Review from Monday

• Recursive definitions and C++ functions
• The mechanism of recursion using activation records
• Recursion vs. iteration
• Rules for writing recursive functions
• Examples:
  – Printing the contents of the vector in reverse order
  – Binary search

Today’s Class: Two Examples

Both of these are hard to design without using recursion:

• Merge sort
• Nonlinear word search

Merge Sort

• Idea:
  – Split a vector in half
  – Recursively sort each half
  – Merge the two sorted halves into a single sorted vector

• We’ll work an example first to see how it is done.

• The recursive code, with everything provided except the merging function, is attached to the notes.
Merge

We will work on the (non-recursive) merge function together in class

- Start from sorted intervals, with subscript ranges \([\text{low}..\text{mid}]\) (the lower interval) and \([\text{mid}+1..\text{high}]\) (the upper interval).

- Merge these by copying the values into

  \[
  \text{scratch[low]} \ldots \text{scratch[high]}
  \]

- Ask yourself, which value can be first if the two intervals are sorted? Which value could be next?

- In a loop, the merging algorithm repeated chooses one value to add to \text{scratch}.

- At each step of the loop, there are only two possibilities: the first uncopied value from the lower interval and the first uncopied value from the upper interval.

- The copying ends when one of the two intervals is exhausted. Then the remainder of the other interval is copied into the scratch vector. Finally, the entire scratch vector is copied back.

- We will complete the code during lecture.

Thinking About Merge Sort

- Exploits the power of recursion! We only need to think about
  - Base case (intervals of size 1)
  - Splitting the vector
  - Merging the results

- We will insert \texttt{cout} statements into the algorithm and use this to try to understand what is happening.
Example: Word Search Without Straight Lines

- Recall the word search problem from the week 11 lab. Here’s some example data:

```
h e a n f u y a a d f j
c r a r n e r a d f a d
c h e n e n s s a r t r
k d f t h i l e e r d r
ch a d u f j a v c z e
d f h o e p r a d l f c
n e i c p e m r t l k f
p a e r m e r o h t r r
d i o f e t a y c r h g
da l d r u e t r y r t
```

- What happens when we no longer require the locations to be along the same row, column or diagonal of the grid, but instead allow the locations to snake through the grid? The only requirements are that

1. the locations are connected along the same row, column or diagonal, and
2. a location can not be used more than once in each word

- Can you find rensselaer? It is there. How about temperature? Close, but nope!

- The implementation of this is very similar to the week 11 lab solution except for the search technique once the first letter of a word is found.

- We will look at the code during lecture, and then consider how to write the recursive function.