**School of Computer Science**

**CIS\*2520: Data Structures**

**Fall 2024, Lab 5**

**Weeks of 8 and 9**

# Search Algorithms

1. Binary search requires the array to be \_\_\_\_\_\_\_\_\_\_\_\_.
2. Binary search algorithm uses which of the following approaches?
   1. Linear
   2. Sort and search
   3. Greedy approach
   4. Divide and conquer
3. What is the main difference between Interpolation Search and Binary Search?
4. Consider the recursive implementation of Binary Search algorithm from the lecture notes. Given the array a=[2,4,7,12,23,34,42,45,67,133,145,156,178] and two values x=42 and y=177. Calculate how many times we need to call the recursive function BinarySearch find x and y in the array a?
5. Consider the following function. It attempts to find element x in a sorted array a of n elements (n>1). This function contains an error. Identify the cases when the function fails and attempt to fix it.

1 void fun(int n, int a[], int x){

2 int i,j,k;

3

4 i= 0;

5 j= n-1;

6 do{

7 k=(i+j) / 2;

8 if (a[k] < x)

9 i = k;

10 else

11 j = k;

12 }

13 while((a[k] != x) && (i < j));

14

14 if (a[k] == x)

16 printf ("x is in the array");

17 else

18 printf ("x is not in the array");

19 }

1. Implement function ceilSearch that is given a sorted array of integers and a searched element x. The function should return a ceiling of the element x.

The ceiling of an element x is the smallest element present in array which is greater than or equal to x. Ceiling is not present if x is greater than the maximum element present in array. For example, if the given array is {12, 67, 90, 100, 300, 399} and x = 95, then the output should be 100.

What can be the minimum worst-case time complexity to find ceiling of a number x in given array?

* 1. O( log(log(n)) )
  2. O( n )
  3. O( log(n) )
  4. O( log(n)\*log(n) )

# Hashing

1. Consider the following hash table and a function h(x)=x%10. What will be the bucket of the x=11?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | 21 | 12 | 33 | 14 |  |  |  | 78 |  |

1. How many different insertion sequences of the key values using the hash function **h(k) = k mod 10** and linear probing will result in the hash table shown below?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  |  | 42 | 23 | 34 | 52 | 46 | 33 |  |  |

1. 10
2. 20
3. 30
4. 40
5. Which of the following hash functions is most likely to cause clustering in a hash table?
   1. h(k) = k % m
   2. h(k) = floor(m \* (kA % 1))
   3. h(k) = k
   4. h(k) = ((k / m) + k \* m) + k % m
6. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?  
   1. h(i) = (12 ∗ i) mod 10
   2. h(i) = (11 ∗ i2) mod 10
   3. h(i) =i3 mod 10
   4. h(i) =i2 mod 10
7. Why it is essential for the search algorithm in the hash table to distinguish between empty-since-start and empty-after-deletion buckets?
8. What will be the state of the hash array with quadratic probing after inserting the following sequence using hash function h(k)=k % 15 (if collision is detected h(k)=k%15+ci2)

Sequence of items to insert: 32, 30, 49, 3, 99, 23, 11, 42, 15, 62

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. Insert the characters of the string **K R P C S N Y T J M** into a hash table of size 10. Use the hash function

h(x) = ( ord(x) – ord("A") + 1 ) mod10

If linear probing is used to resolve collisions, then the following insertion causes collision

* 1. Y
  2. C
  3. M
  4. P

1. Propose a structure that could be used to represent state of a bucket in the hash table with linear probing. Implement a sample using a table of size 10 and a sequence of 5 elements with at least 1 conflict. h(k)=k%10.