(a)

BuildHeap

For each gas station g in h from last to first

    Heapify down

Runtime: O(n) since the work to build it is 2^logn + 1 = 2n

ExtractMin

gas station temp = first element in heap

Delete first element in heap

Return temp

Runtime: O(logn) because delete uses heapify up or heapify down which both have a runtime of O(logn)

Delete

Set index of gas station to be deleted to last gas station in heap.

If that distance of that station is too big

    heapify down

Else

    heapify up

Runtime: O(logn) because heapify up and heapify down have a runtime of O(logn)

ChangingKey

Delete gas station r whose distance needs to be changed from the heap

Change the distance of r to newDist

Insert r into the heap

Runtime: O(logn) since delete and insert which use heapify up and heapify down have a runtime of O(logn)

InsertNode

Add gas station to end of the heap

Heapify up added node

Heapify up (i, in)

If index i of station in to be heapified up is > 0

    Index of parent of station = (i - 1) / 2

    If in.distance < parent.distance

        Swap gas stations

        Heapify up in with new index

Heapify down (i)

Let size = last index of heap

If 2i + 1 > size

    Return with heap unchanged

```
Else if 2i + 1 < size
    Min_index = index of child of element at i with smallest distance
Else if 2i + 1 == size
    Min_index = 2i + 1
Endif

If min_index.distance < i.distance
    Swap elements at min_index and i
    Heapify down (min_index)
```

(b)

findAllReachableStations (start, init_size)

```
Initialize unreachable list to include all stations
Initialize tank to init_size + start.upgrade value
If init_size < 0
    Return empty reachable list
Add start to reachable list
Remove start from unreachable list
Do{
    Size = reachable.size
    For each element r in reachable
        For each element u in unreachable
            If distance from r to u is < tank
                Add u to reachable
                Add u.upgrade value to tank
                Remove u from unreachable
}while(size != reachable.size)
Return reachable
```

Runtime: O(n^2) since the number of elements iterated through in the do while loop is parabolic: n-1 + 2(n-2) + 3(n-3) + … n/2(n - n/2) + … + 3(n-3) + 2(n-2) + n-1 with n/2(n/2) which the largest element in the polynomial which is n^2/4 which is O(n^2).

(c)

Findminimumtanksize(start, dest)

```
Initialize unreached list to include all stations
Initialize tank to start.upgrade
Add start to reachable list
Remove start from unreached list
If distance from start to dest < start.upgradevalue
    Return 0
While reachable !contain dest
    Find station in unreached that is closest to any station in reachable
    Add closest to reachable
    Remove closest from unreached
```

If tank < distance from closest to its closest station in reachable
Diff = distance from closest to its closest station in reachable - tank
Minimum_size += diff
Tank += diff
Tank += closest.upgradevalue
Return minimum_size

Runtime: O(n^2) since the number of elements iterated through in the do while loop is parabolic: n-1 + 2(n-2) + 3(n-3) + … n/2(n - n/2) + … + 3(n-3) + 2(n-2) + n-1 with n/2(n/2) which the largest element in the polynomial which is n^2/4 which is O(n^2).