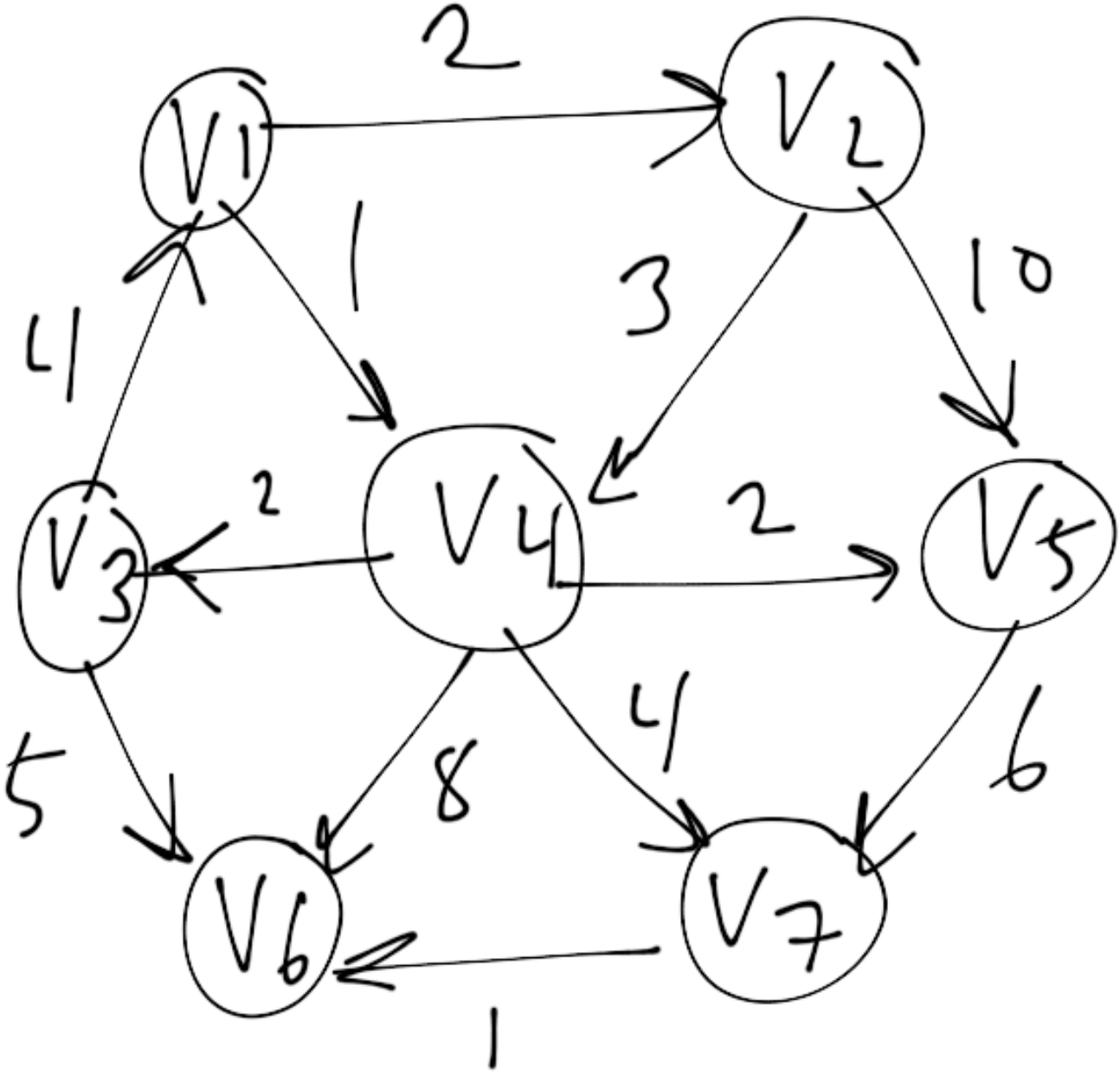


p. 354

- Dijkstra's shortest path:
  - each vertex maintains its own cost - initially these get updated as alg. proceeds
    (in STL implementation, a separate map<> is used)

- algorithm is "greedy"
- initially in both costs are ∞, get adjusted
- in my case, it's better to write down cost and vertex has been "visited"
(cost map stores this info)

- cost map: if vertex is not there, not yet "visited"

maps

step 1

V	cost	path
V1		
V2		
V3		
V4		
V5		
V6		
V7		

priority-q's

(V1, 0)

source vertex

- while priority_queue is not empty {
  - pop off the topmost node (least cost)
  - if not yet visited
(if.costs.count(node.id) == 0)
    - add its cost to lost_miles
    - for each of its adj. nodes
      - push onto priority queue

pushing node onto
priority queue: / not that for pop all

```
push(node_t(node.id,
        adjacent->node.id,
node.cost + adjacent->
                node.cost))
```

take this out for Prim's

Step 2

V	cost	path
V_1	0	V_1
V_2		
V_3		
V_4		
V_5		
V_6		
V_7		

queue

~~$(V_1, 0)$~~

↓

$(V_2, 0+2)$

$(V_4, 0+1)$

note that V_4 is at front of queue

queue

$(V_4, 1) \leftarrow (V_2, 2)$

V_1 V_1

Step 3

V	wt	path
V_1	0	V_1
V_2		
V_3		
V_4	1	V_1
V_5		
V_6		
V_7		

que

~~$(V_4, 1) \leftarrow (V_2, 2)$~~ V_1 V_1

$(V_3, 1+2)$ V_4

$(V_5, 1+2)$ V_4

$(V_6, 1+8)$ V_4

$(V_7, 1+4)$ V_4

que

$(V_2, 2) \leftarrow (V_3, 3) \leftarrow (V_5, 3)$

$\leftarrow (V_7, 5) \leftarrow (V_6, 9)$

step 4

V	cost	path
V_1	0	V_1
V_2	2	V_1
V_3		
V_4	1	V_1
V_5		
V_6		
V_7		

queue

$(V_2, 2) \leftarrow (V_3, 3)$

$\leftarrow \ldots$

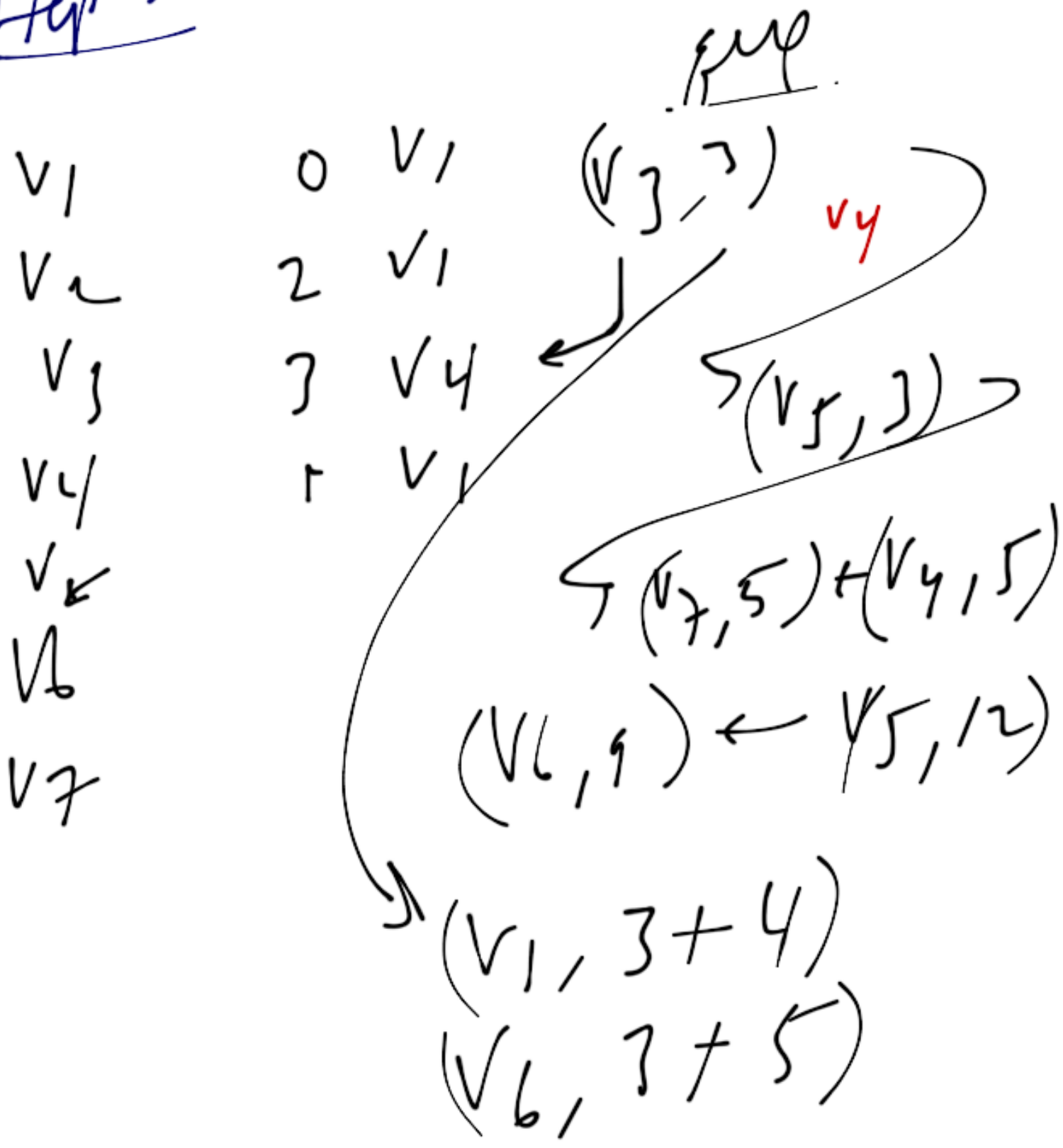
$(V_5, 2+10)$

$(V_4, 2+3)$

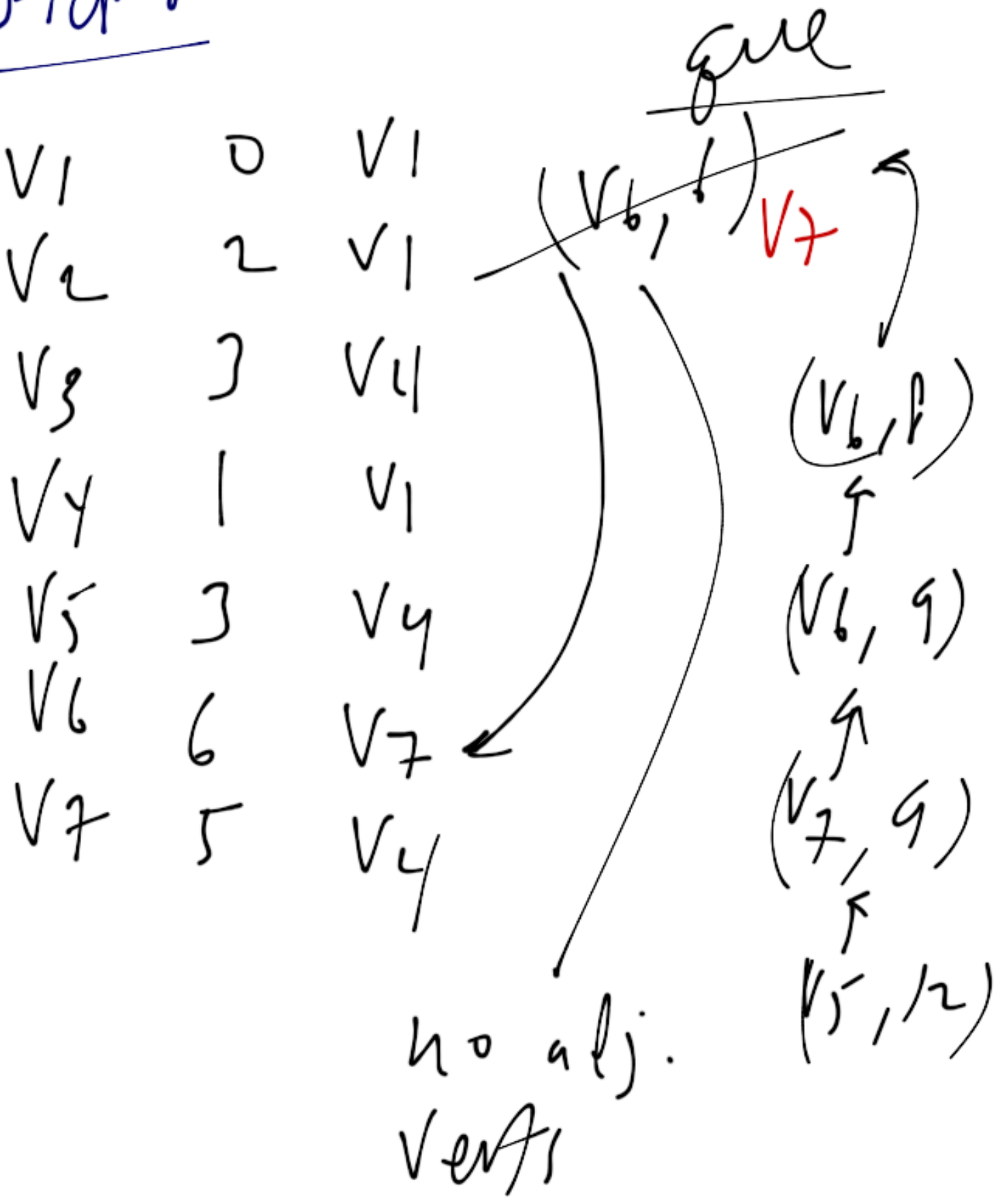
---

not added to queue

Step 5


Pop
V1
V2
V3
V4
V5
V6
V7
0 V1
2 V1
3 V4
1 V1
(V3, 3)
V4
5 (V5, 3)
(V7, 5) + (V4, 5)
(V6, 9) ← (V5, 12)
(V1, 3+4)
(V6, 3+5)

Step 8


queue
V1 0 V1
V2 2 V1
V3 3 V4
V4 1 V1
V5 3 V4
V6 6 V7
V7 5 V4
(V6, 1)
V7
(V6, 8)
(V6, 9)
(V7, 9)
(V5, 12)
no adj. vertex

Step 7

(V6, 8) gets popped off

but costs[V6]

is already there

(non-zero) so

nothing happens

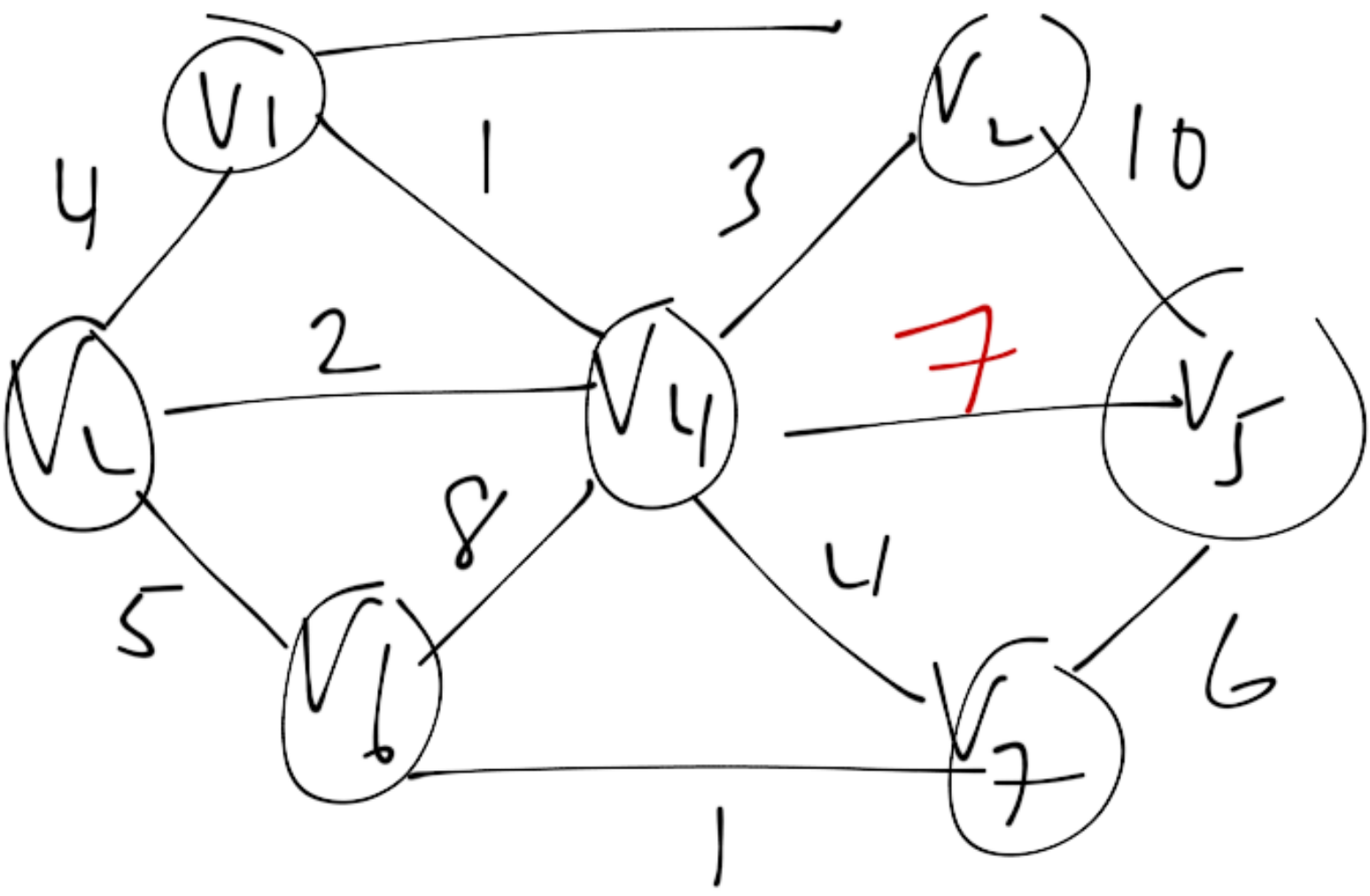
Step 10 - 12

remaining node get

popped off

Prim's Alg: MST

Minimum Spanning Tree for undirected graphs



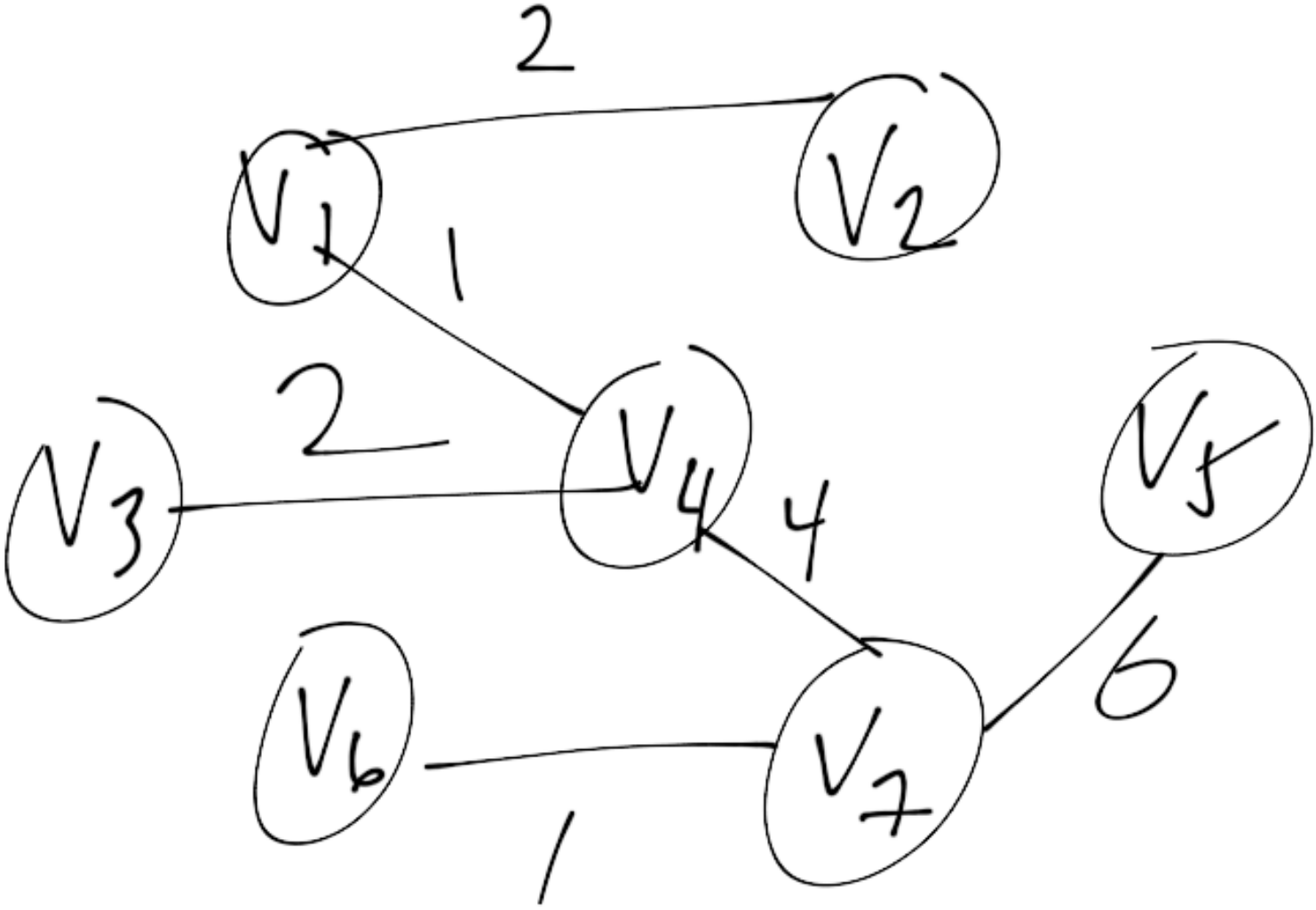
- note undirected graph:

graph[v1.id][v2.id] = 2

graph[v2.id][v1.id] = 2

- Prim's is identical to Dijkstra's except for push statement

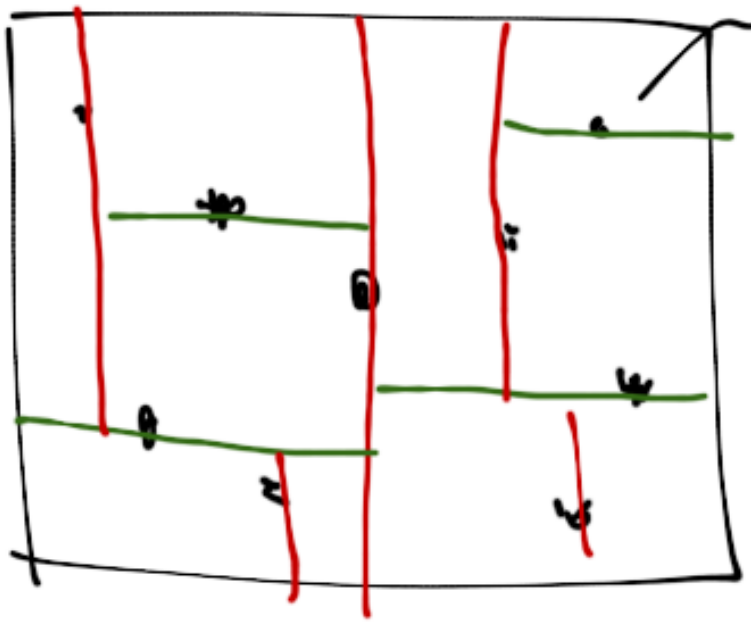
push(node(node.id, adjacent→node.id, ~~cost~~ adjacent→node.cost))



MST of given graph
(V_1 root)

- Kd-trees (§12.6)

- spatial subdivision tree: break up 2D or 3D space



level 0: split on x

level 1: split on y

level 2: split on x again