CST-201

30 November 2018

Green Team

Jackie Adair

Zachary Chambers

BUBBLE SORT:

Is the sort Stable?

**Yes**

Is the sort easily adaptable to singly linked lists?

**Yes, Since the sort iterates up through the list, so it doesn’t require a pointer to the previous node.**

Is the sort easily adaptable to doubly linked lists?

**Yes, the sort iterates up the list therefore doesn’t utilize the previous pointer.**

What is the worst-case distribution for the sort?

**Data reversed from a sorted condition. This would make the complexity O(n2).**

What is the best-case distribution for the sort?

**Data in an already sorted condition. The complexity would be O(n-1)**

Decision Tree: data set [a b c d].

<Jackie is working on this one>

HEAP SORT:

Is the sort Stable?

**No**.

Is the sort easily adaptable to singly linked lists?

**A heap sort could be adapted to singly and doubly linked lists since it is based upon binary trees and creating binary trees using lists is a simple matter of changing the next pointer to right and left pointers. Singly linked lists would not have the benefit of a parent pointer, but it is not necessary**.

Is the sort easily adaptable to doubly linked lists?

**Like stated above, a heap sort could be adapted to a doubly linked list easily since the heap is based upon binary trees. The only benefit a doubly linked list has over the singly linked list architecture is a parent pointer which is not necessary for the most part.**

What is the worst-case distribution for the sort?

**When there are no identical elements and the root element must be moved down to the leaves n times.**

What is the best-case distribution for the sort?

**when there are identical elements and the root element must be moved down to the leaves n/2 times.**

Decision Tree: data set [a b c d].

QUICKSORT:

Is the sort Stable?

**No.**

Is the sort easily adaptable to singly linked lists?

**This sort would require some special handling to adapt to singly linked lists. The fact that you will be splitting the list into sub-lists recursively and that singly linked lists do not have a pointer to the previous node in the list creates a requirement to either create a second pointer to “chase” the iteration through the list and point at the previous node when searched for node is discovered, or iterate through the list using counters and double the iteration count in order to identify that previous node. This will increase complexity.**

Is the sort easily adaptable to doubly linked lists?

**Adapting this algorithm to a doubly linked list would pose no real issues since the list node contain the required pointers to split the list into sub-lists then merge them back together when sorted.**

What is the worst-case distribution for the sort?

**The worst-case distribution for quicksort is if in each call to the algorithm, the lowest or highest element is chosen for the bound. This prevents one of the arrays from participating in the split therefore creates a one-sided sort.**

What is the best-case distribution for the sort?

**The best-case distribution for quicksort is when the bound divides each array into equal sub-arrays. This equalizes the sort and gives the most efficiency.**

Decision Tree: data set [a b c d].

MERGESORT:

Is the sort Stable?

**Yes**

Is the sort easily adaptable to singly linked lists?

**Like the quicksort, the fact that mergesort would split the list into sub-lists creates a problem in that singly linked list have to have extra logic to track the previous nodes.**

Is the sort easily adaptable to doubly linked lists?

**Also like the quicksort, the mergesort would have very few issues being modified into a doubly linked list sorting algorithm simply because all of the pointers are in place to enable the list to be broken into sub-lists**.

What is the worst-case distribution for the sort?

**The worst-case distribution for this sort is if last element of one half precedes only the last element of the other half. This gives a complexity of O(n).**

What is the best-case distribution for the sort?

**The best case distribution for this sort is if the array is in order or if the elements in the right half precede the elements in the left half. O(n/2)**

Decision Tree: data set [a b c d].