برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۳

استاد: دکتر رامتین خسروی

طراحان: دانشور امراللهی، زهرا یزدانی، سروش میرزا سروری، احمدرضا سلیمی، امین ستایش

مهلت تحویل: سه‌شنبه ۲۴ فروردین ۱۴۰۰، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با طراحی بالا به پایین[1] یک مسئله است. با توجه به حجم پروژه لازم است که قبل از شروع پیاده‌سازی زمانی را به طراحی اختصاص دهید. در غیر این صورت در هنگام پیاده‌سازی با مشکل مواجه خواهید شد. بنابراین ابتدا به چگونگی شکستن مسئله به مسائل کوچکتر و پخش کردن مسؤولیت‌ها میان قسمت‌های مختلف برنامه فکر کنید.

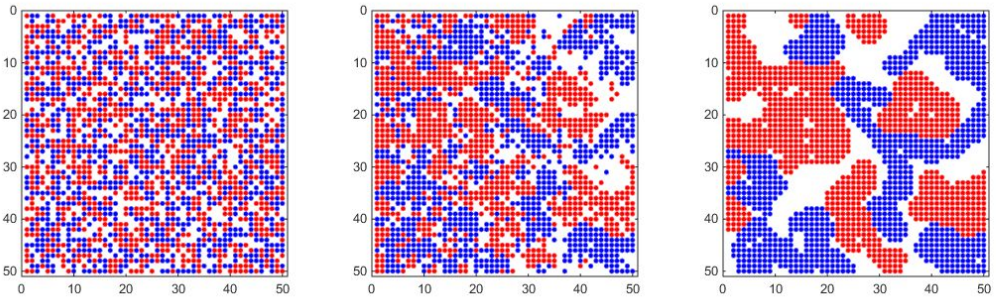
شبیه‌سازی تبعیض نژادی

تبعیض نژادی همیشه یکی از مشکلات اجتماعی در بعضی جوامع بوده است. در سال ۱۹۷۱ اقتصاددان آمریکایی Thomas Schelling آزمایشی طراحی کرد تا تبعیض نژادی و تفکیک شدن افراد از یکدیگر را بررسی کند. در این تمرین به پیاده‌سازی شبیه‌سازی او و بررسی نتایج شبیه‌سازی می‌پردازیم. از طریق این لینک می‌توانید شبیه‌سازی را به صورت آنلاین برای خود اجرا کنید. البته توجه داشته باشید که جزییات پیاده‌سازی شبیه‌سازی در این لینک ممکن است با نحوه پیاده‌سازی در صورت پروژه متفاوت باشد.

شبیه‌سازی به این صورت است که در جهان اولیه دو نوع از موجودات وجود دارند که در جهان پخش شده‌اند. موجوداتی که تعداد اطرافیان مشابه به خودشان از عددی کمتر باشد احساس نارضایتی می‌کنند و به صورت تصادفی در هر مرحله به نقطه دیگری از صفحه پرش می‌کنند تا در پایان شبیه‌سازی احساس رضایت داشته باشند.

[1] Top Down Design

به عنوان مثال، در شکل زیر، نقاط قرمز و آبی نشان‌دهنده دو گونه از موجودات مختلف هستند. در ابتدای شبیه‌سازی تعدادی از این موجودات رضایت کافی ندارند، اما همانطور که در شکل مشخص است پس از اجرای شبیه‌سازی در چند مرحله موجودات به گونه‌ای از یکدیگر تفکیک شده‌اند تا حداقل میزان رضایت را داشته باشند.



نکته جالبی که در این شبیه‌سازی می‌توان به آن اشاره کرد این است که حتی میزان کم تمایل به جدا بودن از دیگران برای هر فرد در نهایت می‌تواند باعث شکل‌گیری جوامع بسیار تفکیک‌شده شود.

در این تمرین از شما می‌خواهیم که مدل ساده شده‌ی این شبیه‌سازی را پیاده سازی کنید.

توضیحات شبیه‌سازی

جهان موجودات

- یک جامعه با یک جدول $n \times m$ متشکل از خانه‌ها نشان داده می‌شود.
- هر خانه از جدول، می‌تواند خالی باشد یا شامل دقیقا یک موجود زنده باشد.
- هر موجود دقیقا متعلق به یکی از گونه‌های قرمز یا آبی است.

رضایت موجودات

یک موجود زنده **ناراضی** است اگر و تنها اگر با حداقل p درصد از همسایه‌های خود، **مشابه نباشد**. مقدار درصد p را **آستانه‌ی رضایت**[2] می‌نامیم. خانه‌های بالا، پایین، چپ و راست هر خانه، همسایه‌های آن خانه به حساب می‌آیند.

[2] happiness threshold

- اگر یک موجود زنده همسایه‌ای خالی داشته باشد، با آن همسایه **مشابه** در نظر گرفته می‌شود.

برای مثال اگر یک موجودی از گونه قرمز باشد و ۴ همسایه‌اش، دو همسایه از گونه قرمز، یک همسایه از گونه آبی و یک همسایه خالی داشته باشد، این موجود با ۷۵٪ از همسایه‌های خود مشابه است. اگر آستانه‌ی رضایت را ۷۰٪ یا ۷۵٪ در نظر بگیریم راضی به حساب می‌آید و اگر ۸۰٪ در نظر بگیریم ناراضی به حساب می‌آید.

قواعد پرش موجودات

- پارامتر زمان در این دنیا گسسته است و مبدا آن $t = 0$ می‌باشد.
- اگر در لحظه t یک موجود ناراضی باشد، در لحظه $t + 1$ به صورت **تصادفی** به خانه دیگری (مقصد) پرش کرده‌است که ویژگی‌های زیر را دارد:
  - خانه مقصد در لحظه t یا خالی بوده یا متعلق به موجود ناراضی دیگری بوده است که از خانه خود به خانه دیگری در این مرحله جابه‌جا شده است.
  - خانه مقصد با خانه مبدا پرش حتما متمایز است. (هیچ موجود ناراضی سر جای خود نمی‌ماند)
  - موجود دیگری به آن خانه پرش نمی‌کند. (قانون هر خانه حداکثر ۱ موجود حفظ می‌شود)

عملکرد برنامه

برنامه شما وضعیت اولیه جامعه را از ورودی دریافت می‌کند و باید بتواند به ۲ حالت شبیه‌سازی را انجام دهد:

1. شبیه‌سازی را تا جایی انجام دهد که همه موجودات زنده راضی باشند.
2. شبیه‌سازی را به تعداد دفعات مشخص (یکی از پارامترهای ورودی) انجام دهد.

برای حالت اول، تضمین شده است که مقدار آستانه‌ی رضایت طوری است که این وضعیت امکان پذیر باشد.

ورودی‌های برنامه

برنامه شما تعدادی ورودی دارد که از طریق آرگومان‌های خط فرمان به برنامه داده می‌شود. برای آشنایی با آرگومان‌های خط فرمان به پیوست ۱ مراجعه کنید. توجه داشته باشید که این آرگومان‌ها می‌توانند **هر ترتیبی** داشته باشند. در ادامه به توضیح هر کدام می‌پردازیم:

- **جدول اولیه**

نام فایل ورودی به همراه پیشوند "f-" به برنامه داده می‌شود و فرمت فایل جدولی از کاراکترها است. برنامه‌ی شما باید این فایل را بخواند و اطلاعات آن را به شیوه‌ی مناسب ذخیره کند.

فایل ورودی وضعیت اولیه جدول را نشان می‌دهد. هر کاراکتر از این فایل دقیقا یکی از مقادیر B، R یا E را دارد که به ترتیب نشان دهنده آبی بودن گونه، قرمز بودن گونه یا خالی بودن خانه متناظر هستند.

تضمین شده است که این جدول یک مستطیل کامل است (یعنی تعداد ستون‌ها در هر سطر برابر است) و همچنین این آرگومان حتما به برنامه داده می‌شود.

برای مثال فایل زیر یک نمونه ورودی با نام map.txt است:

نمونه‌ی map.txt

```
REBRE
REBRE
REBRE
REBRE
REBRE
```

- **میزان رضایت موجودات**

این عدد صحیح که نشان دهنده‌ی درصد آستانه خوشحالی یک موجود است با پیشوند "p-" به برنامه داده می‌شود. وجود این آرگومان **اجباری نیست** و در صورتی که به برنامه داده نشود مقدار پیش‌فرض آن ۳۰ در نظر گرفته شود.

- **حالت شبیه‌سازی**

در صورت وجود این پارامتر، شبیه‌سازی باید به تعداد مشخصی بار اجرا شود و تعداد اجرا به همراه پارامتر "-s" به برنامه داده می‌شود. اگر این پارامتر داده نشود برنامه‌ی شما باید تا زمانی که همه‌ی موجودات به حداقل میزان رضایت داده شده برسند ادامه پیدا کند.

برای مثال اگر نام فایل اجرایی شما a.out باشد، برنامه به شکل زیر اجرا می‌شود:

```
./a.out -f map.txt -s 15 -p 50
```

خروجی‌های برنامه

لازم است برنامه شما موارد زیر را خروجی دهد:

- **وضعیت نهایی جهان**
  - در خروجی stdout به صورت کاراکتر مپ (مانند فرمت ورودی)
  - ذخیره در قالب یک فایل ppm به نام out.ppm به صورتی که قابل مشاهده باشد، برای اطلاع بیشتر از این فرمت فایل، به پیوست ۲ مراجعه کنید.
- **تعداد افراد ناراضی در وضعیت نهایی در خروجی stdout**
  - به صورت یک عدد صحیح در خروجی نمایش داده می‌شود. این عدد باید در یک خط جداگانه قبل از وضعیت نهایی جهان بیاید.

فرمت خروجی stdout به صورت زیر است:

قالب خروجی

```
<unhappy_count>
<world_charachter_map>
```

برای مثال شکل زیر یک نمونه خروجی است که آستانه رضایت موجودات ۳۰ در نظر گرفته شده است:

خروجی نمونه
1 RBBEE RBBRR BRRRR EERRR EEERB

توجه داشته باشید که رنگ قرمز صرفا برای بهتر نشان دادن افراد ناراضی در خروجی نمونه در صورت تمرین استفاده شده است.

نکات پایانی

- قبل از شروع پیاده‌سازی به طراحی برنامه خود زمان کافی اختصاص دهید و به جوانب مختلف آن فکر کنید.
- در این تمرین اجازه استفاده از شیءگرایی را **ندارید**.
- استفاده از وکتورهای موازی خوانایی برنامه‌ی شما را کم می‌کند. سعی کنید به جای آن‌ها از structها برای ذخیره‌ی داده‌های مرتبط استفاده کنید.
- با توجه به اینکه نوشتن فایل تصویر بخشی از تمرین است، استفاده از کتابخانه‌های آماده جهت تولید این فایل‌ها قابل قبول نیست.
- برای بعضی مقادیر آستانه‌ی رضایت، تعداد افراد ناراضی برنامه به صفر نمی‌رسد. نیازی نیست که در این حالات برنامه شما حتما خاتمه بیابد. تضمین می‌شود که در ورودی‌های آزمون این مقادیر برای حالت شبیه‌سازی تا تثبیت جهان (بدون s-) نخواهند آمد.

نحوهٔ تحویل

- کد خود را در قالب یک فایل با نام A3-SID.cpp در صفحهٔ eLearn درس بارگذاری کنید که SID شمارهٔ دانشجویی شماست؛ برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۹۹۹۹۹ باشد، نام پروندهٔ شما باید A3-810199999.cpp باشد که شامل کد شما است.

- برنامهٔ شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++11 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود.

- تمیزی کد، ذخیره کردن اطلاعات در ساختارهای مناسب، شکستن مرحله‌به‌مرحلهٔ مسئله و طراحی مناسب، در کنار تولید خروجی دقیق و درست، بخش مهمی از نمرهٔ شما را تعیین خواهد کرد.

- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

پیوست ۱

آرگومان‌های خط فرمان:

آرگومان‌های خط فرمان آرگومان‌هایی هستند که سیستم‌عامل در زمان اجرای برنامه آن‌ها را به برنامه انتقال می‌دهد. برنامه می‌تواند آن‌ها را نادیده بگیرد و یا از آن‌ها استفاده کند.

برای استفاده از این آرگومان‌ها، تابع main باید به صورت زیر نوشته شود:

```
int main(int argc, char* argv[])
```

دو آرگومان تابع را می‌توان برای دسترسی به آرگومان‌های خط فرمان استفاده کرد:

- argc عدد صحیح؛ تعداد آرگومان‌های خط فرمان داده شده به برنامه

  این مقدار حداقل برابر با یک است؛ زیرا دستور اجرای برنامه (نام پرونده اجرایی) حتماً در زمان اجرای برنامه مورد استفاده قرار می‌گیرد و همواره به‌عنوان آرگومان‌های خط فرمان شماره صفر به برنامه داده می‌شود.

- argv آرایه‌ای از رشته‌های مدل زبان C؛ آرگومان‌های خط فرمان داده شده به برنامه

به عنوان یک مثال ساده برنامه زیر را در نظر بگیرید:

```
#include <iostream>
int main(int argc, char* argv[])
{
    std::cout << "There are " << argc << "arguments:" << std::endl;
    for (int count = 0; count < argc; ++count)
        std::cout << count << " " << argv[count] << std::endl;
    return 0;
}
```

اگر برنامه به شکل زیر اجرا شود:

```
./a.out myFile.txt 100
```

خروجی زیر تولید می‌شود:

```
There are 3 arguments:
0 ./a.out
1 myFile.txt
2 100
```

برای آشنایی بیشتر با نحوه کار آرگومان‌های خط فرمان می‌توانید به این لینک مراجعه کنید.

پیوست۲

قالب فایل ppm

ذخیره سازی در این نوع فایل بدون هیچ گونه فشرده سازی انجام می شود و اطلاعات هر پیکسل به صورت سه تایی (R,G,B) ذخیره می شوند.

اطلاعات درون این نوع فایل از دو قسمت تشکیل شده است.

- هدر که تشکیل شده از سه قسمت است:
  - نوع فایل
  - عرض تصویر
  - طول تصویر
  - اسکیل رنگ (ماکسیمم مقداری که می توان به هر یک از ویژگی های R,G,B نسبت داد.)

با توجه به موارد بالا قالب هدر بدین صورت است:

"P3", width, height, 255

- دیتا که اطلاعات مربوط به رنگ هر پیکسل است به طوری که در هر سطر اطلاعات رنگ پیکسل های آن سطر با فاصله از هم قرار می گیرند.