Amy Nakamura
CPE 349-01/02
Lab 5
04/28/15

Minimizing Wait Time

Psuedo Code:

It's possible to use a simple application of merge sort.

mergeSort(arr[0...n - 1])
// basically mergesort with tweaks
// returns an array of sorted jobs, aka the ordering that minimizes wait time

```
if n = 1 do
        return arr
left ← arr[0...n/2 - 1]
right ← arr[n/2...n - 1]
result ← merge(left, right)
return result
```

merge(left[0...p], right[0...q])
// returns an array of sorted jobs with an input of two arrays

```
i ← 0, j ← 0, k ← 0
while i < p and j < q do
        if left[i] <= right[j] do
                arr[k] ← left[i]
                i ← i + 1
        else do
                arr[k] ← right[j]
                j ← j + 1
        k ← k + 1
if i = p do
        copy right[j...q] into arr
else do
        copy left[i...p] into arr
return arr
```

Performance Evaluation:

The equation of this algorithm is $T(n) = 2T(n/2) + n$, which follows the format of the Master Theorem $T(n) = aT(n/b) + f(n)$ where $f(n) = n^d$.
In this case, $a = 2$, $b = 2$, and $d = 1$.
Plugging the values into the equation $\log_b a$ we get 1, which is equal to d.
This indicates a performance of $O(n * \log n)$.

Proof our algorithm is correct:

This will be a **proof by contradiction**.

1) Assume T to be the ordering of n jobs that minimizes the total wait time, where the order takes the form of an array T[0..j..k..n].

2) Assume there are two jobs $j \in T$ and $k \in T$, that can be exchanged, to reduce the total wait time.
   This implies that the wait time of $k <$ the wait time of j.

3) Exchange j and k, so that the new ordering of jobs T' takes the form of an array T'[0..k..j..n].
   The claim made is that the total wait time of $T' <$ the total wait time of T.
   This is illustrated with the hypothetical example of a T that only has jobs j and k, T[j, k].
   The total wait time of T is $j + (j + k) = 2j + k$
   Now, exchanging j and k on the basis of our above assumptions, we have T'[k, j].
   The total wait time of T' is $k + (k + j) = 2k + j$
   Because $k < j$, $T' < T$ follows.

4) BUT, this contradicts the initial assumption that we started out with an ordering of n jobs that minimizes the total wait time, T.