Sorting race

https://www.toptal.com/developers/sorting-algorithms
CS 2230
CS II: Data structures

Sorting
Brandon Myers
University of Iowa
Today’s big ideas

• How long does it take to sort N items? We’ll use # of comparisons to count the running time.

• Consider various algorithms for sorting

• distinguish between best and worst case running time
Definition of a sort

The operator $\leq$ defines a total order over a collection of items. A sort w.r.t $\leq$ is a permutation of the items such that the order obeys $\leq$.

For example,
If $x_0 \leq x_1$ where $x_0$ is a string that comes before $x_1$ in the dictionary then

"cat" $\leq$ "catnip" $\leq$ "dog"
Wake up your brain

Write: You have a collection of 100 cats with nametags; how would you sort them alphabetically by name?

Clicker: running time for using your algorithm on N cats?

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
Selection sort

for i = 0 to N-1
    find the smallest item in [i, N-1]
    swap it with item i

see demo
http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
Peer instruction

Selection sort

for i = 0 to N-1
    find the smallest item in [i, N-1]
    swap it with item i

Running time to selection sort N elements in an array?

a) \( \Theta(N) \)
b) \( \Theta(N\log N) \)
c) \( \Theta(N^2) \)
d) \( \Theta(N^3) \)
e) \( \Theta(2^N) \)

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
Heap sort

1. For each item, insert it into a min heap.

2. Now call deleteMin, adding the item to the array, until no items are left in the heap.

(use the heap demo to simulate heap sort)
http://www.cs.usfca.edu/~galles/visualization/Heap.html
Heap sort

1. For each item, insert it into a max heap.

2. Now call deleteMax, putting the element in the back of the array, until no elements are left.

Running time to heap sort N elements in an array?

a) \( \Theta(\log N) \)
b) \( \Theta(N) \)
c) \( \Theta(N \log N) \)
d) \( \Theta(N^2) \)
e) \( \Theta(N^3) \)

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
Heap sort best/worst case

Why no better than NlogN?

• a good case? “the input is already sorted”
  • As we build the heap, no bubble up will be necessary, so O(1) per element, so O(N) total for build (so far so good: better than O(NlogN) in the worst case)
  • When we deleteMin, we are forced to swap the root with the last element and bubble it back down, so O(logS) per element (where S is current size). Summation over N elements gives O(NlogN)
Memory usage of heap sort

Requires $O(N)$ additional memory to build the heap (an extra copy of the data)

to use $O(1)$ extra memory instead, you can do heap sort *in-place*

- interpret the original array as a binary tree
- turn it into a valid max binary heap
- when removing elements, utilize the end of the array

demo

http://www.cs.usfca.edu/~galles/visualization/HeapSort.html
Insertion sort

start with an empty output linked list

for i = 0 to N-1
    item = input[i]
    insert item into the linked list in sorted order
Peer instruction

Insertion sort

for i = 0 to N-1
    item = input[i]
    swap item forward until in sorted order

Running time to insertion sort N elements in an array? Best/worst case?

a) \( \Omega(1) \), \( O(N) \)
b) \( \Omega(1) \), \( O(N^2) \)
c) \( \Omega(N) \), \( O(N) \)
d) \( \Omega(N) \), \( O(N^2) \)
e) \( \Omega(N^2) \), \( O(N^2) \)

https://b.socrative.com/login/student/
CS2230A ids 1000-4999
CS2230B ids 5000+
Insertion sort, in-place

for i = 0 to N-1
    item = input[i]
    swap item forward until in sorted order

http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
Mergesort

break array into two halves, recursively mergesort each one

base case swaps two items into order and returns

merge the left and right half
Merge sort in-place

http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
Peer instruction

Merge sort

Running time to mergesort N elements in an array?

a) $\Theta(\log N)$
b) $\Theta(N)$
c) $\Theta(N \log N)$
d) $\Theta(N^2)$
e) $\Theta(N^3)$
Today’s big ideas

• How long does it take to sort N items? We’ll use # of comparisons to count the running time.

• Consider various algorithms for sorting

• distinguish between best and worst case running time