

KELOMPOK X

llllllll

8 QUEENS

GENETIC ALGORITHM

KECERDASAN BUATAN F



KELOMPOK X

llllllllll

NADIF MUSTAFA / 5025211127
FREDERICK YONATAN / 5025211121
NIZAM HAKIM / 5025211209

KECERDASAN BUATAN F



```
int dr[] = {-1, -1, 0, 1, 1, 1, 0, -1};  
int dc[] = {0, 1, 1, 1, 0, -1, -1, -1};  
  
void printBoard(vvi board)  
{  
    cout << endl;  
    for(int i = 0; i < 8; ++i){  
        for(int j = 0; j < 8; ++j)  
            cout << board[i][j] << " ";  
        cout << endl;  
    }  
    cout << endl;  
}
```

Pada bagian awal, terdapat **array dr dan dc** yang merepresentasikan perpindahan baris dan kolom yang dapat dilakukan oleh queen. Kemudian terdapat fungsi **printBoard** yang digunakan untuk mencetak papan catur ke layar.

```
bool validPos(int r, int c)
{
    return (r >= 0 and r < 8 and c >= 0 and c < 8);
}

bool finished(vpsi population)
{
    for(int i = 0; i < 100; ++i){
        if(population[i].sc == 28)
        {
            cout << "Answer :";
            printBoard(translate(population[i].fi));
            return 1;
        }
    }
    return 0;
}

char randomGene()
{
    return '0' + (rand() % 8);
}
```

Pada bagian awal, terdapat fungsi **validPos** yang digunakan untuk memeriksa apakah posisi queen yang diinginkan berada di dalam papan catur (8x8).

Kemudian terdapat fungsi **finished** yang menerima sebuah populasi dan memeriksa apakah ada individu pada populasi yang merupakan solusi dari masalah 8 Queens. Jika ditemukan solusi, maka fungsi akan mencetak papan catur yang merupakan solusi tersebut dan mengembalikan nilai true. Jika tidak ditemukan solusi, maka fungsi akan mengembalikan nilai false.

Terakhir, terdapat fungsi **randomGene** yang digunakan untuk menghasilkan sebuah karakter acak ('0'-'7') yang merepresentasikan posisi queen pada suatu kolom. Karakter tersebut akan digunakan pada tahap inisialisasi populasi dengan menentukan posisi queen pada masing-masing kolom secara acak.

Pada bagian awal, terdapat fungsi **randomChromosome** menghasilkan sebuah string acak yang merepresentasikan suatu kromosom. Kromosom tersebut terdiri dari 8 karakter yang merepresentasikan posisi queen pada masing-masing kolom.

Fungsi **randomPopulation** membuat sebuah populasi acak berisi 100 kromosom dengan 8 gen dalam bentuk string. Setiap kromosom dibuat dengan memanggil fungsi randomChromosome, dan setiap kromosom dipastikan unik dengan menggunakan set appeared. Populasi acak ini kemudian dikembalikan sebagai hasil fungsi.

```
string randomChromosome()
{
    string res = string(8, '0');
    for(int i = 0; i < 8; ++i)
        res[i] = randomGene();
    return res;
}

vs randomPopulation()
{
    vs population(100);
    set <string> appeared;

    for(int i = 0; i < 100; ++i)
    {
        do population[i] = randomChromosome();
        while(appeared.count(population[i]));
        appeared.insert(population[i]);
    }
    return population;
}
```

```
int fitness(string chromosome)
{
    vvi board = translate(chromosome);
    int attacked = 0;

    for(int i = 0; i < 8; ++i){
        for(int j = 0; j < 8; ++j)
        {
            int r = (chromosome[i] - '0') + dr[j];
            int c = i + dc[j];

            while(validPos(r, c) and board[r][c] != 1){
                r += dr[j];
                c += dc[j];
            }
            if(validPos(r, c) and board[r][c] == 1)
                attacked++;
        }
    }
    return 28 - (attacked / 2);
}
```

Fungsi **fitness** adalah fungsi untuk mencari fitness value dari masing-masing chromosome / state, fitness value pada kasus ini adalah banyaknya queen yang tidak saling menyerang.

Fungsi **selection** merupakan implementasi operator seleksi pada algoritma genetika. Fungsi ini menerima satu parameter berupa vektor populasi saat ini dan menghasilkan vektor 10 kromosom terbaik yang dipilih menggunakan metode turnamen.

Metode turnamen ini memilih kromosom dengan fitness score tertinggi dari 10 kromosom acak pada setiap pengulangan turnamen.

Kromosom terbaik pada setiap pengulangan turnamen dimasukkan ke dalam vektor orang tua

.

```
vs selection(vpsi population)
{
    vs parents;
    for(int p = 0; p < 10; ++p)
    {
        int maxFitness = 0;
        string bestParent = population[0].fi;

        for(int k = 0; k < 10; ++k)
        {
            int candidate = rand() % 100;

            if(population[candidate].sc > maxFitness)
            {
                maxFitness = population[candidate].sc;
                bestParent = population[candidate].fi;
            }
        }
        parents.pb(bestParent);
    }
    return parents;
}
```

```
void crossover(vs &parents)
{
    for(int i = 0; i < 9; ++i){
        for(int j = i + 1; j < 10; ++j)
        {
            string parent1 = parents[i];
            string parent2 = parents[j];
            string child1, child2;
            int part = rand() % 7;

            for(int itr = 0; itr <= part; ++itr){
                child1 += parent1[itr];
                child2 += parent2[itr];
            }
            for(int itr = part + 1; itr < 8; ++itr){
                child1 += parent2[itr];
                child2 += parent1[itr];
            }
            parents.pb(child1);
            parents.pb(child2);
        }
    }
}
```

Fungsi **crossover** adalah fungsi untuk mengawinkan 10 parents (chromosome terbaik) yang sudah dipilih sehingga menghasilkan 90 keturunan dan membentuk generasi baru yang terdiri dari 10 parents dan 90 children. Crossover yang digunakan dalam kasus ini adalah single point crossover.

```
void mutation(vs &parents)
{
    for(int i = 0; i < 100; ++i)
    {
        int index = rand() % 8;
        parents[i][index] = randomGene();
    }
}
```

```
vpsi objective(vs parents)
{
    vpsi population(100);
    for(int i = 0; i < 100; ++i)
    {
        int fitnessValue = fitness(parents[i]);
        population[i] = mp(parents[i], fitnessValue);
    }
    return population;
}
```

fungsi **mutation** adalah fungsi yang digunakan untuk memutasi tiap-tiap chromosome pada suatu generasi, mutasi dilakukan dengan cara mengganti salah satu gene menjadi random gene. mutasi dilakukan agar suatu generasi menjadi lebih bervariasi.

Fungsi **objective** menyimpan fitness value dari masing-masing chromosome ke dalam vector `<pair<string, int>>`.

```
void geneticAlgorithm()
{
    int generation = 1;
    vs parents = randomPopulation();
    vpsi population = objective(parents);

    while(true){
        if(finished(population)){
            cout << "Generation : " << generation << endl;
            return;
        }
        parents = selection(population);
        crossover(parents);
        mutation(parents);
        population = objective(parents);
        generation++;
    }
}
```

Fungsi **geneticAlgorithm** merupakan implementasi algoritma genetika untuk menyelesaikan permasalahan 8 Queen. Pada awalnya, fungsi ini akan membuat populasi acak sebanyak 100 kromosom dan mengevaluasi tiap kromosom untuk memperoleh nilai fitness masing-masing. Selanjutnya, dilakukan iterasi pada setiap generasi dengan melakukan seleksi, crossover, dan mutasi pada populasi saat ini. Jika ditemukan kromosom yang memenuhi syarat, fungsi akan berhenti dan mencetak hasilnya.



```
int main()
{
    srand(time(NULL));
    geneticAlgorithm();
    return 0;
}
```

Code di atas merupakan fungsi **main** yang akan dijalankan ketika program dijalankan.

Pertama-tama, fungsi ini memanggil srand untuk menginisialisasi seed random number generator, kemudian memanggil fungsi geneticAlgorithm. Setelah itu, program akan mengembalikan nilai 0 untuk menandakan bahwa program telah berakhir dengan sukses.

CONTOH HASIL OUTPUT

Answer :

```
0 0 1 0 0 0 0 0  
0 0 0 0 0 1 0 0  
0 1 0 0 0 0 0 0  
0 0 0 0 0 0 1 0  
0 0 0 0 1 0 0 0  
1 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 1  
0 0 0 1 0 0 0 0
```

Generation : 3

Answer :

```
0 0 0 0 1 0 0 0  
1 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 1  
0 0 0 1 0 0 0 0  
0 1 0 0 0 0 0 0  
0 0 0 0 0 0 1 0  
0 0 1 0 0 0 0 0  
0 0 0 0 0 1 0 0
```

Generation : 98

Answer :

```
0 0 0 0 0 0 0 1  
0 1 0 0 0 0 0 0  
0 0 0 0 1 0 0 0  
0 0 1 0 0 0 0 0  
1 0 0 0 0 0 0 0  
0 0 0 0 0 0 1 0  
0 0 0 1 0 0 0 0  
0 0 0 0 0 1 0 0
```

Generation : 17



TERIMA KASIH