Assignment 5

Posted on: 3/28/14

Due on: 4/7/14 at 11:55 pm (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**.

The files we will provide to you:

- **BinaryTree.h**
- **BinaryTree.cpp**
- **BinaryTreeNode.h**
- **BinaryTreeNode.cpp**
- **BinarySearchTree.h**
- **BinarySearchTree.cpp**
- **MaxWinnerTree.h**
- **MaxWinnerTree.cpp**
- **BinPacking.h**
- **BinPacking.cpp**
- **main.cpp**
- **inputFile.txt**
- **Makefile**

You are asked to implement **FIRST FIT and BEST FIT** methods in the **BinPacking.cpp**. The detail of these methods are given below:

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.
- Use a Max Winner tree to implement the First Fit Algorithm for the same series of numbers.

The function prototype is given below:

```
void BinPacking::firstFitPack(int *objectSize, int numberOfObjects, int binCa-
pacity)
```

where objectSize is the array of objects to be packed, numberofObjects is the number of Objects to be packed, binCapacity is the capacity of each bin.

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. If there is a tie, beak it with smallest binNumber.
- Use a Binary Search Tree to implement the Best Fit Algorithm for the same series of numbers.

The function prototype is given below:

```
void BinPacking::bestFitPack(int *objectSize, int numberOfObjects, int binCapac-
ity)
```

where objectSize is the array of objects to be packed, numberofObjects is the number of Objects to be packed, binCapacity is the capacity of each bin.
You will test the above two methods on an input given in **inputFile.txt**. In main file, the code for displaying the bin sequence is already written.

Creating and Extracting a tar file:

Linux/ Macintosh

- To create a tar file: **tar cvf (tar file name) (file 1) (file 2) (file 3)...**
- To extract the contents of a tar file: **tar xvf (tar file name)**

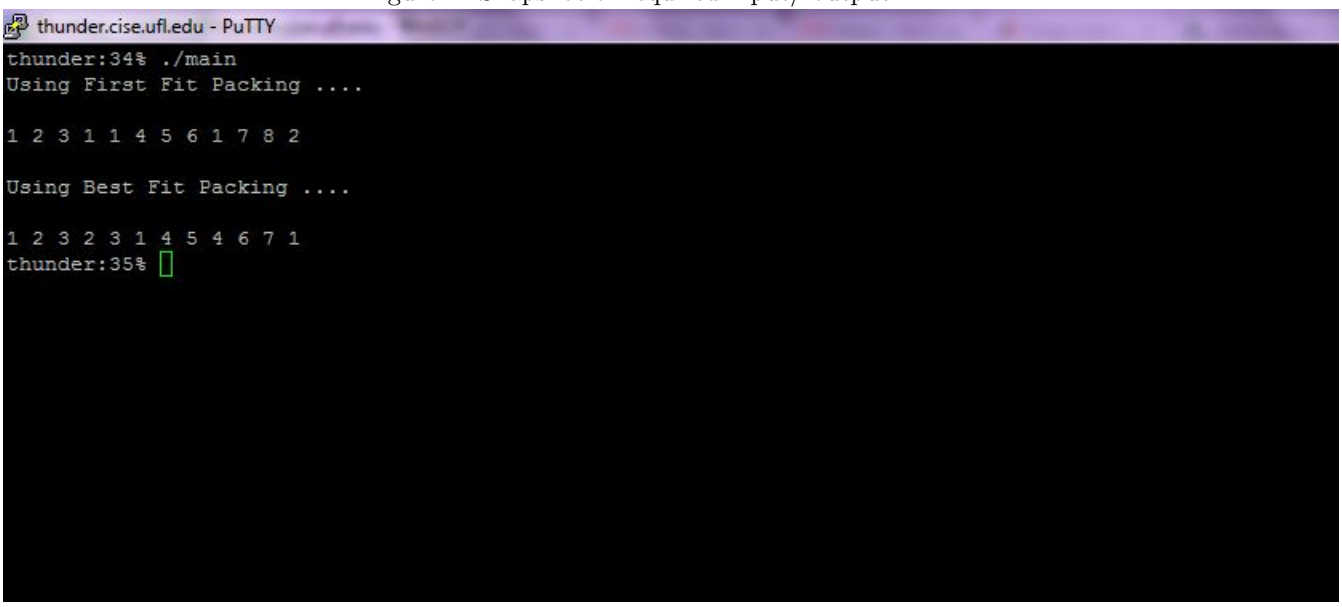
Windows

- You can use **ALZip** to create .tar file.

Things to Remember before Submission:

1. Check for the submission deadline (**both date and time**) and make sure you submit your **.tar file** before the deadline.
2. **LATE SUBMISSIONS ARE NOT ALLOWED.**
3. You should test your code on **thunder machine. TA will run your code on thunder machine and if it fails to compile there, you will be penalized.**

Figure 1: Snapshot of required input/ output



4. TA will only do **make** and then run the executable file, **e.g. ./main**.
5. You should submit **ONLY** a **.tar** file through **SAKAI** consists of **all the .h, .cpp and makefile.** The name of the **.tar** file should contain your name and UFID.
6. The output of your submission should be **exactly like (not "almost like")** the snapshot in Figure 1.
7. You can write any helper/ auxiliary method needed for the implementation of the two methods you are responsible for.