

Data and File Structures Laboratory

Searching Techniques

Malay Bhattacharyya

Assistant Professor

Machine Intelligence Unit
Indian Statistical Institute, Kolkata

September, 2019

- 1 Basics
- 2 Searching on Linear Data Structures
- 3 Searching on Nonlinear Data Structures
- 4 Other Searching Problems

The searching problem

Given a data structure S over a domain of data items D , verify (search) whether a given data item s belongs to D or not. If it belongs, then return the position (index) of s in D with respect to its organization in S .

Searching on Linear Data Structures

- Linear search
- Jump search
- Binary search
- Interpolation search
- Fibonacci search

Searching on Linear Data Structures

Searching Method	On Arrays	On Linked Lists
Linear	Possible	Possible
Binary	Possible	Only possible with alternative implementation (no deletion applied)
Jump	Possible	Only possible with alternative implementation (no deletion applied)
Interpolation	Possible	Only possible with alternative implementation (no deletion applied)
Fibonacci	Possible	Possible with alternative implementation (no deletion applied)

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	↓					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	↓					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		↓				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	↓					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		↓				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$			↓			
Whether $LIST[i] = s$	30	10	20	40	50	FALSE

Linear search

Let $LIST[] = \{30, 10, 20, 40, 50\}$ and $s = 40$.

$i = 0$	↓					
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$		↓				
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$			↓			
Whether $LIST[i] = s$	30	10	20	40	50	FALSE
$i = i + 1$				↓		
Whether $LIST[i] = s$	30	10	20	40	50	TRUE

Return $i = 3$.

Linear search

```
int LinearSearch(int *LIST, int SIZE, int s){
    int i;
    for(i=0; i<SIZE; i++)
        if(*(LIST + i) == s)
            return i; // Data is at the index i
    return; // Data is nowhere
}
```

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre_i = 0, i = b$			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre_i = 0, i = b$			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$pre_i = i, i = i + b$					↓	
Whether $LIST[i] < s$	10	20	30	40	50	FALSE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre_i = 0, i = b$			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$pre_i = i, i = i + b$					↓	
Whether $LIST[i] < s$	10	20	30	40	50	FALSE
$pre_i = pre_i + 1$				↓		
Whether $LIST[pre_i] = s$	10	20	30	40	50	TRUE

Jump search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Further assume that the block size (of jump) is $b = 2$.

$pre_i = 0, i = b$			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$pre_i = i, i = i + b$					↓	
Whether $LIST[i] < s$	10	20	30	40	50	FALSE
$pre_i = pre_i + 1$				↓		
Whether $LIST[pre_i] = s$	10	20	30	40	50	TRUE

Note: The block size in jump search is taken as $b = \sqrt{n}$, where n denotes the size of the list.

Jump search

```
int JumpSearch(int *LIST, int SIZE, int b, int s){
    int pre_i = 0, i = b;
    while(*(LIST + Min_Func(i, SIZE) - 1) < s)
        pre_i = i, i += b;
    if(pre_i >= SIZE)
        return;
    // Linear search between pre_i and i
    for(j=pre_i+1; j<i; j++)
        if(*(LIST + j) == s)
            return i; // Data is at the index j
    return; // Data is nowhere
}

int Min_Func(int x, int y){
    return x < y ? x : y;
}
```

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
Whether $LIST[i] < s$	10	20	↓	40	50	TRUE

Binary search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$l = 0, r = 4, i = l + (r - l)/2 = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
$l = i + 1, i = l + (r - l)/2 = 3$				↓		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Binary search

```
int BinarySearch(int *LIST, int l, int r, int s){
    int i;
    if(r >= l){
        i = l + (r - l)/2;
        if(*(LIST + i) == s)
            return i; // Data is in the middle
        if(*(LIST + i) < s) // Data is in the right
            return BinarySearch(LIST, i+1, r, s);
        // Data is in the left
        return BinarySearch(LIST, l, i-1, s);
    }
    return; // Data is nowhere
}
```

Comparing linear/jump/binary search

	Linear search	Jump search	Binary search
Requirements	Nothing	Ordered list	Ordered list
Backward scan required	No	Once	Multiple times
Time complexity (best case)	$O(1)$	$O(b)$	$O(1)$
Time complexity (average case)	$O(n)$	$O(\sqrt{n})$	$O(\log n)$
Time complexity (worst case)	$O(n)$	$O(\sqrt{n})$	$O(\log n)$

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$i = 0 + (40 - 10) * (4 - 0) / (50 - 10) = 3$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Interpolation search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$.

$$i = 0 + (40 - 10) * (4 - 0) / (50 - 10) = 3$$

$i = 0 + (40 - 10) * (4 - 0) / (50 - 10) = 3$				↓		
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Note: The interpolation search is efficient over binary search for the ordered lists having uniformly distributed data items.

Interpolation search

```
int InterpolationSearch(int *LIST, int SIZE, int s){
    int l = 0, r = SIZE - 1, p;
    while (l<=r && s>=*(LIST + l) && s<=*(LIST + r)){
        // Interpolate the position
        p = l + (s-*(LIST + l)) * (r-l) /
            (*(LIST + r)-*(LIST + l));
        if(*(LIST + p) == s)
            return p;
        if (*(LIST + p) < s) // Data is in the right
            left = p + 1;
        else // Data is in the left
            right = p - 1;
    }
    return; // Data is nowhere
}
```

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = \text{Fibonacci}(f - 2) = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = \text{Fibonacci}(f - 2) = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = \text{Fibonacci}(f - 2) = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
				↓		
$f = 4 - 2 + 1 = 3, i = i + \text{Fibonacci}(f - 2) = 3$						
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Fibonacci search

Let $LIST[] = \{10, 20, 30, 40, 50\}$ and $s = 40$. Note that, the index of largest Fibonacci number that is greater than or equal to the length of given array is $f = 5$.

$i = \text{Fibonacci}(f - 2) = 2$			↓			
Whether $LIST[i] = s$	10	20	30	40	50	FALSE
			↓			
Whether $LIST[i] < s$	10	20	30	40	50	TRUE
				↓		
$f = 4 - 2 + 1 = 3, i = i + \text{Fibonacci}(f - 2) = 3$						
Whether $LIST[i] = s$	10	20	30	40	50	TRUE

Note: Fibonacci search is a suitable replacement of binary search where the list is unbounded. Unlike binary search, it does not use the costly division operation (though avoidable with bitwise shift).

Searching on Nonlinear Data Structures

1 Searching on graphs

- Breadth-first search
- Depth-first search

2 Searching on trees

- Searching in binary trees
- Searching in heaps
- Searching in binary search trees
- Searching in balanced search trees

Other Searching Problems

- Substring search
- Sublist search
- Range search
- Searching in streams

Range search

A database query may ask to find out all the students with GPA between g_1 and g_2 (say 7.5 and 8.5), and salary between s_1 and s_2 (say 90K and 100K).

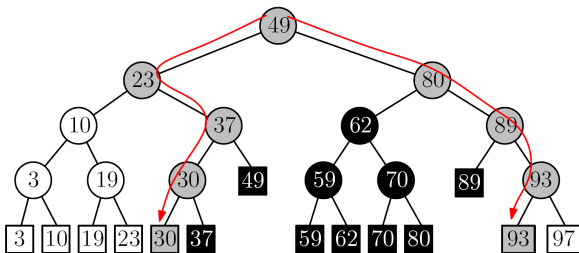
Roll No.	Salary	GPA
1	90K per mensem	7.2
2	80K per mensem	7.7
3	100K per mensem	7.1
4	90K per mensem	7.3
5	70K per mensem	8.2

Different variants of range search

1-D range query problem: Given a set of n points (data items) on \mathbb{R} (real line) and an interval (1-D query range), efficiently find the points that stay within the interval.

Different variants of range search

1-D range query problem: Given a set of n points (data items) on \mathbb{R} (real line) and an interval (1-D query range), efficiently find the points that stay within the interval.



A 1-D range search with query $[25, 90]$ (using binary search tree)

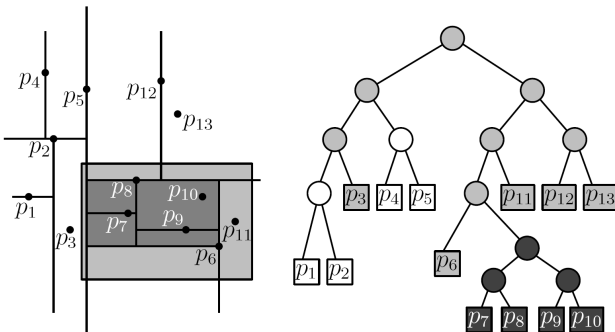
Note: The gray nodes are only visited (traversed).

Different variants of range search

2-D range query problem: Given a set of n points (coordinates) on \mathbb{R}^2 (2-D space) and a boundary (2-D query range), efficiently find the points that stay within the boundary.

Different variants of range search

2-D range query problem: Given a set of n points (coordinates) on \mathbb{R}^2 (2-D space) and a boundary (2-D query range), efficiently find the points that stay within the boundary.



A 2-D range search with query boundary (using k -dimensional tree)

Searching in streams

Definition (Data stream model)

A *data stream model* defines an input stream $\mathcal{S} = \langle s_1, s_2, \dots \rangle$ arriving sequentially, item by item, and describes an underlying signal S , where $S : [1 \dots N] \rightarrow \mathbb{R}$ is a one-dimensional function.

Searching in streams

Definition (Data stream model)

A *data stream model* defines an input stream $\mathcal{S} = \langle s_1, s_2, \dots \rangle$ arriving sequentially, item by item, and describes an underlying signal S , where $S : [1 \dots N] \rightarrow \mathbb{R}$ is a one-dimensional function.

The data stream models can be of the following three types:

- Time Series Model
- Cash Register Model
- Turnstile Model

Different data stream models

The models are classified based on the information about how the input data elements stream in.

1 Time Series Model

- Each s_i equals $S[i]$ and they appear in increasing order of i .
- Observing the traffic at an IP link for each 5 min, or volume estimation of share trading in every 10 min, etc.

2 Cash Register Model

- Perhaps the most popular data model.
- Here s_i 's are increments to $S[j]$'s.
- Monitoring IP addresses that access a web server.

3 Turnstile Model

- This is the most general model.
- Here s_i 's are updates to $S[j]$'s.
- Monitoring the stock values of a company.

Processing streaming data

A useful model for processing streaming data is to work on a window of the most recent N elements received. Alternatively, we consider the elements received within a fixed time interval T .

Processing streaming data

A useful model for processing streaming data is to work on a window of the most recent N elements received. Alternatively, we consider the elements received within a fixed time interval T .

Consider a stream of integers. Further suppose, we want to find out the average of the integers in a window of size N .

For the first N inputs, we can sum and count the integers and calculate the average. After that, for each of the new input i received, add $\frac{(i-j)}{N}$ to the previous average value, where j is the oldest integer in the window.

Streaming algorithms

In streaming algorithms, the data is available as a stream and we would like –

- the per-item processing time
- storage and
- overall computing time

to be simultaneously $O(N, t)$, preferably $O(\text{polylog}(N, t))$, at any time instant t in the data stream.

Note: $O(\text{polylog}(N, t))$, often written as $\text{polylog}(N, t)$, means $O((\log n)^k)$ or $O(\log^k n)$ for some k .

Problems – Day 16

- 1 Suppose a pair of linked lists in which the elements are sorted in ascending order are given. Write a program to efficiently find out whether one of them is a sublist of the other.

Input format:

```
1 3 5 7 9 11 13 15 17 19 21 23 # List 1
5 7 9 11 # List 2
```

- 2 You are given a robotic hammer and some metal sheets. You can manually set the force (in Newton) imparted by the hammer as integral values. If a force cannot break the metal sheet cannot even make it weaker. Write a program to explore the maximum tolerable force (in Newton) of the metal sheet in minimum trials. The input (treat it as a black box) is the maximum tolerance level of the metal sheet and the outputs are the forces imparted by the robotic arm in successive trials.

Problems – Day 16

- Let us define a sequence $a_0, a_1, a_2, \dots, a_{n-1}$ as Λ -bitonic if there exists a j , $0 \leq j < n$, such that $a_0 < a_1 < \dots < a_j > a_{j+1} > \dots > a_{n-1}$. Consider an $m \times n$ matrix A consisting of integer entries, such that each row and each column of the matrix forms a Λ -bitonic sequence. Write a program to efficiently find the largest element of the matrix.
- Suppose you are given a pair of ordered lists one of which has the salaries of ISI MTech students of 2016-18 batch in descending order and the other of 2017-19 in ascending order. Write a program to find out the median salary of all the students taken together involving time logarithmic to the (maximum) size of those lists.
- Write a program to implement Fibonacci search on a generic array having ordered list of data items. Consider that the comparator() function works on various data types in a usual sense.

Problems – Day 16

- 6** Suppose the marks of your Assignment 1 are available in ascending order. You have to figure out the students whose marks are same as their ranks (1 to n). Write a program to achieve this in no more than $\lfloor \log_2 n + 1 \rfloor$ trials.
- 7** Let π be a permutation of $\{1, \dots, n\}$ and π_{-1} symbolizes π with one element missing. Consider that the elements from the set $\pi_{-1}[i]$ appears as a stream in the increasing order of i , one by one. Write a program to determine the missing integer at the end. You are allowed to store only $O(\log n)$ bits of memory to determine the missing integer. Try to further restrict it to exactly $\log n$ bits of memory.

Input format:

8 # The value of n

1 4 6 2 8 5 7 # The set of values streaming in