1

Part I – Questions (20 points each unless noted otherwise)

1. (40 points) If you are pair programming in part II, use pair programming

on this question as well. Other questions in Part I must be done without

help from your pair programming partner.

Write a Python module newtonraphson.py that contains functions to find

the roots of an equation for a polynomial in x using the Newton-Raphson

method. Your solution should have the following signature:

def NewtonRaphson(fpoly, a, tolerance = .00001):

"""Given a set of polynomial coefficients fpoly

for a univariate polynomial function,

e.g. (3, 6, 0, -24) for 3x^3 + 6x^2 +0x^1 -24x^0,

find the real roots of the polynomial (if any)

using the Newton-Raphson method.

a is the initial estimate of the root and

starting state of the search

This is an iterative method that stops when the

change in estimators is less than tolerance.

"""

coefficients should be an iterable (tuple or list) of coefficients for the

powers of x. As an example:

7𝑥𝑥 4 + 3𝑥𝑥 3 − 5𝑥𝑥 2 + 32𝑥𝑥− 7𝑥𝑥 0 = 0

would be represented as [7, 3, −5, 32, −7]. Show the answer s for the

equation above given a starting points of x=5 and x = -50:

NewtonRaphson( [7, 3, -5, 32, -7], 5) and

NewtonRaphson( [7, 3, -5, 32, -7], -50)

Note that only real roots will be returned when we start the search with a

real value. You may not use any library functions for taking

derivatives or evaluating polynomials. Write auxiliary functions:

def polyval(fpoly, x):

"""polyval(fpoly, x)

Given a set of polynomial coefficients from highest order to x^0,

compute the value of the polynomial at x. We assume zero

coefficients are present in the coefficient list/tuple.

Example: f(x) = 4x^3 + 0x^2 + 9x^1 + 3 evaluated at x=5

polyval([4, 0, 9, 3], 5))

returns 548

"""

def derivative(fpoly):

"""derivative(fpoly)

Given a set of polynomial coefficients from highest order to x^0,

compute the derivative polynomial. We assume zero coefficients

are present in the coefficient list/tuple.

Returns polynomial coefficients for the derivative polynomial.

Example:

derivative((3,4,5)) # 3 \* x\*\*2 + 4 \* x\*\*1 + 5 \* x\*\*0

returns: [6, 4] # 6 \* x\*\*1 + 4 \* x\*\*0

"""

2. Show that the Manhattan distance is consistent for N-puzzles if one

assumes that the cost of a move is 2 (swap one tile each way)

Ans:

T is a tile misplaced in a configuration. Let ht() be Manhattan distance which moves t to the goal place.

Assume x1 and x2 configurations. Also, c(a, b) is real cost of cheapest path between a and b.

There are three cases.

Case 1: T misplaced in x1 and x2:

🡺 Contributes 1 to ht(x1)

🡺 Contributes 1 to ht(x2)

🡺 Contributes 0 to c(x1, x2)

Thus, ht(x1) <= ht(x2) + c(x1, x2)

Case 2: T misplaced in x1:

🡺 Contributes 1 to ht(x1)

🡺 Contributes 0 to ht(x2)

🡺 Contributes 2n to c(x1, x2), n <=0

Thus, ht(x1) <= ht(x2) + c(x1, x2)

Case 3: T misplaced in x2:

🡺 Contributes 0 to ht(x1)

🡺 Contributes 1 to ht(x2)

🡺 Contributes 2n to c(x1, x2), n <=0

Thus, ht(x1) <= ht(x2) + c(x1, x2)

3. Avena Cereal Company makes a cereal called Crunchy Oat Clusters. Each cluster has must weigh no less than 1 gram and no more than 2 grams and is composed of a combination of oats, rice, and honey. Avena, hires you, the renowned genetic algorithm expert to optimize their cereal. Their food scientists have developed a function that can predict how crunchy clusters will be given the amounts of oats, rice, and honey: fcrunch(oats, rice, honey). Describe a set of crossover and mutation functions that do not violate the production constraints.

Ans:

Crossover and mutation functions:

Let a list whose length is 20. There are 4 types of element in the list: 1. Oats, 2. Rice, 3. Honey, 4. None. Each element in list represents 0.1 gram.

For example, The cluster contains 0.7g oats, 0.5g rice, and 0.3g honey, the list well be like.

[o, o, o, o, o, o, o, r, r, r, r, r, h, h, h, n, n, n, n, n]

Crossover:

Random shuffle lists of two clusters. Random select an index to separate a list into two parts. After separated, swap first parts of two clusters. we can get two brand new lists, but they may violate the constraints, that is, the ‘None’ elements are more than 10. Therefore, we have to check the number ‘None’ when we swap elements.

Crossover Algorithm:

1. Get the amount of ‘None’ in second parts.

2. Swap index= n element. (n start from 0)

3. If the element is ‘None’, the amount of ‘None’ add 1.

4. If the amount of ‘None’ is more than 10 🡪 Replace the ‘None’ with the original element, and the amount of ‘None’ subtract 1.

5. n++. If the n < half length, go back to step 2. Otherwise, crossover done.

Mutation:

For each cluster, randomly select an index and replace it with another elements.

4. Consider the fairly simple problem of selecting 2 matching cards from a deck of

cards with pictures on them. The pictures consist of moon, sun, and stars, and

there are two instances of each card (6 cards in total). Legal moves are pick a

card that has not been picked, and the goal state is matching the first card drawn.

Sketch an and-or search tree for this problem. You do not need to draw the full

tree, just enough to make it clear that you understand what the full tree would

look like.

Ans: 