**GA Phase 1 Requirements**

​

Each student will be randomly assigned of the following 10 problems (from the KARP21, but renumbered on the left for the purpose of making the random assignment easier):

|  |  |  |
| --- | --- | --- |
| 0 | Clique | Karp#3 |
| 1 | Max cut | Karp#21 |
| 2 | Partition | Karp#20 |
| 3 | Satisfiability with at most 3 literals per clause | Karp#11 |
| 4 | Hitting Set | Karp#15 |
| 5 | Chromatic number | Karp#12 |
| 6 | 0-1 integer programming | Karp#2 |
| 7 | Directed Hamilton circuit | Karp#9 |
| 8 | Undirected Hamilton circuit | Karp#10 |
| 9 | Knapsack | Karp#18 |

All three phases of the semester project will involve your assigned problem. Make sure you work ONLY on the problem assigned to you. Submitting a phase based on a different problem will NOT be given credit.

**Phase 1 (P1):** Solve your assigned problem using an exhaustive search (“brute force”) method. First, decide the data structure (type and size) necessary to store a proposed solution. Second, determine what type of mathematical or data structure the problem and/or its solution resembles. It will be this structure that you will be generating (“populating”) an exponential number of times in order to find the correct solution. Third, construct a verification function that tests for any proposed solution whether or not it is a correct solution to the problem with its given input data. Fourth, how can we evaluate proposed solutions that are very good even if not perfect? Those solutions are worthwhile exploring as well. To satisfy this goal, you will need to describe a fitness function in a text document with pseudo-code or English and not actual code necessary, but must make reference to the actual structure name(s) in your code. A **fitness function** can evaluate a proposed solution and measure it with a score for the purposes of comparison between different proposed solutions. The reason it is stated this way is that one does not always know ahead of time what score value the perfect solution has. Later in the semester we will officially define fitness and other such related functions. Fifth, write your code so that it generates all possibilities of the above structure and test each proposed solution for correctness to the given problem. Sixth, select/find three sets of input data (described below) to test your code with each set of data (either found on web or created on your own) for your problem and find the solution to the problem given that set of data. This must compile and run from the command line using the jdk javac and java commands.

Comment your code appropriately and use meaningful names for classes, methods, variables and constants. You are expected to report on the number of lines of code using cloc found at <https://github.com/AlDanial/cloc/releases/tag/v1.92> This command must be run on the command line (cmd.exe). When you collect all .java files into the same temporary directory (see Phase1\_Req1 below), then the following command will report on the contents of each .java file. This cloc report is part of the project submission. The command to obtain the report data is as follows (assuming you are using the above version):

cloc-1.92.exe --by-file \*.java

The output of your code should include name of the problem assigned with its KARP#, the input data, the correct solution, and the number of solutions you had to generate in order to find this solution. Do this for each of the three input sets and all of this should be copied into a text (.txt) file which you will upload to **GA.PH1.Task#5** (below).

The project should provide the user of your program the option to choose (ie **provide at command line**) what names for the input (containing the necessary data for your assigned problem) and output files (text file which the program will output the brute-force solutions to your data sets for your given problem) of the project execution. You will accomplish this by the use of command line flags. See a separate file discussing this. If the user does not provide a name for the input and/or output files then the project should use the default names of input.txt and output.txt respectively.

The program will be run three times, each with a different sized data set (small, medium, large). Let **problemSize** be the size of your problem, then the sizes for small, medium and large as follows:

10 <= **problemSize** // Small

16 <= **problemSize** // Medium

24 <= **problemSize** // Large

and that Small <= Medium <= Large.

The outputFile should contain within the name of your NP problem, the input data to the problem that you tested and the solution to your problem, one for each of three problem sizes.

Submitting will be done by uploading the necessary files to Blackboard in the correct part requesting the specific files:

**GA.P1.Task#1:** Compress by (g)zip or jar the source code Java files ONLY without the .o files or software tools packaging files, nor their directories. Collect together all the .java files into a temporary directory and then compress this one directory, which you will then Upload to the assignment **GA.P1.Task#1**.

**GA.P1.Task#2:** Upload a Readme.txt explaining how to compile and run your program from command line to the Blackboard system for **GA.P1.Task#2**.

**GA.P1.Task#3:** Upload the input files necessary to test your system to the Blackboard system for **GA.P1.Task#3**. You should write a InputFormat.txt file that explains in detail exactly how to interpret the input files. Compress these files in a manner similar to that explained for GA.P1.Task#1.

**GA.P1.Task#4:** Upload the output files that your system produced corresponding to the input files that you uploaded to GA.P1.Task#3. These will be used to test your system with your input files in addition to any of my own input. Upload these files to the Blackboard system for **GA.P1.Task#4.**

**GA.P1.Task#5:** Upload the report generated by cloc program (mentioned above) which will report on the number of actual java code lines, blank lines, comment lines for each of the .java files that is part of your project. Upload this report to the Blackboard system for to **GA.P1.Task#5.**

**GA.P1.Task#6:** Describe the Problem you were assigned with the KARP# in a text document. Upload this description to the Blackboard system for **GA.P1.Task#6.**

**GA.P1.Task#7:** Provide the Pseudo-Code that described the Fitness function related to your problem in a text document. Upload this document to the Blackboard system for **GA.P1.Task#7.**

**GA.P1.Task#8:** Describe the mathematical structure that is most closely related to your problem in a text document. Upload this document to the Blackboard system for **GA.P1.Task#8.**

**GA.P1.Task#9:** In order to find the correct solution, the project code had to have a loop and/or nesting of loops that generate all possible proposed solutions. Identify the lines of code in your project that accomplished this and now upload the requested code to the Blackboard system for **GA.P1.Task#9.** (If there are multiple files, then prepare that code in a manner described in GA.P1.Task#1 beforeuploading to **GA.P1.Task#9**.)

**GA.P1.Task#10.** In order to verify that a proposed solution is the perfect solution, the project code had to have a method that can validate the solution. Identify the lines of code in your project that accomplished this and now upload it to **GA.P1.Task#10.** (If there are multiple files, then prepare that code in a manner described in GA.P1.Task#1 beforeuploading to **GA.P1.Task#10**.)

**See Syllabus for the exact Phase1 Submitting Deadline. This one deadline applies to all of the above tasks.**