Lists Stacks & Queues

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

May 20, 2017

Overview I

1. Lists
    - Comparisons
    - Amortized Analysis

2. Stacks
    - Intro
    - Array implementation
    - Linked-list implementation
    - Parenthesis checking

Comparisons

- Is a linear data structure.
- A List is a collection where the elements are ordered and therefore each element has an index which is the position in the list. Allows duplicates.
- A Set is an unordered collection in which no two elements are identical.
- A Bag is an unordered collection in which can have duplicates.

	Linked List	Array based
Access	$\Theta(n)$	$\Theta(1)$
Insertion	$O(n)$	$O(1)$
Deletion	$O(n)$	$O(n)$

Amortized Analysis

- In Amortized analysis, the time taken to execute a sequence is averaged over all the operations executed.
- Even though one of the operations in the sequence might take a long time, the average time taken over all operations is small.
- This is not the same as the average case.
- This guarantees the average performance of each operation in the worst-case.

Intro

- It's a list structure where all operations occur on one end of the list, known as the top of the stack.
- To add an element is called a push operation.
- To remove an element is called a pop operation.
- To get the element at the top of the stack is called a top operation.

Array implementation

- Requires a means of handling array overflow, i.e. double size of array when full.
- `push()` has an amortized complexity of $O(1)$ in the worst case.
- `top()` does not alter the stack at all and simply gives the top element, this is $O(1)$ in the worst case.
- `pop()` only alters the last element, nothing is shifted and therefore has complexity $O(1)$ in the worst case.

Linked-list implementation

- `push()` has a complexity of $O(1)$ in the worst case, this is NOT amortized due to the lack of array overflow requirement.
- `top()` has $O(1)$ complexity in the worst case.
- `pop()` has $O(1)$ complexity in the worst case.

Parenthesis checking

- Stacks can be used to determine is parenthasis match correctly or not. e.g. $[a(b+c)da/c+e]/b$
- Use a stack to push the left side of the parenthesis and when the right side has been found, pop the parenthasis.
- When an item is popped, it is compared to the found parenthesis and if they are of the same type, then it's matching.

The End