19.1 Quickselect

This section has been set as optional by your instructor.

Quickselect is an algorithm that selects the smallest element in a list. Ex: Running quickselect on the list (15, 73, 5, 88, 9) with k = 0, returns the smallest element in the list, or 5.

For a list with N elements, quickselect uses quicksort's partition function to partition the list into a low partition containing the X smallest elements and a high partition containing the N-X largest elements. The smallest element is in the low partition if k is ≤ the last index in the low partition, and in the high partition otherwise. Quickselect is recursively called on the partition that contains the element. When a partition of size 1 is encountered, quickselect has found the smallest element.

Quickselect partially sorts the list when selecting the smallest element.

The best case and average runtime complexity of quickselect are both O( ). In the worst case, quickselect may sort the entire list, resulting in a runtime of O( ).

Figure 19.1.1: Quickselect algorithm.

```
// Selects kth smallest element, where k is 0-based
Quickselect(numbers, first, last, k) {
   if (first >= last)
      return numbers[first]

   lowLastIndex = Partition(numbers, first, last)

   if (k <= lowLastIndex)
      return Quickselect(numbers, first, lowLastIndex,
k)
   return Quickselect(numbers, lowLastIndex + 1, last,
k)
}
```

PARTICIPATION ACTIVITY 19.1.1: Quickselect.

1) Calling quickselect with argument k equal to 1 returns the smallest element in the list.

○ True

○ False

2) The following function produces the same result as quickselect, albeit with a different runtime complexity.

```
Quickselect(numbers, first,
last, k) {
   Quicksort(numbers, first,
last)
   return numbers[k]
}
```

○ True

○ False

3) Given k = 4, if the quickselect call `Partition(numbers, 0, 10)` returns 4, then the element being selected is in the low partition.

○ True

○ False

CHALLENGE ACTIVITY | 19.1.1: Quickselect.

489394.3384924.qx3zqy7

Start

What is returned when running quickselect on (62, 13, 74, 20, 55, 80, 57) with k = 5?

Ex: 10

1 | 2 | 3

Check Next

19.2 Bucket sort

This section has been set as optional by your instructor.

Bucket sort is a numerical sorting algorithm that distributes numbers into buckets, sorts each bucket with an additional sorting algorithm, and then concatenates buckets together to build the sorted result. A ***bucket*** is a container for numerical values in a specific range. Ex: All numbers in the range 0 to 49 may be stored in a bucket representing this range. Bucket sort is designed for arrays with non-negative numbers.

Bucket sort first creates a list of buckets, each representing a range of numerical values. Collectively, the buckets represent the range from 0 to the maximum value in the array. For buckets and a maximum value of , each bucket represents —— values. Ex: For 10 buckets and a maximum value of 49, each bucket represents a range of —— = 5 values; the first bucket will hold values ranging from 0 to 4, the second bucket 5 to 9, and so on. Each array element is placed in the appropriate bucket. The bucket index is calculated as —— . Then, each bucket is sorted with an additional sorting algorithm. Lastly, all buckets are concatenated together in order, and copied to the original array.

Figure 19.2.1: Bucket sort algorithm.

```
BucketSort(numbers, numbersSize, bucketCount) {
   if (numbersSize < 1)
      return

   buckets = Create list of bucketCount buckets

   // Find the maximum value
   maxValue = numbers[0]
   for (i = 1; i < numbersSize; i++) {
      if (numbers[i] > maxValue)
         maxValue = numbers[i]
   }

   // Put each number in a bucket
   for each (number in numbers) {
      index = floor(number * bucketCount / (maxValue +
1))
      Append number to buckets[index]
   }

   // Sort each bucket
   for each (bucket in buckets)
      Sort(bucket)

   // Combine all buckets back into numbers list
   result = Concatenate all buckets together
   Copy result to numbers
}
```

PARTICIPATION ACTIVITY 19.2.1: Bucket sort.

Suppose BucketSort is called to sort the list (71, 22, 99, 7, 14), using 5 buckets.

1) 71 and 99 will be placed into the same bucket.

- ○ True
- ○ False

2) No bucket will have more than 1 number.

- ○ True
- ○ False

3) If 10 buckets were used instead of 5, no bucket would have more than 1 number.

○ True
○ False

Bucket sort terminology

The term "bucket sort" is sometimes used to refer to a category of sorting algorithms, instead of a specific sorting algorithm. When used as a categorical term, bucket sort refers to a sorting algorithm that places numbers into buckets based on some common attribute, and then combines bucket contents to produce a sorted array.