به نام خدا

پروژه ی اول: بالاترین درجه اتصال در میدان های شهر :

میدان های یک شهر با استفاده از خیابان های متصل به آنها به یکدیگر متصل می شوند. اتصال این میدان ها را به صورت یک **شبکه** در نظر می گیریم. وقتی که یک خیابان بین دو میدان وجود دارد هر دو به یکدیگر اتصال دارند.

برای هر دو میدان **حداقل درجه گام** را به صورت حداقل خیابان مورد نیاز برای رفتن از یک میدان به میدان دیگر تعریف می کنیم. برای یک شبکه، **حداکثر درجه گام های شبکه** را به صورت ماکسیمم درجه از بین حداقل درجه گام ها به ازای هر دو میدان در شبکه تعریف می کنیم(در واقع از بین حداقل درجه گام ها بین هر دو میدان در شبکه، مقدار ماکسیمم را انتخاب می کنیم). اگر برای دو میدان در شبکه هیچ مسیری نتوان یافت شبکه را به صورت "DISCONNECTED " تعریف می کنیم.

در این سوال ما می خواهیم **حداکثر درجه گام های شبکه** را پیدا کنیم. در زیر دو مثال مختلف نشان داده شده است.

**ورودی**

خط اول دو عدد در بردارد :P() ، که نشان دهنده تعداد از میدان های شهر است و عدد R()نشان دهنده تعداد از ارتباطات در شبکه است. مثلا در شکل سمت چپ بالا 4 میدان و 4 اتصال وجود دارد. در شکل سمت راست 4 میدان و دو اتصال وجود دارد.

Emam

Saadi

Shohada

bahman

Emam

Saadi

Shohada

bahman

حداکثر درجه گام در این شبکه : 2

این شبکه Disconnected است

خط بعدی R ارتباطات را تعریف می کند. هر ازتباط به صورت زوج های دوتایی از نام میدان ها تعریف می شود(از اول خط به صورت جفت جفت در نظر بگیرید). نام شهرها منحصر به فرد است و هیچ فضای خالی در داخل یک نام وجود ندارد. نام هر میدان با یک خط فاصله از هم جدا می شوند. چون یک شخص ممکن است بیش از یک ارتباط داشته باشد پس یک نام ممکن است بیش از یکبار در این خط ذکر شود

**خروجی**

برای شبکه مورد نظر در ورودی حداکثر درجه گام های شبکه را چاپ می کنیم. اگر شبکه Disconnected است کلمه “DISCONNECTED” را چاپ می کنیم.

|  |  |
| --- | --- |
| خروجی | ورودی |
| Network: 2 | 4 4  Saadi Bahman Emam Bahman Shohada Emam Bahman Shohada |
| Network: DISCONNECTED | 4 2  Saadi Emam Shohada Bahman |

به نمونه ی آزمایشی زیر توجه کنید:

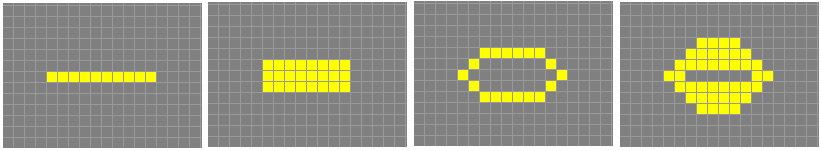
|  |  |
| --- | --- |
| Test case 1 | |
| Output | Input |
| Network: 2 | 4 4  Saadi Bahman Bahman Emam Emam Shohada Saadi Shohada |
| Network: 3 | 6 5  Khayam Saadi Bahman Saadi Bahman Keshvari Bahman Shoada Madar Bahman |
| Network: 2 | 4 4  Saadi Bahman Emam Bahman Shohada Emam Bahman Shohada |
| Network: DISCONNECTED | 4 2  Saadi Emam Shohada Bahman |
| Network: 6 | 11 14  Golha Defa Defa Milad Milad Enghelab Enghelab Shohada Enghelab Saadi Saadi Shohada Saadi Bahman Milad Keshvari Keshvari Bahman Keshvari Madar Madar Bahman Bahman Emam Emam Shohada Emam Arghavan |

پروژه ی دوم: بازی زندگی

این بازی توسط کامپیوتر انجام می شود. تعدادی سلول بصورت یک ماتریس n\*n وجود دارند. بعضی از سلول ها زنده (رنگ زرد) و بعضی مرده (رنگ خاکستری) هستند. بعد از گذشت یک واحد زمانی (مثلاً یک ثانیه) وضعیت سلولها ها طبق قوانین زیر عوض می شود. بازی با یک هسته ی اولیه شروع می شود و مراحل رشد و نمو و مرگ و میر سلول باید مرحله به مرحله نشان داده شود.

* هر سلول زنده که 1 همسایه ی زنده دارد یا هیچ همسایه ی زنده ای ندارد، از تنهایی می میرد.
* هر سلول زنده ای که 4 یا بیشتر همسایه ی زنده داشته باشد از کمبودهای ناشی از جمعیت زیاد می میرد.
* هر سلول زنده با 2 یا 3 همسایه ی زنده، به حیات خود ادامه می دهد.
* هر سلول مرده که 3 همسایه ی زنده داشته باشد، به دنیا می آید (یا زنده می شود)

مراحل رشد و نمو از چپ به راست نشان داده شده اند.



حداقل 10 حالت جالب که منجر به تولید اشکال زیبا بشود را پیدا کنید و به کاربر اجازه دهید که آنها را بارگذاری کند. باید زمان بین تغییر حالتها قابل کنترل باشد (مثل آهسته، معمولی و سریع). حداقل سایز آرایه را 20 در 40 در نظر بگیرید. اما استفاده از اندازه های خیلی بزرگتر توصیه می شود. تعداد همسایه ها در مرزها کمتر است.

پروژه ی سوم: حمله ی اعداد اول : کسایی-بهادری

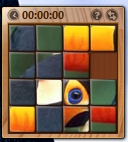
در این بازی یک عدد طبیعی با سرعت ثابت از بالای پنجره به سمت پایین حرکت می کند و کاربر باید عاملهای اول موجود در آنرا با کلیک روی دکمه های مناسب پیدا کند. در قسمت پایین برنامه تعدادی دکمه وجود دارد که شامل اعداد 2، 3، 5، 7، 11، 13 و ... هستند. مثلاً اگر عدد 24 به سمت پایین حرکت کند و کاربر دکمه ی 2 را بزند، عدد به 12 تبدیل می شود. بعد از زدن دکمه ی 3، عدد به 4 تبدیل می شود. کاربر باید دو بار دیگر دکمه ی 2 را بزند تا عدد به 2 و بعد به 1 تبدیل شود و عدد بعدی شروع به حرکت کند.

اعداد به صورت تصادفی انتخاب می شوند و سرعت حرکت آنها به تدریج زیاد می شود. اگر کاربر عامل اول را اشتباه حدش بزند، این عدد به صورت آزاد سقوط کرده و عدد بعدی شروع به حرکت می کند. اگر تعداد اعداد سقوط کرده در یک قسمت از بازی از یک حد مشخص (مثلاً 5) بگذرد، بازی خاتمه پیدا می کند.

در مورد گرافیک بازی می توانید از بازی معروف tetris ایده بگیرید.

پروژه ی چهارم: پازل دستی اعداد - سرابی

این بازی شامل 16 خانه است که اعداد 1 تا 15 به صورت تصادفی در آنها چیده شده اند و یکی از خانه ها خالی است. کاربر باید با جابجایی خانه خالی اعداد را مرتب کند. برای آشنایی بیشتر می توانید یکی از ویجتهای ویندوز را ببینید. در صورت امکان از یک تصویر به جای اعداد استفاده کنید. برنامه باید قابلیت ثبت بهترین رکوردها (از لحاظ زمانی) را داشته باشد. هم چنین کاربر باید بتواند تا 5 حرکت به عقب برگردد.



پروژه ی پنجم: مخفی کردن تصویر

در این پروژه یک فایل bmp و یک فایل متنی وجود دارند. شما باید فاییل bmp را باز کنید و در اطلاعات مربوط به پیکسلها فایل متنی را جایگذاری کنید. بدین ترتیب که هر پیکسل رنگی شامل سه رنگ و 24 بیت یا 3 بایت است. هر بایت یک عدد بین 0 تا 255 است که میزان رنگ مربوطه را مشخص می کند. کدام از بایتها می توانند 1 بیت را در خود حمل کنند. به عنوان مثال، اگر کد رنگ برابر 155 باشد، کم ارزش ترین بیت برابر 1 است. اگر ما بخواهیم بیت 0 را در این رنگ جایگذاری کنیم، باید کم ارزشترین بیت را برابر 0 قرار دهیم. لذا کد رنگ به 154 تغییر می کند. اما اگر بیت مورد نظر ما برای کدگذاری 1 باشد، تغییری در کد این رنگ نمی دهیم.

در حالت کلی، ما فایل متنی را به صورت کاراکتر به کاراکتر می خوانیم. هر کاراکتر شامل 8 بیت است و هر کدام از بیتها در یک کد رنگ قرار می گیرند. لذا، برای کدگذاری سه کاراکتر به هشت پیکسل رنگی نیاز مند هستیم.

پروژه ی ششم: بازی سودوکو

در فایل <sudoku.txt> پنجاه پازل مختلف قرار دارند که از آسان به سخت مرتب شده اند. اعداد 0 در این فایل نشانگر جاهای خالی هستند که باید برنامه شما آنرا پیدا کند. برنامه‌ای بنویسید که یکی از این پازلها را انتخاب و نمایش دهد و سپس آنرا حل کند.

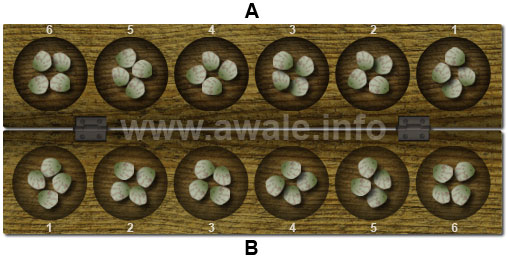
پروژه ی هفتم: فیلتر بلوم

در این پروژه فیلتر بلوم را پیاده می‌کنید. فیلتر بلوم به درد پایگاه داده‌های خیلی بزرگ می‌خورد و مشخص می‌کند که آیا داده‌ی مورد نظر در پایگاه داده موجود است یا نه. اما مشکلی که دارد این است که ممکن است فیلتر اعلام کند یک داده در پایگاه داده موجود است در حالی که این گونه نیست. البته بر عکس آن امکان پذیر نیست. یعنی، اگر فیلتر بگوید که داده در پایگاه داده موجود نیست قطعاً درست خواهد بود. نکته‌ی بعدی این که می توانید دقت فیلتر را کنترل کنید.

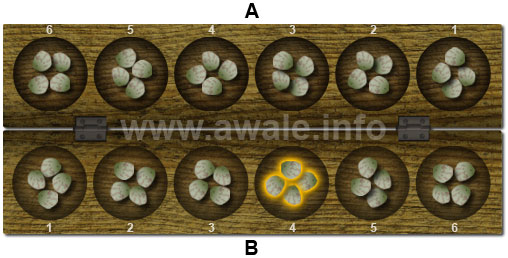
فیلتر بلوم دارای یک آرایه از بیتها است که در ابتدا همه‌ی آنها برابر صفر هستند. هر بار که یک داده را وارد پایگاه داده کنیم، داده را به یک تابع درهم‌سازی Hash می‌دهیم تا یک عدد تولید شود. حال، تمام بیتهای متناظر با عدد در آرایه را برابر “1” تنظیم می‌کنیم. اگر بخواهیم وجود یک داده در پایگاه داده را چک کنیم بدین صورت عمل می کنیم که Hash داده را محاسبه می‌کنیم. اگر بیتهای متناظر در آرایه تنظیم شده بودند به احتمال خیلی زیاد داده در پایگاه داده موجود است. اگر هنگام ورود داده به پایگاه داده، Hash آن طوری باشد که تمام مکانهای متناظر قبلاً تنظیم شده باشند می گوییم که برخورد رخ داده است. مشکل عدم دقت فیلتر بلوم همین امکان وقوع برخورد است. کلمات وارد شده در پایگاه داده از نوع رشته هستند. لطفاً برای مشخص کردن تابع Hash به اینجانب مراجعه کنید.

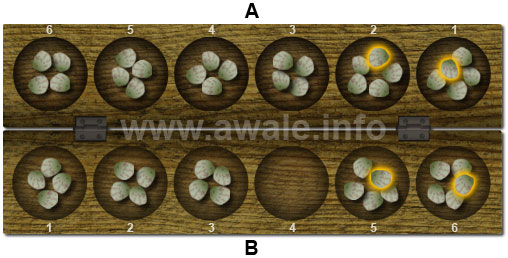
پروژه ی هشتم: بازی آواری

بازی آواری یکی از بازیهای بومیان آفریقا است که از طریق سنگ و سطل انجام می‌شود. این بازی شامل ۱۲ سطل و ۴۸ سنگ است. سطلهای ۱ تا ۶ مال بازیکن A و سطلهای 7 تا ۱۲ مال بازیکن B است. هدف این بازی این است که سنگهای بیشتری جمع‌آوری شود. اگر بازیکنی بیشتر از ۲۴ سنگ جمع کند می‌تواند پیشنهاد خاتمه ی بازی را بدهد. یا می توان بازی را تا وقتی که نتوان سنگ دیگری را خارج کرد ادامه داد. در ابتدا هر کدام از سطلها دارای ۴ سنگ است.



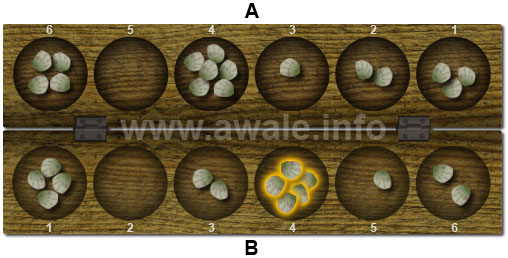
قانون اول: هر بار که نوبت به یک بازیگر رسید، آن بازیگر یکی از خانه های خود را انتخاب می کند و سنگهای آنرا بر می‌دارد و آنها در جهت خلاف حرکت عقربه های ساعت در سطلها بخش می کند. به هر سطل فقط یک سنگ اضافه می‌شود و در صورتی که سطلهای خودمان تمام شود از سطلهای حریف استفاده می کنیم. به شکل زیر دقت کنید:

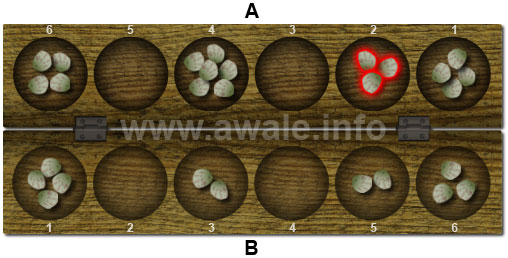
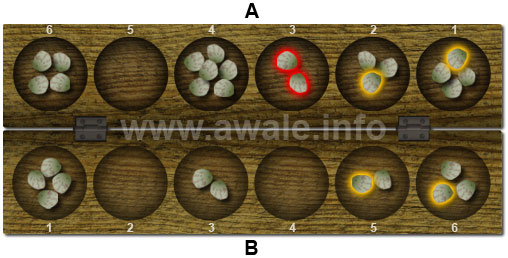




در اینجا، بازیگر B سطل ۴ را انتخاب و سنگهای آنرا پخش کرده است.

قانون دوم: پس از این که بازیگر سنگهای خود را پخش نمود، اگر آخرین خانه ای که به آن سنگ اضافه کرده است متعلق به حریف بود و تعداد سنگهای ۲ یا ۳ بود، آن سنگها را برای خود بر می‌دارد. سپس، به خانه ی قبلی نگاه می‌کند. اگر تعداد سنگهای این سطل هم ۲ یا سه بود آنها را بر می‌دارد. اینکار را آنقدر تکرار می کنیم که به خانه ای برسیم که تعداد سنگهای ۲ یا ۳ نباشد. به شکل زیر دقت کنید:





قانون سوم: اگر سطلی که سنگهای آنرا برداشته‌اید دوازده سنگ یا بیشتر داشته باشد، در اینصورت می‌توان یک دور در تمام خانه ها سنگ قرار داد. اما حق ندارید سنگ دوازدهم را در سطل خالی شده بگذارید و از آن سطل صرفنظر کنید.

قانون چهارم: حق ندارید در یک حرکت تمام سطلهای حریف را خالی کنید. انجام اینکار به منزله ی قبول شکست از طرف شما است.

قانون پنجم: اگر ادامه ی بازی عاقلانه نباشد یا امکان برداشتن سنگ دیگری نباشد، دو طرف می‌توانند بازی را خاتمه دهند و کسی که سنگ بیشتری جمع کرده است برنده است.