# TIME/SPACE COMPLEXITY LIST!

(still a work in progress)

Hey guys, I'm probably wrong about some of these. If you see a mistake, feel free to correct me. Also, there’s probably plenty missing at this stage, feel free to add it to the list as well.

| ADT/Thing | Operation | Time (includes best/avg/worst) | Space (if applicable) |
| --- | --- | --- | --- |
| Stack | push | O(1) |  |
|  | pop | O(1) |  |
| Vector (array) | insert | O(n) |  |
|  | delete | O(n) |  |
|  | access | O(1) |  |
| Vector (list) | insert | O(1) |  |
|  | delete | O(1) |  |
|  | access | O(n) |  |
| Heap | insert | O(log n) |  |
|  | min | O(1) |  |
|  | remove min | O(log n) |  |
| Insertion-sort | insert | O(n) |  |
|  | find/remove | O(1) |  |
| Selection-sort | insert | O(1) |  |
|  | find/remove | O(n) |  |
| Map (list-based) | put | O(n) (checks for key) |  |
|  | get | O(n) |  |
| Hash Map | insert | O(n) worst, O(1) expected |  |
|  | search | O(n) worst, O(1) expected |  |
|  | remove | O(n) worst, O(1) expected |  |
| Unsorted List Dictionary | insert | O(1) |  |
|  | remove | O(n) |  |
|  | find | O(n) |  |
| Sorted Array Dictionary (search table) | insert | O(n) |  |
|  | remove | O(log n) |  |
|  | find | O(log n) |  |
| Skip List | insert | O(log n) | O(n) expected |
|  | find | O(log n) |  |
|  | remove | O(log n) |  |
| Priority queue (list, unsorted) | insert | O(1) |  |
|  | removeMin/min | O(N) |  |
| Priority queue (list, sorted) | insert | O(N) |  |
|  | removemin/min | O(1) |  |
| Binary Search Tree | find | O(n) if max height | O(n) |
|  | insert | O(n) |  |
|  | remove | O(n) |  |
| AVL Tree | restructure | O(1) | O(n) |
|  | find | O(log n) because balanced |  |
|  | insert | O(log n) |  |
|  | remove | O(log n) |  |
| Splay | splaying | O(h) worst, O(log n) amort.  Freq. revisits mean < log n |  |
|  | add, remove, find | same as splaying |  |
| (2,4) Trees | search | O(log n) |  |
|  | insert | O(log n) |  |
|  | delete | O(log n) |  |
| Red-Black Trees | search | O(log n) |  |
|  | insert | O(log n) |  |
|  | delete | O(log n) |  |
| (a, b) Tree | operations | O(log n / log a)  (log n / log b) |  |
| Merge Sort | sort | O(n log n) |  |
| Quick-Sort | sort | O(n log n) expected |  |
| Bucket-Sort | sort | O(n + N) (N: bucket list size) |  |
| Lexicographic Sort | sort | O(dT(n)) F(n) is stable sort fn |  |
| Radix Sort | sort | O(dT(n + N)) |  |
| Quick-Select | select | O(n) expected  O(n^2) worst case |  |
| Brute-Force Pattern Matching |  | O(nm)  n = size of text  m = size of pattern |  |
| Boyer-Moore | Boyer-Mooring | O(mn + s)  s = size of alphabet |  |
| Knuth-Morris-Pratt |  | O(n + m) |  |
| Trie/Suffix Trie | insert/delete/search | O(dm)  m = size of string param.  d = size of alphabet | O(n)  n = number of strings |
| Depth-first search | search | O(n + m) |  |
| Dijkstra’s Algorithm | path-finding | O((n+m) log n)  (don’t know what these mean atm) |  |