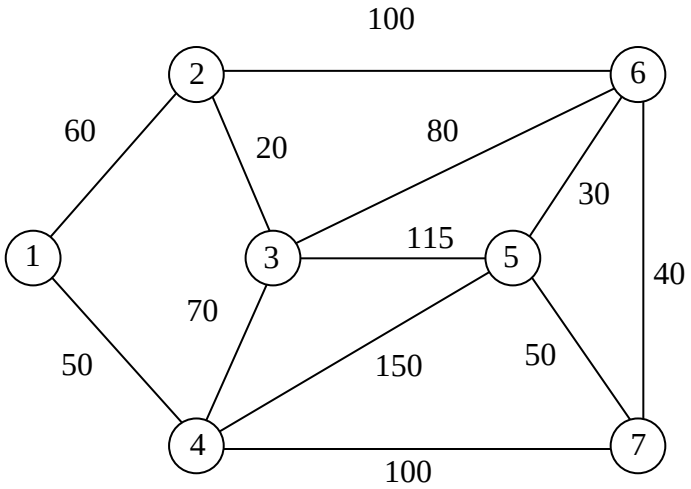
WESTERN NEW ENGLAND UNIVERSITY
COLLEGE OF ARTS AND SCIENCES
CS 366 Design and Analysis of Algorithms November 21, 2022

**Homework 7:** Greedy Algorithms and Nifty Data Structures
**Due Date**: December 7, 2022

1. Write an algorithm to be added to the end of **`dijkstra`** that takes an ending vertex **`target`** (assumed to be an integer in the range 1 to n), and uses the values stored in **`predecessor`** to output the list of vertices for a shortest path, in order, from **`start`** to **`target`**, along with the total distance from **`start`** to **`target`**.

2. A weighted graph is shown below. Trace through Dijkstra's algorithm, with starting vertex 1. Show your work on pages 2 – 5 by giving the data at the end of each iteration of the **for loop** as follows:
   - fill in the values for the heap that results at the end of the iteration - after the `delete` operation and all `decreaseKey` operations have been performed for that iteration (You should trace the intermediate swaps on scrap paper.)
   - fill in the values in the table for the `key` (which keeps track of total weight of shortest path from start = 1), whether or not the vertex is `in the heap`, and the `predecessor`

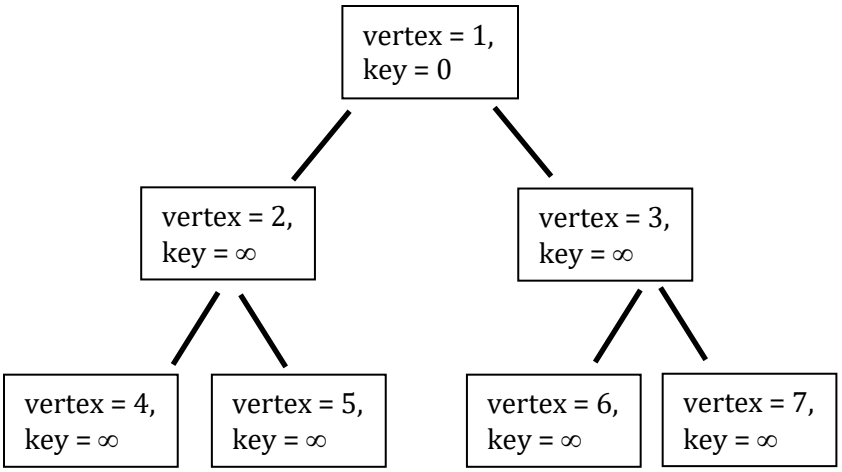
   The initialization step and the first iteration have been done for you.




3. For the same weighted graph above, trace through **`Kruskal`**'s algorithm. Show your work by giving the data at the end of each iteration of the **while loop** as follows:
   - draw the heap of edges at the end of the iteration - after the `delete` operation has been performed
   - draw the disjoint set data structure and give the values in the arrays `parent` and `height` at the end of each iteration – after the `union` operation has been performed (use a disjoint set forest implementation with union by height)
   - list the edges belonging to the array S at the end of each iteration

   The initialization step and the first iteration have been done for you (see pages 6 and 7).

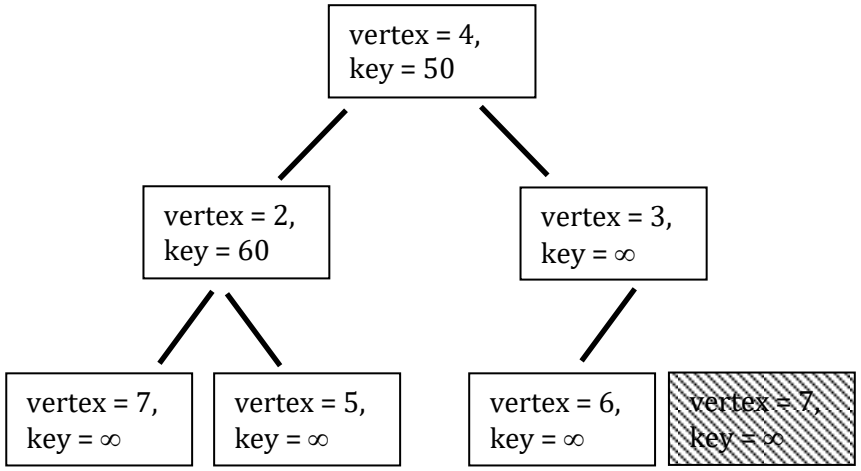
**For problem 2**: After the initialization phase of the algorithm




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | 0 | ∞ | ∞ | ∞ | ∞ | ∞ | ∞ |
| **in heap** | T | T | T | T | T | T | T |
| **predecessor** | 1 | null | null | null | null | null | null |

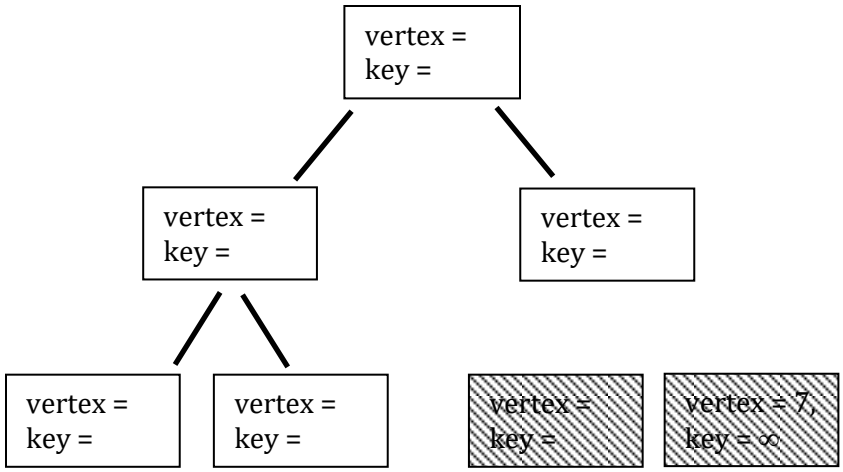
---

At the end of the iteration `i = 1`:




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | 0 | 60 | ∞ | 50 | ∞ | ∞ | ∞ |
| **in heap** | F | T | T | T | T | T | T |
| **predecessor** | 1 | 1 | null | 1 | null | null | null |

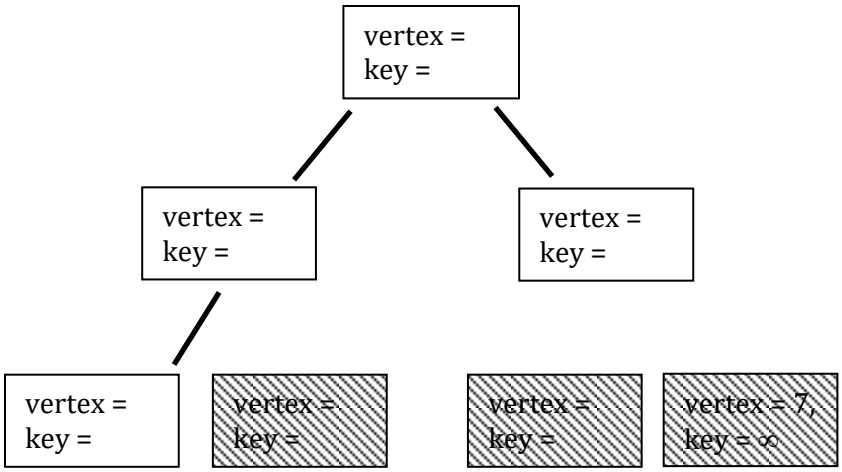
At the end of the iteration `i = 2`:




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

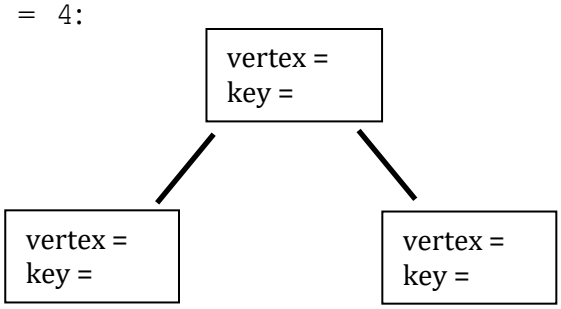
---

At the end of the iteration `i = 3`:




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

At the end of the iteration `i = 4`:

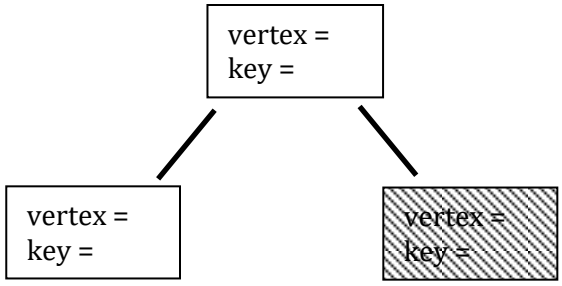



vertex = key = | vertex = key = | vertex = key = | vertex = 7, key = ∞

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

---

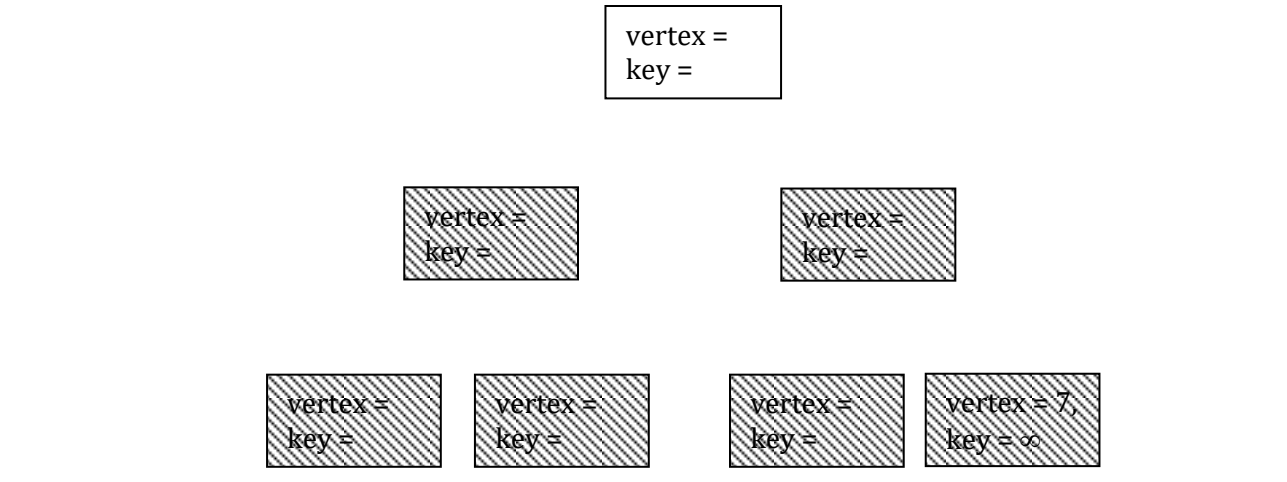
At the end of the iteration `i = 5`:




vertex = key = | vertex = key = | vertex = key = | vertex = 7, key = ∞

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

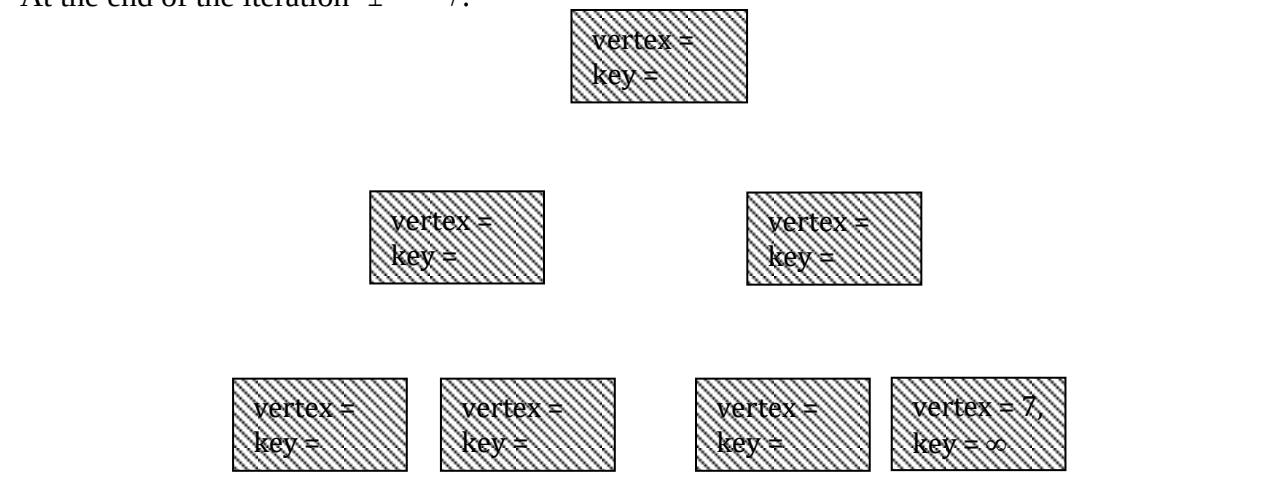
At the end of the iteration `i = 6`:




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

---

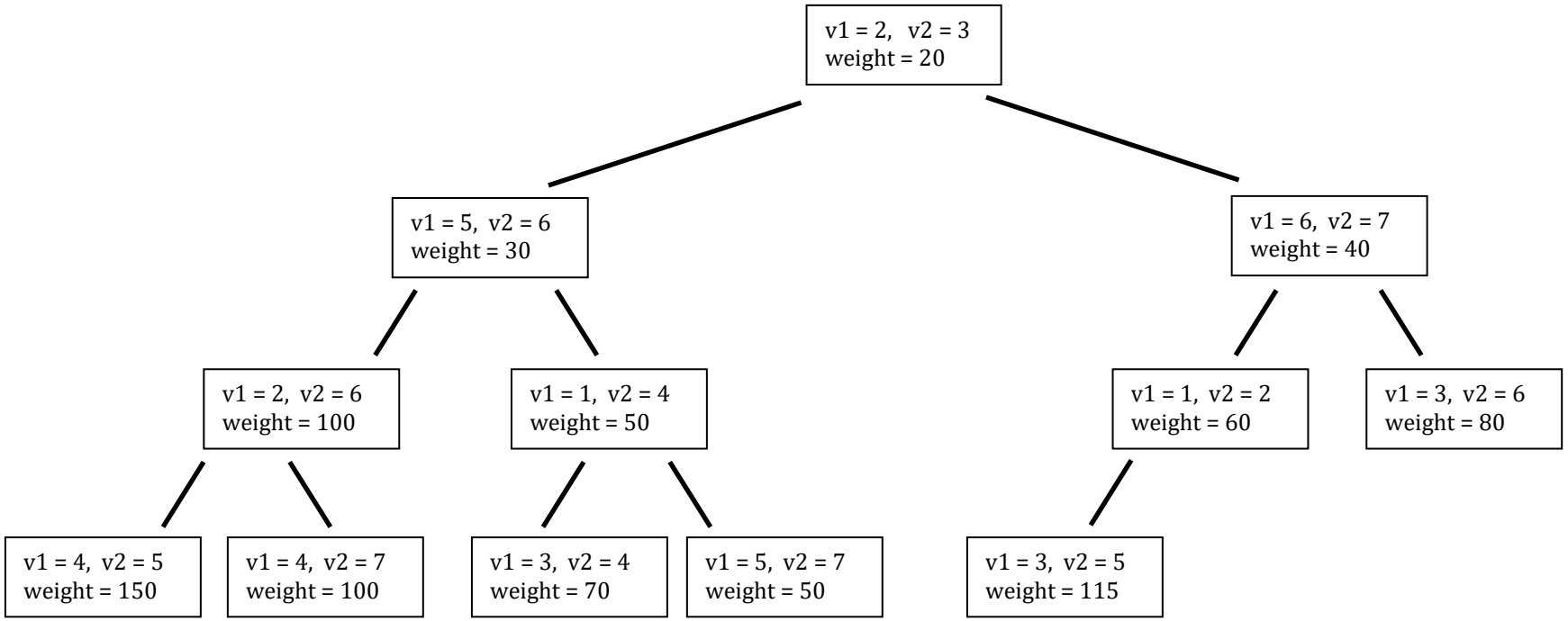
At the end of the iteration `i = 7`:




| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| **key** | | | | | | | |
| **in heap** | | | | | | | |
| **predecessor** | | | | | | | |

**For problem 3**: After initialization

**Heap of edges**




**Disjoint Set Data Structure**:

`parent = [1, 2, 3, 4, 5, 6, 7]` (parent of `i` is itself)

`height = [0, 0, 0, 0, 0, 0, 0]`

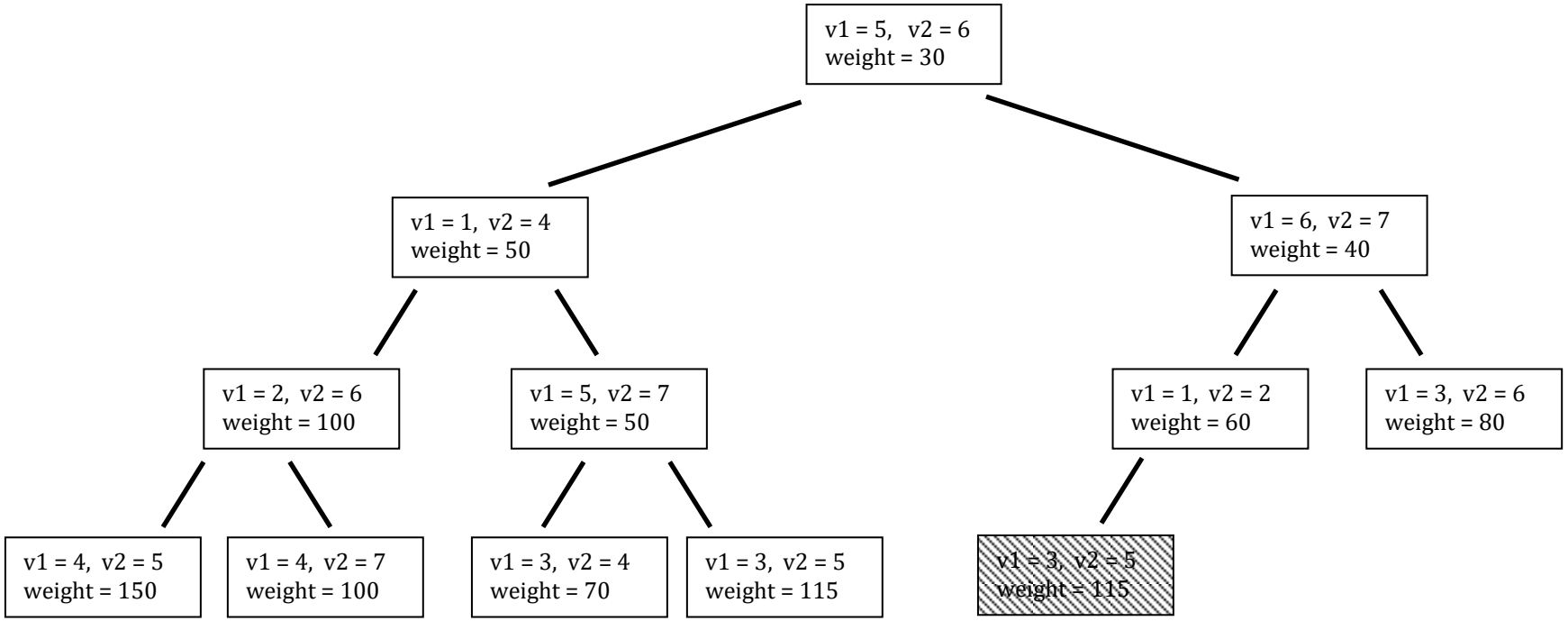

**edges in MST**

`S = [ ]` (empty)

`j = 0 (number of edges)`

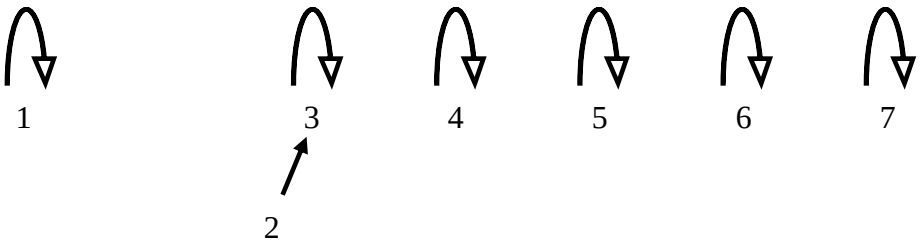
After first iteration of while loop:

**Heap of edges**




**Disjoint Set Data Structure**:

```
parent = [1, 3, 3, 4, 5, 6, 7]
height = [0, 0, 1, 0, 0, 0, 0]
```




**edges in MST**

```
S[1] =
```

v1 = 2, v2 = 3
weight = 20

```
j = 1   (number of edges)
```