COMP9024 25T1 Data Structures and Algorithms

Week 8 Practical
Search Tree Data Structures

1. (Tree properties)

   a. The Binary Search Tree ADT (`BSTree.h`, `BSTree.c`) from the lecture contains a code stub for the function:

   ```
   int TreeHeight(Tree t) { ... }
   ```

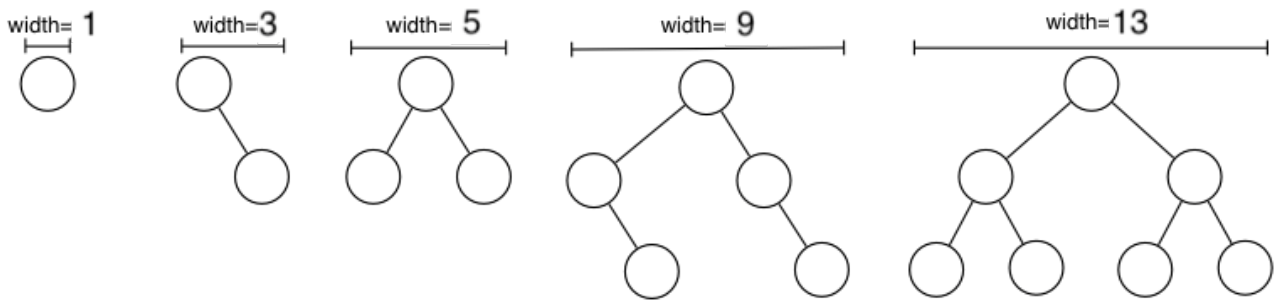
   to compute the height of a tree.

   Implement this function, but do not modify the interfaces of any functions, or the implementations of any of the other functions.

   Hint:

   - The function should return `-1` for an empty tree.

   b. Computing the height/depth of trees is useful for estimating their search efficiency. For *drawing* trees, we're more interested in their *width*. For some simple trees the following diagrams show a useful definition of tree width if you want to keep reasonable spacing:



   Derive a formula for the width of a tree that generalises from the examples and add a new function to the Binary Search Tree ADT which computes the width of a tree. Use the following function interface:

   ```
   int TreeWidth(Tree t) { ... }
   ```

   You will need to add this interface as a function prototye in `BSTree.h` and implement the function within `BSTree.c`. Once again, do not modify the prototypes or the implementation of any of the other functions.

   Hint:

   - The function should return `0` for an empty tree.

An example use of the Binary Search Tree ADT is shown below. This assumes a test program has been written to insert command line arguments as leaf nodes into an initially empty tree, and then print out the `TreeHeight()` and `TreeWidth()`.

```
prompt$ ./BSTreeTest 13 3 4 12 14 10 5 1 8 2 7 9 11 6 18
Tree Height: 8
Tree Width: 29
prompt$
```

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find the files named* `BSTree.c` *and* `BSTree.h` *with your implementation for* `TreeHeight()` *and* `TreeWidth()` *in the current directory. You can use dryrun as follows:*

```
prompt$ 9024 dryrun BSTree
```

Due Date

Tuesday, 15 April, 11:59:59. Late submissions will not be accepted.

Submission

You should submit your files using the following `give` command:

```
prompt$ give cs9024 week8 BSTree.c BSTree.h
```

Alternatively, you can select the option to "Make Submission" at the top of this page to submit directly through WebCMS3.

Important notes:

- Make sure you spell all filenames correctly.
- You can run `give` multiple times. Only your last submission will be marked.
- Where you're expected to submit multiple files, ensure they form a single submission. If you separate them across multiple submissions, each submission will replace the previous one.
- Whether you submit through the command line or WebCMS3, it is your responsibility to ensure it reports a successful submission. Failure to submit correctly will not be considered as an excuse.
- You cannot obtain marks by e-mailing your code to tutors or lecturers.

Assessment

- The exercise is worth 2 marks.
- Submissions will be auto-marked shortly after the deadline has passed.
- It is important that the output of your program follows exactly the format of the sample output, otherwise auto-marking will result in 0 marks.
- Ensure that your program compiles on a CSE machine with the standard options, i.e. `-Wall -Werror -std=c11`. Programs that do not compile will receive 0 marks.
- Auto-marking will use test cases different to `dryrun`. (Hint: do your own testing in addition to running `dryrun`).

Collection

Once marking is complete you can collect your marked submission using the following command:

```
prompt$ 9024 classrun -collect week8
```

You can also view your marks using the following command:

```
prompt$ 9024 classrun -sturec
```

You can also collect your marked submission directly through WebCMS3 from the "Collect Submission" tab at the top of this page.

Plagiarism

Group submissions will not be allowed. Your programs must be entirely your own work. Plagiarism detection software will be used to compare all submissions pairwise (including submissions for similar assessments in previous years, if applicable) and serious penalties will be applied, including an entry on UNSW's plagiarism register.

- ***Do not copy ideas or code from others***
- ***Do not use a publicly accessible repository or allow anyone to see your code***

Please refer to the on-line sources to help you understand what plagiarism is and how it is dealt with at UNSW:

- Plagiarism and Academic Integrity
- UNSW Plagiarism Policy Statement
- UNSW Plagiarism Procedure