Nalongsone Danddank Student ID : 14958950 StarID: jf3893pd
Email: nalongsone.danddank@my.metrostate.edu

Deliverable D Project Report

Introduction:

To solve Hamiltonian Cycle problem, I have tried three Local Search's algorithms like, Generic algorithm, Simulated Anneling algorithm, and the algorithm which combine Generic and Simulated Anneling, to experiment and find out the best solution.

For all three algorithms, I initial time to limited the program running ,

```
int limitedTime = 3;// seconds, more time more better solution.
```

For Generic algorithm: set maximum of number step to prevent run out of memory,

```
int maxNumberStep = 400000;// number iterators looping.
```

For both Simulated Anneling and Generic+Simulated algorithm:

```
double temp = 100000.0;
double coolingRate = 0.0003;
```

I would like to set these number very high because I have a limited time already, so we are be able to set more time what we need before temp done.

Generic algorithm:

```
    localSearch_genericAlgorithm(limitedTime, maxNumberStep);
```

The algorithm start with a random start select path, generate initial to be the beginning of path and find out index of Node Goal "G" and node Start "S":

```
Path path = initialPath();
```

Before run the loop, take a number of steps for looping, as we know that if we want to find out the correct path we need to run (n-2)! time, which the n is the number of node in the graph. So that means if we have graph 12 nodes path we have to run (12-2)! = 10! = 3,628,800 time to find out the final answer path. So if the path is small like 6 nodes we just run 4! = 24 time we don't need prevent, but if the path is bigger them 9!, that is too long and large number than my computer to run and finished it. So I create a method to handle this problem:

```
int numberSteps = numberStepByFactory(grahpSize, maxNumberStep);
```

Then, create a available to get the better path, "bestPath". Create a trace list path to remember each path, and to prevent calculating a duplicate one. And create a offspring to process the next step of generation.

```
ArrayList<ArrayList<Integer>> tracePaths = new
        ArrayList<ArrayList<Integer>>();

tracePaths.add(path.pathIndexs);

ArrayList<Integer> offSpring = null;
```

Next, looping by the number of step that we initial in beginning. And inside the looping we have to evaluate between two path, the best path that we have in the last iterator to the new path that create from new offspring.

```
bestPath = evaluatePath(path, bestPath, i);
```

The new offspring comes from parent which build from random selection that take the previous path. So we build a nest loop to create a new offspring:

```
ArrayList<Integer> parent1 = randomSelection(path.pathIndexs);
```

```
        ArrayList<Integer> parent2 = randomSelection(path.pathIndexs);
```

Then, processing crossover of two parents to create a new offspring. And processing the mutation step of offspring.

```
        offSpring = crossover(parent1, parent2);
        mutation(offSpring);
```

In the end of nest loop, I test the new offspring, it is in my tabu list "tracePaths" or not if there be long there go to loop again, create a new one until get a new offspring.

```
        } while (isNewOffSpring(offSpring, tracePaths));
```

Next, when we get a new offspring we add to the tabu list, and take the new offspring path to compare with the old best path that we got from the previous one if it better we take the new one, if not, keep the old one.

```
        bestPath = evaluatePath(path, bestPath, i);
```

And go looping until run out the number of step we have initial or run out of time that we set on the beginning.

Finally we will get the best path and print out the best path.

The result of best paths as I have tried for each test files by limited time 300 seconds:

```
F1d.txt -->Best cycle: A C B D E , dist = 13
F1dd.txt -->Best cycle: A D C B E , dist = 29
F2d.txt -->Best cycle: LA SF Sea Den Min Chi Atl Dal Mia Was NY Bos , dist
             = 6847
F2dd.txt -->Best cycle: LA SF Sea Den Dal Mia Atl Was NY Chi Min Bos , dist
             = 7378
F3d.txt -->Best cycle: GPo Crk SPt Mkt Owa Nfd FFl DLk Bau EGF LFa Skp Mvo
             LCr Frb ALe SCl Wrt Clq Dul Wnd Fmt Wlm Mar NUl Hut Rch Aus
             Mpl Alx Bem SPl Lit GRa Win RWi Mor Bra Stl IFl Hib Ely Vir
             TRF Noy Mrh Was Pip Luv , dist = 7145

F3dd.txt -->Best cycle: Unfortunately, run out of time!
F4d.txt -->Best cycle: c00 c21 c42 c40 c16 c18 c29 c17 c51 c11 c50 c53 c34
             c26 c03 c38 c14 c48 c13 c05 c47 c24 c31 c28 c15 c49 c08 c19
             c41 c35 c27 c20 c04 c52 c09 c46 c54 c56 c37 c30 c06 c22 c07
             c36 c44 c33 c25 c45 c32 c39 c12 c57 c23 c43 c10 c02 c55 c01 ,
             dist = 88061
```

Simulated Anneling algorithm:

```
localSearch_simulatedAnnelingAlgorithm(limitedTime, temp, coolingRate);
```

The algorithm start with a random start select path, generate initial to be the beginning of path and find out index of Node Goal "G" and node Start "S":

```
        Path path = initialPath();
```

Then, create a available to get the better path, "bestPath". And create a trace list path to remember each path, and to prevent calculating a duplicate one. And create a offspring to process the next step of generation.

```
        ArrayList<ArrayList<Integer>> tracePaths = new
                ArrayList<ArrayList<Integer>>();
        tracePaths.add(path.pathIndexs);
```

Next, looping by follow the temperate and cooling rate which we set in the beginning. Until the temperate go to 0. inside the loop we build a nest loop that we create a new path by random selection of previous path and looping until we get a new one that not including in tabu list or "tracePath" list.

```
        newPathIndexs = randomSelection(path.pathIndexs);
        while (isNewOffSpring(newPathIndexs, tracePaths));
        tracePaths.add(newPathIndexs);
```

When we get a new path. We get energy of solutions and decide if we should accept the neighbour or not by taking a acceptance Probability function, $e^{(h(a) - h(a'))/T}$.

```
        int currentDistance = path.dist;

        int neighbourDistance = newPath.dist;

        double rand = randomDouble();

        if (acceptanceProbability(currentDistance, neighbourDistance,
              temp) > rand) { path = newPath;}
```

Then keep track of the best solution:

```
        bestPath = evaluatePath(path, bestPath, temp);
```

The end of loop we have to cool down the temperate before go to the next looping.

```
        temp *= 1 - coolingRate;
```

Go the next one of looping until the temperate to down to 0 or we find out the correct answer or we terminate by run out of time. Finally we should get the best path and print out.

The result of best paths as I have tried for each test files by limited time 300 seconds:

```
F1d.txt -->Best cycle: A C B D E , dist = 13

F1dd.txt -->Best cycle: A D C B E , dist = 29

F2d.txt -->Best cycle: LA SF Sea Den Dal Atl Chi Min Was Mia NY Bos , dist
             = 7525

F2dd.txt -->Best cycle: LA Sea SF Den Min Dal Mia Atl Was NY Chi Bos , dist
             = 8171

F3d.txt -->Best cycle: GPo LFa Frb ALe GRa Vir Hib Noy Crk Mar Bem Mrh EGF
             Dul Bra DLk Wlm Mvo FFl IFl Skp Wrt Bau Mor Alx TRF Aus Clq
             Ely Win SPt Lit Stl Owa LCr NUl Was Hut Fmt Wnd Nfd Rch SCl
             Mpl SPl Mkt RWi Pip Luv , dist = 7383

F3dd.txt -->Best cycle: Unfortunately, run out of time!

F4d.txt -->Best cycle: c00 c46 c35 c51 c30 c32 c14 c26 c22 c54 c19 c29 c24
             c57 c56 c07 c39 c04 c50 c17 c20 c05 c34 c53 c11 c33 c18 c38
             c02 c37 c08 c31 c12 c52 c43 c23 c49 c03 c47 c55 c10 c40 c13
             c27 c48 c25 c42 c16 c41 c06 c09 c44 c28 c45 c15 c21 c36 c01 ,
             dist = 87467
```

Combine Generic and Simulated Anneling algorithm:

```
localSearch_genericAndSimulatedAnnelingAlgorithm(limitedTime, temp, 
coolingRate);
```

This combination algorithm, I use the idea of generic algorithms to generate a new offspring by random selection of parent, crossover, and mutation to create a new offspring.

```
        ArrayList<Integer> parent1 = randomSelection(path.pathIndexs);

        ArrayList<Integer> parent2 = randomSelection(path.pathIndexs);

        offSpring = crossover(parent1, parent2);

        mutation(offSpring);
```

And I use the idea of energy of solution and the acceptance probability of Simulated Anneling algorithm, $e^{(h(a) - h(a'))/T}$ to select the better path and the concept of cooling from high temperate to be low.

```
        double rand = randomDouble();

        if (acceptanceProbability(currentDistance, neighbourDistance,
                temp) > rand) {path = newPath;}
```

Then keep track of the best solution:

```
        bestPath = evaluatePath(path, bestPath, temp);
```

The end of loop we have to cool down the temperate before go to the next looping.

```
        temp *= 1 - coolingRate;
```

Go the next one of looping until the temperate to down to 0 or we find out the correct answer or we terminate by run out of time. Finally we should get the best path and print out.

The result of best paths as I have tried for each test files by limited time 300 seconds:

```
F1d.txt -->Best cycle: A C B D E , dist = 13

F1dd.txt -->Best cycle: A D C B E , dist = 29

F2d.txt -->Best cycle: LA SF Sea Den Min Dal Chi Atl Mia Was NY Bos , dist
             = 6914

F2dd.txt -->Best cycle: LA Den SF Sea Min Dal Chi Was Atl Mia NY Bos , dist
             = 8580

F3d.txt -->Best cycle: GPo Fmt Wnd Pip Hut Bem IFl Vir Ely GRa Stl NUl Was
             Clq Mrh Noy TRF Crk Alx Owa Wrt Win LCr Mpl Hib Dul EGF RWi
             Nfd Wlm Mor Skp ALe Frb SCl SPl Bra Lit LFa Mkt FFl SPt Aus
             Mar Mvo DLk Bau Rch Luv , dist = 7204

F3dd.txt -->Best cycle: Unfortunately, run out of time!

F4d.txt -->Best cycle: c00 c52 c40 c50 c13 c16 c48 c18 c36 c53 c34 c56 c41
c37 c30 c06 c09 c51 c27 c44 c33 c17 c07 c32 c20 c15 c39 c03 c12 c05 c08 c21
c23 c54 c42 c57 c29 c25 c35 c22 c38 c46 c45 c14 c28 c47 c49 c43 c24 c02 c04
c31 c19 c55 c26 c10 c11 c01 , dist = 90588
```

Conclusion:

As I have experiment with three algorithms, When I set the parameters and limited time to low to expect algorithms run fast and get the best path quickly. The simulated Anneling algorithm is perform better to get the a better path. However, When I need the best answer with set the parameters and limited time to be high, the best algorithm is Generic algorithm. Just like the result I show above for each algorithm. And the algorithm which often get lucky to find out the best path for f2dd.txt file is also Generic algorithm. The last algorithm which to combine of Generic and Simulated Anneling, it does not performs better then two original one, I was disappointed to combine them. So I think I would like to try **Generic algorithm** if need in the future project.

Reference:

Poole, et. al. *Artificial Intelligence, 2nd ed*, ISBN 978-1107195394. http://artint.info/index.html.

Jay Chiruvolu, Menlo School. *JOURNAL OF ANALYSIS OF APPLIED MATHEMATICS* Volume 11, 2018 pages: 88-105 .

Marco Dorigo, Senior Member, IEEE, and Luca Maria Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem* IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 1, NO. 1, APRIL 1997

S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, *Optimization by Simulated Annealing,* Science, New Series, Vol. 220, No. 4598 (May 13, 1983), pp. 671-680