ASSIGNMENT 3

Subject : Linked-list Implementation

Advisor : : Res. Assist. Selim YILMAZ

Programming Language: C

Due Date: 02.01.2020 — 18:59:59

Name:İsmail ATEŞ

Number:21626953

INTRODUCTION / AIM:

In this assignment, we are expected to gain knowledge on C language including linked-list implementation and dynamic memory allocation. Prior knowledge on basic C syntax is assumed.

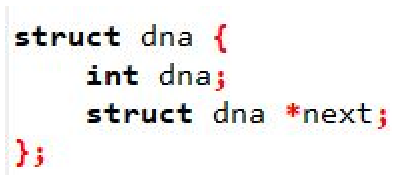
PROBLEM DEFINITION:

Our task is, storing selection, mutate, population, xover files on program.Program arguments are problem size, population size, generation size.

The detailed version of your GA implementation is given as follows:

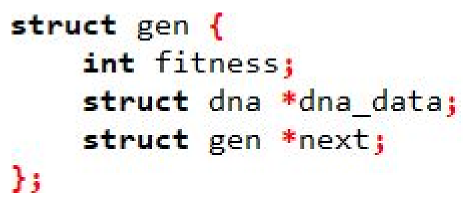
```
Read command line arguments;
Load all the parameters including selection, mutation, crossover;
Initialize population w.r.t. population file;
Evaluate the fitness and ranking of each chromosome;
Sort the population in ascending order;
Store the best chromosome;
Print the population with the best chromosome found so far;
repeat
        Apply selection;
        Apply crossover;
        Apply mutation;
        Evaluate the fitness and ranking of each chromosome;
        Sort the population in ascending order;
        Update the best chromosome if found better;
        Print the population with the best chromosome found so far;
until MAX GEN is reached;
return best chromosome;
```

STRUCTS THAT I USED:



```c
struct dna {
    int dna;
    struct dna *next;
};
```

dna struct stores:
- dna(gene)
-link of next dna



```
struct gen {
    int fitness;
    struct dna *dna_data;
    struct gen *next;
};
```

gen struct stores:
- fitness
-link of next dna
-link of next gen
#its chrosome's struct

FUNCTIONS THAT I USED:

void readFiles(struct gen **head, int dna_size, int pop_size, char bestChromosome[])
Function read 4 file which are population, selection, xover, mutate. Function call addGen() then for each line in selection file it call evolution(),doMutate(),calculateFitness() functions.

void addGen(struct gen **head, char genList[], int n)
Function creates new nodes of gen then call addDnaToGen() function for each dna.

void addDnaToGen(struct gen *head, int dna)
For each dna function creates new node of dna.

void sort(struct gen **head)
Function sort the Dna's for their fitness.

void calculateFitness(struct gen *head,int n, int generation, char bestChromosome[])
Function calculate the fitness of dna's.
void doMutate(struct gen *head, int mutateIndex)
Function mutates the gen.
void doXover(struct gen **head, int genIndex1, int genIndex2, int dnaIndex1, int dnaIndex2)
Function cross the gens of given dna's.
void evolution(struct gen *head, char selectionLine[], char xoverLine[], int pop_size)
Function calls format the selection and xoverline then call doXover()
void Display(struct gen *head, int generation, char bestChromosome[])
It's print function
void freeList(struct gen* head)
It's free the structsç
int main(int argc,char* argv[])
Main gets arguments, then call readFiles()

MY ALGORİTHM

My program starts with readFiles() function. “readFiles()” read the files line by line. First while population file==NULL, readFiles() call addGen. And Then AddGen() calls addDnaToGen(). Then readFiles() call evolution(), doMutate(),calculateFitness() functions respectively. evolution() calls doXover(). calculateFitness() call sort() then Display(). So I crate my first generation with my files first line. When readFiles() get other line, this steps repeated by program.