Problem 4: Merge vs. Selection Sort

* Merge Sort:

```
if (array.size() == 1) return;
int mid = array.size() / 2;
vector<int> left(mid);
vector<int> right(array.size() - mid);
for(int i = 0; i < mid; i++) left[i] = array[i];
for(int i = mid, i < array.size(); i++)
        right[i - mid] = array[i];
mrgSort(left);
mrgSort(right);
merge(array, left, right);
```

we continually halve our subarrays until we hit our base condition. This process of halving our subarrays will take at most log(N) operations, where N = size of array.

Thereafter, we merge our halves back together. This process takes N operations because we iterate over our entire array as we fill it. Thus, the merging operations are O(N) operations and the halving operations are O(Log N) operations.

Therefore, we can conclude that the merge sort is O(NLogN). Everytime we halve our arrays, we have to merge them back together.

* Selection sort:

```
for(int i=0; i<p; i++){
    int minIndex = i;
    for(int j=i+1; j< array.size(); j++){
        if( array[j] < array[minIndex])
            minIndex = j;
    }
    swap(array[i], array[minIndex]);
}
```

Let:

O_i = operations in the outer loop.

O_j = operations in the inner loop.

$POmin$ = operations to find the min. value.

O_s = swap operations.

Note: There is a probability P attached to O_{min} because these operations take place when a condition is satisfied.

Then:

$$\sum_{i=0}^{p-1} O_i + O_s + \sum_{j=i+1}^{N-1} O_j + POmin .$$

* $\sum_{i=0}^{N} 1 = (N-0+1) * 1 = N+1 .$

- Let $O_j + POmin = O_{jmin}$.

$$\Rightarrow \sum_{j=i+1}^{N-1} O_j + POmin = (N-1-(i+1)+1) * O_{jmin}$$
$$= (N-i-1) * O_{jmin}$$

$$\Rightarrow \sum_{i=0}^{p-1} O_i + O_s + N O_{jmin} - i * O_{jmin} - O_{jmin}$$

* $\sum_{i=0}^{N} i = N * \frac{(N+1)}{2} .$

$$\Rightarrow \sum_{i=0}^{p-1} Oi + Os + NOjmin - i * Ojmin - Ojmin.$$

$$= p(Oi + Os + NOjmin - \frac{(p-1)}{2} p * Ojmin - Ojmin)$$

$$= pOi + pOs + pNOjmin - \frac{(p^2-p)}{2} Ojmin - pOjmin.$$

Let $f(N) = C'N + C$.

$$= (pOjmin)N + (pOi + pOs - \frac{(p-1)}{2} p * Ojmin - pOjmin)$$

Thus, we can conclude that our selection sort is $O(N)$ for $p \ll N$.

Conclusion: Our selection sort outperforms the merge sort for $p \ll N$. As $p \to N$, the selection sort behaves more and more like an $O(N^2)$ sort. In this case, the merge sort outperforms our selection sort. As long as p is taken to be $p \ll N$, our selection sort behaves like an $O(N)$ sort; thus, beating out the merge sort which is $O(N \log N)$.