|  |  |
| --- | --- |
| Samuel Kang  2/22/2023 | |
| Searching & Sorting Algorithms | * Linear search – go from first to last in array, doesn’t have to be sorted * Linear search time complexity – O(n) * Binary search – array has to be sorted in input and compare elements searching for and mid * In binary search, if search value is lower than middle, ignore the right side (vice versa if search value higher) * Choose element at index (0 + array.length)/2 (rounded to whole number) * Binary search time complexity: log(n), how many while loops * Best case time complexity: O(1) * Binary Search using Binary Recursion * int BinarySearch(int key, int[] A, int LI, int HI) {   if (LI > HI) then return -1;  mid = (LI + HI)/2;  if (key == A[mid]) return mid;  else if (key < A[mid]) return BinarySearch(key A, LI, mid -1);  else return BinarySearch(key, A, mid +1, HI);  }   * Base case – minimum possible case in recursion * Recursion is computationally expensive than simple solution * Multiple activation frames created, stacks * Stack – LIFO (last in first out) nodes   Sorting Algorithms   * Arrange list of elements in some order * Nested for loops – time complexity: O(n^2), sorting – O(nlog(n)) * Types: * Bubble Sort – the worst algorithm to think of * Selection Sort * Insertion Sort * Merge Sort * Quick Sort * In place algorithm – don’t need other arrays to store stuff; don’t need more space   Bubble Sort – traverse a collection of elements   * Move from front to the end * “Bubble” the largest value to the end using pair-wise comparisons and swapping * Time Complexity of Bubble Sort – O(n^2)   Selection Sort: Informal   * Worst case of Selection Sort: time complexity of O(n^2)   Insertion Sort   * Divide two parts: sorted and unsorted |