CS280 – Splay Trees
January 17, 2016

Sudoku Questions

http://azrael.digipen.edu/~mmead/www/Courses/2016/winter/cs280/project3/index.html

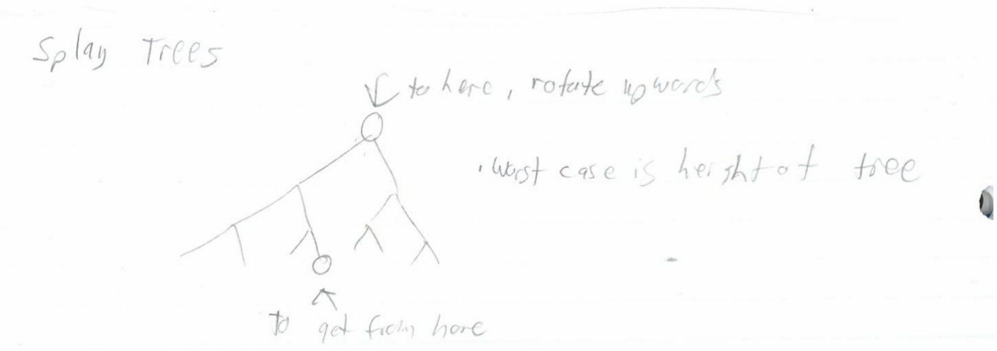
- Make sure place_value returns a boolean
- We should call recirsove function for every space on the board
  - Only if there is no conflict
- States of the program
  - Conflicts
    - Backing up
  - No conflicts
    - Moving forwards
- Abort check probably won't be tested, mostly for debugging
- <u>**Midterm next Wednesday**</u>

<u>***http://azrael.digipen.edu/~mmead/www/Courses/CS280/Trees.html***</u>

Trees

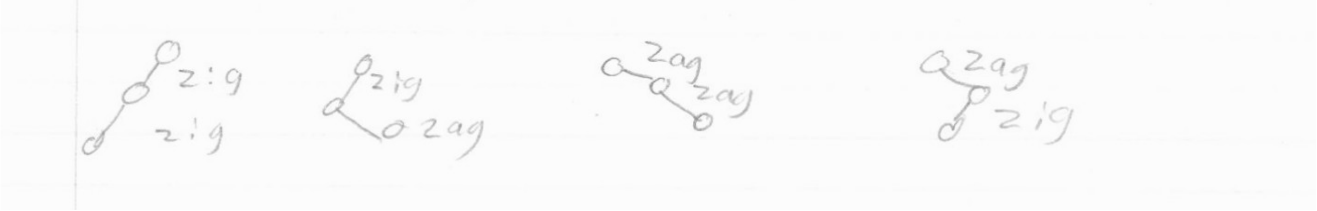
- Rotating nodes is similar to a linked list
  - Just pointer manipulation
- Order of the data matters as it will affect the tree structure (unless you manually balance the data as you go)
- Worst case for a BST (Binary Search Tree) is sorted data
  - Degenerates into a linked list

Splay Trees



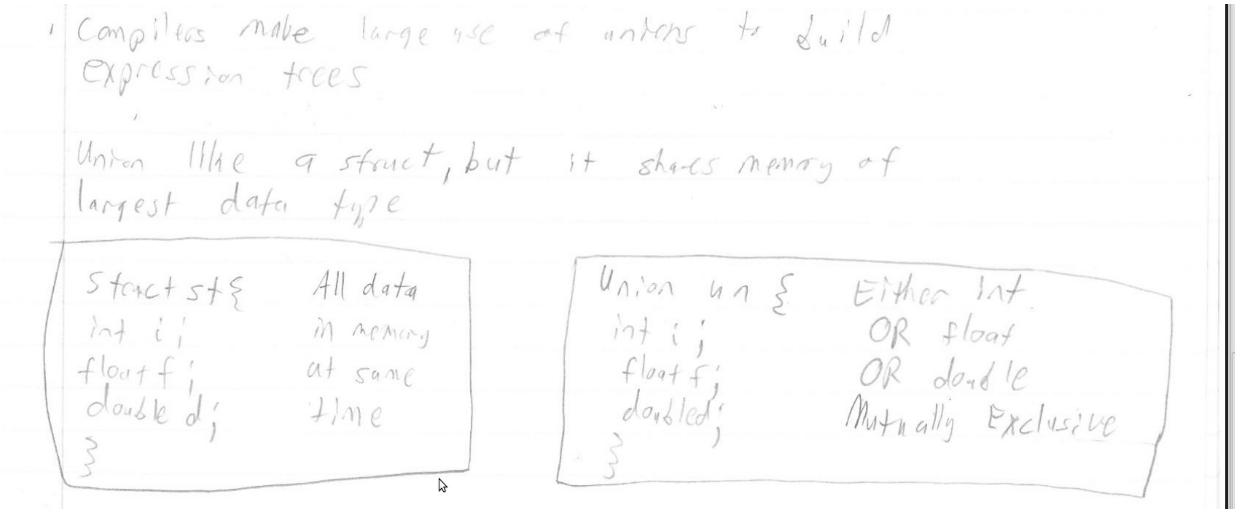
'Splaying' a node

- Moving a newly inserted node into the correct spot is called 'splaying' the node
- Four different types of patterns from grandparent to grandchild
  - zig-zig
  - zig-zag
  - zag-zag
  - zag-zig



- Splaying trees has good average performance
- Meant to be used on large sets of data
- Splaying algorithm is trivial if not keeping the tree balanced
  - Keeping it balanced is another matter entirely however

- Key to expression trees is **post order traversal**
- Compilers make large use of unions to build expression trees
  - A Union is like a struct, but it shares the memory of the largest data type.
  - 'Mutually Exclusive struct'



Compilers make large use of unions to build expression trees

Union like a struct, but it shares memory of largest data type

```
Struct st{     All data
int i;         in memory
float f;       at same
double d;      time
}
```

```
Union un {     Either int
int i;          OR float
float f;        OR double
doubled;       Mutually Exclusive
}
```

- Unions are a great way to save space
  - Think of a networking bottleneck
- const
  - *const int i = get(); //i is constant but only known at runtime*
- constexpr
  - *constexpr int foo(); //can be evaluated at compile time*
- Midterm exam will cover up to what we went over on the 17th