Merge Sort

Code

`merge`

```
export function merge(A: number[], start: number, middle: number, end: number) {
    // create 2 arrays with extra slot
    const L = new Array(middle - start + 1);
    const R = new Array(end - middle + 1);
    // copy elements to subarrays
    for (let l = 0; l < L.length - 1; l++) L[l] = A[start + l];
    for (let r = 0; r < R.length - 1; r++) R[r] = A[middle + r];
    // fill extra slot with sentinal
    L[L.length - 1] = Number.MAX_VALUE;
    R[R.length - 1] = Number.MAX_VALUE;
    let l = 0;
    let r = 0;
    // compare elements to order original array
    for (let i = start; i < end; i++) {
        if (L[l] < R[r]) {
            A[i] = L[l];
            l++;
        } else {
            A[i] = R[r];
            r++;
        }
    }
}
```

`mergeSort`

```
export function mergeSort(A: number[], start: number, end: number) {
    if (start < end - 1) {
        let middle = Math.floor((start + end) / 2);
        mergeSort(A, start, middle);
        mergeSort(A, middle, end);
        merge(A, start, middle, end);
    }
}
```

Design

- uses a divide-and-conquer approach which are usually **recursive** in structure

Runtime Analysis

TODO