CSC 535/635 – HW 4: Solving the Jump-It Game Problem with a Genetic Algorithm  
Due Date: **Tuesday, May 1, 2018 (beginning of class)**

**Assignment Description:**

The game of Jump-It consists of a single player and a board of integers, were each integer represents the cost to move to that integer’s position. The game boards themselves consist of a set of *n* integers where the first element in the board is always 0 and the rest of the integers are always positive. A sample game board, where *n*=6, is given below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 3 | 80 | 6 | 57 | 10 |

Fig. 1: A sample Game Board.

The main objective of the game is to move the player character from the first cell to the last cell with the lowest total cost. The player character will always start the game in the first cell, with a total cost of 0, and will have two types of moves available: The player can either move to the adjacent cell or jump over the adjacent cell to land on the next cell. The total cost of a game of Jump-It is the sum of the costs of the visited cells.

In the board shown above, there are several ways to get to the end. Starting in the first cell, our cost so far is 0. We could jump to 80, then jump to 57, then move to 10 for a total of: . This path is shown in Fig. 2. However, a cheaper path would be to move to 3, jump to 6, then jump to 10, for a total cost of: as shown in Fig. 3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 3 | 80 | 6 | 57 | 10 |

Fig. 2: More expensive path

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 3 | 80 | 6 | 57 | 10 |

Fig. 3: Less expensive Path

**For this homework assignment you will implement a genetic algorithm (GA) solution for the Jump-It game problem. Follow the detailed algorithm given on slides 24 – 27 of GA lecture slides. Except, for Step 3 of the algorithm where you must use only one iteration, where the next generation is generated from the current generation by replacing the two least fit individuals of the current population with the newly generated children. Your solution must be compared with the Dynamic Programming solution provided and must achieve an accuracy of at least 80%. In addition to the programming portion of the assignment you must write a report in the report template format (found on trace).**

**Program Requirements:**

The output from your program **must** display the following items for **each** game solution:

* The game board
* The minimum cost if the board was solved with DP
* The optimal path chosen by DP.
* The minimum cost if the board was solved with GA
* The optimal path chosen by GA.

After running your solution on all game boards in the input file, your code **must** then output the following:

* The accuracy of your GA implementation. (This must be found by comparing the output from the DP program to your GA).

The **console** output from your code **must** follow the format provided in the following samples:

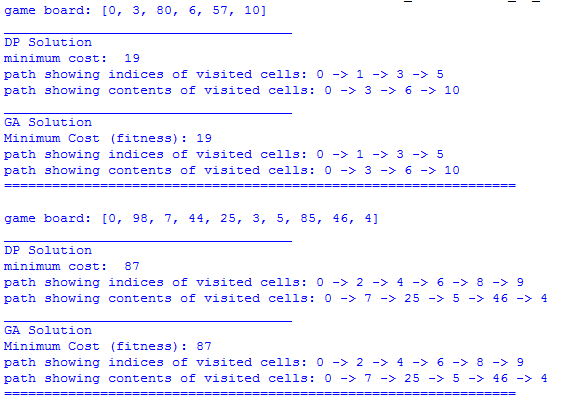
****

Fig. 4: Sample format 1

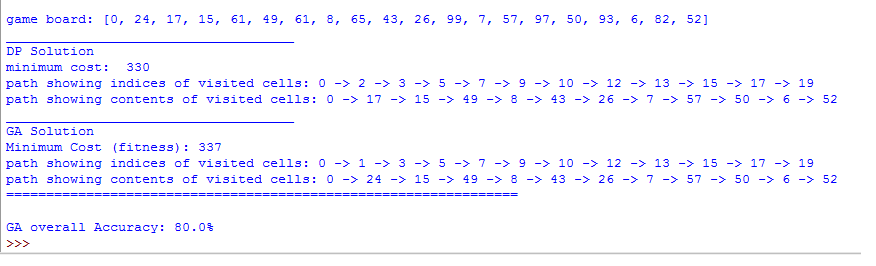


Fig. 5: End of the program output format.

**Report Requirements:**

* Your report **must** follow the template report provided on trace. The sections in the template are provided for a reason, and we are only looking for couple paragraphs per section, which should result in a 2+ page report. Feel free to write more if you want to (thorough reports will receive bonus points). If you have any questions about the report fell free to email or speak to the TA.
* **In your report you must address the following items:**
  + What representation did you choose for a candidate solution? What are its Phenotype and Genotype?
  + How do you generate a genome?
  + How do you measure the fitness of a particular genome?
  + What method did you choose for parent selection and why?
  + What method did you use for crossover/mutation and why?
  + How does the accuracy of the GA compare to Dynamic Programming after multiple runs?
  + What stopping criteria did you use and why?

**Remarks:**

* You may encode a board with 1s and 0s, where a 1 corresponds to a visited cell and a 0 corresponds to an unvisited cell. These can be used as the “genomes” in your GA.
* In your GA solution, you can strip out the first cell on the board since 0+*I*=*I*.
* Students are advised to follow the **Roulette Wheel** method for selecting parents as shown in the slides. However, you can change this for extra credit (see below).
* A very common bug that we will be checking for occurs when recombining parents at a crossover point, p1, if one parent has a 0 digit before p1 and the other parent has a 0 digit after p1, then a generated child would have two consecutive 0s. This would result in an invalid genome (corresponding to a board with a move of a jump over two adjacent cells, which is not allowed in the Jump-It game). You need to prevent this from happening in your code. First, you can try different crossover points. However, if no valid crossover point can be found for two parents, then you may either clone one of the two parents or select a different parent and redo recombination.
* Two sample input files are available on the course website to help you test your code. One is small which is great for helping you develop your code. The other is much larger to provide a good test sample. Be aware that we will use different files to test your code.

**Submission instructions:**

* For this assignment you may either work alone or in groups of 2. Graduate and undergraduate students may **not** work together in the same group (e.g. form groups of only undergraduates or graduates.).
* You are allowed to use Python, IPython, R, Java or C++ for your implementation. Please provide sufficient instructions in your programs doc-string about how to run your code.
* Upload your files including a copy of your report to your CSC 535-635 TRACE folder in a subfolder named **HW4.** Only one submission per group is needed. Please turn in a **stapled** hard copy of your report collection at the beginning of class on the due date.
* Name your program **hw4.extension**. Replace **extension** by the appropriate file extension based on your choice of programming language
* Name the input file containing the dataset: **input.txt**
* **What to turn in?** Upload to TRACE copies of your source code and report (preferably) in MS word. Please make sure to follow the format of the template report provided on TRACE. **Please make sure that your code is well organized and properly documented and commented.** Turn in hard copies of your report and code at the beginning of class on the due date. Please remember to include the name of your trace folder with your code. Please turn in only one report per group.

**Extra Credit [up to 25 points]:**

For extra credit, students may choose to alter their basic algorithm from one of the following options:

1. Find your own way to significantly improve the performance of the GA discussed in class.
2. Implement and test different parent selection methods (i.e. Boltzmann method, tournament method, sigma scaling selection).
3. Implement and test different crossover methods (i.e. Real-valued difference or other arithmetic methods, single point/multi point/uniform crossover, simplex 3-parent crossover).

To earn any extra credit points your submission **must** meet the following three requirements:

1. You write about the item done for extra credit in a separate section titled: **Extra Credit** after the conclusion. In this section you must compare your extra credit implementation and your basic implementation and show the results in a table. Offer some good explanation as to the differences seen in the comparison table.
2. Your extra credit implementation must be done in a separate file named: **hw4\_modified.extension.** Replace **extension** by the appropriate file extension based on your choice of programming language. Your code must run with the default input used for grading and be free of errors.
3. Your code must work and be free from errors or bugs.