Heaps
Example of Complete Binary Tree for a Vector
Maximum and Minimum Heaps Example

(A) Maximum Heap (9 nodes)

(B) Maximum Heap (4 nodes)

(C) Minimum Heap (9 nodes)

(D) Minimum Heap (4 nodes)
Example of Heap Before and After Insertion of 50

(a) (b)
Example of Reorder the tree in pushHeap()

Step 1: Compare 50 and 25 (Exchange v[10] and v[4])

Step 2: Compare 50 and 30 (Exchange v[4] and v[1])

Step 3: Compare 50 and 63 (50 in correct location)
Example of Exchanging elements in popHeap()

Before a deletion

After exchanging the root and last element in the heap
Example of Adjusting the heap for popHeap()

Step 1: Exchange 18 and 40

Step 2: Exchange 18 and 38
Example of Implementing heap sort

```c
int arr[] = {50, 20, 75, 35, 25};
vector<int> v(arr, 5);
```

Heapified Tree
Example of Implementing heap sort (Cont. . .)

Calling `popHeap()` with last = 5
deletes 75 and stores it in h[4]

Calling `popHeap()` with last = 4
deletes 50 and stores it in h[3]

Calling `popHeap()` with last = 3
deletes 35 and stores it in h[2]

Calling `popHeap()` with last = 2
deletes 25 and stores it in h[1]
Example of Heapifying a Vector

Initial Vector

adjustHeap() at 4 causes no changes
(A)
Example of Heapifying a Vector (Cont…)

adjustHeap() at 3 moves 30 down
(B)

adjustHeap() at 2 moves 17 down
(C)

adjustHeap() at 1 moves 12 down two levels
(D)

adjustHeap() at 0 moves 9 down three levels
(E)
Heap Summary Slide 1

- an array-based tree that has heap order
- maximum heap: if $v[i]$ is a parent, then $v[i] \geq v[2i+1]$ and $v[i] \geq v[2i+2]$ (a parent is $\geq$ its children)
- root, $v[0]$, is the maximum value in the vector
- minimum heap: the parent is $\leq$ its children.
- $v[0]$ is the minimum value
- Insertion: place the new value at the back of the heap and filtering it up the tree.
Heap Summary Slide 2

- Deletion: exchanging its value with the back of the heap and then filtering the new root down the tree, which now has one less element.
- Insert and delete running time: \( O(\log_2 n) \)
- Heapifying: apply the filter-down operation to the interior nodes, from the last interior node in the tree down to the root
- Running time: \( O(n) \)
- The \( O(n \log_2 n) \) heapsort algorithm heapifies a vector and erases repeatedly from the heap, locating each deleted value in its final position.