Daniel Cender

Data Structures and Algorithms in C++

Spring 2019

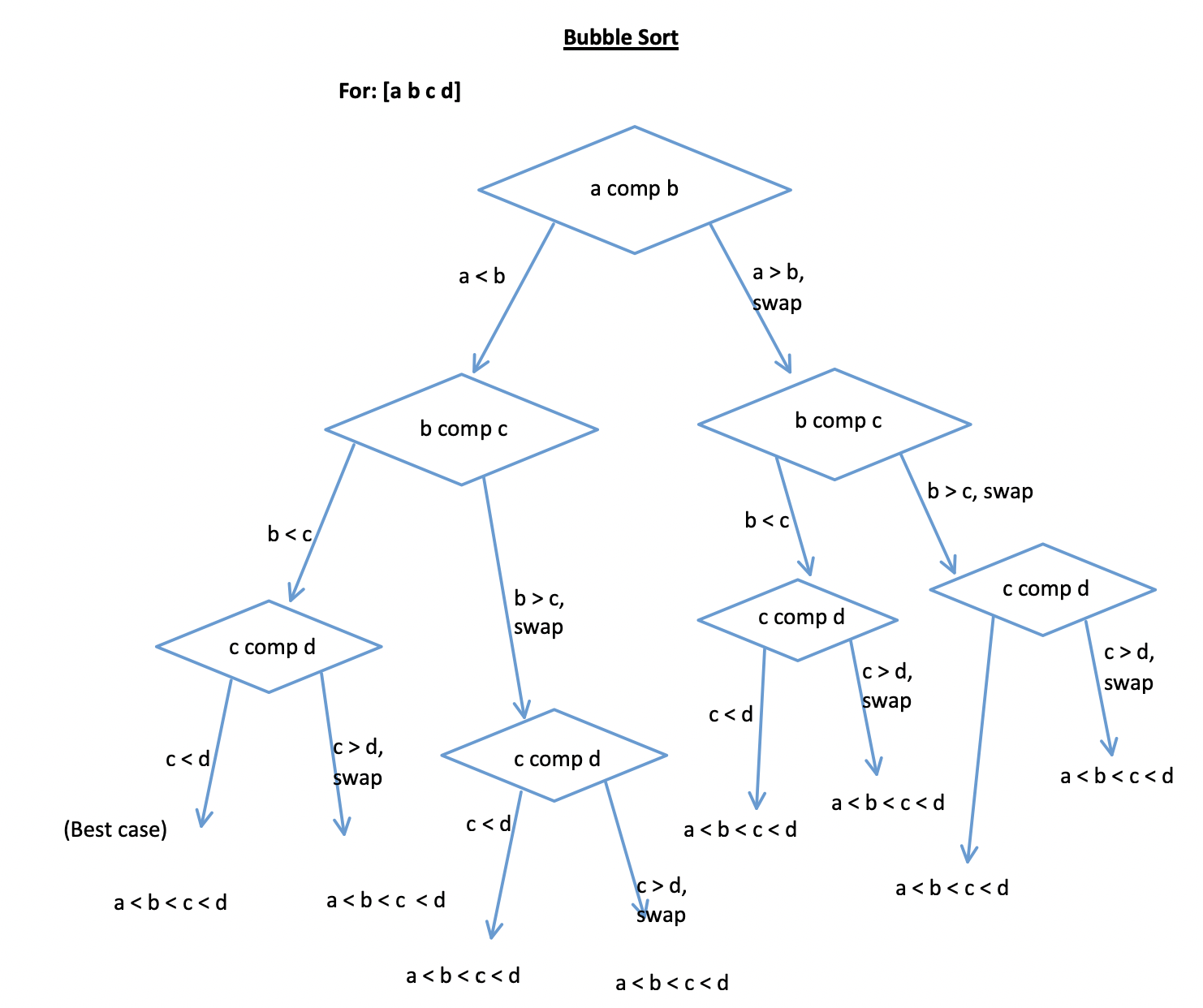
Week 6 CLC

I was assigned to answer question 3 for all sorts and build a decision tree for each sort for dataset: [a b c d].

**Question:** Is the sort easily adaptable to doubly linked lists?

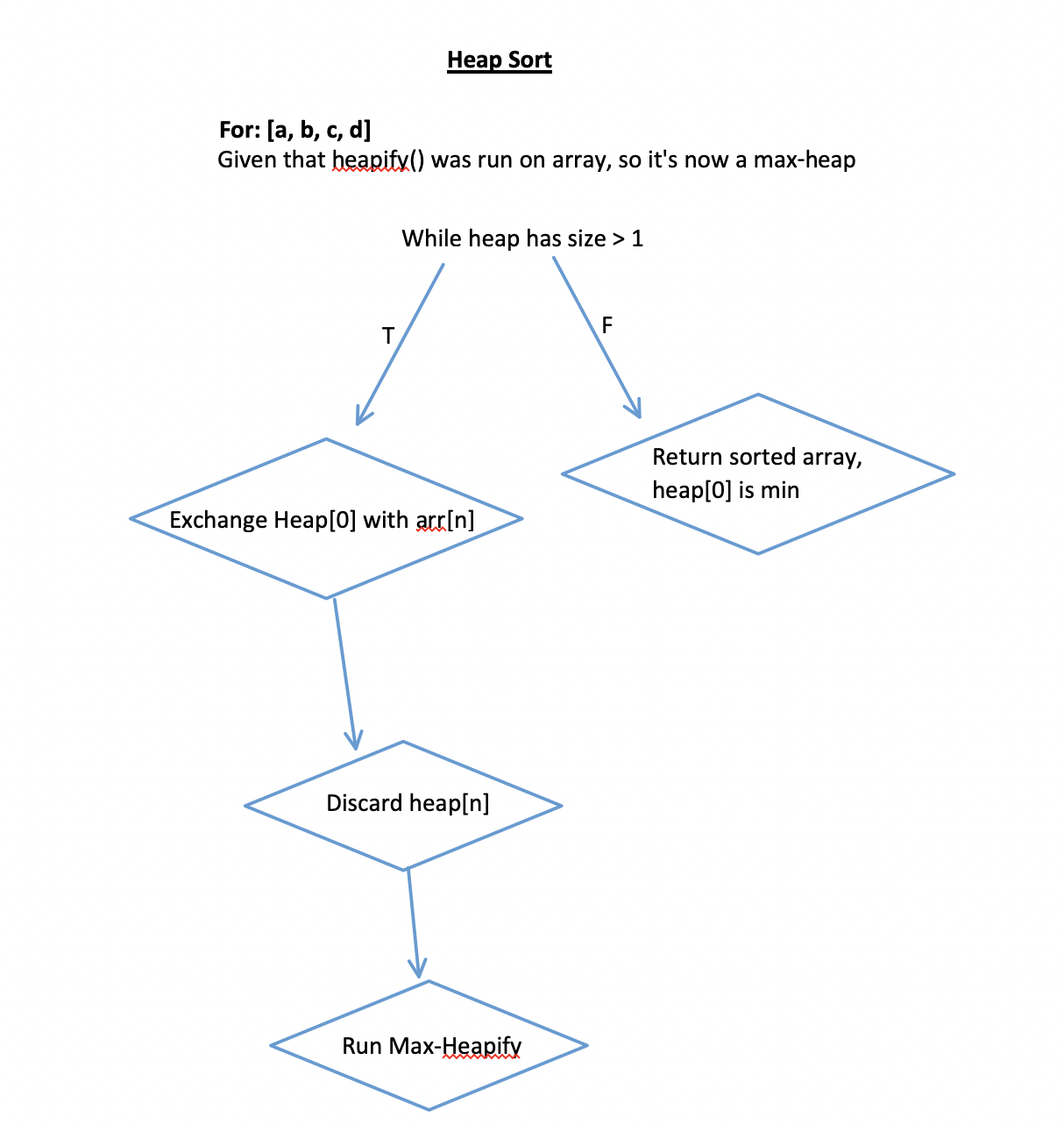
**For bubble sort:** This sort could easily be adapted to work with doubly linked lists. Having the linked pointers to previous and next nodes would reduce the overhead required by this algorithm if it ran against a large linked list.

**Decision Tree:**



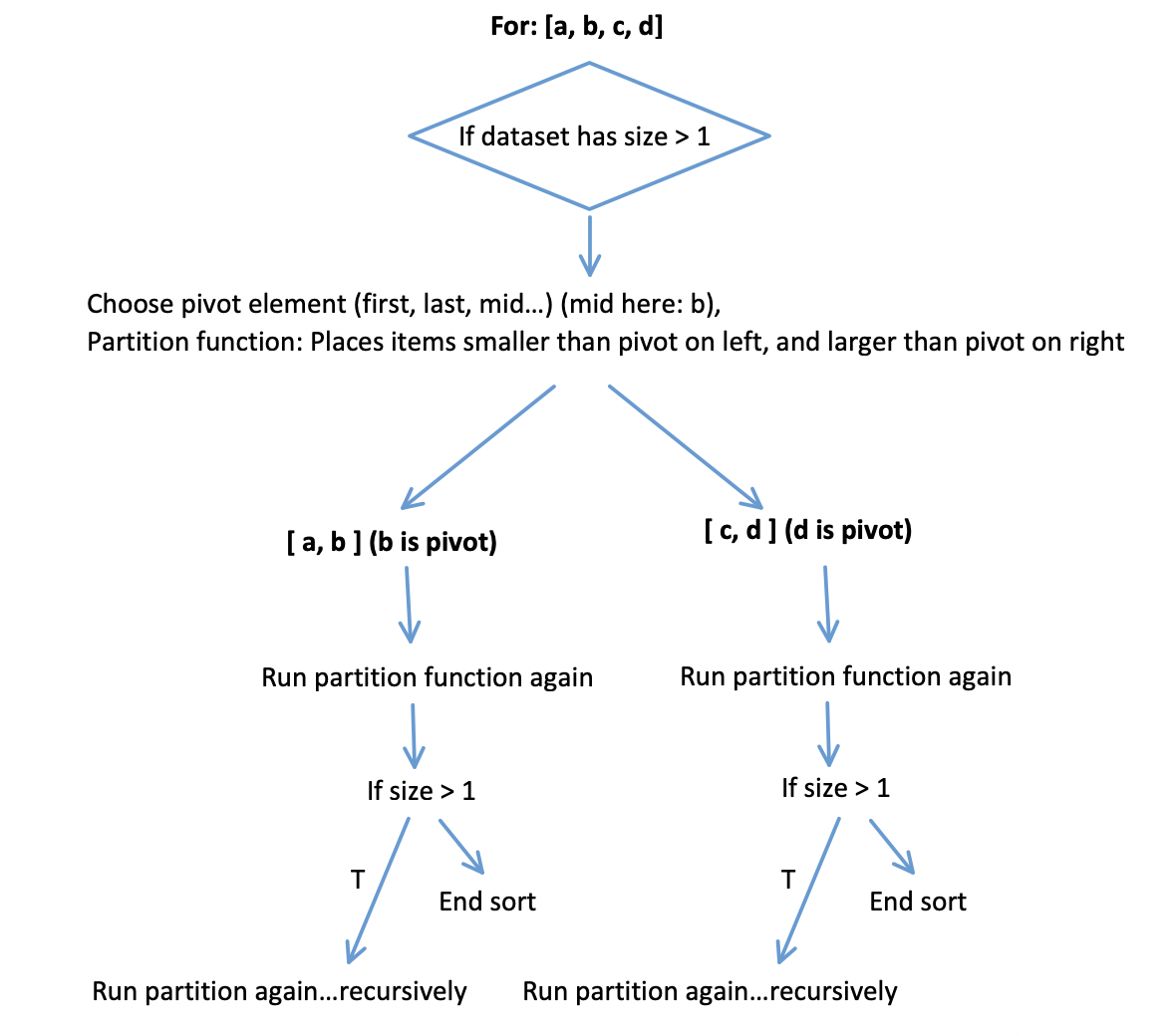
**For heap sort:** Linked lists are fairly-easily restructured into heaps of many kinds, so I could see this algorithm applying well to a doubly linked list.

**Decision Tree:**

****

**For quicksort:** I can see how it could be adapted to accommodate a doubly linked list, but that would seem to require more memory overhead and traversal than would give us the best runtime for this algorithm. Referencing [this GeeksForGeeks article](https://www.geeksforgeeks.org/quicksort-for-linked-list/), it seems that a doubly linked list can be sorted with Quicksort at the same worst-case runtime as an unsorted array.

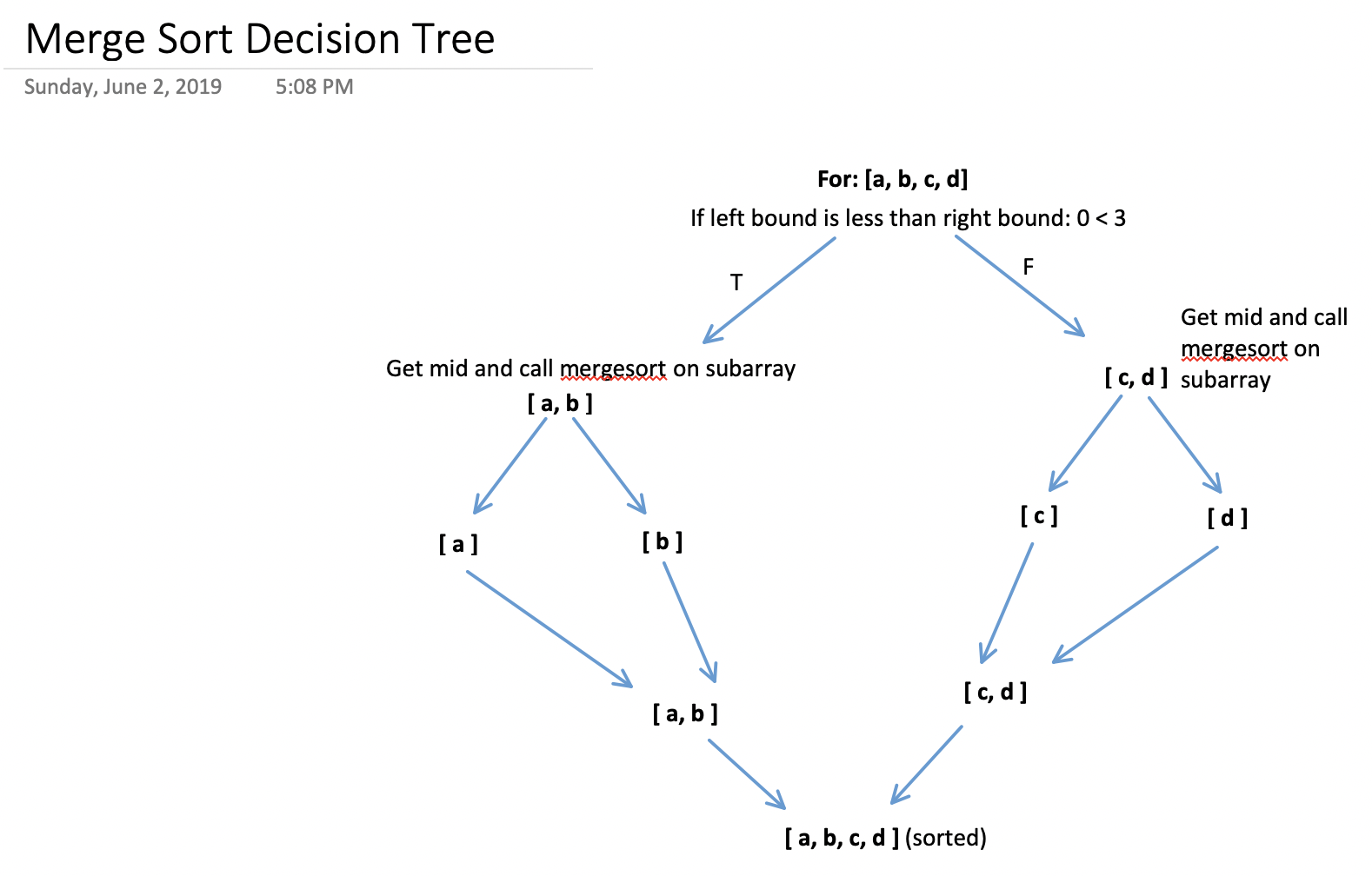
**Decision Tree:**

****

**For mergesort:** A merge sort can easily be adapted to work with linked lists, both singly and doubly linked, because this method requires sequential access, not random access to elements.

Reference content regarding this usage [here.](https://www.geeksforgeeks.org/merge-sort/)

**Decision Tree:**

****