Stephen Bleden

Chris Ruiz

Meghan Haukaas

**Problem 3 - Bin Packing: First Fit and Best Fit**

**Problem**

This problem asks us to place elements of a given size into bins of a given size, using two different algorithms. Both of these algorithms must have a time class of θ(N log N), where N is the number of elements to be inserted. Each algorithm solves a different problem; first-fit inserts elements one-by-one as soon as they are known, and best-fit inserts elements after the entire set of elements to be inserted is known.

We need to design a container that represents a bin. We also need a way of creating new bins of a defined or default size on demand. To accurately test the time complexity of these algorithms, we will need a way to generate random test cases of size N. To test specific scenarios and edge cases, we will need a way to define a bin packing problem in a file and load that file into our algorithms.

**Problems with the Problem**

The naive approach has time complexity θ(N2). Optimizing our algorithms to process N elements in θ(N log N) time will be a significant challenge. To test this time complexity fully, we will need to design clever test cases that push the algorithms away from their most efficient states. We will also need to design intelligent methods of verifying this time class.

*The end.*

But there's antoher the end, end-ier.

Fixed the one below. :)

MY FONT IS CORRUPTED :<

This section still in the stone age of space-filling fonts.

Wow fail page break doesn't even show me how much of a page is left.

None pages. Hooray

Our paper is so tiny, but I like it.

(Obviously we'll need to determine and define these states then if we'll be using them as a point of reference.) Those are the test cases we define in task 3 I'm just trying to come up with a way to fill this haha :3 THIS SHALL BE ORANGE

lol what more is there to say then~ HOW DARE YOU

Haha it was like lol bitch please

Wow shit that one ctrl z undid a lot........ Even reset the font I'm typing in.

It might be interesting to see what cases cause first fit to be better than best fit and vice versa. I vote w put that off until we're sure we have time for that. I don't think it's part of the required problem? Our test cases will need to account for that. Not inherently haha, fair enoug. Probly part oof the discussion.

One of the main tasks we will face is the implementation of the desired time complexity from our starting case (the naïve approach) of O(N^2) (for N elements). I feel like that sentence is as organized as my sleep schedule :[ You know my pain too then! :DDDDDDD Designing two algorithms that are capable of this efficiency will be a challenge.

Betterrrrr!

Uhhhhhhhhhhhhhhhhhhhh code

Time complexity

stufffffffffffffffff :D

The book says the dumb algorithm is N^2, since is scans down each bin every time. That must mean N insertions counts. So it does actually want us to insert N elements in N log N.

Getting the algorithm all shiny is gonna be crucial in this one if we're timing it. Good thing the book tells us exactly what to do. LOL

I'm guessing that we're assuming doubles in this? Maybeeeee NAHHH, I'm thinkin' unsigned shorts.

Put the things in the things, with some decisions. Fine by me.

Uhhhh time complexities of the bad versionssss of the sorts?

Not the wait wait okay so I thought those were the best cases though, isn't it significantly worse? LOL notessss

N log N is average for good sorts, like quick and merge.

Quick's worse case is 2^n

Merge and quick both have a best case of N log N too I think.

Some other sorts have a best case of N if the list is already sorted, like butter. I mean bubble. Wat. Why. I was listening to something about bread in the background.

OBJECTION! - Relevance?

LOL no worries I am distracted too haha okay

Says the guy that was distracted by bread talk ;P

\*fixed it.

*The end. Part 2.*