of Dijkstra's Algorithm

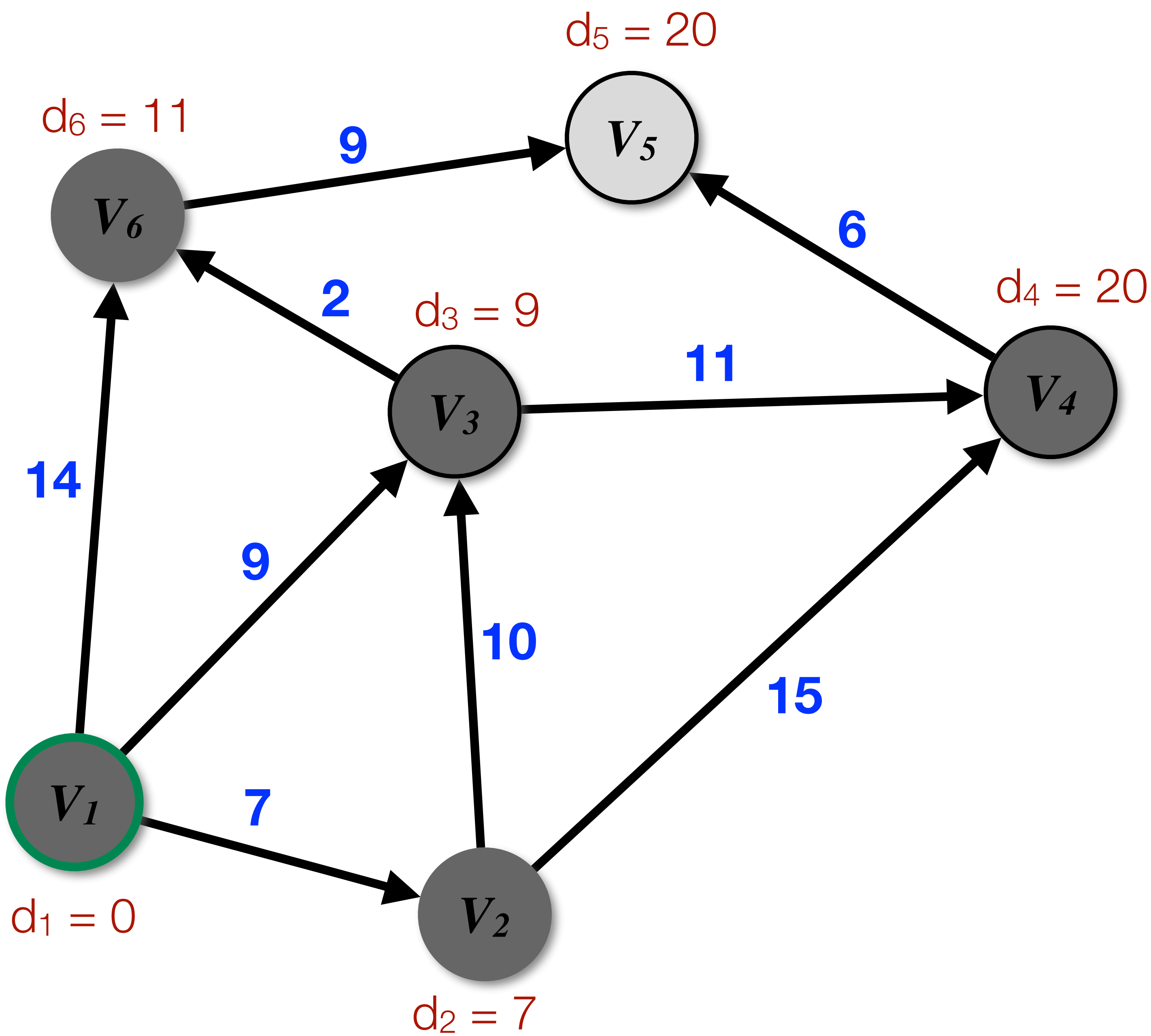


- Done processing node V_4

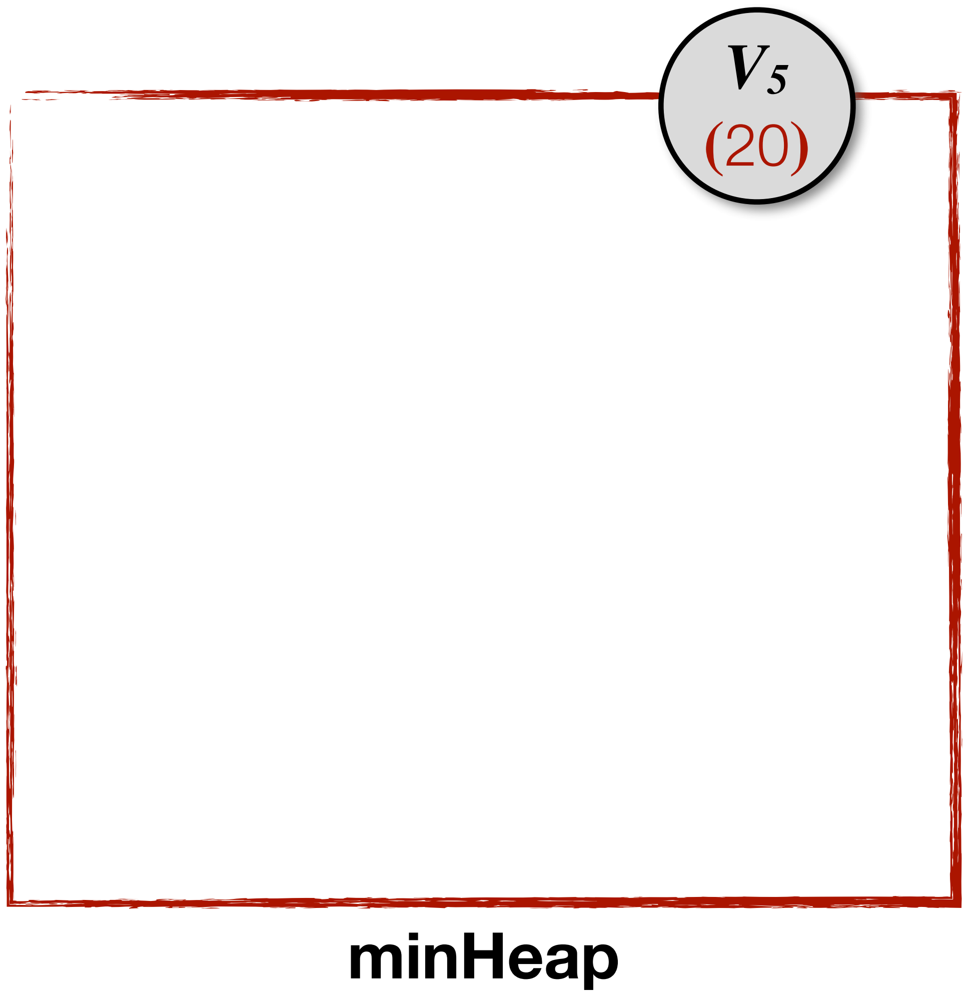


minHeap

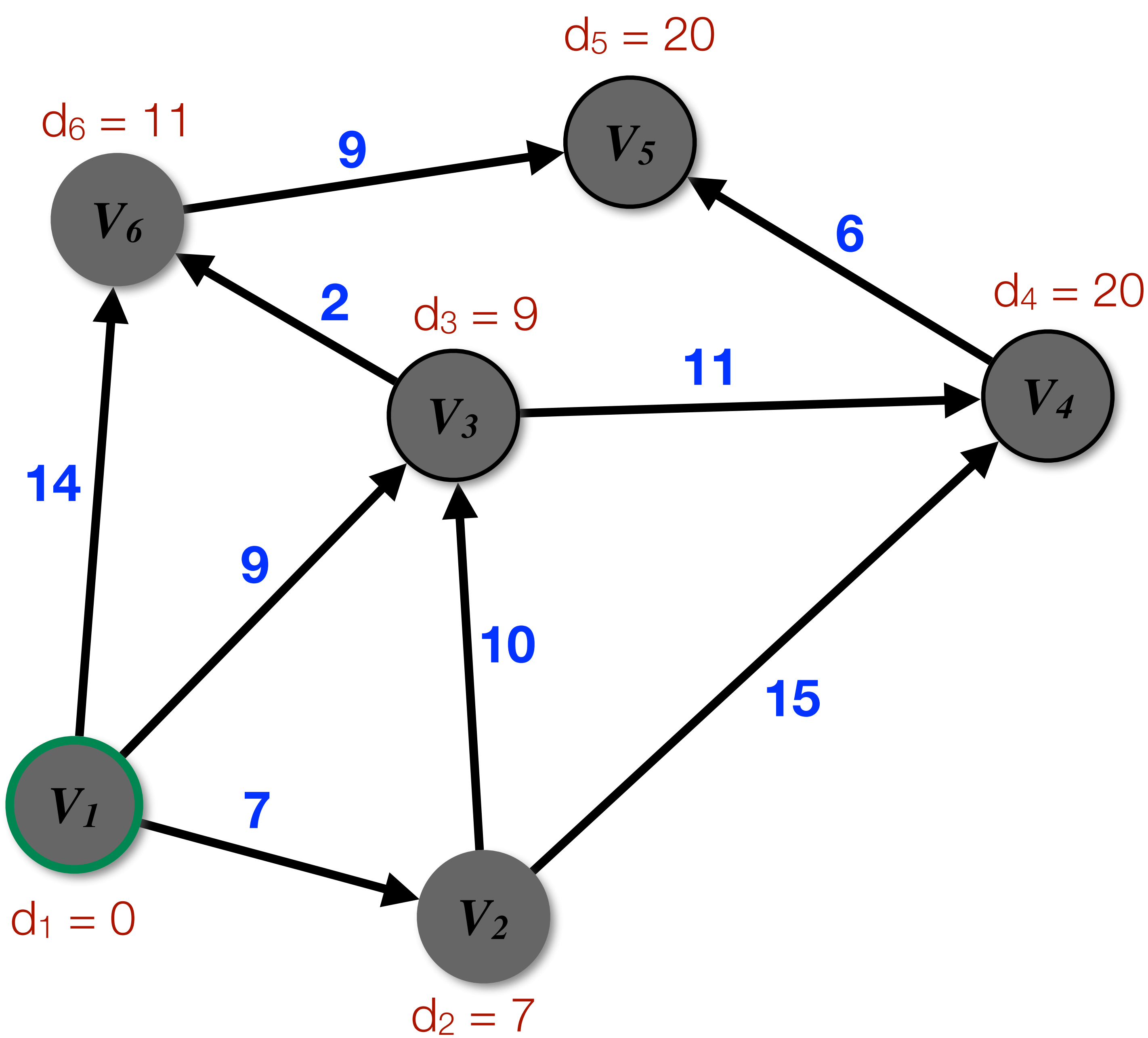
Example of Dijkstra's Algorithm



- **Pull the next minimum node from the minHeap and process it**



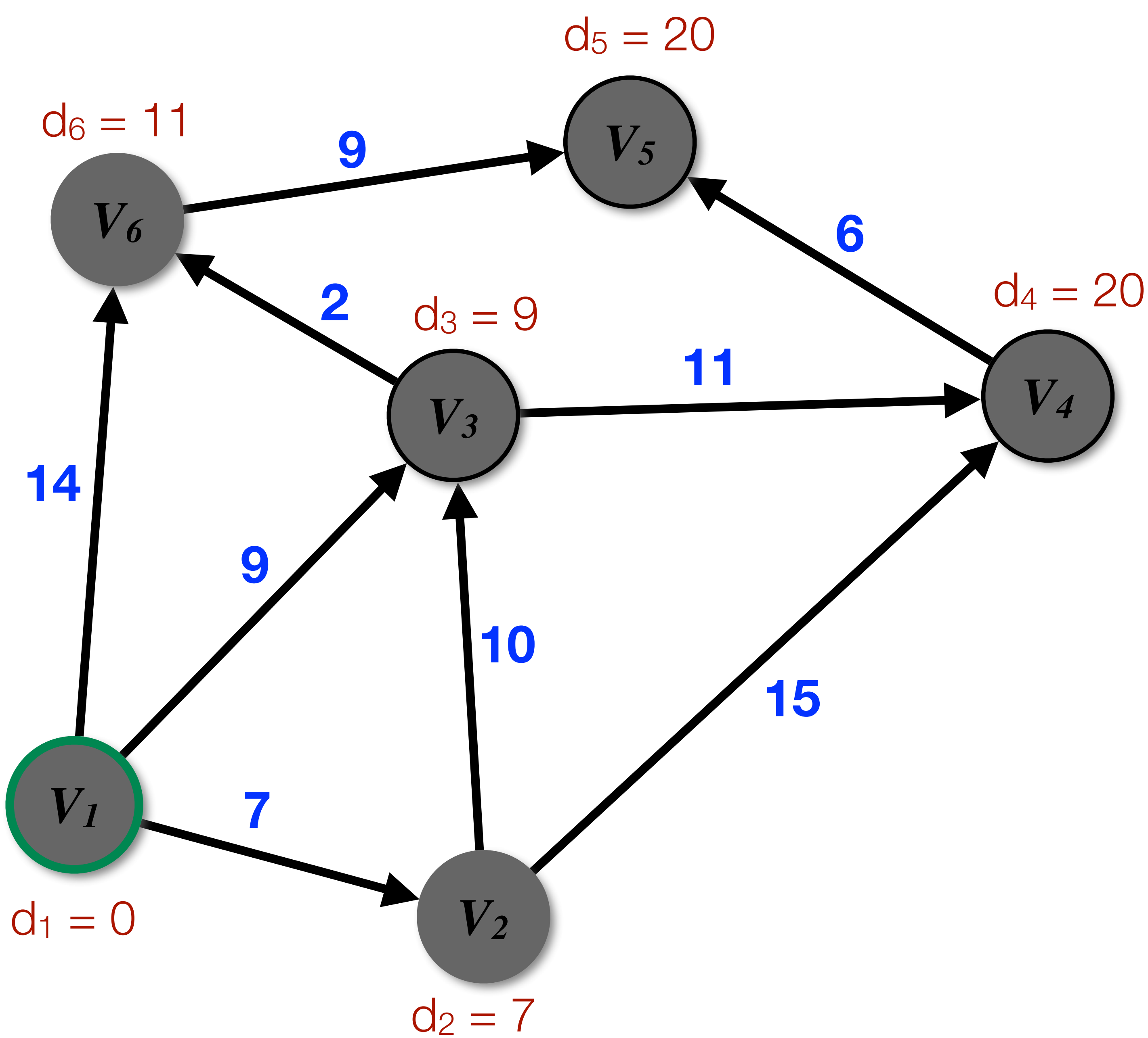
Example of Dijkstra's Algorithm



- **Node V_5 has no adjacent nodes and is therefore complete**

minHeap

Example of Dijkstra's Algorithm



- No more nodes exist in the minHeap
- The distance value at each node now represents its minimum distance from the source node