COP 3530 Fall 2013

Data Structures and Algorithms

Assignment 6. Due: Nov. 20, 11:55 p.m. via Sakai

In this problem you will solve the Bin Packing Problem using two algorithms/ approaches: First Fit and Best Fit. You will use a Max Winner tree to implement First Fit and Binary Search Tree to implement Best Fit using a pointer-based implementation of both trees.

The main points we shall be covering are:

* Pointer Based Data Structures
* Tournament Trees, Binary Search Trees
* Continuing use of makefiles, abstract data types in C++, and separate compilation

Tournament trees

A tournament tree is a complete binary tree. Winner trees are most easily visualized as a Tennis match where the two children nodes play a match and the winner is promoted.

A Binary Search Tree is a binary tree where each node has a (key, value) pair and for every node x, all keys in the left subtree of x are smaller or equal to that in x. Also, for every node x, all keys in the right subtree of x are greater or equal to that in x

The Bin Packing Algorithms are defined below in the problem statement. You are to accomplish the following two tasks:

1]

Use the max winner tree to implement the First Fit Algorithm to solve the bin packing problem for a series of integers. In the First Fit Algorithm, objects are considered for packing in the order in which they are entered. Object i is packed into the left-most bin into which it fits.

**First Fit**

* Bins are arranged in left to right order.
* Items are packed one at a time in the given order.
* The current item is packed into leftmost bin into which it fits.
* If there is no bin into which current item fits, start a new bin.

2]

Use a Binary Search Tree to implement the Best Fit Algorithm for the same series of numbers.

**Best Fit**

* Items are packed one at a time in the given order.
* To determine the bin for an item, first determine the set S of bins into which the item fits.
* If S is empty, then start a new bin and pack the item into this new bin.
* Otherwise, pack the item into the (in S) that has the least available capacity.

You will use a pointer-based tree implementation this time. For First Fit using a max winner tree. You will need to keep track of the path down the tree in order to reverse it and ascend the tree to replays matches.

Files provided by instructor

* myException.cpp and myException.h. Same as in previous Assignments.

Files you must develop

* main.cpp
* BinPacking.cpp and BinPacking.h (this drives both bin packing techniques)
  + Include the methods for First Fit and Best Fit
* BinaryTree.cpp and BinaryTree.h
* MaxWinnerTree.cpp and MaxWinnerTree.h
* Makefile. Each .cpp file must compiled separately. No templates. Be sure to provide “clean” and ‘tar” targets in the make file.

Some sample output is as follows:

thunder:8% ./main

Please choose the algorithm for test: 1. Firstfit; 2. Bestfit; 3. Quit?1

Enter number of objects and bin capacity for First Fit

12

6

Enter space requirement of object 1

2

Enter space requirement of object 2

5

Enter space requirement of object 3

5

Enter space requirement of object 4

1

Enter space requirement of object 5

1

Enter space requirement of object 6

3

Enter space requirement of object 7

4

Enter space requirement of object 8

6

Enter space requirement of object 9

2

Enter space requirement of object 10

5

Enter space requirement of object 11

6

Enter space requirement of object 12

1

Pack object 1 in bin 1

Pack object 2 in bin 2

Pack object 3 in bin 3

Pack object 4 in bin 1

Pack object 5 in bin 1

Pack object 6 in bin 4

Pack object 7 in bin 5

Pack object 8 in bin 6

Pack object 9 in bin 1

Pack object 10 in bin 7

Pack object 11 in bin 8

Pack object 12 in bin 2

Please choose the algorithm for test: 1. Firstfit; 2. Bestfit; 3. Quit? 2

Enter number of objects and space requirements for Best Fit

Enter number of objects and bin capacity

12

6

Enter space requirement of object 1

2

Enter space requirement of object 2

5

Enter space requirement of object 3

5

Enter space requirement of object 4

1

Enter space requirement of object 5

1

Enter space requirement of object 6

3

Enter space requirement of object 7

4

Enter space requirement of object 8

6

Enter space requirement of object 9

2

Enter space requirement of object 10

5

Enter space requirement of object 11

6

Enter space requirement of object 12

1

Pack object 1 in bin 1

Pack object 2 in bin 2

Pack object 3 in bin 3

Pack object 4 in bin 3

Pack object 5 in bin 2

Pack object 6 in bin 4

Pack object 7 in bin 1

Pack object 8 in bin 5

Pack object 9 in bin 4

Pack object 10 in bin 6

Pack object 11 in bin 7

Pack object 12 in bin 4

• Your submission MUST run on thunder.cise.ufl.edu. That is where it will be run and graded.

The simplest syntax of the tar command is as follows.

tar cvf (tar\_file\_name) (file 1) (file 2) (file 3)...

To extract the contents of a tar file:

tar xvf (tar\_file\_name)

As before, use the Makefile to create the tarfile.

It is STRONGLY recommended you verify your submission is successful by downloading it from Sakai (into a separate directory), untarring it, and running make.