Today

- Merge and Quick Sort over Linked Lists
- Algorithm Analysis (cont)

Assignments

- HW 4 out
- No class Thursday (10/12)
**Merge Sort over Linked Lists**

- We are performing merge sort “in place” by splicing and reattaching

We’ll use a helper function for the merge sort implementation:

```
Node* merge_sort(Node* left, int len)
```

This function will do **both** the splitting and merging

- `left` is the left-most node in the list to consider
- `len` is the number of nodes to consider

**The general (high-level) algorithm**

```
Node* merge_sort(Node* left, int len)
1. if len <= 1 return left
2. compute mid length and set up right pointer
3. left := merge_sort(left, mid)
4. right := merge_sort(right, len - mid)
5. merge the two lists left and right
6. return pointer to first node of merged list
```

```
void merge_sort()
1. head := merge_sort(head, node_count)
2. update tail ptr (by looping to last node)
```
Quick Sort over Linked Lists

• Again, implement quick sort by splicing and reattaching nodes

We'll use a helper function:

Node* quick_sort(Node* start, int len)

Function does both partitioning and recursive step

General High-Level Algorithm

Node* quick_sort(Node* start, int len)
1. if len <= 1 then return start
2. separate first node from rest of list (pivot node)
3. move nodes into a smaller and larger list (partition)
4. smaller := quick_sort(smaller, smaller_len)
5. larger := quick_sort(larger, larger_len)
6. attach smaller, pivot, and larger together
7. return head node of resulting list

void quick_sort()
1. head := quick_sort(head, node_count)
2. update tail pointer (by looping to last node)
Algorithm Analysis Example 3:

```cpp
bool duplicates(const int A[], int n)
{
    for (int i = 0; i < n; ++i)
        for (int j = i + 1; j < n; ++j)
            if (A[i] == A[j])
                return true;
    return false;
}
```

Q: Is there a “best case” and “worst case”?

- Worst case when no duplicates

Q: Only counting array comparisons, what is the worst-case as $T(n)$?

- Notice that:
  - $i = 0$, inner loop performs $n - 1$ comparisons
  - $i = 1$, inner loop performs $n - 2$ comparisons
  - ...
  - $i = n-1$, inner loop performs $n - n$ comparisons
- Giving (*)
  \[
  T(n) \leq \sum_{i=0}^{n-1} i = \frac{(n-1)n}{2} = \frac{1}{2}n^2 - \frac{1}{2}n
  \]

This is also worst-case “quadratic time” (i.e., $n^2$)

(*) Note: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$