BRAC UNIVERSITY

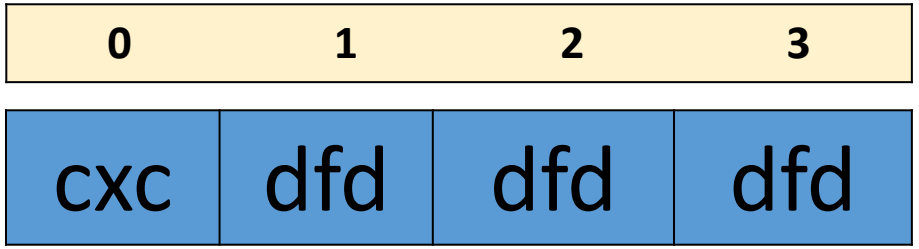
Inspiring Excellence

CSE221

Algorithms: *Linear Search*

Linear Search

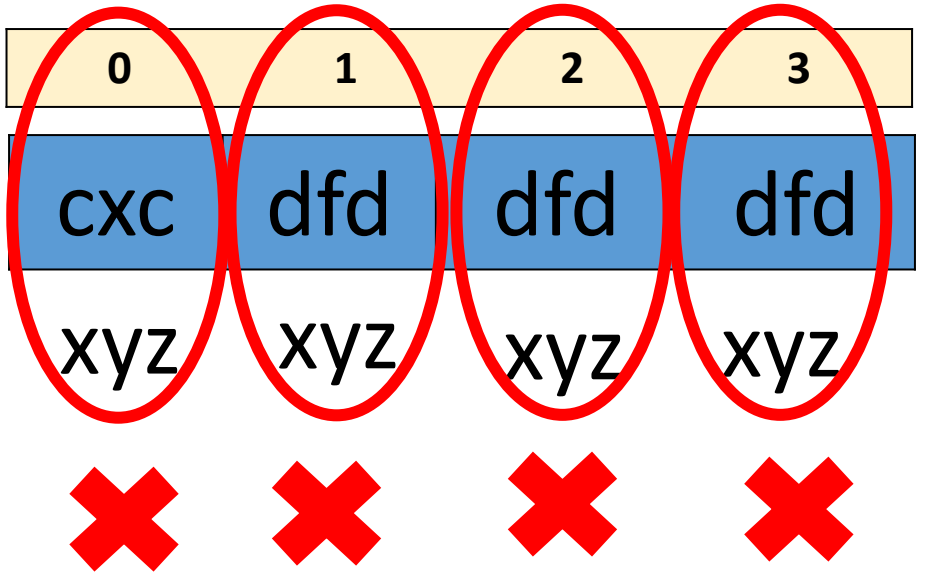
- The most basic searching algorithm
- Given an item to search from a bucket of items, we search ONE BY ONE.
- If we find then bingo! Else we look till the end of bucket.
- Example: Given an array of items. We will try to find “xyz” from it using linear search.



Linear Search

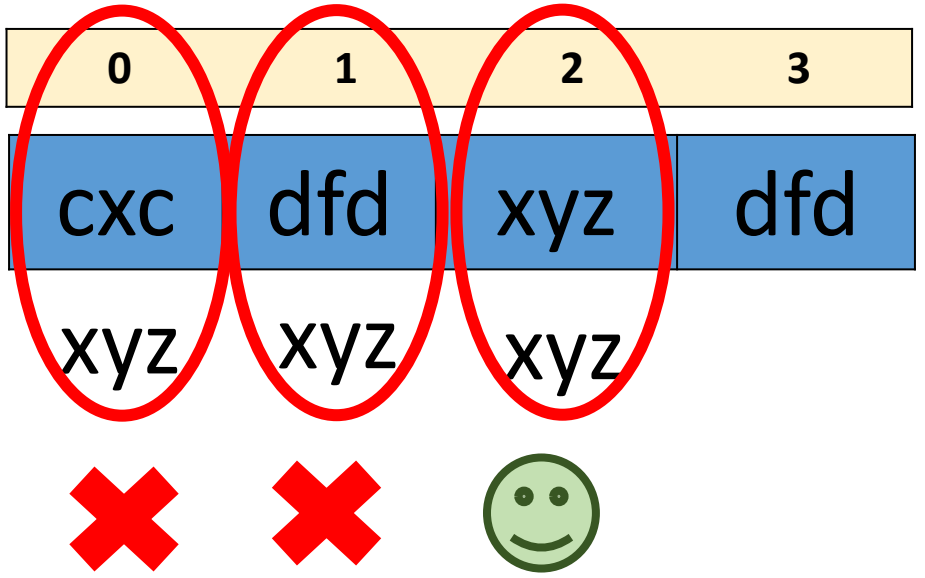
Simulation (Iterative and Recursive)

- As mentioned in the previous slide linear search is done one by one hence we start searching for “xyz” from index 0. If the items match we stop else we look for the next item in the array.



Linear Search

Simulation (Iterative and Recursive)



Linear Search

Pseudo code (Iterative)

```
boolean linearSearch(A[], item){

    int i = 0;
    while(i < A.length){
        if (A[i] matches item){
            return true;
        }
        i++;
    }
    return false;
}
```

Starting from index 0 of the array A

Loop to traverse till the end of the array

Checking if the *ith* element of the array matches with *item*

If matches, then return true, not searching further.

If mismatch in the if condition, this line will run moving to the next index of the array.

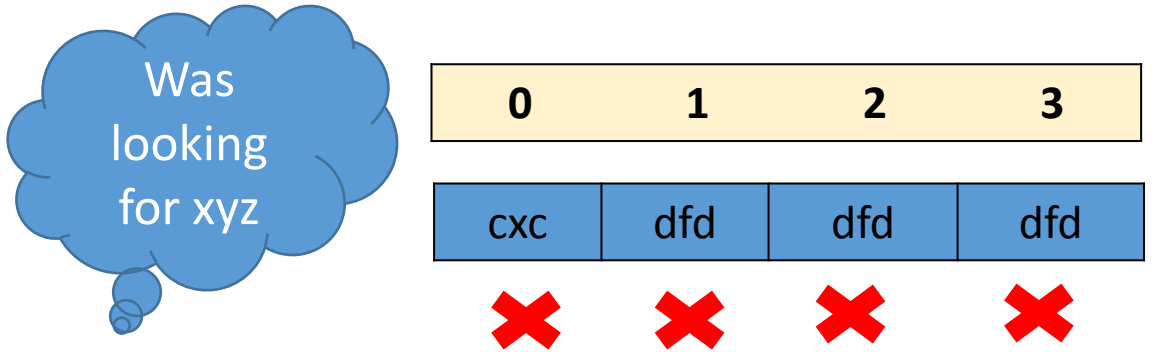
The program will reach this line after while loop is complete. This means all indices are traversed and item could not be found. False is returned.

Linear Search

Time Complexity (Iterative)

```
boolean linearSearch(A[], item){

    int i = 0;
    while(i < A.length){
        if (A[i] matches item){
            return true;
        }
        i++;
    }
    return false;
}
```



The worst case scenario occurs when the loop runs from index 0 to the end of the array. That is the item is found in the last index or not found at all. Recall the above scenario.

If there are *n* indices in an array, the worst case would be O(n)

Now for the best case to occur the item we are searching for appears at the beginning as a result the number of search is just 1. This is an once is a blue moon situation must be ignored.

Linear Search

Pseudo code (Recursive)

```
boolean linSearch(A[], i, key){
        if (i>=A.length){
                return false;
        }else{
                if (A[i]==key){
                        return true;
                }else{
                        i++;
                        return linSearch(A, I, key);
                 }
        }
}
```

Note that the **CORE** of the algorithm is the same. Using recursion we replacing the loop with method call.

Linear Search

Recursion Tracing

Tracing recursive algorithms are not easy like the iterative ones. Now how to know if a recursive algorithm is correct? Simple, look at the recurrence equation [the logic of the program]. The logic of linear search is, searching starts from index 0 and proceed by one index until you find the element or you reach the end of the array.

You will not get it right on the first day. Practice and patience. Recursion is fun!!

Linear Search

Recursion Tracing

Let A = | 5 | 6 | 7 | 8 |

i = 0 and key = 7. linSearch() method invoked with parameters

Every method call is saved in stack, with current values

```
boolean linSearch(A[], i, key){
        if (i>=A.length){
1.              return false;
        }else{
                if (A[i]==key){
2.                      return true;
                }else{
                        i++;
3.                      return linSearch(A, i, key);
                }
        }
}
```

Step no.	Variable Values (Stack)	Scope
1	A unchanged i = 0 key = 7	Scope 3 i becomes 1 recursive call
2	i =1	Scope 3 i becomes 2 recursive call
3	i = 2	Scope 2 return true

Linear Search

Recursion Simulation

Refer to slide number 3 and 4.