Array (as Data Structure)

Hsuan-Tien Lin

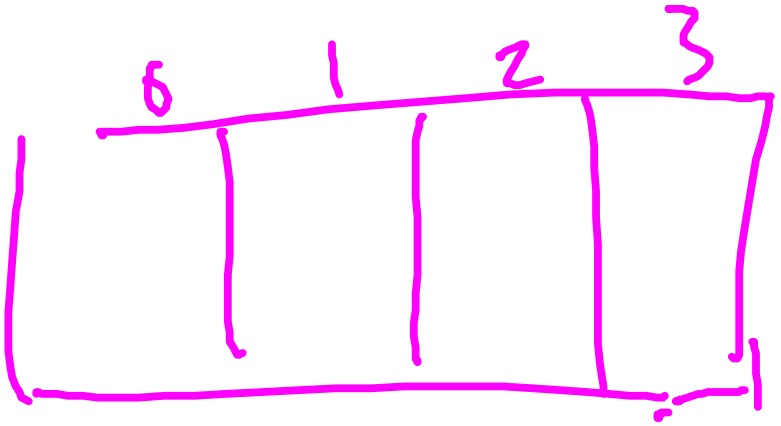
Dept. of CSIE, NTU

March 2, 2021

Array

What is Array?

wikipedia: *a collection of elements, each identified by one array index*



array: numbered lockers

Memory is (Generally Viewed as) Array

address

mem[address]

arr[index]

pointer: stores index to memory array

Array as Memory Block in C/C++

access

- `data getByIndex(index):`

  `arr[index]`, which means

  `memory[arr + index * sizeof(data)]`

arr 11266211

3

maintenance

- `construct(length):`
  - `malloc(sizeof(data)*length)` in C
  - `new data[length]` in C++
- `updateByIndex(index, data):`

  `arr[index] = data`

desired property: fast computation of address from `index`

$\Longrightarrow$ fast random access

Array as Abstract Data Structure

access

- `data getByIndex(index)`
- `insertByIndex(index, data)`

maintenance

- `construct(length)`
- `updateByIndex(index, data)`
- `removeByIndex(index)`

implicit assumption:
`index` to address done by fast math formula

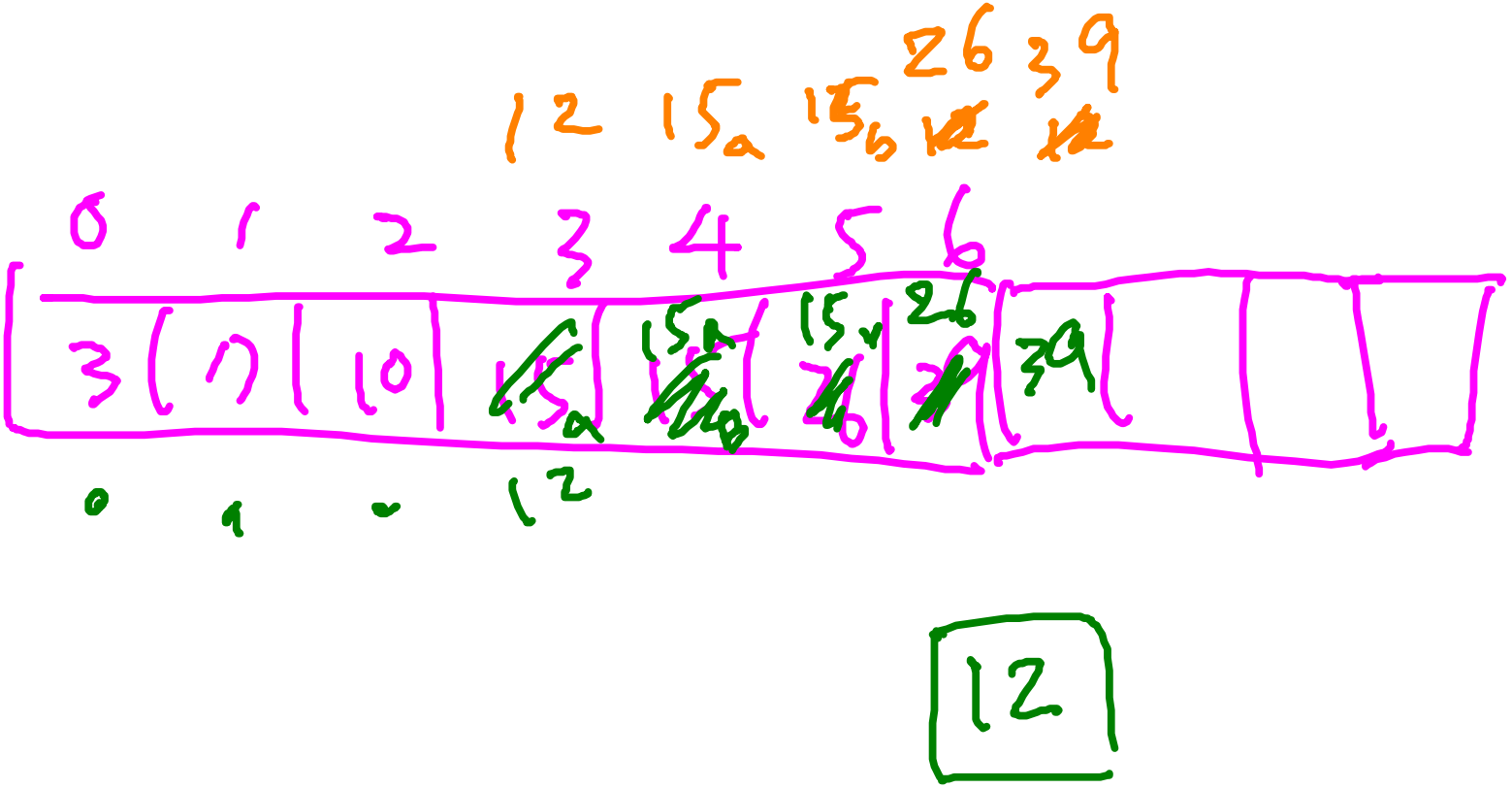
Ordered Array

Definition of Ordered Array

0	1	2	3	4	5	6
3	7	10	15	15	26	39

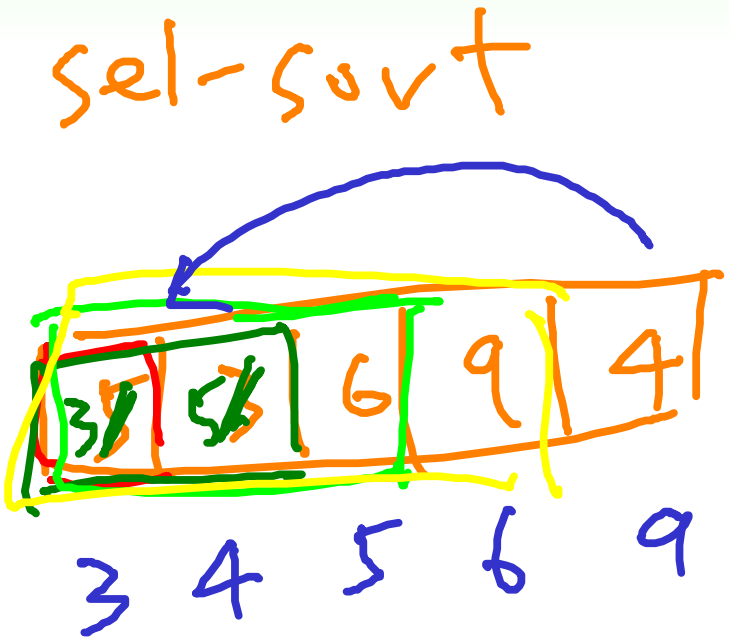
an array of consecutive elements with ordered values

`insert` of Ordered Array



"cut in" from the back

`construct` of Ordered Array



insertion sort: construct with multiple `insert`

`update` and `remove` of Ordered Array

maintenance

- `updateByIndex(index, data)`: rotate up or down
- `removeByIndex(index)`: fill in from the back

ordered array: more maintenance efforts

Binary Search within Ordered Array

Application: Book Search within (Digital) Library

3	5	6	7	10	19	24

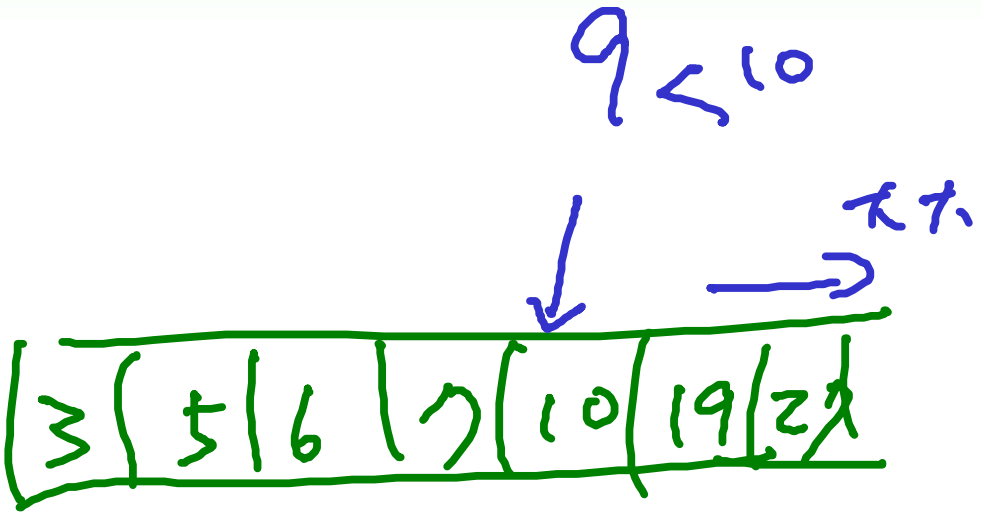
comparable elements: book IDs

Sequential Search Algorithm

```
for i ← 1 to A.len
    if (is A[i] what I want?)
        return A[i]
return null;
```

similar to `getMinIndex`

Ordered Array: Sequential Search Algorithm with Cut



ordered: possibly easier to declare not found