Subject: Heap Sorting Date: / /

② heap sorting algorithm is a comparison based sorting algorithm that uses a binary heap (binary tree) to sort a sequence of numbers. We perform 3 steps: ① building a binary heap of n objects in A where the min (max) object located at the top of heap o(nlogn.

② Call "retrive and remove" for extracting the largest object remaining in the heap and repeat it until heap is empty.

③ Saveing these objects in array, it will take o(n).

algorithm step by step:

① heapify: this is a helper function used to maintain the heap property for a subtree rooted at index i.

② "heap-sort": the main heapsort function that use the build-maxheap and repeatedly extracts the maximum. It swap the root element with the last element and then calls heapify to rebuild the heap. So the time complexity for this step is o(nlogn)

③ build_max_heap: it is a function iterates through the elements from the elements from n/2 to 0 and for each element it calls heapify. Since heapify take o(logn) the overall time complexity for building the heap is o(n)

- the overall complexity for the heap sorting process is o(nlogn)

Subject: Date: / /

Kruskal's algorithm

Kruskal's algorithm is a greedy algorithm used to find the minimum Spanning Tree (MST) of a connected, weighted graph. it works by sorted the edges of the graph and adding them to the MST one by one, ensuring no cycles are formed.

1. Sort the edges of the graph in non-decreasing order of their weight
2. initalize the MST as an empty set.
3. For each edge in the sorted edge list; check if adding the edge to the mst forms a cycle using a union-find data strucure
   - if no cycle is formed, add the edge to the MST
4. repeat until the MST contains V-1 edges, where V is the number of vertices in the graph.

- Sorting the edges: Sorting all edges takes $O(E \log E)$ where E is the number of edges.
- union-find operations: each find set or union operation runs is almost constant time, $O(\alpha(V))$ where α is inverse ackermann function which grows very slowly
- overall Complexity: we process each edge the overall complexity is dominated where $O(E \log E + E \alpha(V))$
  $= O(E \log E)$.

- it works for connected, unidirected, weighted graphs.