# Assignment 3

Hyoungshick Kim

November 12, 2018

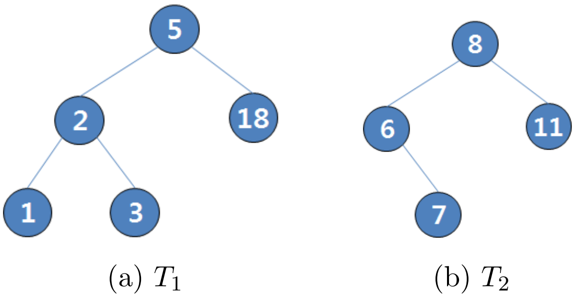
## 1 Union of Binary Search Tree

This is a programming assignment to test your understanding of Binary Search Tree.

- Your goal is to merge two given *binary search trees* into one united binary search tree to minimize the costs incurred in changing edges to construct a new merged binary search tree.

- You will write a code in the C programming language to merge two binary search trees $T_1$ and $T_2$. We assume that all key values in $T_1$ and $T_2$ are unique. For each tree, the number of nodes and a set of edges in the tree are given in the input file named '`hw3_input.txt`'. For each edge "$u$ $v$" in the input file, $u$ is the parent of $v$. That is, you need to build one united binary search tree $T_3$ from two binary search trees $T_1$ and $T_2$. Please write (1) the root in the tree $T_3$, (2) the edges in the tree $T_3$, (3) the number of newly added edges $N_A$ (i.e., the edges which are included in $T_3$ but not included in $T_1$ and $T_2$) and (4) the number of deleted edges $N_D$ (i.e., the edges which are included in either $T_1$ or $T_2$ but not included in $T_3$) into the output file named '`hw3_output.txt`'.

- The following is an example of input and output files:



(a) $T_1$ (b) $T_2$

Figure 1: Input trees ($T_1$ and $T_2$)

```
[Input file: hw3_input.txt]
5
5 2
5 18
2 1
2 3
4
8 6
8 11
6 7

[Output file: hw3_output.txt]
5
5 2
5 18
2 1
2 3
18 8
8 6
8 11
6 7
1
0
```

The example input instances can be visually represented in Figure 1.

The merged tree $T_3$ can be visually represented in Figure 2

- You will be judged by (1) the correctness of the results returned by your submitted program, (2) the costs incurred in changing edges to construct a new merged binary search tree (i.e., $N_A + N_D$), (3) the actual running time of the program and (4) the well-written document to explain your source code and the performance analysis of your algorithm. For test, we will use $5 \leq n, m \leq 1000$ where $n$ and $m$ are the numbers of nodes in $T_A$ and $T_B$, respectively. Please test your code extensively with several inputs, so you are sure it works correctly.

- We assume that all nodes in the trees $T_A$ and $T_B$ have unique key values.

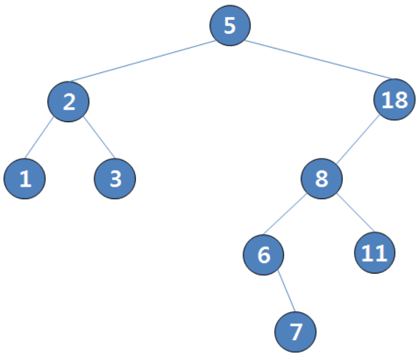


Figure 2: Output tree ($T_3$)

- You MUST use your own tree datastructure.
- Please upload your source code (c files), instructions to illustrate how your source code works, document to explain your code and the performance analysis to iCampus. Submit your assignment by midnight, Sunday November 26; this is the **firm deadline**; late submissions are allowed with a penalty. Each 24 hours (or part thereof) late will cost you 20%.
- **Your assignments must be your own original work.** We will use a tool to check for plagiarism in assignments.