Understanding of data structures

• Stack
  • Notation, -: pop and print, everything else: push
  • Input: it was - the best - of times - - - it was - the - -

What’s the printed output? What’s left in the stack?

• Tips: trace the stack very carefully, and keep in mind that stack is Last In First out and only has one access point: the top
  • During the class, we traced the pushes and pops until the 2nd was, highlighted in red in the input above. By that point:
    • printed output: was best times of the
    • Stack: it, it, was (top)

• Finishing the whole input sequence will lead to
  • printed output: was best times of the was the it
  • What’s left in the stack: it (top)
Understanding of data structures

• Queue
  • Notation, -: dequeue and print, everything else: enqueue
  • Input: it was - the best - of times - - - it was - the - -

What’s the printed output? What’s left in the queue?

• Tips: trace the queue very carefully, and keep in mind that queue is First In First out and has two access points: the front and the end
• During the class, we traced the enqueues and dequeues until the 2nd was, highlighted in red in the input above. By that point:
  • printed output : it was the best of
  • queue: (front) times it was (end)

• Finishing the whole input sequence will lead to
  • printed output : it was the best of times it was
  • What’s left in the queue: (front) the (end)
Tracing the Code

Tips: trace the stack \( s \) and the updated value of \( n \) carefully.

What is the printed output for
- \( n = 6? \)
  - 110
- \( n = 7? \)
  - 111
- \( n = 8? \)
  - 1000

How is the printed output related to \( n? \)
- It is the binary representation of the integer value \( n \).

Questions: What do we use a stack here? Why not a queue?
Tracing the Code

Tips: trace the queue $q$ and the values of $a$ and $b$ carefully.

What is the printed output?

0
1
1
2
3
5
8
13
21
34

What is the functionality of this code segment?

- Compute and print the Fibonacci numbers
Tracing the Code

Suppose x is a linked list and t is a node.

What does the following code segment do?

```java
    t.next = x.next;
    x.next = t;
```

How about the following code segment?

```java
    x.next = t;
    t.next = x.next;
```
Tracing the Code

• Trace the recursion carefully, and keep track of the current node x. The next slide shows the detailed tracing that we did in class.

• Starting with node R, What is the printed output?
  
  R
  H
  C
  E
  L
  J
  V

• Is this traversal pre-order, in-order, post-order, or level-order?
  This is pre-order traversal.
Tracing the Code

```
private static void traverse(Node x) {
    if (x == null) return;
    StdOut.println(x.key);
    traverse(x.left);
    traverse(x.right);
}
```

- `x: R`
  - print `R`
  - `x.left: H` (next `x`)
    - print: `H`
    - `x.left: C`
      - print `C`
      - `x.left: null, return`
      - `x.right: E`
        - print `E`
        - `x.left: null`
        - `x.right: null`
  - `x.right: L`
    - print `L`
    - `x.left: J`
      - print `J`
      - `x.left: null`
      - `x.right: null`
    - `x.right: null`
  - `x.right: V`
    - print `V`
    - `x.left: null`
    - `x.right: null`
Problem Solving

3-SUM (REVISITED)

3-SUM. Given n distinct integers, find three such that \( a + b + c = 0 \).

Goal. \( \Theta(n^2) \) expected time; \( \Theta(n) \) extra space.

• Recall: Our earlier solution to this problem, discussed in the first part of the course, is to use a 3-layer nested loop to check all triples, and increase the count for each triple with a sum of 0.

Initialized count to 0

for \( i \) in 0 ... \( n - 1 \) //\( n \) is the length of the data array

    for \( j \) in \( i+1 \) to \( n-1 \)

        for \( k \) in \( j+1 \) to \( n-1 \)

            if (data[\( i \)] + data[\( j \)] + data[\( k \)] == 0)

                increase count by 1

• Running time: Theta(\( n^3 \)), Extra space (not counting the data array): Theta (1)
Problem Solving

3-SUM (revisited)

3-SUM. Given \( n \) distinct integers, find three such that \( a + b + c = 0 \).

Goal. \( \Theta(n^2) \) expected time; \( \Theta(n) \) extra space.

• To get an algorithm with \( \Theta(n^2) \) expected running time, we cannot do three nested loops. We can do two, and we can try to use the allowed extra space to make this happen.

• Key Observations:
  • given two numbers \( \text{data}[i] \) and \( \text{data}[j] \), the third number needed to make the sum to be zero is
  \(- (\text{data}[i] + \text{data}[j])\), i.e. the negation of the sum of \( \text{data}[i] \) and \( \text{data}[j] \).
  • In our original solution, we use the inner most loop controlled by \( k \) to check every number after \( j \) to essentially check if \( \text{data}[k] \) is \(- (\text{data}[i] + \text{data}[j])\)
  • Instead of using a loop, let us put all the data entries and their indices to a hashtable: key – Integer type, data entries, Value – Integer type, each data entry’s index. For example:
    • E.g. Data: -2, 4, 0, 3, -1, -3 (keys)
    • Index: 0, 1, 2, 3, 4, 5 (values)
  • This hashtable takes extra linear space and can determine in \( \Theta(1) \) expected time whether a specified int value, e.g. \(- (\text{data}[i] + \text{data}[j])\), is a contained key and its corresponding index.
Problem Solving

3-SUM (REVISITED)

3-SUM. Given $n$ distinct integers, find three such that $a + b + c = 0$.
Goal. $\Theta(n^2)$ expected time; $\Theta(n)$ extra space.

New solution -- main idea:
Use an int variable count to keep track of the number of triples with a sum of 0.
Use a hash table to store the data entries and their corresponding indices.
Use a two-level nested loop to go through all pairs of the data. For each pair
data[i] and data[j] with $j > i$, check if $-(data[i] + data[j])$ is a key contained in the
hash table. If it is contained and its index (conceptually k) is greater than j, then
increase the count by 1, since data[i], data[j], and data[k] sum to zero.
Problem Solving

New solution -- pseudo code:

Initialized count to 0
Create a hash table to store all the data entries and their corresponding indices
for i in 0 ... n-1    //n is the length of the data array
    for j in i+1 to n-1
        if -(data[i] + data[j]) is a key contained in the hash table and its index is > j
            increase count by 1

• Running time: Theta(n^2) expected time, Extra space: Theta (n)