Runtime Complexity

• Algorithms take time to run
• Clock time varies
  • Can vary based on input
  • Can vary based on number and kind of steps
• Typically talk about runtime in an abstract sense
  • Big $O$ – worst case
  • Big $\Omega$ - best case
  • Big $\theta$ - average case
Different Times

\(O(1)\) constant
\(O(\log n)\) logarithmic
\(O(n)\) linear
\(O(n \log n)\) "n log n"
\(O(n^2)\) quadratic
\(O(n^3)\) cubic
\(n^{O(1)}\) polynomial
\(2^{O(n)}\) exponential
Code Example: O(1)

```c
struct node* push (struct node * head, int n) {
    struct node *temp = malloc(sizeof(struct node));
    temp -> val = n;
    temp->next = head;
    head = temp;
    return head;
}
```
Code Example: O(n)

```c
int length(struct node *head) {
    int n=0;
    while(head != NULL) {
        n++;
        head = head->next;
    }
    return n;
}
```
Code Example: Code $O(n^2)$

```c
void bubble_sort(struct node *head, int size) {
    ...  
    int iteration, i;
    for(iteration=1; iteration<size; iteration++) {
        for(i=0; i<size-iteration; i++) {
            if(current->val > current->next->val) {
                //swap values
            }
            //move current to next node
        }
        current = head;
    }
}
```
int binarySearch(const int list[], int length, int item) {
    int first = 0, last = length - 1, mid;
    int found = 0;
    while (first <= last && found == 0) {
        mid = (first + last) / 2;
        if (list[mid] == item)
            found = 1;
        else if (list[mid] > item)
            last = mid - 1;
        else
            first = mid + 1;
    }
    if (found == 1)
        return mid;
    else
        return -1;
} //end binarySearch
Binary Search – CS Unplugged

• https://www.youtube.com/watch?v=iDVH3oCTc2c
Final and Sorting

• Due Thursday, June 14, 2 pm to Canvas
• Posted Sunday
• Four sorting algorithms
  • Merge Sort, Selection Sort, Bubble Sort, Insertion Sort
  • Benchmark and compare
• Reflection survey
• https://www.youtube.com/watch?v=kPRA0W1kECg