SEARCHING AND SORTING

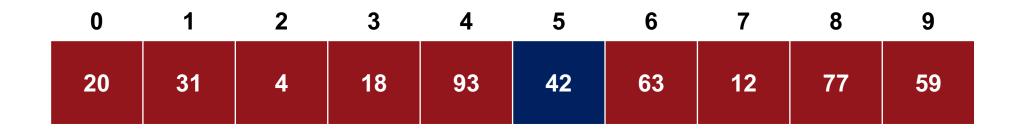
Searching

Concept

to locate a value within a data structure (a collection of values)

Example

```
int a[10] = { 20, 31, 4, 18, 93, 42, 63, 12, 77, 59 } // an array of 10 values cout << search(a, size, 42); // return the index 5
```



Sorting

Concept to organize values within a data structure

Example int a[10] = { 20, 31, 4, 18, 93, 42, 63, 12, 77, 59 } // an array of 10 values sort(a, 10); // some sort function

	0	1	2	3	4	5	6	7	8	9
UNSORTED	20	31	4	18	93	42	63	12	77	59
	0	1	2	3	4	5	6	7	8	9
SORTED	4	12	18	20	31	42	59	63	77	93

Linear Search

Concept

search for the number 42

0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59
0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59
0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59
0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59
0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59
0	1	2	3	4	5	6	7	8	9
20	31	4	18	93	42	63	12	77	59

Linear Search

Concept

search from the start to the end of the array for a value O(n) time complexity

Example

```
int getIndex(int *a, const int &SIZE, int value) {
    for(int i=0; i<SIZE; ++i) {
        if(a[i] == value) { return i; }
        return -1;
        // if no value found return -1</pre>
// if no value found return -1
```

Binary Search

Example search for the number 77 in a sorted array by repetitively halving the search zone

start	middle	enc
0	4	9
5	7	9
8	8	9

77 found at index 8

0	1	2	3	4	5	6	7	8	9
4	12	18	20	31	42	59	63	77	93
					5	6	7	8	9
					42	59	63	77	93
					5		7	8	9
								77	93

Binary Search

```
Concept
           search a sorted array by repetitively halving the search zone
           O(log n) time complexity
Example
           int getIndex(int *a, const int &SIZE, int value) {
                                                                   // parameters array, size, value
                                                                   // set start/end of zone
               int start=0, end=size-1;
               while( start <= end ) {
                                                                   // repetitively halve zone
                   int middle = ( start + end ) / 2;
                                                                       find middle
                   if( a[middle] = value ) { return middle; }
                                                                   // if value is middle return
                   else if( a[middle] < value) { start = middle+1; }//
                                                                       halve zone to the right
                   else { end = middle-1; }
                                                                       halve zone to the left
               return -1;
                                                                   // value not found
```

Insertion Sort

Concept progressively insert elements into a sorted subset

	0	1	2	3	4
SORT 12	73	12	43	25	9
	0	1	2	3	4
SORT 43	12	73	43	25	9
	0	1	2	3	4
SORT 25	12	43	73	25	9
	0	1	2	3	4
SORT 9	12	25	43	73	9
	0	1	2	3	4
SORTED	9	12	25	43	73

Insertion Sort

```
Concept
            progressively insert elements into a sorted subset
            O(n<sup>2</sup>) time complexity
Example
            void sort(int *a, int size) {
                                                             // value and index variables
                int v, index;
                for(int i=1; i<size; ++i) {
                                                             // iterate from second to end
                    v = a[i];
                                                             // store current in v
                                                             // store i in index
                    index = i:
                                                             // repeat if previous > v and index > 0
                    while( index>0 && a[index-1]>v ) {
                        a[index] = a[index-1];
                                                             // shift previous to current
                                                             // decrement index
                        --index;
                    a[index] = v;
                                                             // insert value at index
```

Selection Sort

Concept

progressively swap minimum value from unsorted to sorted subset

	0	1	2	3	4
SORT 9	73	43	12	9	25
	0	1	2	3	4
SORT 12	9	43	12	73	25
	0	1	2	3	4
SORT 25	9	12	43	73	25
	0	1	2	3	4
SORT 43	9	12	25	73	43
	0	1	2	3	4
SORTED	9	12	25	43	73

Selection Sort

```
progressively swap minimum value from unsorted to sorted subset
Concept
            O(n<sup>2</sup>) time complexity
            void sort(int *a, int size) {
Example
                 for(int i=0; i<SIZE-1; ++i) {
                                                  // iterate from start to second to last (0 to size-2)
                     int min = i;
                                                   // set minimum index to current index
                     for(int j=i+1; j<SIZE; ++j) { //
                                                       iterate from i+1 to last
                                                           find minimum value of i+1 to last
                         if( a[j] < a[min] ) { //
                             min = j;
                     if( min != i ) {
                                                   // if min index is not current index
                                                           swap min and current
                         int temp = a[i];
                                                   //
                         a[i] = a[min];
                         a[min] = temp;
```

Insertion vs. Selection Sort

Insertion Sort efficiency increases when data is previously partially sorted

generally faster than selection sort

Selection Sort optimal for data stored on flash memory due to less writing of data