Ryan Hawkins

Program 2 write-up

AI Bagger

Discretionary Note: I have had a rather tough time getting a final working model together. I wanted to make sure I turned in my write-up and what I did have even if it still got me a 0 I would at least have felt better than turning in nothing, even though I have worked many hours on this project. When I started my write-up, I saved my file and had a computer update. The next day when I started word and went to open my write-up, it had said there was some issue and the file was not available. I looked everywhere on my computer and cannot find it anywhere. I still however decided to start rewriting my write-up, because I had over 3 pages of material on my program, how I was writing it, the testing I was doing, and exactly where I was in the program. I only write this out of self-frustration for specific circumstances I cannot control, however I still wanted my effort to be seen, even if I did not have a working program. Due to finding out my write-up was missing (even though the link to the file where it originally saved was still visible in the recent docs on word) it really forced me to not have last minute time to try more things on my program. My program will not bag items, but will create the bag objects requested in the input file, and will create objects of items from the input in the file.

I started from the ground up on this project and started with a parser that would make an array list of each line, from that array list I could start parsing each line and making objects of the items in the list that need to be bagged. The first two lines of the file allowed me to make my bag objects. Knowing lines 3 to n was items that would need to be bagged it was easy to grab each line. When creating

my objects I start by saving the entire line, this way I can go back and add my constraints later. The reason I cannot do it when my object is created is due to the fact that we have not created future items that there can be conflicts with.

To keep track of what items can go with other items, I decide to do what was discussed in class and make the object have a bit set. Originally no bits will be set, after the objects are made we will go through any constraints the item may have and set the bit that matches the constrained items mask. If an item can only go with 1 item we just set that one bit, if it can go with any we set all. I never made a constraint bit set for the bags, maybe that is why I was unsuccessful in getting an MRV search. I am still confused on if two different variables were to be considered or just 1 and how exactly to save those. My initial thought was to save them in order of cardinality but I cannot seem to get a proper comparison of 2 cardinality numbers, even without any errors on eclipse.

How I tested my program: I put a lot of effort into ensuring I had the correct setup on my way to my search program. I wanted to make sure that I felt very confident in my program doing exactly what I wanted. I started testing in each chunk of the process, beginning with the file parser, printing out each line to ensure we were getting them all but nothing out of bounds. Print statements are my favorite for ensuring I am getting to places in my code, and then making test files that I can test and see how they do. Since I did not get a bagger working I did not attempt anything big. I did test to make sure I would catch for mess-ups in a file, and also if it would catch a 1 item bag and return the correct output. So for every 1 item case I will succeed at least, or if there is 1 item but the item is too big. Testing my program for my bit set was a little confusing to start, because it will print the number of the set bit, and not a visual representation of 0's and 1's

like my brain would have loved. Due to time constraints for me I did not take any time to write a toString, because unless I can find something that easily converts it to a visual binary I would have to make it for infinite bits since the number of items I could have are in the 10 thousand.

My program is successful in the following situations; if the program only has 1 item, when the file input has some sort of error like no weight for an item (except for if a char is where there should be an int), if the item is too heavy for your bag set it will not waste your time and just end when that item is made.

I did enjoy writing this program, I was disappointed in myself when I hit the wall I did. I know that I started 2 weeks ahead of the due date, and spent a lot of nights sitting in front of my computer working on this trying to just move forward. Which I did great at until the search algorithm where I failed horribly. I did work alone, that could be a serious contribution to not completing my work on time as the time it took me to test some parts was more tedious. Specifically the object making, and ensuring that my items were being correctly made took time because I had a small bug that I made in my code and after I found out why it was not correctly making the objects things went a lot more smooth.