Cheatography

Sorting algorithms Cheat Sheet

by pryl via cheatography.com/66402/cs/16808/

Sorting algorithms and Methods

Sorting algorithms	*Methods*
Bubble sort	Exchanging
Heapsort	Selection
Insertion sort	Insertion
Introsort	Partitioning & Selection
Merge sort	Merging
Patience sorting	Insertion & Selection
Quicksort	Partitioning
Selection sort	Selection
Timsort	Insertion & Merging
Unshuffle sort	Distribution and Merge

Best and Worst Case

Algorithms	*Best Case*	*Worst Case*
Bogosort	n	∞
Bubble sort	n	n^2
Bucket sort (uniform keys)	-	n^2k
Heap sort	n log n	n log n
Insertion sort	n	n^2
Merge sort	n log n	n log n
Quick sort	n log n	n^2
Selection sort	n^2	n^2
Shell sort	n log n	$n^{4/3}$
Spreadsort	n	n(k/s+d)
Timsort	n	n log n
Unshuffle sort	n	kn

Insertion sort

```
function insertionSortR(array A, int n)
     if n>0
        insertionSortR(A,n-1)
        x ← A[n]
        j ← n-1
        while j >= 0 and A[j] > x
            A[j+1] ← A[j]
            j ← j-1
        end while
        A[j+1] ← x
     end if
 end function
```

Merge sort

```
function merge_sort(list m)
    // Base case. A list of zero or one elements is
sorted, by definition.
    if length of m ≤ 1 then
        return m
    // Recursive case. First, divide the list into
equal-sized sublists
    // consisting of the first half and second half of
the list.
    // This assumes lists start at index 0.
    var left := empty list
    var right := empty list
    for each x with index i in m do
        if i < (length of m)/2 then
            add x to left
        else
            add x to right
    // Recursively sort both sublists.
    left := merge_sort(left)
    right := merge_sort(right)
    // Then merge the now-sorted sublists.
    return merge(left, right)
```

Bogosort

```
while not isInOrder(deck):
    shuffle(deck)
```

Bucket sort

```
function bucketSort(array, n) is
  buckets ← new array of n empty lists
  for i = 0 to (length(array)-1) do
    insert array[i] into buckets[msbits(array[i], k)]
  for i = 0 to n - 1 do
    nextSort(buckets[i]);
  return the concatenation of buckets[0], ....,
buckets[n-1]
```

Resources

https://en.wikipedia.org/wiki/Sorting_algorithm#Comparison_of_algorithms

http://bigocheatsheet.com

Cheatography

Sorting algorithms Cheat Sheet

by pryl via cheatography.com/66402/cs/16808/

Sorting algorithm complexities

Algorithms	*Average Case*	*Memory complexity*
Bitonic sorter	$\log^2 n$	$n \log^2 n$
Bogosort	$n \times n!$	1
Bubble sort	n^2	1
Bucket sort (uniform keys)	n+k	nk
Burstsort	n(k/d)	n(k/d)
Counting sort	n+r	n+r
Heap sort	n log n	1
Insertion sort	n^2	1
Introsort	n log n	log n
LSD Radix Sort	n(k/d)	$n+2^d$
Merge sort	n log n	n
MSD Radix Sort (in-place)	n(k/d)	2^d
Patience sort	-	n
Pigeonhole sort	$n+2^k$	2^k
Quicksort	n log n	log n
Selection sort	n^2	1
Shell sort	Depends on gap sequence	1
Spaghetti sort	n	n^2
Spreadsort	n(k/d)	$(k/d)2^d$
Stooge sort	$n^{(\log 3/\log 1.5)}$	n
Timsort	n log n	n

Bubble sort

```
procedure bubbleSort( A : list of sortable items )
    n = length(A)
    repeat
        swapped = false
        for i = 1 to n-1 inclusive do
            if A[i-1] > A[i] then
                swap(A[i-1], A[i])
                swapped = true
            end if
        end for
        n = n - 1
    until not swapped
```

Bubble sort (cont)

```
end procedure
```

Quicksort

```
algorithm quicksort(A, lo, hi) is
    if lo < hi then
        p := partition(A, lo, hi)
        quicksort(A, lo, p - 1 )
        quicksort(A, p + 1, hi)
algorithm partition(A, lo, hi) is
    pivot := A[hi]
    i := lo
    for j := lo to hi - 1 do
        if A[j] < pivot then
            swap A[i] with A[j]
            i := i + 1
    swap A[i] with A[hi]
    return i
```

Selection sort

```
procedure selection sort
   list : array of items
   n : size of list
   for i = 1 to n - 1
   / set current element as minimum /
      min = i

      / check the element to be minimum /
      for j = i+1 to n
         if list[j] < list[min] then
            min = j;
         end if
      end for
      / swap the minimum element with the current
element/
      if indexMin != i then
         swap list[min] and list[i]
      end if
   end for

end procedure
```