UCLA Samueli Computer Science

CS32: Introduction to Computer Science II

Discussion Week 4

Junheng Hao, Arabelle Siahaan

Apr. 26, 2019

Announcements

- Homework 2 is due on 11PM Tuesday, April 30.
- Midterm 1:)

Outline Today

- Linked List (One last question)
- Stack
- Queue
- Homework 3: Guide

Linked List Problem: Reverse

Leetcode questions #206

UCLA Samueli Computer Science

Question: How to reverse a (single) linked list?

Example:

```
Input: 1->2->3->4->5->NULL
Output: 5->4->3->2->1->NULL
```

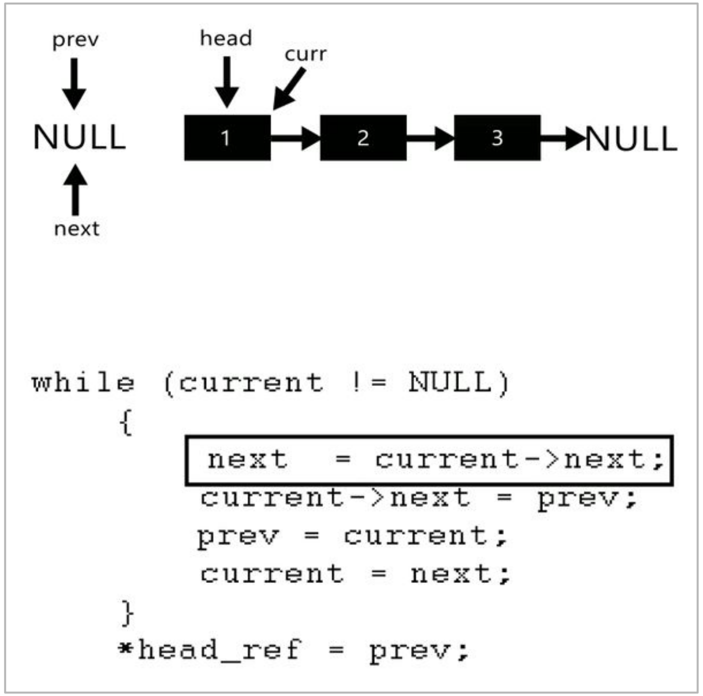
```
// One possible solution
Node* reverseList(struct ListNode* head)
{
    Node *prev=NULL,*current=head,*next;
    while(current) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    return prev;
}
```

Problem: Reverse Linked List

Leetcode questions #206

UCLA Samueli Computer Science

Let's see what happens in these lines of codes! [Link]



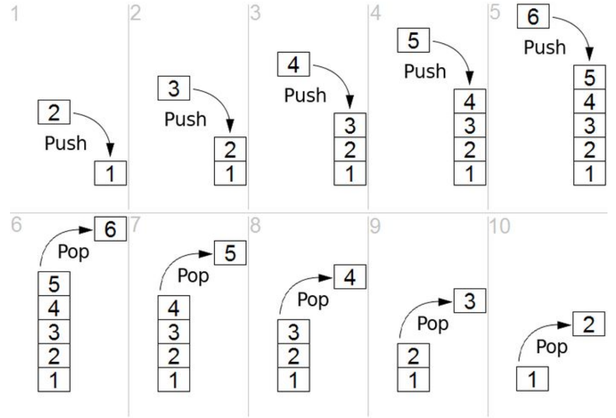
```
while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
```

Stack: FILO

Review: Basics

- FILO: First In, Last Out
- A standard stack implementation
  - `Push()` and `pop()`
  - Other methods: `top()`, `count()`
- Applications:
  - Stack memory: function call
  - Check expressions: matching brackets
  - Depth-first graph search
- Question: How do you implement stack with linked list / (dynamic) arrays?

```
class Stack
{
public:
        bool push(const ItemType& item);
        ItemType pop();
        bool empty() const;
        int count() const;
private:
        // some features
};
```



Stack

Implement stack with linked list

- Container: linked list
- Functions:
  - `push()`: Insert node before head.
  - `pop()`: Remove head and return the head value.
  - `top()`: Read head node.
  - `count()`: Maintain a private `int` member.

- Question: How about using arrays?

Stack
Examples

- Given a math expression or text sequence:
  - 6+((5+2)*3-(7+11)*5)*6 → Consider calculation of Reverse Polish Notation
  - Latex: f^{\text{DNN}}(X,\mathbf{W}) = \sigma \left(\mathbf{W} \cdot X + \mathbf{b} \right)

$$\boxed{f^{\text{DNN}}(X,\mathbf{W}) = \sigma \left(\mathbf{W} \cdot X + \mathbf{b} \right)}$$

- How to check the brackets of all types are valid in the sequence? How to calculate expression in Reverse Polish Notation (RPN)?

Regular Expression: 2 + 3 * (5 - 1)

Reverse Polish Notation (RPN): [2] [3] [5] [1] [-] [*] [+]

Stack
Examples

- Given a math expression or text sequence:
  - 6+((5+2)*3-(7+11)*5)*6 → Consider calculation of Reverse Polish Notation
  - Latex: f^{\text{DNN}}(X,\mathbf{W}) = \sigma \left(\mathbf{W} \cdot X + \mathbf{b} \right)

$$f^{\text{DNN}}(X,\mathbf{W}) = \sigma \left(\mathbf{W} \cdot X + \mathbf{b} \right)$$

- How to check the brackets of all types are valid in the sequence? How to calculate expression in Reverse Polish Notation (RPN)?

Regular Expression: 2 + 3 * (5 - 1)

Reverse Polish Notation (RPN): [2] [3] [5] [1] [-] [*] [+]

Stack

Application: Algebraic Expressions

- **Infix Notation**
  - Operators are written in between their operands → X + Y
  - Ambiguous - needs extra rules built in about operator precedence and associativity and parentheses
- **Postfix Notation**
  - Operators are written after their operands → X Y +
  - Operators are evaluated left-to-right. They act on the two nearest values on the left.
- **Prefix Notation**
- **Tasks**
  - Evaluating Postfix Expressions
  - Converting Infix to Postfix Expressions

Stack

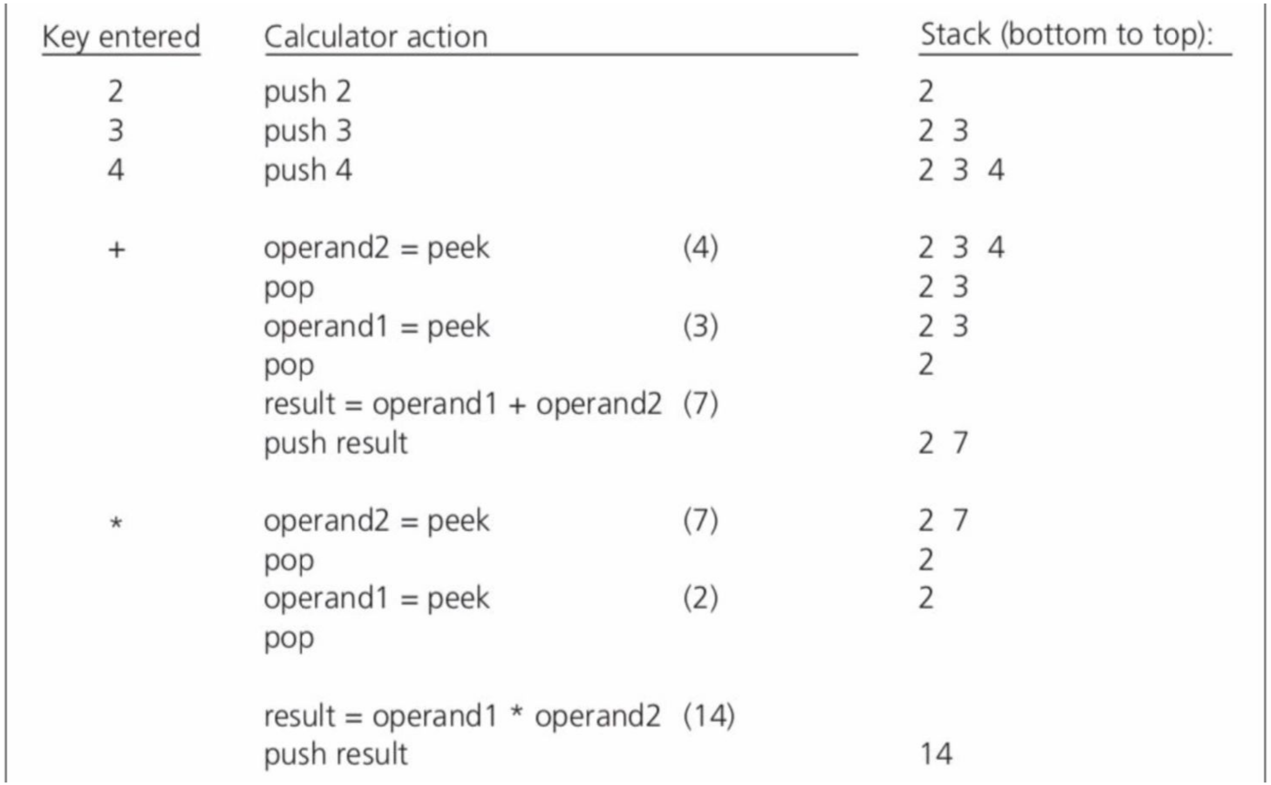
Application: Evaluating Postfix Expressions

Postfix Expression

2 3 4 + *

Infix Expression

2 * (3 + 4)

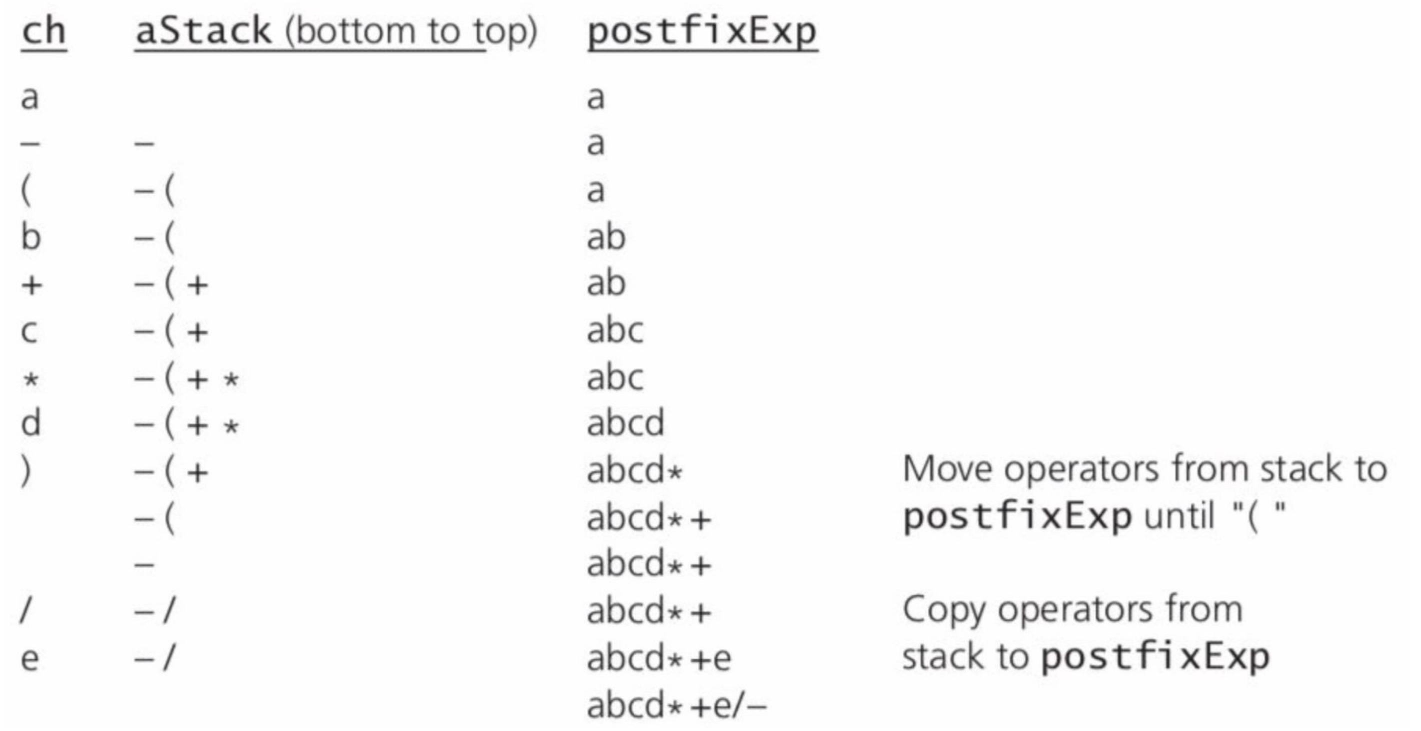


Key entered	Calculator action		Stack (bottom to top):
2	push 2		2
3	push 3		2 3
4	push 4		2 3 4
+	operand2 = peek	(4)	2 3 4
	pop		2 3
	operand1 = peek	(3)	2 3
	pop		2
	result = operand1 + operand2	(7)	
	push result		2 7
*	operand2 = peek	(7)	2 7
	pop		2
	operand1 = peek	(2)	2
	pop		
	result = operand1 * operand2	(14)	
	push result		14

Stack

Application: Converting Infix to Postfix Expressions

Infix expression: `a - ( b + c * d ) / e`

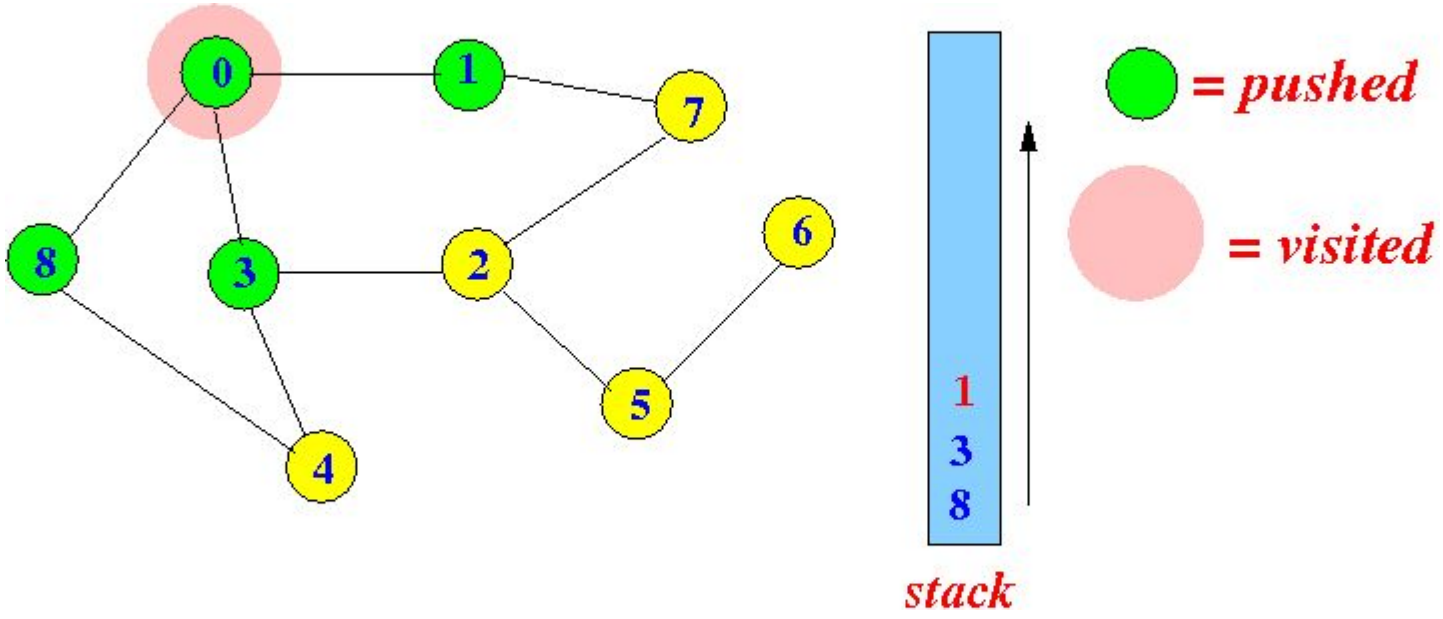


ch	aStack (bottom to top)	postfixExp	
a		a	
–	–	a	
(	– (	a	
b	– (	ab	
+	– (+	ab	
c	– (+	abc	
*	– (+ *	abc	
d	– (+ *	abcd	
)	– (+	abcd*	Move operators from stack to
	– (	abcd*+	postfixExp until "("
	–	abcd*+	
/	– /	abcd*+	Copy operators from
e	– /	abcd*+e	stack to postfixExp
		abcd*+e/–	

Stack

Example: stack and depth-first search

- Depth-first Search (DFS) on graph (will be later lectures or CS180)



Stack*

Example: Use stack to implement DFS [Link]

UCLA Samueli Computer Science

```
void Graph::DFS(int s)
{
    vector<bool> visited(V, false);    // Initially mark all vertices as not visited
    stack<int> stack; // Create a stack for DFS
    stack.push(s);          // Push the current source node
    while (!stack.empty())
    {
        s = stack.top(); // Pop a vertex from stack and print it
        stack.pop();
        // Print the popped item only if it is not visited.
        if (!visited[s])  { cout << s << " "; visited[s] = true;  }
        for (auto i = adj[s].begin(); i != adj[s].end(); ++i)
            if (!visited[*i])
                stack.push(*i);
    }
}
```

Note: Get all adjacent vertices of the popped vertex s.
If the adjacent has not been visited, then push it to stack .

Stack*

Problem: Find largest rectangle all-1sub-matrix

Question:

Given a binary matrix, find the area of maximum size rectangle binary-sub-matrix with all 1's.

Level: (Super) Difficult

Example 1:

```
Input:     0 1 1 0
           1 1 1 1
           1 1 1 1
           1 1 0 0

Output:    8
```

Example 2:

```
Input:     0 1 1 0
           1 1 1 1
           1 1 1 1
           1 1 0 0
           1 1 0 1
           1 1 1 0

Output:    10
```

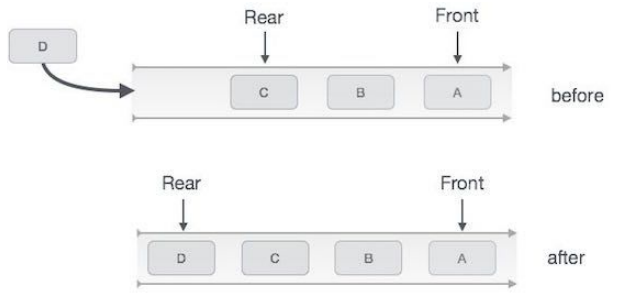
```
// Your solution here

// Think about how you can use stack to solve
this problem and compare with brute-forth
methods

// Tips: Think about the problem of Largest
Rectangular Area in a Histogram solved by
using stack

// One possible solution: [Link]
```

Queue: FIFO

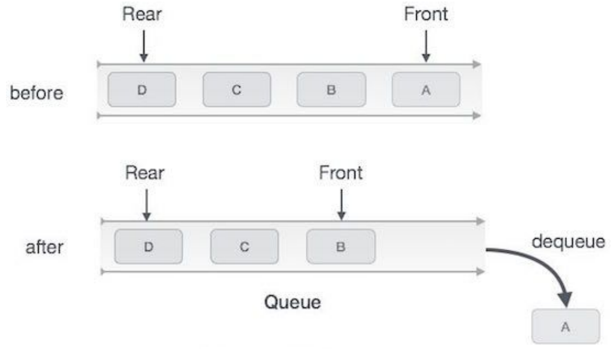
Review: Basics

UCLA Samueli Computer Science

- FIFO: First In, First Out
- Basic methods:
  - `enqueue(), dequeue()`
  - `front(), back()`
  - `count()`
- Applications
  - Data streams
  - Process scheduling (DMV service request)
  - Breadth-first graph search
- How to implement queue with linked lists or dynamic arrays?



Queue Enqueue



Queue Dequeue

Queue

Extension: Deque (double-ended queue)

UCLA Samueli Computer Science

```
class Deque
{
    public:
        bool push_front(const ItemType& item);
        bool push_back(const ItemType& item);
        bool pop_front(const ItemType& item);
        bool pop_back(const ItemType& item);
        bool empty() const; // true if empty
        int count() const; // number of items
    private:
        int size; // Some data structure that keeps the items.
};
```

Question: How to implement `class Deque` with linked lists?

Queue

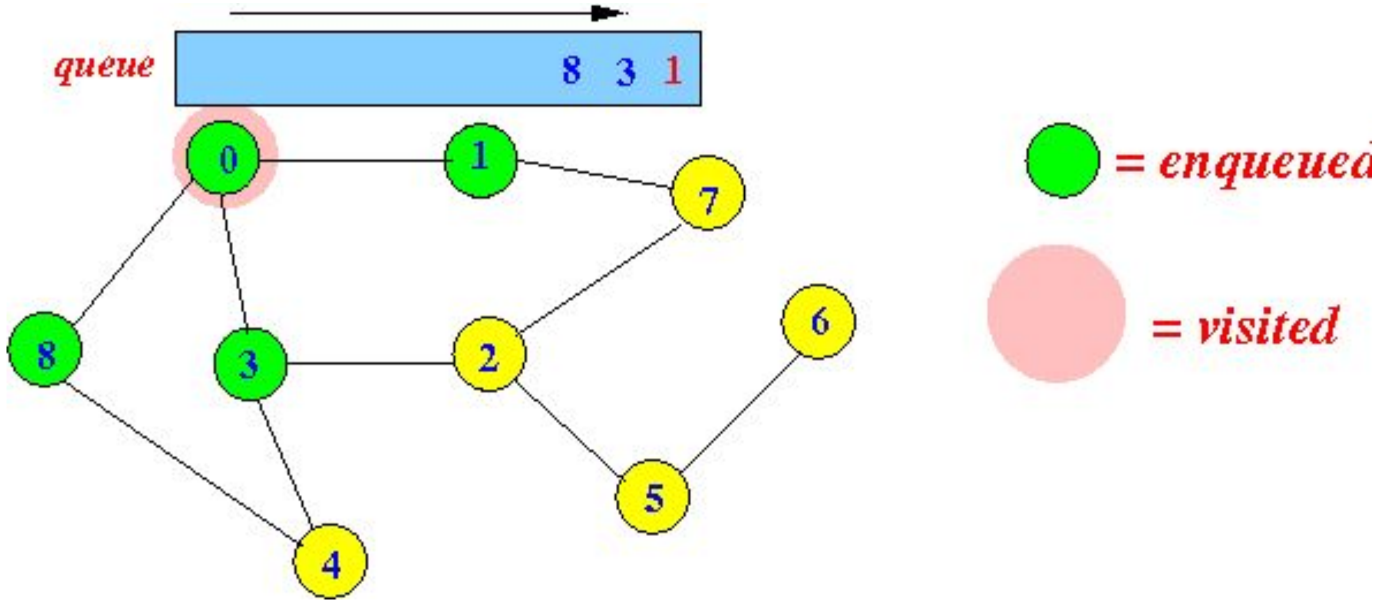
Extension: Priority queue

- Data: A finite number of objects, not necessarily distinct, having the same data type and ordered by priority
- Operations:
  - Add a new entry to the queue based on priority
  - Remove the entry with the highest priority from the queue
- We will learn priority queue (and heap) later this quarter after tree!

Queue*

Example: queue and breadth-first search

- Breadth-first Search (BFS) on graph (will be later lectures or CS180)



Queue*

Example: Use queue to implement BFS [Link]

UCLA Samueli Computer Science

```
void Graph::BFS(int s)
{
    bool *visited = new bool[V];  // Mark all the vertices as not visited
    for(int i = 0; i < V; i++) { visited[i] = false; }
    list<int> queue; // Create a queue for BFS
    visited[s] = true; // Mark the current node as visited and enqueue it
    queue.push_back(s);
    list<int>::iterator i;
    while(!queue.empty()) {
        s = queue.front(); // Dequeue a vertex from queue and print it
        queue.pop_front();
        for (i = adj[s].begin(); i != adj[s].end(); ++i) {
            if (!visited[*i]) {
                visited[*i] = true;
                queue.push_back(*i);
            }
        }
    }
}
```

Note: Get all adjacent vertices of the dequeued vertex s. If a adjacent has not been visited, then mark it visited and enqueue it.

Suggestions on Stack and Queue

- Drawing pictures and carefully tracing the current status through your code, updating the picture with each statement, can help you find bugs in your code.
- Infix to postfix conversion is very important!
- We have shown that stack and queue can be used for traversal on graphs. They can also be applied on Trees (topics later in this quarter). Understand different data structures you use will result in different traversal order.
- You can use the given Standard Template Library (STL) to implement stack and queue.

```
#include <stack>
#include <queue>
std::stack<type> variableName;
std::queue<type> variableName;
```

Hints for Homework 2 (P1-4)

UCLA Samueli Computer Science

Task: Use stack or queue to implement the `pathExists()` function in the maze.

```
// Homework 2 (P1-4): One possible solution
class Coord{}; // member: m_r, m_c

void explore(string maze[], stack<Coord>& toDo,
int r, int c){
  // push (r,c) to stack if OPEN (not SEEN)
}

bool pathExists(string maze[], int nRows, int
nCols, int sr, int sc, int er, int ec){
  // Some basic case to return false??
  // initialize stack or queue
  // Push starting coordinate (sr,sc)
  // Loop & check (when stack/queue is non-empty)
  // Return true or false
}
int main(){ … } //test your code
```

Note & Reminders:

1. Follow the pseudocode if you still have no idea. The question itself is somewhat similar to BFS and DFS.
2. How to mark the `(r,c)` as visited?
3. Using stack and queue will result **different** Coord visit order (in Problem 2 and Problem 4), in your `hw.txt` to be submitted.
4. Reminder: You can use the given stack or queue class in C++ or your implementation.

Hints for Homework 2 (P5)

Task: *This problem will appear later.*

```
// Homework 2 (P5)
// Yeah, will appear later. :)
```

Note & Reminders:

1. The hints will appear later.

UCLA Samueli Computer Science

Break Time! (5 minutes)

Q & A

Group Exercises: Worksheet 1

- Exercise problems from **Worksheet 3** (see “LA worksheet” tab in CS32 website). Answers will be posted after all discussions.
- Questions for today:
    - TBD

Thank you!

Q & A