Sorting race

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

Comparison sorting
Brandon Myers
University of Iowa
Today’s Learning Objectives

• Execute several *comparison sorting* algorithms
• Analyze the running time several comparison sorting algorithm
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?
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/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
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
Execute a variety of comparison sorting algorithms

5 6 9 27 15 24 22 29 14 16 13

We’re somewhere in the middle of running selection sort on this array. What two items will be swapped next?

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Analyze the running time several comparison sorting algorithm

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/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
for fun

https://www.youtube.com/watch?v=kPRA0W1kECg
more serious

http://sortbenchmark.org/

<table>
<thead>
<tr>
<th>App</th>
<th>Metric</th>
<th>2016, TB(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GraySort</td>
<td>Sort rate (TBs / minute) achieved while sorting a very large amount of data (currently 100 TB minimum).</td>
<td>100 TB in 134 Seconds</td>
</tr>
<tr>
<td>CloudSort</td>
<td>Minimum cost for sorting a very large amount of data on a public cloud. (currently 100 TB).</td>
<td>100 TB for $144</td>
</tr>
<tr>
<td>MinuteSort</td>
<td>Amount of data that can be sorted in 60.00 seconds or less.</td>
<td>37 TB</td>
</tr>
<tr>
<td>JouleSort</td>
<td>Amount of energy required to sort either $10^8$, $10^9$, $10^{10}$, or $10^{12}$ records (10 GB, 100 GB, 1 TB, or 100TB).</td>
<td>168,242 Joules for $10^{10}$</td>
</tr>
<tr>
<td>PennySort</td>
<td>Amount of data that can be sorted for a penny's worth of system time.</td>
<td>2011, 286 GB</td>
</tr>
</tbody>
</table>

solutions to this competition consider both hardware and software together

512 servers in parallel

- Data into segments
- Radix sort each segment, 1 per core (20 cores)
- Send each record to a server based on its range
- Merge sort to finish
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

6 27 5 15 24 22 29 14 16 13 9
Analyze the running time several comparison sorting algorithm

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/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
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) *additional* 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
Insertion sort, in-place

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

http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
Execute a variety of *comparison sorting algorithms*

17 23 30 1 10 12 2 26 28 8 7 21

Run in-place insertion sort on this array.

Clicker: How many *swaps* total were required?

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Analyze the running time several comparison sorting algorithm

Insertion sort, in place

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

Running time to insertion sort N elements in an array?

Best case, worst case

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

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Mergesort

break array into two halves, recursively mergesort each one

base case is two items

*merge* the left and right half
Merge sort in-place

http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
Execute several *comparison sorting* algorithms

Run in-place merge sort on this array. Write a new line each time elements change position.

12 2 26 28 8 7 21 19 25 11 18 3 4 20
Analyze the running time several comparison sorting algorithm

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) $\Theta(N^3)$

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
quicksort

1. pick a “pivot”
2. for each item less than “pivot” put it in the left side of the array
3. for each item greater than or equal to “pivot” put it in the right side of the array
4. call quicksort on left, call quicksort on right
quick sort visualization (uses in-place quick sort)

http://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
We called quicksort on the above array. Let’s suppose we always *pick the second element of the array as the pivot* (i.e. 80 above). What will be the next two recursive calls to quicksort (assuming we call it on left then right)?

- a)  
  50  38  12  44  
  99  

- b)  
  50  38  12  44  
  80  99  

- c)  
  50  38  12  44  
  12  

- d)  
  12  38  44  50  
  12  44  

- e)  
  12  38  44  50  
  80  99  

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Peer instruction

What value would make the best pivot?

a) A random value
b) The minimum of the items
c) The mean of the items
d) The value of the first item
e) The median of the items

https://b.socrative.com/login/student/
room CS2230X ids 1000-2999
room CS2230Y ids 3000+
Running time of quick sort

assuming the worst pivot every time?

assuming the best pivot every time?
Picking a good pivot cheaply?

many proposals

These three are okay and take care of the sorted/almost sorted cases
- median of three
- random index
- middle index of the partition
- ...

What if there are many duplicates of a value (this is common!)
What if you are sorting Retweets by original poster?)
- create a third bucket for the common value; don’t need to sort it since it is already sorted
Today’s Learning Objectives

• Execute several *comparison sorting* algorithms
• Analyze the running time several comparison sorting algorithm