Topics for this week:
- Merge Sort
- Quick Sort
- their time-complexity (introducing $O(n \log n)$)

Announcements/discussion:
- project part 3 published
- questions about the hashing lab?
Problem: Creating a book index

We need to create a book index, i.e., an **alphabetically sorted** list of words from the book, along with the page numbers.

Note: we’ll work with text files and instead of page numbers, we will print out line numbers.

```
if txt file:
    Hello, how are you?  Are you doing line?  line 1
    Are you in Rochester?  line 2
```

Parts of the idea:

1) for every word, find the lines the word is at  
   just like last week
   when we computed word counts,
   we can now compute a list of locations

2) sort the words  
   want a fast sort
   $O(n^2)$ like bubble and insert sort
   would be slow for large files
   key, value
   words, list of locations

   are → lines 1, 2
   doing → line 1
   line → line 1
   hello → line 1
   how → line 1
   in → line 2
   Rochester → line 2
   you → lines 1, 2
**Merge Sort**

We redefined the problem: having a (very long) list of words, how do we efficiently sort them alphabetically?

A *divide-and-conquer* idea:

Suppose 

\[ L = [1, 7, 6, 2, 4, 5, 8, 3] \]

\( \Rightarrow \) we'll be working with numbers bec. I am lazy

def MergeSort(L):
  if L is of size \( \leq 1 \):
    return L
  divide L into two lists, A, B
  of the same size (or one
  is longer by exactly 1 elem)

  Asorted = MergeSort(A)
  Bsorted = MergeSort(B)

  merge Asorted and Bsorted
  into Lsorted

How to divide:

let \( n = \) length of \( L \)
we can get \( A, B \) by slicing:

\[ A = L \text{ sliced up to } \left\lfloor \frac{n}{2} \right\rfloor \]
\[ B = L \text{ sliced from } \left\lceil \frac{n}{2} \right\rceil +1 \]

Note: the lecture notes split the list differently
(easier to implement on linked lists)

How to merge (Asorted, Bsorted):

Lsorted = empty list

while both Asorted and Bsorted are non-empty:
  if the first elem. of Asorted < first elem. of Bsorted:
    append it to Lsorted and remove the elem. from Asorted
  else:
    append the first elem. of Bsorted to Lsorted and remove it from Bsorted

return Lsorted
Merge Sort

Stack trace diagram for input [5,7,1,3,4,8,6,2]:

Running time:

- \( \leq 20 \cdot n \) steps in split + merge
- \( \leq 2 \cdot (20 \cdot \frac{n}{2}) = 20n \)
- \( \leq 4 \cdot (20 \cdot \frac{n}{4}) = 20n \)

\( O(n \log n) \) overall
Quick Sort - an alternative (the most commonly used) sorting algorithm
Quick Sort

Running time: