3137 Data Structures and Algorithms in C++

Lecture 3

July 12 2006

Shlomo Hershkop

Announcements

- Homework 2 out tonight
- Please make sure you complete hw1 asap
  - if you have issues, please contact me
- will be doing more hands on code today to help you get started
- please do the reading to keep up with the theory

Outline

- lists applications
- Stacks
  - code
  - applications
- Queues
- Trees

- Reading: Chapter 3,4 (beginning)

3

Lists

- remember we discussed a DS called a list
- group of items
- set of operations
- depending on implementation, different set of run times
- you choose appropriate implementation depending on task
  - given a project constraint…

4

Dynamic memory allocation

- one of the nice things about C/C++ is the ability to allocate memory on the fly
- one of the bad things about C/C++ is the ability to allocate memory on the fly

5

memory types

- generally memory comes from either stack or heap
- anyone know the difference ?

6

- we start with fixed memory allocation for things we know ahead of time
- stack builds up in memory
- heap builds down
- here it is visually:

7

- so because stack builds up, its easy to keep it neat
  - why is that ??
- heap:
  - you can allocate/de-allocate in any pattern so how do you keep track of everything ??

8

lists

- linked list of memory blocks which allow you to traverse to see where I can allocate memory from
  - called free list
- malloc/new
  - remove memory from free list and pass it back to program
- free/delete
  - add it back to the free list

9

definition

- Garbage collection
  - is the process of reclaiming unused free memory space
  - each node of memory has a bit called the “mark” bit
    - 0 = free
    - 1 = reserved
- initialization:
  - each memory node set to free
- marking:
  - if memory in use set to reserved
- gathering:
  - all node marked free are linked into a list which represents all your free memory

10

- since memory can be allocated in different sizes
- you have many choices of how to take a block from the free list
- any ideas what works best ??

11

First Fit Algorithm

- when we need memory of size X
- starts traversing the free list as soon as we find a block of size X or greater, allocate, and put the “change” back on free list
- any ideas on run time ?

12

Best Fit Algorithm

- either exact fit or min leftover

13

Definition

- Memory Fragmentation
  - free space becomes divided into many small pieces
  - need memory of size X but although it exists do not have continuous piece of memory of size X
  - this is not file fragmentation
    - anyone know what this is ??
- so what can we do about it ??

14

solution

- is to go through memory combining all allocated memory on one side and free memory on other side
- need to figure out which memory blocks can be freed
- use of sweeping algorithm
  - white, black, grey lists
  - reachability calculations

15

Stack ADT

- Restricted list
- 2 basic operations:
  - Push
  - Pop
- LIFO DS
- view of only top item in the list

16

- might think its pretty simple
- second most fundamental DS in computer science after arrays

17

- Any idea on runtime ??
- implementation ?

18

implementation

- linked list
  - how this works
- array
  - how this works

19

coding

- what would the code look like ??
- lets start with simple non template version
- say we want to implement an integer stack

20

functions

- isEmpty()
- Push
- Pop

21

Application

- balancing symbols
- lets write some code to see if we have balanced symbols in a block of text

22

functions

- isOpenParent()
- isClosedParent()
- isParent()

- rule:
- if(isParent()){
  - if(isOpenParent())
    - push
  - else
    - pop and compare
    - now what ??

23

- Anyone know what reverse polish notation is ??

24

Postfix Notation

- postfix
- reverse polish notation

- infix

- prefix

25

Example 2

- This is on the next homework

- can you write a palindrome detector

- Too bad, I hid a boot

26

- mentioned it during recursive programming
- A calls B calls C
- LIFO

27

Queue DS

- List with specific property:
  - FIFO
- enqueue
- dequeue
- run time ??

28

Implementation

- Any ideas on how to implement ??
- what will the run time be for each ??

29

Simple

- Array
  - use a simple array
- Linked List
  - use a linked list with head/tail

30

circular list

- alternative is to use an array but with two extra indices showing where the head/tail are located
- What will be the run time ?
- how to determine size??

31

- priority queue
- this queue also keeps track of how important something is…higher importance should go first

32

Reverse

- can you think of a fast way to reverse a queue ??

33

Elevator simulation

- Anyone know what software are run on elevators ??

- how to write code ?
- how to test ?

34

Switch

- ok, lets leave basic data structures and start on more complicated ones….

35

Definition

- Tree
  - a collection of nodes consisting of a root node and zero ro more non empty subtrees each of whose roots are connected by a directed edge from the primary root

36

Definition

- Parent node
  - node at the beginning of a directed edge
- Child node
  - node at the end of a directed edge
- Internal node
  - non root with children

37

- basic tree
  - just root with no children
- leaf
  - tree with no children

38

Definition

- A Path from $node_i$ to $node_k$ consists of a sequence of nodes n_i, n_{i+1},..,n_k such that n_i is the parent of n_{i+1} and $i \leq k$

39

Question

- So what can trees represent ??

- family relationships
- file systems
- organizations
- game strategy
- dictionary

40

- so given a tree structure
- What would be the runtime for finding an element ??
- inserting ?

41

- if the tree has up to M number of children per node then its called a M-ary tree
  - for homework will be coding dictionary, an ideas on how to do it ??
- Binary Tree
  - either single node or node with up to 2 children which are binary trees
  - left, right
- Because trees are recursively defined, the algorithms to manipulate them are usually coded recursively

42

code

```
class TreeNode {
  int item;
  TreeNode *left;
  TreeNode *right;
};

Class BinaryTree{
  TreeNode *root;
};
```

43

definition

- Binary Search Tree
  - tree in which every element to the left is less than current node, and every element to the right is greater than current node

44

- lets do some code
- can you fill this in ??

```
class BST {



};
```

45

functions

- ?constructors ??
- insert
- find
- delete
- print ???

46

Printing a tree

- There are three general ways of printing a tree structure

- inorder
- preorder
- postorder

47

- inorder
  - inorder(left), root, inorder(right)
- preorder
  - root, preorder(left), preorder(right)
- postorder
  - postorder(left), postorder(right), root

48

```
void printinorder(TreeNode *node){
  if(node == NULL)
      return;

  printinorder(node->left);
  printItem(node);
  printinorder(node->right);
}
```

49

- tree height is the path length
- how would you define the height function??

50

```
int height(TreeNode *node) {

   if(node == NULL)
        return -1;

   int lefth = height(node->left);
   int righth = height(node->right);

   if(lefth > righth)
        return 1 + lefth;
   else
        return 1 + righth;
```

51

- find ??
- how does find work ?
- what is the running time ?

52

helper find function

```
bool findh(const int n, TreeNode *node) {
  if (node == NULL)
      return false;
  else if (n < node->item)
      return findh(n, node->left);
  else if (n > node->item)
      return findh(n, node->right);
  else
      return true;
}
```

53

- how do you find the minimum on the BStree ??
- what about the maximum ??

54

```
TreeNode * findmax(TreeNode *node){

   if(node != NULL)
      while(node->right != NULL)
        node = node->right;

      return node;
      }
```

55

- what would insert look like ??

56

```
void insert(int n, TreeNode* node) {

if(node == null)
  node = new TreeNode(n,NULL,NULL);

??????


}
```

57

Next time

- wrap up homework
- get eclipse working ☺
  - pay attention to your version and the c++ plug in version
- reading chapter 3,4

58