```
RESULTS

------ arrays of 100 items ------

Sorting arr3ItemsShuffled100 (native)

time = 0.028999999999996362ms

Sorting arrFullyShuffled100 (native)

time = 0.016899999999992588ms

Sorting arr3ItemsShuffled100 (bubble)

number of iterations = 100; time = 0.23730000000000473ms

Sorting arrFullyShuffled100 (bubble)

number of iterations = 2523; time = 0.5636999999999972ms

Sorting arr3ItemsShuffled100 (quick)

time = 0.6376999999999953ms

Sorting arrFullyShuffled100 (quick)

time = 0.671999999999997ms

Sorting arr3ItemsShuffled100 (bucket)

number of iterations = 110; time = 0.19089999999999918ms

Sorting arrFullyShuffled100 (bucket)

number of iterations = 110; time = 0.08100000000000307ms

------ arrays of MANY items ------

Sorting arr3ItemsShuffled (native)

time = 2.0553000000000026ms

Sorting arrFullyShuffled (native)

time = 24.931200000000004ms

Sorting arr3ItemsShuffled (bubble)

number of iterations = 10334; time = 382.345ms

Sorting arrFullyShuffled (bubble)

number of iterations = 2491875416; time = 18686.897ms

Sorting arr3ItemsShuffled (bucket)
```

```
number of iterations = 100010; time = 14.758900000000722ms

Sorting arrFullyShuffled (bucket)

number of iterations = 100010; time = 7.908899999998539ms
```

CODE

```
//PARTLY GENERATED WITH AI (GITHUB COPILOT)

const size = 100_000;
const arr3ItemsShuffled = Array.from({length: size}, (_, i) => i + 1);

const pickDistinctIndices = (n, count) => {
  const indices = new Set();
  while (indices.size < count) indices.add(Math.floor(Math.random() * n));
  return [...indices];
};

const [a, b, c] = pickDistinctIndices(arr3ItemsShuffled.length, 3);
const temp = arr3ItemsShuffled[a];
arr3ItemsShuffled[a] = arr3ItemsShuffled[b];
arr3ItemsShuffled[b] = arr3ItemsShuffled[c];
arr3ItemsShuffled[c] = temp;

const arrFullyShuffled = Array.from({length: size}, (_, i) => i + 1);
for (let i = arrFullyShuffled.length - 1; i > 0; i--) {
  const j = Math.floor(Math.random() * (i + 1));
  [arrFullyShuffled[i], arrFullyShuffled[j]] = [arrFullyShuffled[j], arrFullyShuffled[i]];
}

const arr3ItemsShuffled100 = Array.from({length: 100}, (_, i) => i + 1);
const [a100, b100, c100] = pickDistinctIndices(arr3ItemsShuffled100.length, 3);
const temp100 = arr3ItemsShuffled100[a100];
arr3ItemsShuffled100[a100] = arr3ItemsShuffled100[b100];
arr3ItemsShuffled100[b100] = arr3ItemsShuffled100[c100];
arr3ItemsShuffled100[c100] = temp100;
```

```
const arrFullyShuffled100 = Array.from({length: 100}, (_, i) => i + 1);
for (let i = arrFullyShuffled100.length - 1; i > 0; i--) {
  const j = Math.floor(Math.random() * (i + 1));
  [arrFullyShuffled100[i], arrFullyShuffled100[j]] = [arrFullyShuffled100[j],
arrFullyShuffled100[i]];
}

function bubbleSort(arr) {
  let swapped = true;
  let numberOfIteration = 0;

  const start = performance.now();

  while (swapped) {
    swapped = false;

    for (let i = 0; i < arr.length - 1; i++) {
      if (arr[i] > arr[i + 1]) {
        const tmp = arr[i + 1];
        arr[i + 1] = arr[i];
        arr[i] = tmp;

        numberOfIteration++;

        swapped = true;
      }
    }
  }

  const timeTaken = performance.now() - start;
  console.log("number of iterations = " + numberOfIteration + "; time = " + timeTaken +
"ms");
}

function quickSort(arr) {
  if (arr.length < 2) return arr;

  const chosenItem = arr[0];
  const leftArray = [];
  const rightArray = [];
  for (let i = 1; i < arr.length; i++) {
    const item = arr[i];
    if (item <= chosenItem) {
      leftArray.push(item);
    } else {
      rightArray.push(item);
    }
  }

  const leftSorted = quickSort(leftArray);
  const rightSorted = quickSort(rightArray);

  const result = [];

  for (let i = 0; i < leftSorted.length; i++) {
    result.push(leftSorted[i]);
  }

  result.push(chosenItem);

  for (let i = 0; i < rightSorted.length; i++) {
    result.push(rightSorted[i]);
  }

  return result;
}

function quickSortMeasure(arr) {
  let start = performance.now();
  quickSort(arr);
  const timeTaken = performance.now() - start;
  console.log("time = " + timeTaken + "ms");
}

function bucketSort(arr) {
  const buckets = [];
  const num_buckets = 10;
  const start = performance.now();
  let numberOfIteration = 0;

  for (let i = 0; i < num_buckets; i++) {
    buckets.push([])
  }
  const minVal = Math.min(...arr);
  const maxVal = Math.max(...arr);
  // place elements into buckets
  let bucketSize = Math.floor((maxVal - minVal) / num_buckets);
  if (bucketSize < 1) {
    bucketSize = 1;
  }
  for (let i = 0; i < arr.length; i++) {
    const elem = arr[i];
    const bucket_index = parseInt((elem - minVal) / bucketSize);
    if (bucket_index >= num_buckets) {
      // put the max value in the last bucket
      buckets[num_buckets - 1].push(elem);
    } else {
      buckets[bucket_index].push(elem);
    }
    // consider this as one operation
    numberOfIteration++;
  }

  const sortedArr = [];
  for (let i = 0; i < num_buckets; i++) {
    const bucket = buckets[i];
    // TODO performance varies with different sorting algorithms here
    bucket.sort(function (x, y) {
      return x - y
    });
    for (let j = 0; j < bucket.length; j++) {
      sortedArr.push(bucket[j]);
    }
    // consider this as one operation
    numberOfIteration++;
  }

  const timeTaken = performance.now() - start;
  console.log("number of iterations = " + numberOfIteration + "; time = " + timeTaken +
"ms");

  return sortedArr;
```

```
}

function nativeSort(arr) {
   const start = performance.now();
   // note: toSorted because we don't want to mutate the original array for other sorting
algorithms
   arr.toSorted((a, b) => a - b);
   const timeTaken = performance.now() - start;
   console.log("time = " + timeTaken + "ms");
}


console.log("------ arrays of 100 items ------");

console.log("Sorting arr3ItemsShuffled100 (native)");
nativeSort(arr3ItemsShuffled100);

console.log("Sorting arrFullyShuffled100 (native)");
nativeSort(arrFullyShuffled100);

console.log("Sorting arr3ItemsShuffled100 (bubble)");
bubbleSort(arr3ItemsShuffled100);

console.log("Sorting arrFullyShuffled100 (bubble)");
bubbleSort(arrFullyShuffled100);

console.log("Sorting arr3ItemsShuffled100 (quick)");
quickSortMeasure(arr3ItemsShuffled100);

console.log("Sorting arrFullyShuffled100 (quick)");
quickSortMeasure(arrFullyShuffled100);

console.log("Sorting arr3ItemsShuffled100 (bucket)");
bucketSort(arr3ItemsShuffled100);

console.log("Sorting arrFullyShuffled100 (bucket)");
bucketSort(arrFullyShuffled100);

console.log("------ arrays of MANY items ------");

console.log("Sorting arr3ItemsShuffled (native)");
```

```
nativeSort(arr3ItemsShuffled);

console.log("Sorting arrFullyShuffled (native)");
nativeSort(arrFullyShuffled);

console.log("Sorting arr3ItemsShuffled (bubble)");
bubbleSort(arr3ItemsShuffled);

console.log("Sorting arrFullyShuffled (bubble)");
bubbleSort(arrFullyShuffled);

/* STACK OVERFLOW ERROR (too much recursion)
console.log("Sorting arr3ItemsShuffled (quick)");
quickSortMeasure(arr3ItemsShuffled);

console.log("Sorting arrFullyShuffled (quick)");
quickSortMeasure(arrFullyShuffled);
*/

console.log("Sorting arr3ItemsShuffled (bucket)");
bucketSort(arr3ItemsShuffled);

console.log("Sorting arrFullyShuffled (bucket)");
bucketSort(arrFullyShuffled);
```