John Kucera

Prof. Suzanna Schmeelk

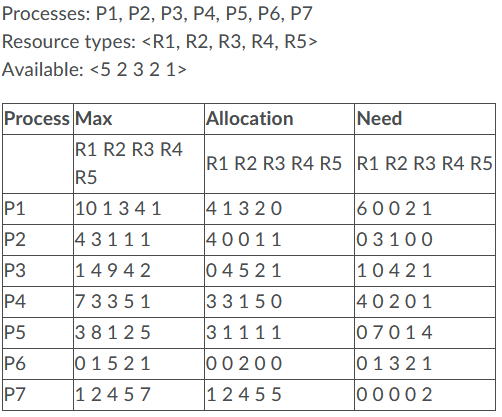
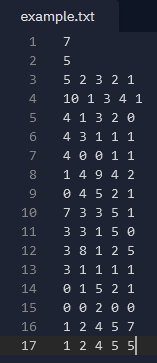
CMSC 412 Operating Systems

6 February 2020

Week 4: Homework 4 with Execution Example from my Week 4 Discussion

**Banker’s Algorithm: BACKTRACKING Approach (Java)**

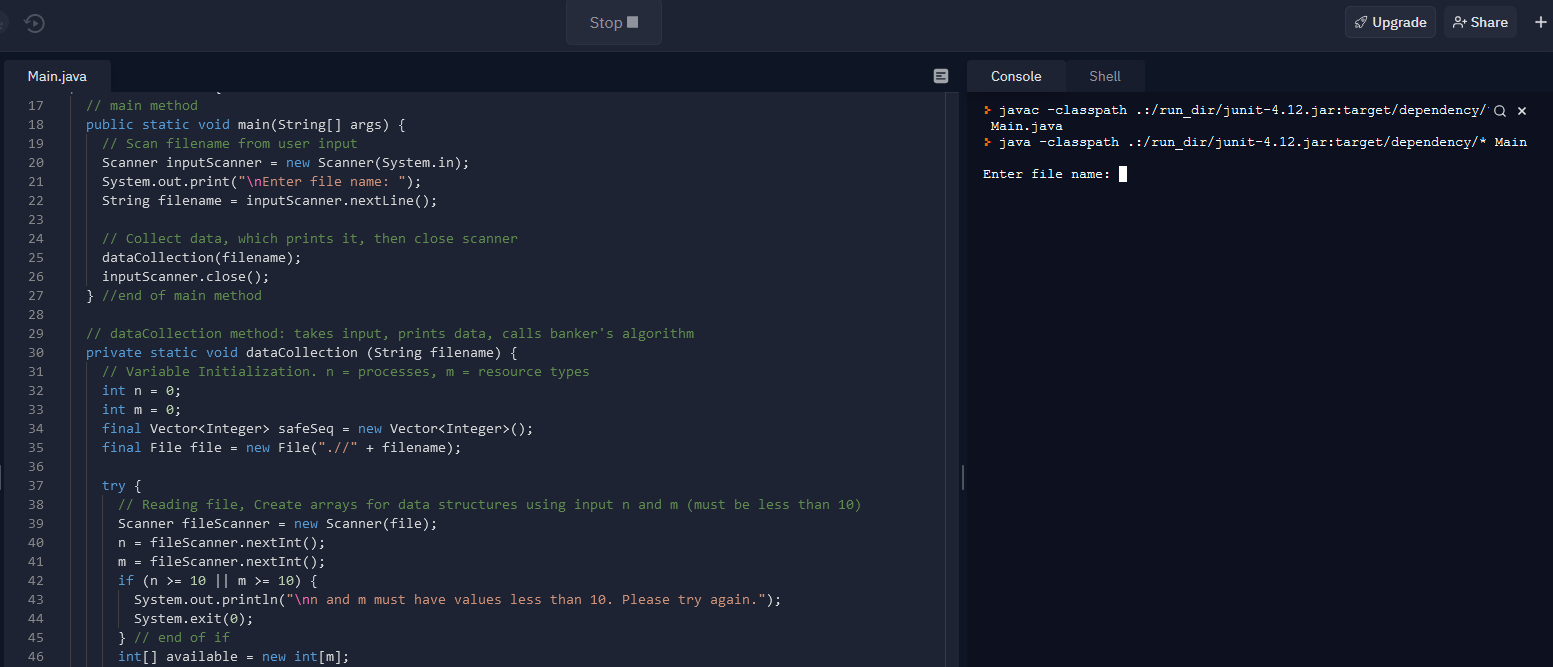
**Solution Description**: In my Java program, I used conditional statements and recursion (for a backtracking approach) to find all solutions to a given set of N processes and M resource types. When the program is run, I have a scanner that waits for user input to type in the name of the file. If the input does not match any file, IOException is raised and the program ends. Additionally, if N or M are 10 or greater, the user is notified of this and the program ends.

The text file is to be formatted as integers with spaces in between, which must be in order of: n, m, available values, and max + allocation values for each process. Here is a comparison of my text file (on the right) to the table I made in Week 4 Discussion (on the left):

The dataCollection method takes these values and sorts them into arrays, while printing them. It then uses the max and allocation values to calculate need values, then sorting those into arrays as well and printing them. A Java Vector of type integer is created to hold the safe sequences. Then, bankersAlgorithm method is called and all of this data is used as the parameters for this method.

My bankersAlgorithm method starts by determining, for all processes n, if the current process has not yet been executed and if it CAN be executed with the available resources + allocation. canBeExecuted method evaluates the latter by comparing available to need arrays. Then, the process is appended to the safe sequence Vector and available resources is added to the allocated resources since they have been freed up. bankersAlgorithm is then called again (inside itself, being recursive) to find all possibilities. As this goes on, the sequences are continuously printed, the Vector gets cleared so it can be used to store other safe sequences, and by the end it prints all solutions.

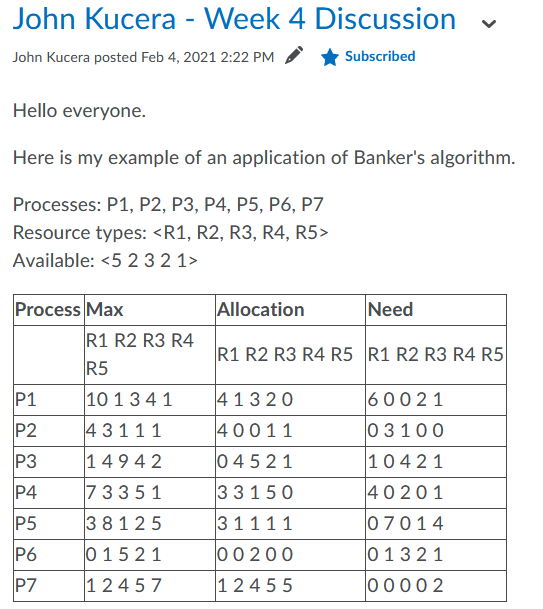
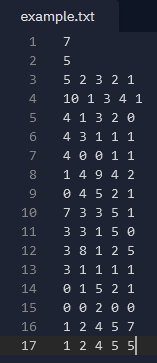
**Main.java running in repl.it:**



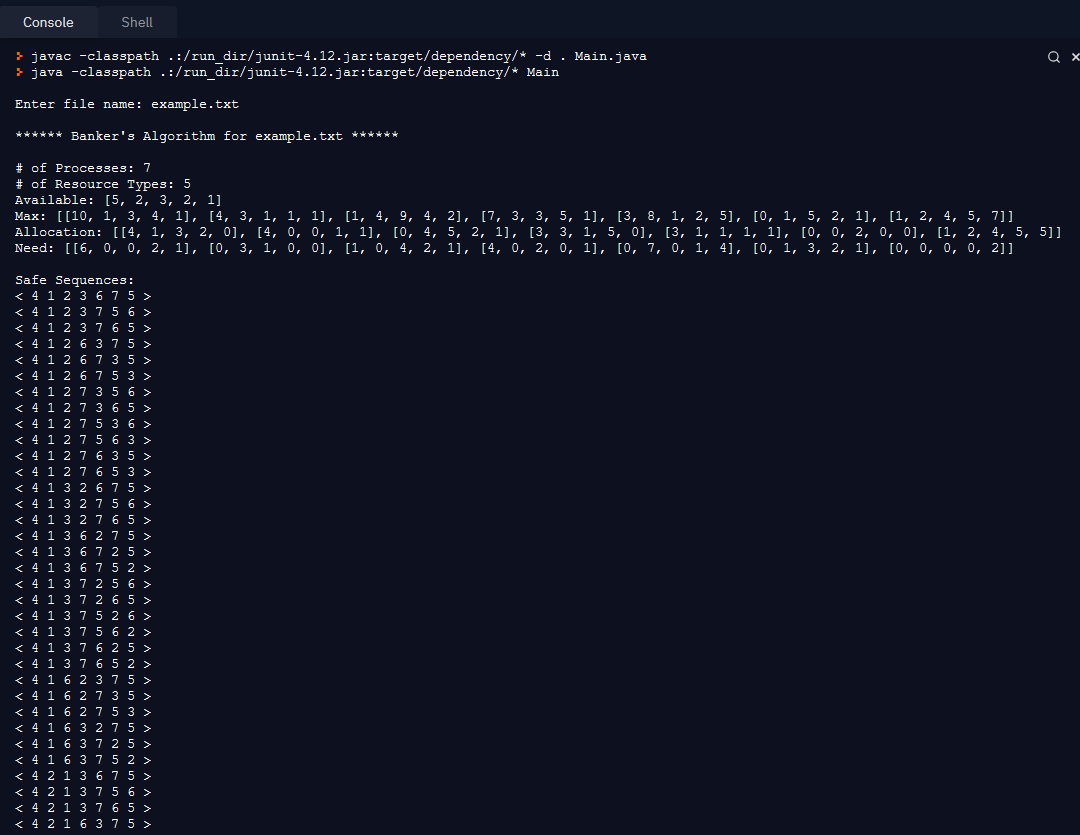
**Test Runs for error handling**

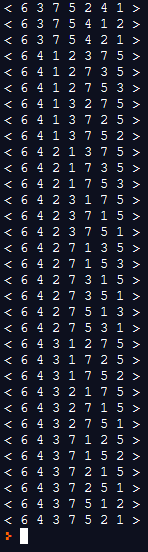
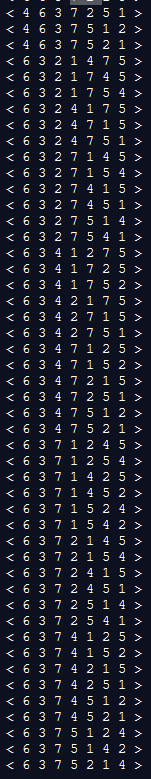
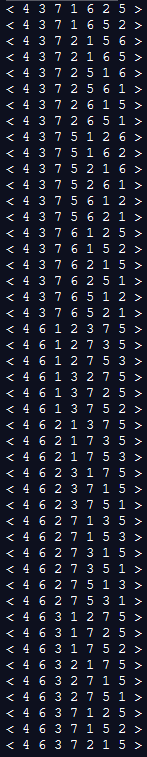
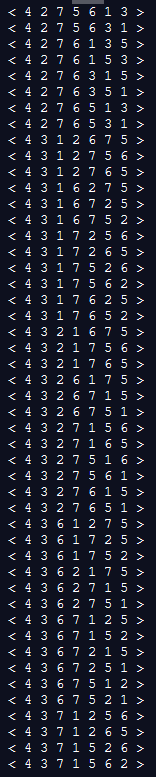
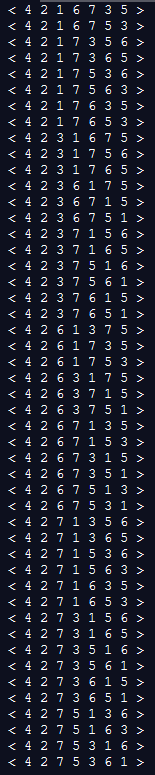
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aspect Tested** | **Input** | **Expected Output** | **Actual Output** | **Pass?** |
| Input does not match any file | no | Print “File not found. Please try again.” and end program |  | **YES** |
| n cannot be greater than 10 | n = 11: | Print “n and m must have values less than 10. Please try again.” and end program |  | **YES** |
| m cannot be greater than 10 | m = 20: | Print “n and m must have values less than 10. Please try again.” and end program |  | **YES** |

**Successful Test Run with my Week 4 Discussion example**:



**OUTPUT**:





**Result: PASS**

**List of safe sequences in plain text**:

< 4 1 2 3 6 7 5 >

< 4 1 2 3 7 5 6 >

< 4 1 2 3 7 6 5 >

< 4 1 2 6 3 7 5 >

< 4 1 2 6 7 3 5 >

< 4 1 2 6 7 5 3 >

< 4 1 2 7 3 5 6 >

< 4 1 2 7 3 6 5 >

< 4 1 2 7 5 3 6 >

< 4 1 2 7 5 6 3 >

< 4 1 2 7 6 3 5 >

< 4 1 2 7 6 5 3 >

< 4 1 3 2 6 7 5 >

< 4 1 3 2 7 5 6 >

< 4 1 3 2 7 6 5 >

< 4 1 3 6 2 7 5 >

< 4 1 3 6 7 2 5 >

< 4 1 3 6 7 5 2 >

< 4 1 3 7 2 5 6 >

< 4 1 3 7 2 6 5 >

< 4 1 3 7 5 2 6 >

< 4 1 3 7 5 6 2 >

< 4 1 3 7 6 2 5 >

< 4 1 3 7 6 5 2 >

< 4 1 6 2 3 7 5 >

< 4 1 6 2 7 3 5 >

< 4 1 6 2 7 5 3 >

< 4 1 6 3 2 7 5 >

< 4 1 6 3 7 2 5 >

< 4 1 6 3 7 5 2 >

< 4 2 1 3 6 7 5 >

< 4 2 1 3 7 5 6 >

< 4 2 1 3 7 6 5 >

< 4 2 1 6 3 7 5 >

< 4 2 1 6 7 3 5 >

< 4 2 1 6 7 5 3 >

< 4 2 1 7 3 5 6 >

< 4 2 1 7 3 6 5 >

< 4 2 1 7 5 3 6 >

< 4 2 1 7 5 6 3 >

< 4 2 1 7 6 3 5 >

< 4 2 1 7 6 5 3 >

< 4 2 3 1 6 7 5 >

< 4 2 3 1 7 5 6 >

< 4 2 3 1 7 6 5 >

< 4 2 3 6 1 7 5 >

< 4 2 3 6 7 1 5 >

< 4 2 3 6 7 5 1 >

< 4 2 3 7 1 5 6 >

< 4 2 3 7 1 6 5 >

< 4 2 3 7 5 1 6 >

< 4 2 3 7 5 6 1 >

< 4 2 3 7 6 1 5 >

< 4 2 3 7 6 5 1 >

< 4 2 6 1 3 7 5 >

< 4 2 6 1 7 3 5 >

< 4 2 6 1 7 5 3 >

< 4 2 6 3 1 7 5 >

< 4 2 6 3 7 1 5 >

< 4 2 6 3 7 5 1 >

< 4 2 6 7 1 3 5 >

< 4 2 6 7 1 5 3 >

< 4 2 6 7 3 1 5 >

< 4 2 6 7 3 5 1 >

< 4 2 6 7 5 1 3 >

< 4 2 6 7 5 3 1 >

< 4 2 7 1 3 5 6 >

< 4 2 7 1 3 6 5 >

< 4 2 7 1 5 3 6 >

< 4 2 7 1 5 6 3 >

< 4 2 7 1 6 3 5 >

< 4 2 7 1 6 5 3 >

< 4 2 7 3 1 5 6 >

< 4 2 7 3 1 6 5 >

< 4 2 7 3 5 1 6 >

< 4 2 7 3 5 6 1 >

< 4 2 7 3 6 1 5 >

< 4 2 7 3 6 5 1 >

< 4 2 7 5 1 3 6 >

< 4 2 7 5 1 6 3 >

< 4 2 7 5 3 1 6 >

< 4 2 7 5 3 6 1 >

< 4 2 7 5 6 1 3 >

< 4 2 7 5 6 3 1 >

< 4 2 7 6 1 3 5 >

< 4 2 7 6 1 5 3 >

< 4 2 7 6 3 1 5 >

< 4 2 7 6 3 5 1 >

< 4 2 7 6 5 1 3 >

< 4 2 7 6 5 3 1 >

< 4 3 1 2 6 7 5 >

< 4 3 1 2 7 5 6 >

< 4 3 1 2 7 6 5 >

< 4 3 1 6 2 7 5 >

< 4 3 1 6 7 2 5 >

< 4 3 1 6 7 5 2 >

< 4 3 1 7 2 5 6 >

< 4 3 1 7 2 6 5 >

< 4 3 1 7 5 2 6 >

< 4 3 1 7 5 6 2 >

< 4 3 1 7 6 2 5 >

< 4 3 1 7 6 5 2 >

< 4 3 2 1 6 7 5 >

< 4 3 2 1 7 5 6 >

< 4 3 2 1 7 6 5 >

< 4 3 2 6 1 7 5 >

< 4 3 2 6 7 1 5 >

< 4 3 2 6 7 5 1 >

< 4 3 2 7 1 5 6 >

< 4 3 2 7 1 6 5 >

< 4 3 2 7 5 1 6 >

< 4 3 2 7 5 6 1 >

< 4 3 2 7 6 1 5 >

< 4 3 2 7 6 5 1 >

< 4 3 6 1 2 7 5 >

< 4 3 6 1 7 2 5 >

< 4 3 6 1 7 5 2 >

< 4 3 6 2 1 7 5 >

< 4 3 6 2 7 1 5 >

< 4 3 6 2 7 5 1 >

< 4 3 6 7 1 2 5 >

< 4 3 6 7 1 5 2 >

< 4 3 6 7 2 1 5 >

< 4 3 6 7 2 5 1 >

< 4 3 6 7 5 1 2 >

< 4 3 6 7 5 2 1 >

< 4 3 7 1 2 5 6 >

< 4 3 7 1 2 6 5 >

< 4 3 7 1 5 2 6 >

< 4 3 7 1 5 6 2 >

< 4 3 7 1 6 2 5 >

< 4 3 7 1 6 5 2 >

< 4 3 7 2 1 5 6 >

< 4 3 7 2 1 6 5 >

< 4 3 7 2 5 1 6 >

< 4 3 7 2 5 6 1 >

< 4 3 7 2 6 1 5 >

< 4 3 7 2 6 5 1 >

< 4 3 7 5 1 2 6 >

< 4 3 7 5 1 6 2 >

< 4 3 7 5 2 1 6 >

< 4 3 7 5 2 6 1 >

< 4 3 7 5 6 1 2 >

< 4 3 7 5 6 2 1 >

< 4 3 7 6 1 2 5 >

< 4 3 7 6 1 5 2 >

< 4 3 7 6 2 1 5 >

< 4 3 7 6 2 5 1 >

< 4 3 7 6 5 1 2 >

< 4 3 7 6 5 2 1 >

< 4 6 1 2 3 7 5 >

< 4 6 1 2 7 3 5 >

< 4 6 1 2 7 5 3 >

< 4 6 1 3 2 7 5 >

< 4 6 1 3 7 2 5 >

< 4 6 1 3 7 5 2 >

< 4 6 2 1 3 7 5 >

< 4 6 2 1 7 3 5 >

< 4 6 2 1 7 5 3 >

< 4 6 2 3 1 7 5 >

< 4 6 2 3 7 1 5 >

< 4 6 2 3 7 5 1 >

< 4 6 2 7 1 3 5 >

< 4 6 2 7 1 5 3 >

< 4 6 2 7 3 1 5 >

< 4 6 2 7 3 5 1 >

< 4 6 2 7 5 1 3 >

< 4 6 2 7 5 3 1 >

< 4 6 3 1 2 7 5 >

< 4 6 3 1 7 2 5 >

< 4 6 3 1 7 5 2 >

< 4 6 3 2 1 7 5 >

< 4 6 3 2 7 1 5 >

< 4 6 3 2 7 5 1 >

< 4 6 3 7 1 2 5 >

< 4 6 3 7 1 5 2 >

< 4 6 3 7 2 1 5 >

< 4 6 3 7 2 5 1 >

< 4 6 3 7 5 1 2 >

< 4 6 3 7 5 2 1 >

< 6 3 2 1 4 7 5 >

< 6 3 2 1 7 4 5 >

< 6 3 2 1 7 5 4 >

< 6 3 2 4 1 7 5 >

< 6 3 2 4 7 1 5 >

< 6 3 2 4 7 5 1 >

< 6 3 2 7 1 4 5 >

< 6 3 2 7 1 5 4 >

< 6 3 2 7 4 1 5 >

< 6 3 2 7 4 5 1 >

< 6 3 2 7 5 1 4 >

< 6 3 2 7 5 4 1 >

< 6 3 4 1 2 7 5 >

< 6 3 4 1 7 2 5 >

< 6 3 4 1 7 5 2 >

< 6 3 4 2 1 7 5 >

< 6 3 4 2 7 1 5 >

< 6 3 4 2 7 5 1 >

< 6 3 4 7 1 2 5 >

< 6 3 4 7 1 5 2 >

< 6 3 4 7 2 1 5 >

< 6 3 4 7 2 5 1 >

< 6 3 4 7 5 1 2 >

< 6 3 4 7 5 2 1 >

< 6 3 7 1 2 4 5 >

< 6 3 7 1 2 5 4 >

< 6 3 7 1 4 2 5 >

< 6 3 7 1 4 5 2 >

< 6 3 7 1 5 2 4 >

< 6 3 7 1 5 4 2 >

< 6 3 7 2 1 4 5 >

< 6 3 7 2 1 5 4 >

< 6 3 7 2 4 1 5 >

< 6 3 7 2 4 5 1 >

< 6 3 7 2 5 1 4 >

< 6 3 7 2 5 4 1 >

< 6 3 7 4 1 2 5 >

< 6 3 7 4 1 5 2 >

< 6 3 7 4 2 1 5 >

< 6 3 7 4 2 5 1 >

< 6 3 7 4 5 1 2 >

< 6 3 7 4 5 2 1 >

< 6 3 7 5 1 2 4 >

< 6 3 7 5 1 4 2 >

< 6 3 7 5 2 1 4 >

< 6 3 7 5 2 4 1 >

< 6 3 7 5 4 1 2 >

< 6 3 7 5 4 2 1 >

< 6 4 1 2 3 7 5 >

< 6 4 1 2 7 3 5 >

< 6 4 1 2 7 5 3 >

< 6 4 1 3 2 7 5 >

< 6 4 1 3 7 2 5 >

< 6 4 1 3 7 5 2 >

< 6 4 2 1 3 7 5 >

< 6 4 2 1 7 3 5 >

< 6 4 2 1 7 5 3 >

< 6 4 2 3 1 7 5 >

< 6 4 2 3 7 1 5 >

< 6 4 2 3 7 5 1 >

< 6 4 2 7 1 3 5 >

< 6 4 2 7 1 5 3 >

< 6 4 2 7 3 1 5 >

< 6 4 2 7 3 5 1 >

< 6 4 2 7 5 1 3 >

< 6 4 2 7 5 3 1 >

< 6 4 3 1 2 7 5 >

< 6 4 3 1 7 2 5 >

< 6 4 3 1 7 5 2 >

< 6 4 3 2 1 7 5 >

< 6 4 3 2 7 1 5 >

< 6 4 3 2 7 5 1 >

< 6 4 3 7 1 2 5 >

< 6 4 3 7 1 5 2 >

< 6 4 3 7 2 1 5 >

< 6 4 3 7 2 5 1 >

< 6 4 3 7 5 1 2 >

< 6 4 3 7 5 2 1 >