

Problems | Name: _____

2.1 Asymptotic Notation

Reference example

```
int count = 0;
int n = 4;
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++) count++;
System.out.println(count);
```

2.1.1 - Explain

Compare $3n + 8$ with n squared as input size grows.

2.1.2 - Explain

A loop repeatedly halves a positive problem size until it reaches 1. What growth class describes its iteration count?

Problems | Name: _____

2.2 Comparison Sorts

Reference example

```
int[] a = {4, 1, 3};
for (int i = 1; i < a.length; i++) {
    int key = a[i], j = i - 1;
    while (j >= 0 && a[j] > key) { a[j + 1] = a[j]; j--; }
    a[j + 1] = key;
}
System.out.println(Arrays.toString(a));
```

2.2.1 - Explain

Why is insertion sort linear on already-sorted input in this implementation?

2.2.2 - Explain

Does the greater-than comparison in the inner loop preserve equal-item order?

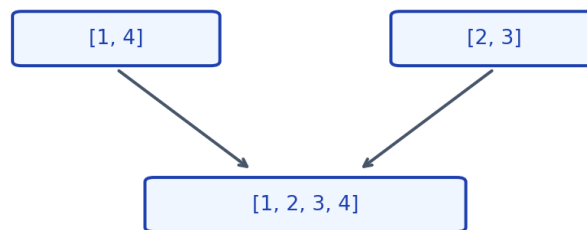
Problems | Name: _____

2.3 Merge Sort and Quick Sort

Reference example

Merge sorted runs [1, 4] and [2, 3]
Choose smallest front: 1, then 2, then 3
Append remaining 4
Result: [1, 2, 3, 4]

Merging sorted runs



Take the smaller available front item, then append any remainder.

2.3.1 - Explain

What is the merge result for sorted runs [2, 5] and [1, 3, 6]?

2.3.2 - Explain

Why can already sorted input be a bad case for quicksort that always uses an endpoint pivot?

Problems | Name: _____

2.4 Binary Search

Reference example

```
int[] a = {2, 5, 8, 11, 14};
int low = 0, high = a.length - 1, found = -1;
while (low <= high) {
    int mid = low + (high - low) / 2;
    if (a[mid] == 11) { found = mid; break; }
    if (a[mid] < 11) low = mid + 1;
    else high = mid - 1;
}
System.out.println(found);
```

2.4.1 - Explain

On [2, 5, 8, 11, 14], what midpoint indices are checked when searching for 2?

2.4.2 - Write code

Implement iterative binary search for a sorted int array. Demonstrate a present target and an absent target.
