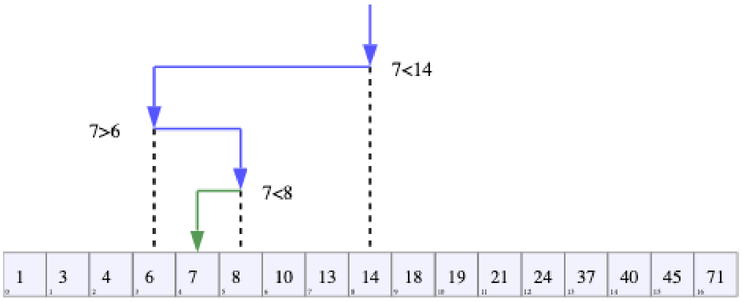
CSC-122: Data Structures and Algorithms
Written Assignment 2
Spring 2024

Question 1. Binary Search



Input: A sorted array `A[0.. n−1]` and a target value `Q`
Output: an index `i` such that `Q = A[i]` or `−1` if no such index exists

1. Set `Low` to `0` and `High` to `n−1`.
2. If `Low > High`, the search terminates as unsuccessful. Return `−1`.
3. Set `Mid` to the floor of `(Low + High) / 2`.
4. If `A[Mid] < Q`, set `Low` to `Mid + 1` and go to step 2.
5. If `A[Mid] > Q`, set `High` to `Mid - 1` and go to step 2.
6. If `A[Mid] == Q`, the search is done; return `Mid`.

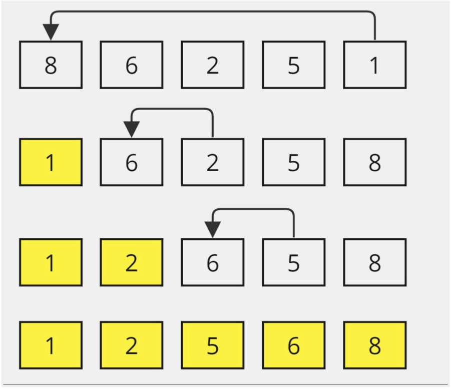
Use the binary search algorithm above to create a **trace table** for the sets of inputs specified in 1.1. to 1.3.

Following is an example trace table for binary search algorithm. Feel free to add/remove/edit rows or columns as you deem best.

Line #	Low	Mid	High	A[Mid]	A[Mid] == Q

1.1. `A = [10, 20, 30, 40, 50],` `Q = 20`
1.2. `A = [97, 98, 99],` `Q = 99`
1.3. `A = [1, 2, 4, 5],` `Q = 3`

Question 2. Selection Sort



Inputs: Given a list of numbers `L` of length `n`

Output: A sorted list of numbers `L` of length `n`

1. Initialize an iterator `i` to `0`.
2. Identify the smallest element in the unsorted sublist (from `i` to `n-1`).
3. Swap the smallest element with the element at index `i`.
4. Increment `i` by `1`.
5. Repeat steps 2-4 until `i` is equal to `n-1`.

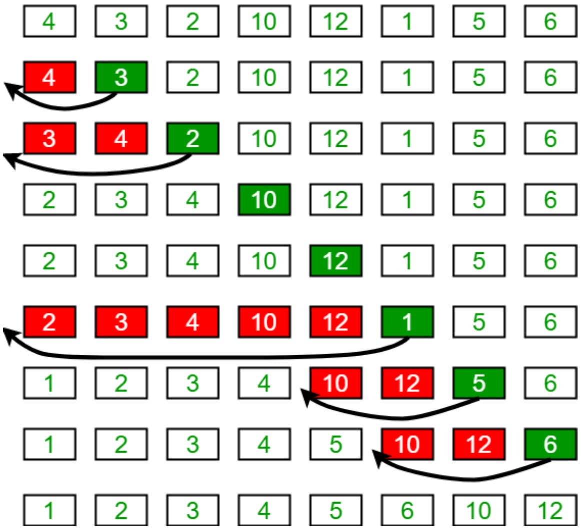
Using the selection sort algorithm above to create a **trace table** for the sets of inputs specified in 2.1. to 2.3.

Following is an example trace table for selection sort algorithm. Feel free to add/remove/edit rows or columns as you deem best.

`Line #`	`L`	`i`	`minimum(A[i:n])`

2.1. `A = [5, 4, 3, 2, 1]`
2.2. `A = [3, 2, 5, 1, 4]`
2.3. `A = [1, 5, 2, 4, 3]`

Question 3. Insertion Sort



4 3 2 10 12 1 5 6

4 3 2 10 12 1 5 6

3 4 2 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

2 3 4 10 12 1 5 6

1 2 3 4 10 12 5 6

1 2 3 4 5 10 12 6

1 2 3 4 5 6 10 12

Inputs: Given a list of numbers `L` of length `n`

Output: A sorted list of numbers `L` of length `n`

```
1. Initialize a iterator i to 1.
2. Initialize a variable key to L[i].
3. Iterate over the list using j  from i-1 to 0 and do the following
        4. If L[j] > key, set L[j+1] to L[j].
        5. Else set L[j+1] to key.
6. Increment i by 1.
```

Use the Insertion Sort algorithm above to create a **trace table** for the sets of inputs specified in 3.1. to 3.3.

Following is an example trace table for binary search algorithm. Feel free to add/remove/edit rows or columns as you deem best.

Line #	L	key	i	j

3.1. `A = [5, 4, 3, 2, 1]`

3.2. `A = [3, 2, 5, 1, 4]`

3.3. `A = [1, 5, 2, 4, 3]`