<http://bigocheatsheet.com/>

<https://www.geeksforgeeks.org/radix-sort/>

**From:** Matt Emerson   
**Sent:** Thursday, April 12, 2018 5:48 PM  
**To:** Matt Emerson  
**Subject:** RE: Sorting

Need to specify space as well

**From:** Matt Emerson   
**Sent:** Thursday, April 12, 2018 5:48 PM  
**To:** Matt Emerson  
**Subject:** Sorting

<https://www2.cs.duke.edu/courses/spring05/cps100/class/15_Sorting/sorting_cheat.html>

|  |  |
| --- | --- |
| Term | Definition |
| Average Case Time | Given an arbitrary input, expected run time |
| Worst Case Time | For a particular generate case, how bad will the algorithm perform |
| Best Case Time | For a particularly benevolent input case, best case performance  Always O(n); cause you have to compare each number once |
| Situations where useful | For what inputs and applications |
| Situations after ith pass | What the vector looks like after going through the vector I times |

Basic Sorts

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | What is it? | Situation After ith pass | Average Case Time | Worst Case Time | Best Case Time | Situation where useful |
| Insertion | Insert an as-yet-unprocessed into a sorted list of records processed so far  (NOTE: does it at the end) | First I elements sorted | N^2 | N^2 ; reverse sorted order takes about twice as long | N  When already in sorted order | Simple sort, easy to implement, good when the number of inversions is small |
| Shell’s Sort | Like insertion sort, but allows you to stride through the vector to find where to go. | First I elements sorted | N^1.5  When we divide the stride by 2.2 each time | With stride of 1, it’s just insertion sort | N  Stride=1 and sorted list | Good for general purpose sort when insertion sort has trouble.  A bit more overhead involved |
| Selection Sort | Find the minimum of the remaining elements each itme | First I elements are sorted in the proper position | N^2 | N^2  Not sensitive to input! | N^2 | Insensitive to data; easy to estimate how long it will take |
| Bubble Sort | On each pass, if two elements are out of order swap them.  You’re done when you can go through the vector witht no swaps. | Last I sorted and in the proper position. | N^2  Sweeping through the first n-I each time regardless | N^2 | N  When sorted or mostly sorted (elements a place or two away) | Probably none |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Divide and Conquer Sorts

Let’s do better than N^2 or even N^1.25!

BASE CASE: end of the recursion

DIVIDE STEP: Split up the problem into more manageable pieces

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Base Case | Divide Step | Average Case Time | Worst Case Time | Best Case Time | Situation where useful |
| Merge Sort | 0 or 1 element list; just return the list | Mergesort each half and then merge the two in O(n) time | Nlogn  We split the problem in two every time regardless | nlogn | nlogn | Good for working in parallel, since every mergesort is working on a different portion of the vector, not sensitive to input |
| Quick Sort | Is our portion of the vector close to sorted?  If so, just use insertion sort | Choose a pivot and divide the vector into elements smaller then the pivot and elements greater than it | Nlogn  On everage, the pivot will evenly divide the vector | N^2  The pivot gives us uneven partitions each time | N  If given a pretty well sorted vector, just call insertion sort on it immediately | Fastest practical sort, finely tuned versions exist for most systems. |

Any sort which relies on just comparisons, you will always have to do nlogn comparisons worst case.  Why is this?

“Restricted Sorts”

If we know something about the data

RESTRICTION: in what form do the input elements need to be

K-FACTOR: how does the form of the input elements affect running time

**Distribution set**

Useful when records are in a small range of integers of other cardinal values.

Pseudo-code

Basically, define a vector in the right sorted order, convert the thing to be counted into an integer and start incrementing values on the integer vector.  Can just displacement from the value.  Ultimately, you create a new vector of the incoming set and put the sorted pieces into it.

For each input, put it into a bucket (b), and then expand the buckets back into the output (k)

**Radix Sort**

* For each digit going least significant to most significant
* For all n numbers
  + Place number in proper bin
* Distribute sort each bin
* Concatenate sorts results from each bin and repeat

Nd = is the initial iteration, iterate over each digit (d) in each input value (n)

Dk = for each digit, concatenate the values.

**Bucket Sort**

Bucket sort assumes that the input is generated by some random process which uniformly distribute the numbers over some interval.  Divide interval into n equally sized buckets.

* For each element in the original vector
  + Insert sorted into proper bucket O(1+length of list in bucket)
* Concatenate the lists from all of the buckets together

If we have an input of size z, we have n buckets.  On average each bucket will have 1 element so the length of the list in each bucket is 1.  We can then do insert sort for a particular bucket in constant time.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Restriction | K Factor | Average Case Time | Worst Case Time | Best Case Time | Situation where useful |
| Distribution Sort | Elements in integral range L to U | K=U-L | K+N  Not sensitive to input beyond values of L and U | K+N  Remember that k can be quite large.  Continue 2^32 for ints. | K+N | Have restricted range of inputs like radix sort |
| Radix Sort | The keys are sequences of fixed-size “digits” | D is the number of digits.  K is the range of each digit.  For example, for decimal numbers under 1000, d=3, k=10. | Dn+kd | Dn+kd | Dn+kd | Things like bounded integers of bounded length strings. |
| Bucket Sort | Uniformly distributed floating point numbers in some interval | K=length of list in bucket so general running time is O(1+k) | n | N^2  (everything in the same bucket) | n | When we’re sorting random numbers of randomly generating values of some kind |