CSCI2100C Lab 3

Prepared by:

Song BIAN

Reminders

- Please remember your password for the online judge
- Please start your assignment early
  - And report any issues as early as possible. Some issues regarding the registration of the OJ were reported not until some hours before the deadline. For such cases of failure of code submission, we may not grade your lab.
  - Penalty:
    - -10 marks/day pro rata for first two days after deadline
    - -10 marks/hour pro rata afterwards (so you get 0 marks if you submit 2 day 8 hours after the deadline)
- Grading is based on the **last** submission
- Write your own code
  - We will check your code
  - Suspected cases of plagiarism will be reported
- Questions?

Agenda

- Merge Sort
- BST (binary search tree)
- Overview of Lab 3 Problems

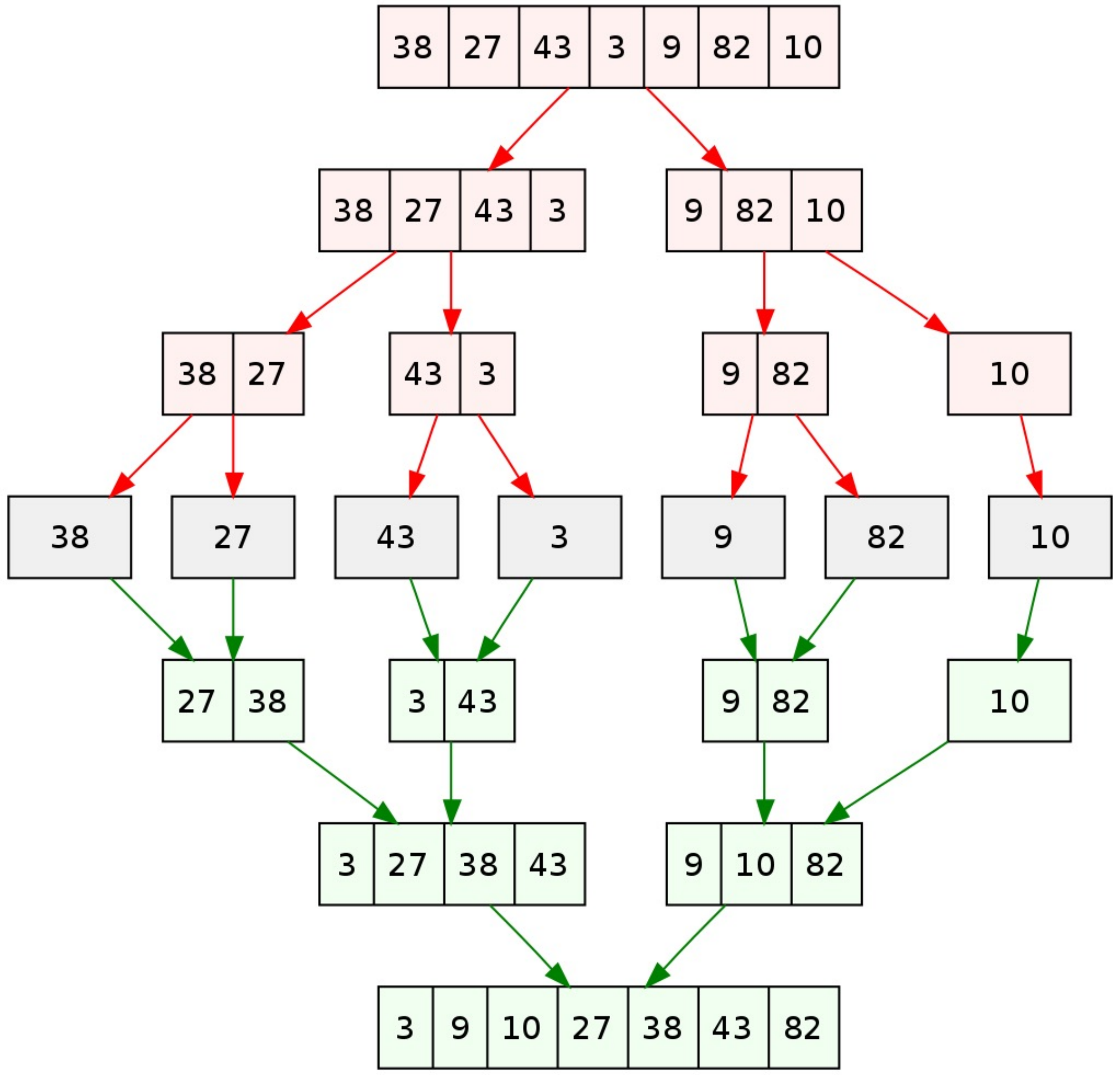
Agenda

- **Merge Sort**
- BST (binary search tree)
- Overview of Lab 3 Problems

Merge Sort

Properties

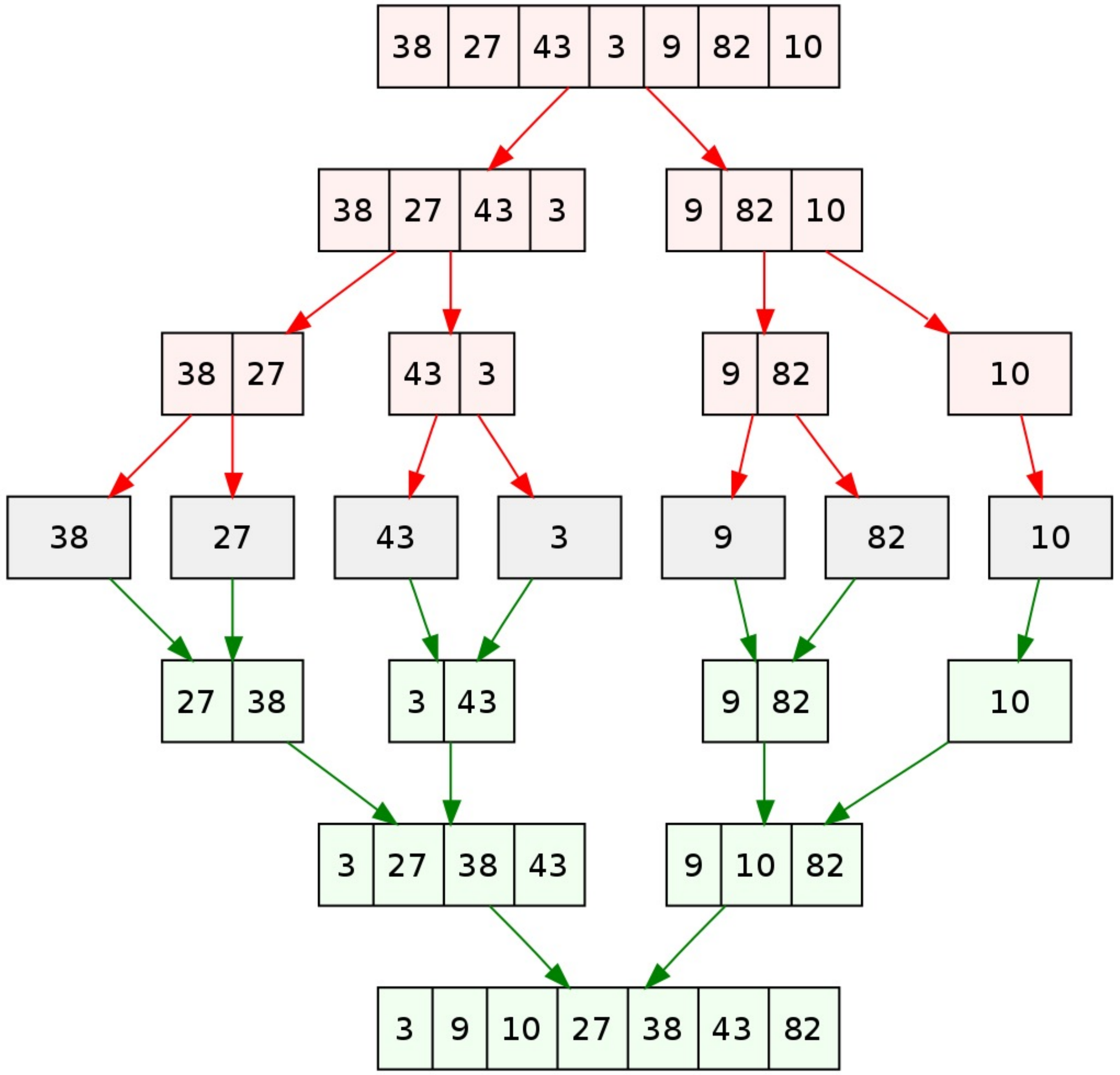
- As one may understand from the right image, at each step a list of size M is being divided into 2 sublists of size M/2, until no further division can be done. To understand better, consider a larger array A containing the elements (38, 27, 43, 3).
- At the first step this list of size 4 is divided into 2 sublists the first consisting of elements (38, 27) and the second one being (43, 3). Now, the first list consisting of elements (38, 27) is further divided into 2 sublists consisting of elements (38) and (27) respectively.



Merge Sort

Properties

- As no further breakdown of this list can be done, as each sublist consists of a maximum of 1 element, we now start to merge these lists. The 2 sub-lists formed in the last step are then merged together in sorted order using the procedure mentioned above leading to a new list (27, 38). Backtracking further, we then need to merge the list consisting of element (3, 43) too with this list, leading to the new sorted list (3, 27, 38, 43)
- Time complexity is O(NlogN)



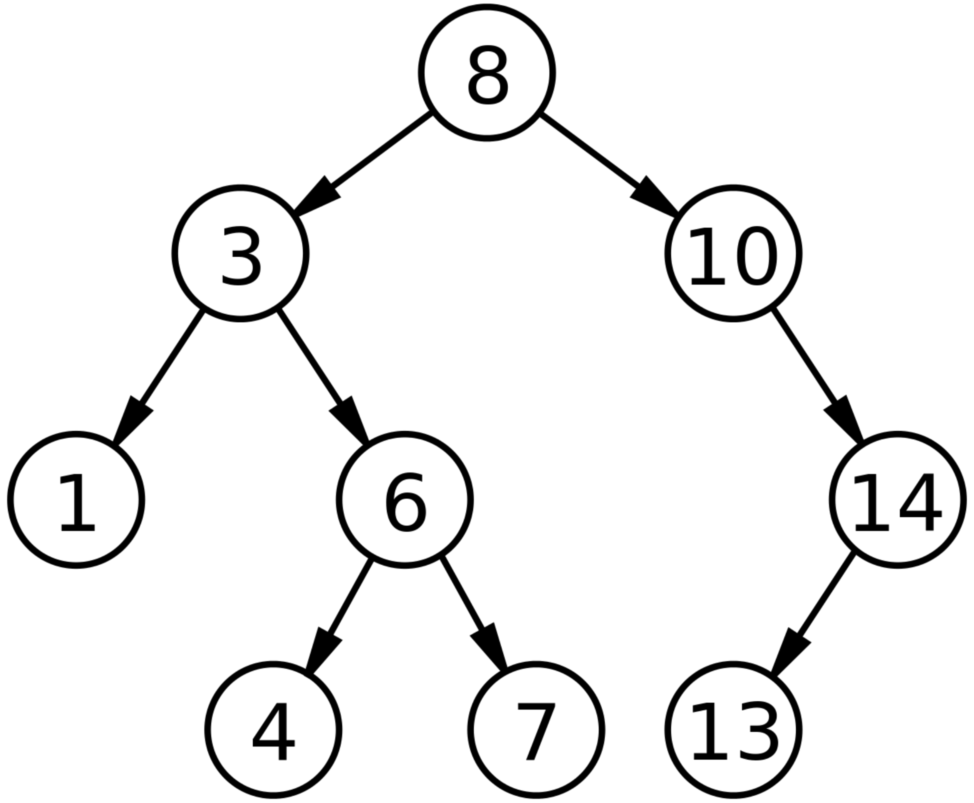
Agenda

- Merge Sort
- BST (binary search tree)
- Overview of Lab 3 Problems

BST (binary search tree)

Properties

- Binary search trees keep their keys in sorted order, so that lookup and other operations can use the principle of binary search: when looking for a key in a tree (or a place to insert a new key), they traverse the tree from root to leaf, making comparisons to keys stored in the nodes of the tree and deciding, on the basis of the comparison, to continue searching in the left or right subtrees. On average, this means that each comparison allows the operations to skip about half of the tree, so that each lookup, insertion or deletion takes time proportional to the logarithm of the number of items stored in the tree



Agenda

- Merge Sort
- BST (binary search tree)
- Overview of Lab 3 Problems

Lab 3

Problem 1

- Straight forward implementation of merge sort

Code segment to complete:

```
void merge(int array[], int left, int middle, int right) {
    int lp = left;
    int rp = middle + 1;
    int buffer[right - left + 1];
    // write your code here
}

void mergeSort(int array[], int left, int right) {
    if (right - left + 1 <= 1) {
        return;
    }
    int middle = left + (right - left)/ 2;
    mergeSort(array, left, middle);
    mergeSort(array, middle + 1, right);
    merge(array, left, middle, right);
}
```

Merge Sort

Description:

Given an unsorted array of N elements, output the sorted array.

Input:

N, followed by an unsorted integer array

Output:

A sorted integer array

Sample

input:

```
5
1 5 3 2 4
```

output:

```
1 2 3 4 5
```

Lab 3

Problem 2

- Straight forward implementation of BST search and min operation

Code segment to complete:

```
struct TreeNode * search(struct TreeNode * root, int key) {
    // write your code here
}

struct TreeNode * min(struct TreeNode * root) {
    // write your code here
}
```

BST search and min

Each input file contains one test case. For each case,

the first line contains a positive integer N (≤20) which is the total number of keys to be inserted.

Then N distinct integer keys are given in the next line. All the numbers in a line are separated by a space.

Input a search key

Output the search result and min-value of the BST.

Input:

The number of elements N.

An integer array.

Output:

Search result: If key in the BST, output key's value. Otherwise, output NULL. Output the min-value in the tree.

Sample 1

input:

5

1 10 25 16 98

10

output:

10

1

Algorithm: ***search(root, key)***

```
if isEmpty(root)  //base case: empty tree
    return NULL
data ← Data(root)
if key == data.key //equal to the search key, return the node
    return root
elseif key<data.key //larger than the search key,check left subtree
    search(Lchild(root), key)
else // smaller than the search key, only check the right subtree
    search(Rchild(root), key)
```

Algorithm: ***min(root)***

```
node = root
While !isEmpty(node)
and !isEmpty(Lchild(node))
    node = Lchild(node)
return node
```

Last but not Least

- Please add this declaration on top of (commented as shown) all your codes submitted to the OJ.

```
/*
I, <Your Full Name>, am submitting the assignment for
an individual project.
I declare that the assignment here submitted is original except for
source material explicitly acknowledged, the piece of work, or a
part
of the piece of work has not been submitted for more than one
purpose
(i.e. to satisfy the requirements in two different courses) without
declaration. I also acknowledge that I am aware of University
policy
and regulations on honesty in academic work, and of the
disciplinary
guidelines and procedures applicable to breaches of such policy and
regulations, as contained in the University website
http://www.cuhk.edu.hk/policy/academichonesty/.
It is also understood that assignments without a properly signed
declaration by the student concerned will not be graded by the
teacher(s).
*/
```

Questions?