Test #1

- Next week (Feb 9/10), in your registered tutorial timeslot
- Room assignments will be sent out on Monday
- Coverage: Weeks 1 - 4 (Induction, Recurrences)
Outline

Wrap up RecBinSearch

MergeSort

Divide and Conquer

Notes
Recursive Binary Search

Upper bound on $T(n)$
Recursive Binary Search

$T(n)$ is non-decreasing
Recurrence for MergeSort

```
MergeSort(A, b, e):
    if b == e: return
    m = (b + e) / 2
    MergeSort(A, b, m)
    MergeSort(A, m+1, e)
    # merge sorted A[b..m] and A[m+1..e] back into A[b..e]
    for i in [b,...,e]: B[i] = A[i]
    c = b
    d = m+1
    for i in [b,...,e]:
        if d > e or (c <= m and B[c] < B[d]):
            A[i] = B[c]
            c = c + 1
        else:  # d <= e and (c > m or B[c] >= B[d])
            A[i] = B[d]
            d = d + 1
```
Recurrence for MergeSort
Unwind $T(n)$
Unwind $T(n)$
Merge Sort
Lower bound on $T(n)$
Merge Sort

Lower bound on $T(n)$
Merge Sort

Lower bound on $T(n)$
Merge Sort

Upper bound on $T(n)$
Merge Sort

Upper bound on $T(n)$
Class of algorithms: partition problem into $b$ *roughly* equal subproblems, solve, and recombine:

$$T(n) = \begin{cases} 
k & \text{if } n \leq B 
\end{cases}$$

$$a_1 T([n/b]) + a_2 T([n/b]) + f(n) \quad \text{if } n > B$$

where $B, k > 0$, $b > 1$, $a_1, a_2 \geq 0$, and $a = a_1 + a_2 > 0$. $f(n)$ is the cost of splitting and recombining.
If $f$ from the previous slide has $f \in \theta(n^d)$, then

$$T(n) \in \begin{cases} 
\theta(n^d) & \text{if } a < b^d \\
\theta(n^d \log n) & \text{if } a = b^d \\
\theta(n^{\log_b^a}) & \text{if } a > b^d 
\end{cases}$$
Applying the Master Theorem

MergeSort
Applying the Master Theorem
RecBinSearch