**Assignment 2**

Name : Shivani Bimavarapu

UIN: 01025992

**Instructions on how to compile and execute each program:**

I have coded everything in python. You should be able to execute them on putty or any python editor. On putty, to execute the file, type “python filename.py”. for example “python ackley\_hillclimb.py” for task1.

**Problem 1 Task1: Hill climbing- N queens**

Output: The output will be automatically saved in the external file “newfile.txt”. The newfile.txt contains the 100 Local minima of Ackley function which results in the following chart. The Global minima in 100 runs is -20.

**Output of Problem 1 Task 1:**

The output of task 1 generated inside the text file is as follows:

The three columns below are the Local minima, x and y values, 100 each generated for first 100 runs:

-11.433857741252805 4.909480849681881 -0.7503589302968412

-10.957916473712856 -4.714204181791677 2.107183900259969

-10.981650312752397 -1.8949540212323752 -4.465881285479721

-13.134319163326406 4.068662794392913 -1.985142930316919

-13.104120861486525 -1.8820130781044488 3.958005366145308

-14.101227575463572 -3.200171901939552 -0.19253704155626672

-11.95489533672944 -3.8175925041865373 -3.0641684390993613

-13.618075601417067 2.9187255213372123 2.9975571316012686

-13.738598749150691 -3.100229798071108 -0.49389737421025065

-14.835013354366044 -0.18092545939410035 -2.4743292744107106

-11.938817416055644 -0.17511087387771218 -4.303835938512742

-12.880854596127712 3.8128940044282498 -1.9946865860039864

-11.025722589773814 2.911541366105263 4.846973954829302

-11.46304528181485 -3.4833659790922344 -2.9677439760216826

-11.142883963639136 -4.2622858960528935 1.5927088509633973

-9.144735891438863 -4.2330069305196645 4.731685899614556

-12.53210053024341 0.35597826590150083 -3.8184940959860363

-9.388468708548139 -4.093121905135403 -4.492293764855203

-12.739425795594977 2.0402905143628676 3.7949265393628444

-10.89687881596168 -4.533119578351098 -0.6189612914419342

-10.914195920366867 3.932762284462546 4.1676655812750605

-12.069734922110875 2.3177882262890224 -3.2811605529493635

-15.230971331984644 -0.5252001604019051 -2.2015065357821864

-10.25862464065046 4.5915974703218945 2.2909754602602717

-18.772242406605713 -0.9825220298207701 -1.0877387624349808

-11.294604760879668 3.7551456434454327 -2.4167125248564716

-13.737623798524313 1.7770934856790905 -3.0907856337059103

-10.827346450425624 -4.572496359289064 0.5631040499391915

-10.832319194432614 0.4115396151753039 -4.727325266988856

-12.762585123270695 2.7079796331808605 -2.9435560993374974

-15.359631717479807 -2.8763003605573236 0.9817059609933946

-14.754887163328068 -1.9888066822609507 2.9776609848449196

-10.543925511511684 -4.395915002698137 -2.9670114897916435

-9.125414780776277 -4.88006832099143 -4.763270451232017

-13.678447126973 1.8329531191738857 -2.50061715204866

-12.215398800891865 -1.855189891714969 3.5252453043745344

-15.243683978096094 1.5317514618362549 -1.438577759395266

-9.948497337670792 4.742652166733203 2.5345915403880843

-17.222636479145347 -0.7551677039887431 -1.2621027904692146

-11.875633980013646 -0.9683394373048924 -4.34792035162785

-11.542449276487826 -3.371049341207313 -2.4439568389047217

-11.236213302154503 -3.6579095968117916 -2.4268663082761823

-15.522247929336785 -2.006960204360478 -1.786161145528978

-12.284219539391149 2.894176785188739 -4.047853990083212

-18.336152536863025 -0.41443293970502637 -0.9354634799911024

-10.844719337756562 -4.8421900776194695 2.223036042536068

-10.949788666575243 1.7718724177509195 4.799940938007505

-11.9313437961973 -4.0737123955644 -1.659447267059352

-14.628165326675983 -2.8986084265290457 1.9756825697002096

-14.249856249846129 -0.38658714122801285 2.9002704554054137

-14.46392061167406 -2.7147159201062565 1.1281073673164528

-22.19912558762714 -0.07604695935981276 0.06221397689527741

-17.770185287985758 -1.129885847225458 -0.42180321177032987

-9.989613085775103 -2.4577118360667627 -4.727827843633056

-10.969348775048013 -4.377993063330696 -1.357759692269168

-10.445598599721782 -4.416423479959763 3.0238817032587413

-15.330683280674695 2.314115322295681 0.20954366168407645

-10.02190908183633 4.5698033213018725 -2.4316528750169377

-12.458330665552198 1.0017493967432065 -4.969198648965313

-16.47179148073025 0.28833962827819865 -1.7787008993934883

-15.187381706515227 1.7637917269331354 -2.0876834655412035

-11.6341937776937 -1.1915849754505108 5.004620154737167

-10.275904034569574 -4.6541194833481745 2.861681565481881

-12.342895106728214 -2.2045153921920346 3.3127799502503237

-12.847197171187878 -2.675601365144974 -2.2936871828279526

-13.34009161175745 -0.053641393230363654 -4.1401098411298385

-13.035183031538033 -3.8481560659262124 1.971724254686654

-13.943138021026385 -3.1599915099702227 -1.9608418506054468

-16.441412507210842 -0.9127230109344575 -1.6241734421823937

-14.027895608772678 -1.0387038947417457 -3.2414018997951772

-11.807213437876266 -0.5162121330292615 4.1717069808764355

-12.550648086987051 -3.487637608176616 0.634719799271318

-13.434675641766036 1.5698783907253224 2.608046314106762

-11.19865184451028 3.746472879491777 2.6591693233781446

-16.547135590535255 0.7730462412921905 -1.4402518430204319

-13.15839172865893 3.405813196204763 -0.18229393871034194

-13.048233026463937 -2.026217846780191 4.0782611191041696

-13.036795395945802 -3.6217707146259057 0.09446044575632584

-10.558887592455827 4.534286544155373 1.4844270528215588

-14.560310276613132 0.9010947659965272 2.5736955402631843

-13.246683599891787 4.027535700774462 1.94603065753258

-9.55996469577227 -3.323302394539853 4.4928209566023085

-10.370540827116786 -3.8441174010793366 3.5870972409235407

-13.758258981843701 2.033468849082588 -2.4518326083905335

-10.06452774038389 3.9337570611099633 4.31261307111618

-10.88882741298739 -1.8925545463300013 -4.535160397301665

-18.018927265412295 0.6019057587376785 -1.0185850001369063

-15.26744129168481 -0.0817766611571475 -2.8015517326430563

-11.605120802187164 3.8721959865745923 2.347002594182039

-12.920128435182427 -0.7467991501334896 -3.9477768852367436

-16.036126035576583 0.5210309056650484 1.9249509972992236

-18.792881840977383 -0.2741844873416799 0.7527862126750602

-12.842420351712267 4.117660359445024 1.9599620818817733

-17.05700840537898 -0.05000300778274307 -1.6710670592535817

-13.082471069568467 -1.1827086600030237 4.016125790058766

-19.931175987445613 -0.027328390743638728 0.8505781522161485

-14.55357606838463 1.0318431407160418 2.5861149151122165

-9.94227346718773 4.639596096655728 2.658586428829361

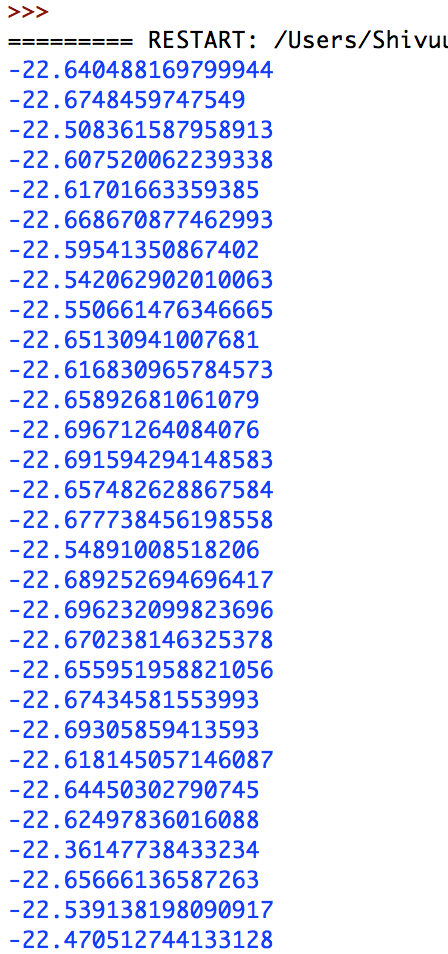
-11.510479487299984 -3.0398891699255817 3.6858712280814387

-10.563291786497883 -1.6986522432790891 4.615444195346088

**Problem 1 Task2: Differential evolution- N queens**

You can see the output on the python console. I did not save the output automatically to an external file in this case. I have plotted the output in the following chart. The minimum of first 100 runs can be seen some where at 22.72

**Output of Problem 1 task 2:**

****

This is just part of the output. It continues until 100 runs with 100 minima. README would be so long if I put all my output.

**Problem 1 Task 3: (Task 4 in the description) Analyzing the results obtained in the above two tasks:**

In the hill climbing approach, we can see that the global minima for 100 runs is ranging from -20 to -7.5

In differential evolution approach, all the global minima for 100 runs is ranging some where around -22 (example 1: -22.53913, example 2: -22.3614778)

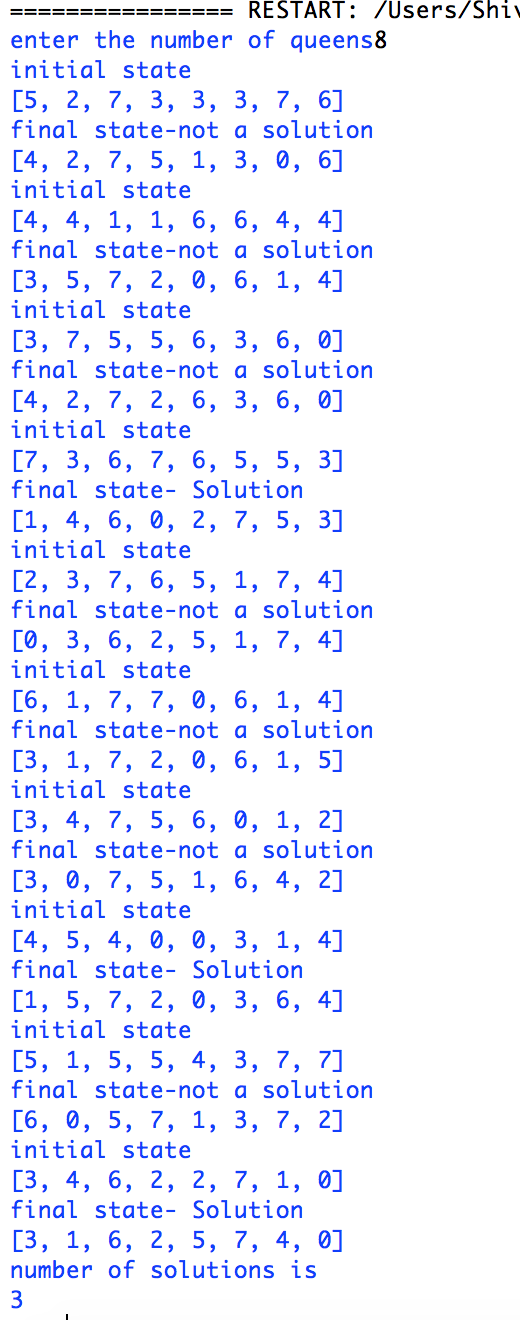
So, differential evolution is definitely preferable to find the global minima of the ackely function.

**Problem 2 Task 1: Outputs are as follows:**

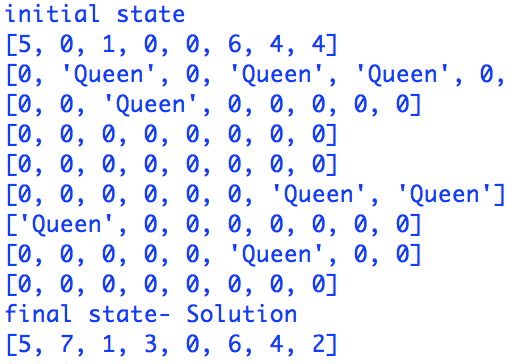
**N=8 (8 queens)**

I was not sure how to plot the initial state and final state in a graph since they are multiple row and column values for a chess board.

So, if I only print the lists containing row numbers of queens in the initial and final state of the chess board, my output for first 10 times looks as follows:

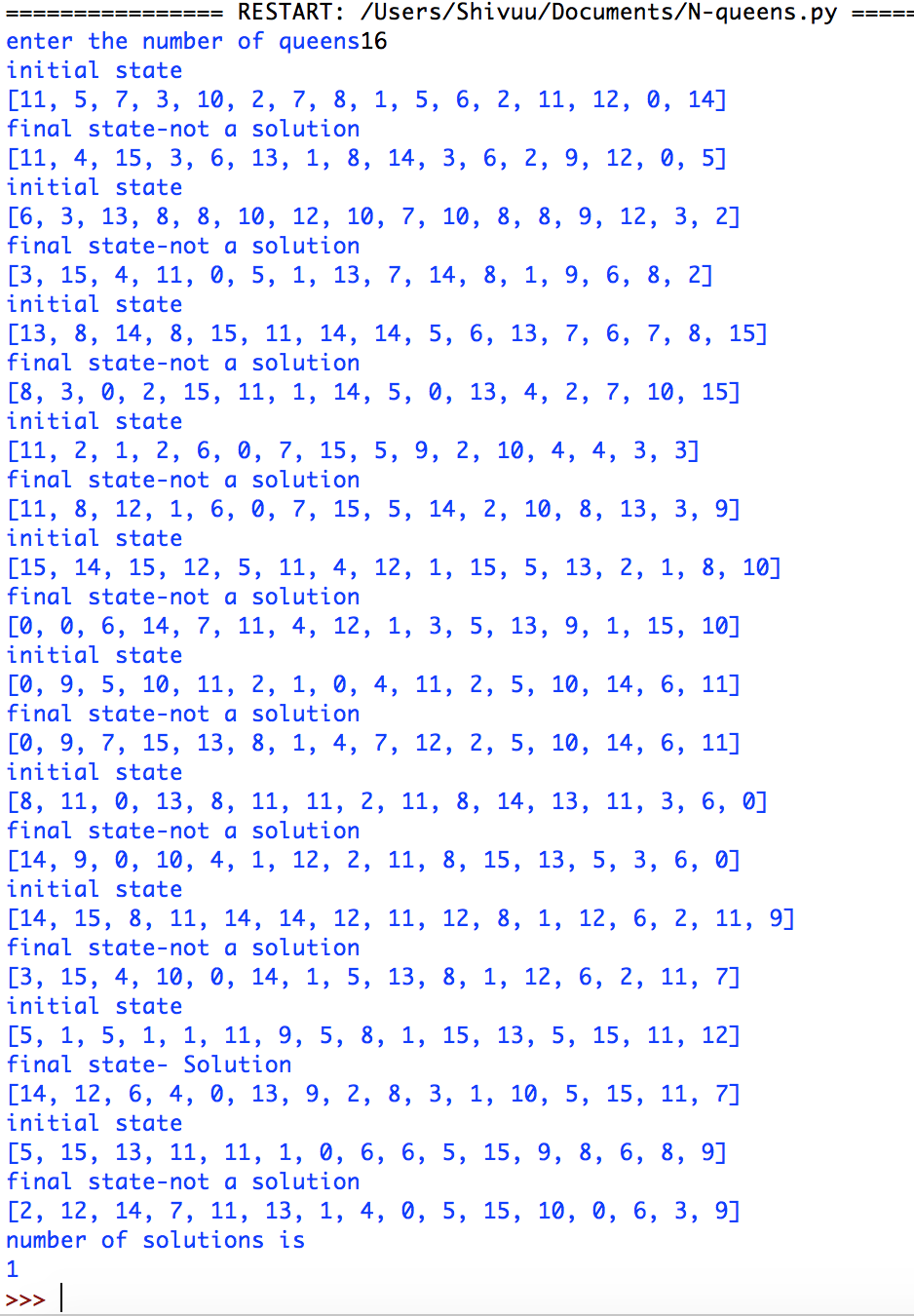


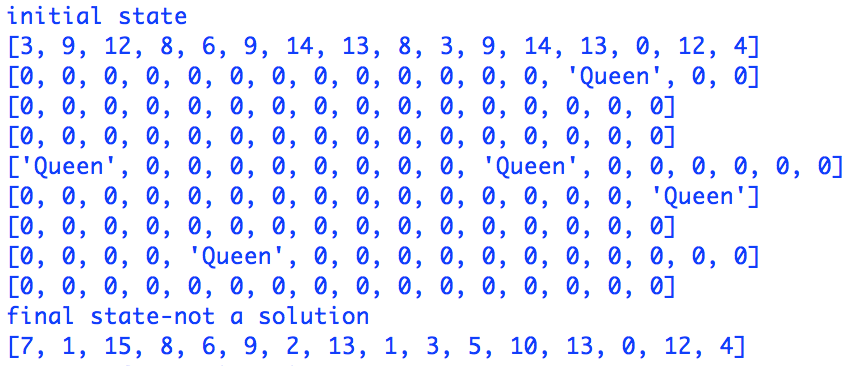
If I print the matrix kind of version of my output, for single run, it looks as follows:



**N=16 (16 queens)**

if I only print the lists containing row numbers of queens in the initial and final state of the chess board, my output for first 10 times looks as follows:

matrix kind of version of my output, for single run**:**

****

**Problem 2 Task 2: N-queens genetic algorithm:**

I have chosen population size as 100

**N=8 (8 queens)**

Output:

Enter number of queens8

[3, 1, 6, 2, 5, 7, 0, 4]

[3, 0, 4, 7, 5, 2, 6, 1]

[6, 2, 0, 5, 7, 4, 1, 3]

[3, 6, 0, 7, 4, 1, 5, 2]

[4, 7, 3, 0, 6, 1, 5, 2]

[4, 6, 1, 3, 7, 0, 2, 5]

[3, 6, 2, 7, 1, 4, 0, 5]

[5, 2, 6, 3, 0, 7, 1, 4]

[3, 7, 4, 2, 0, 6, 1, 5]

[5, 3, 6, 0, 7, 1, 4, 2]

[4, 6, 1, 5, 2, 0, 3, 7]

[6, 3, 1, 7, 5, 0, 2, 4]

[4, 1, 7, 0, 3, 6, 2, 5]

[2, 5, 7, 1, 3, 0, 6, 4]

[2, 0, 6, 4, 7, 1, 3, 5]

[5, 7, 1, 3, 0, 6, 4, 2]

[0, 6, 3, 5, 7, 1, 4, 2]

[3, 1, 6, 2, 5, 7, 4, 0]

[4, 1, 7, 0, 3, 6, 2, 5]

[4, 1, 7, 0, 3, 6, 2, 5]

[4, 0, 3, 5, 7, 1, 6, 2]

[4, 1, 7, 0, 3, 6, 2, 5]

[7, 1, 4, 2, 0, 6, 3, 5]

[3, 1, 6, 2, 5, 7, 4, 0]

[5, 2, 6, 1, 3, 7, 0, 4]

[3, 1, 6, 4, 0, 7, 5, 2]

[6, 3, 1, 4, 7, 0, 2, 5]

[3, 1, 7, 4, 6, 0, 2, 5]

[4, 6, 3, 0, 2, 7, 5, 1]

[3, 6, 0, 7, 4, 1, 5, 2]

[4, 7, 3, 0, 2, 5, 1, 6]

[2, 5, 7, 0, 4, 6, 1, 3]

[3, 0, 4, 7, 5, 2, 6, 1]

[7, 2, 0, 5, 1, 4, 6, 3]

[6, 4, 2, 0, 5, 7, 1, 3]

[3, 1, 7, 5, 0, 2, 4, 6]

[5, 2, 4, 6, 0, 3, 1, 7]

[5, 2, 4, 6, 0, 3, 1, 7]

[1, 5, 0, 6, 3, 7, 2, 4]

[3, 1, 6, 2, 5, 7, 4, 0]

[4, 2, 0, 6, 1, 7, 5, 3]

[3, 7, 0, 4, 6, 1, 5, 2]

[3, 6, 4, 2, 0, 5, 7, 1]

Total minimum fitness is : 0

Total number of solutions are : 43

>>>

**N=16 (16 queens)**

================== RESTART: /Users/Shivuu/Documents/gan.py ==================

Enter number of queens16

[9, 12, 3, 8, 2, 13, 1, 10, 15, 5, 0, 4, 14, 7, 11, 6]

[2, 13, 6, 14, 9, 5, 12, 0, 3, 7, 11, 1, 15, 10, 8, 4]

Total minimum fitness is : 0

Total number of solutions are : 2