EE6227 Assignment 4

1. Requirements

This code is written in C++. Thus if you want to build from source code, your system should have a C++ compilation environment. For convenience, I have also provided binary files that can be run under **macOS, Windows or Ubuntu**. Next I will describe the two methods of running from binary files and building from source code, respectively, under different operating systems.

a. Run from binary files

In the `bin` directory there are my pre-compiled executable files, you can select the executable file that corresponds to your system to run.

Windows

Double-click the `GA4_island_Windows.exe` file

macOS / Ubuntu

Open a terminal in the `bin` directory

```
./GA4_island_macOS  # if you use intel-based Mac
./GA4_island_Ubuntu # if you use Ubuntu
```

Note: These binaries are only valid for testing in my environment, if they fail to run, please build from source code !!!

b. Build from source code

Windows

```
g++ islandGA.cpp Genetic.cpp -o GA4_island -std=c++11 -static
.\GA4_island.exe
```

macOS / Ubuntu

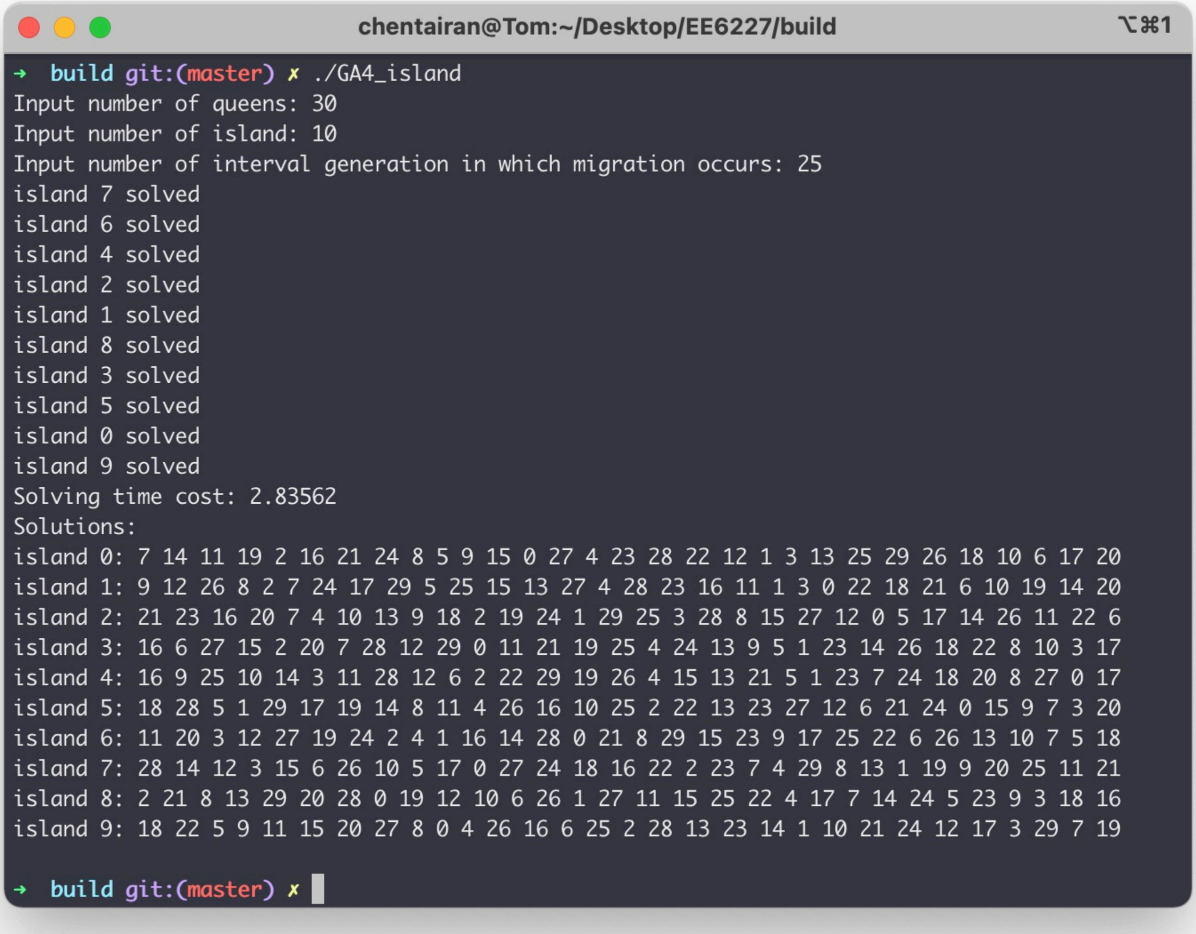
Option1: Using g++ compiler

```
g++ islandGA.cpp Genetic.cpp -o GA4_island -std=c++11
./GA4_island
```

Option2: Using cmake & make

```
cd /path/to/this/code  # Change to your own code path
mkdir build
cd build
cmake ..
make
./GA4_island
```

The following images show the results of the run:



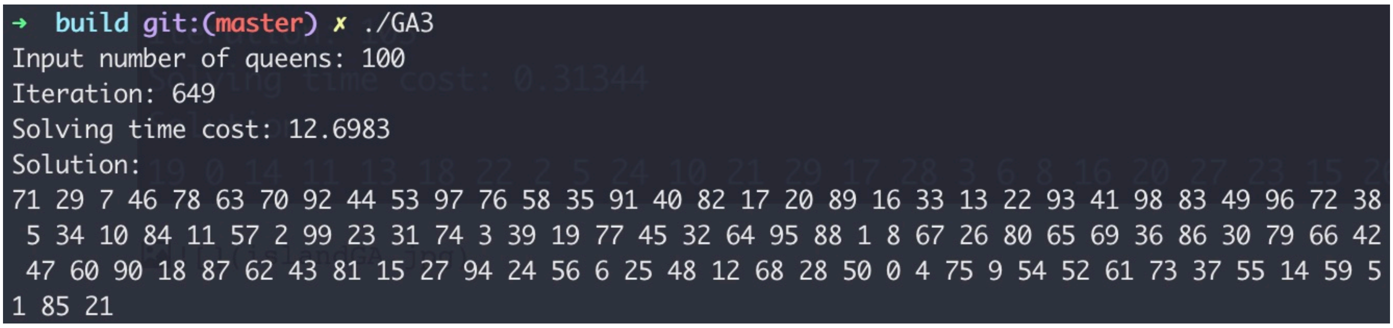
```
➜  build git:(master) ✗ ./GA4_island
Input number of queens: 30
Input number of island: 10
Input number of interval generation in which migration occurs: 25
island 7 solved
island 6 solved
island 4 solved
island 2 solved
island 1 solved
island 8 solved
island 3 solved
island 5 solved
island 0 solved
island 9 solved
Solving time cost: 2.83562
Solutions:
island 0: 7 14 11 19 2 16 21 24 8 5 9 15 0 27 4 23 28 22 12 1 3 13 25 29 26 18 10 6 17 20
island 1: 9 12 26 8 2 7 24 17 29 5 25 15 13 27 4 28 23 16 11 1 3 0 22 18 21 6 10 19 14 20
island 2: 21 23 16 20 7 4 10 13 9 18 2 19 24 1 29 25 3 28 8 15 27 12 0 5 17 14 26 11 22 6
island 3: 16 6 27 15 2 20 7 28 12 29 0 11 21 19 25 4 24 13 9 5 1 23 14 26 18 22 8 10 3 17
island 4: 16 9 25 10 14 3 11 28 12 6 2 22 29 19 26 4 15 13 21 5 1 23 7 24 18 20 8 27 0 17
island 5: 18 28 5 1 29 17 19 14 8 11 4 26 16 10 25 2 22 13 23 27 12 6 21 24 0 15 9 7 3 20
island 6: 11 20 3 12 27 19 24 2 4 1 16 14 28 0 21 8 29 15 23 9 17 25 22 6 26 13 10 7 5 18
island 7: 28 14 12 3 15 6 26 10 5 17 0 27 24 18 16 22 2 23 7 4 29 8 13 1 19 9 20 25 11 21
island 8: 2 21 8 13 29 20 28 0 19 12 10 6 26 1 27 11 15 25 22 4 17 7 14 24 5 23 9 3 18 16
island 9: 18 22 5 9 11 15 20 27 8 0 4 26 16 6 25 2 28 13 23 14 1 10 21 24 12 17 3 29 7 19

➜  build git:(master) ✗
```

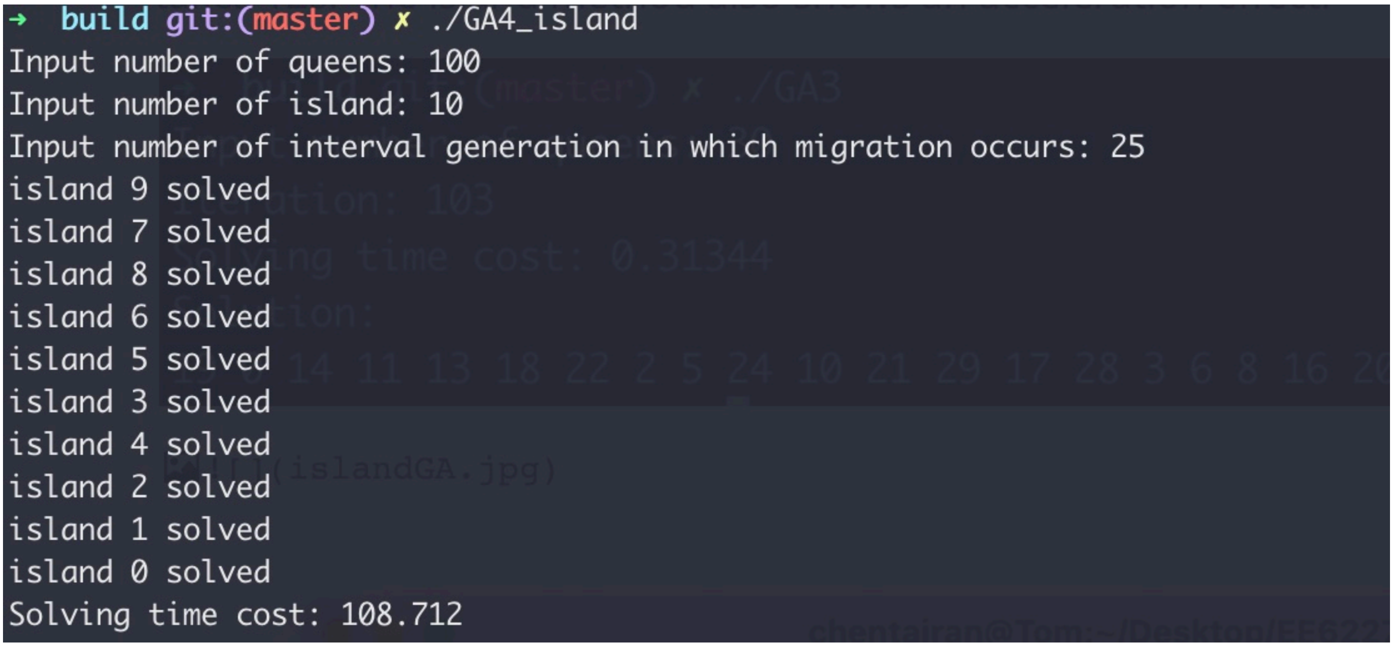
2. Compare islandGA with Normal GA

I compare island GA with normal GA. For the normal GA, the time taken to solve for an n=30 solution is 0.31344s. And the time taken for island GA to solve for **island=10, n=30 is 2.83562s, averaging 0.283s per solution**. The island method speeds up the solver.

I also test the case when **n=100**. **Island GA takes 108.712s (averaging 10.8712s per solution)**, and **normal GA takes 12s.** The island method also shows an acceleration effect. The following figures show the results of the two GA algorithms respectively.



```
➜  build git:(master) ✗ ./GA3
Input number of queens: 100
Iteration: 649
Solving time cost: 12.6983
Solution:
71 29 7 46 78 63 70 92 44 53 97 76 58 35 91 40 82 17 20 89 16 33 13 22 93 41 98 83 49 96 72 38
 5 34 10 84 11 57 2 99 23 31 74 3 39 19 77 45 32 64 95 88 1 8 67 26 80 65 69 36 86 30 79 66 42
 47 60 90 18 87 62 43 81 15 27 94 24 56 6 25 48 12 68 28 50 0 4 75 9 54 52 61 73 37 55 14 59 5
1 85 21
```



```
➜  build git:(master) ✗ ./GA4_island
Input number of queens: 100
Input number of island: 10
Input number of interval generation in which migration occurs: 25
island 9 solved
island 7 solved
island 8 solved
island 6 solved
island 5 solved
island 3 solved
island 4 solved
island 2 solved
island 1 solved
island 0 solved
Solving time cost: 108.712
```

3. Island GA implementation

The algorithm is implemented as follows:

- Firstly, **k** GA solvers are constructed, and they are randomly initialized.
- Based on the iterations of the original GA, **m** chromosomes are randomly selected from the solver every **t** iterations and put into the next solver.
- Stop iterating when all solvers have found a solution

Implementation details can be found in `islandGA.cpp` and `Genetic.cpp`