Michał Gleba, Grzegorz Godycki Ćwirko, Maryia Mankevich, Joanna Raczek

Komputer w labiryncie

Programiści piszą programy, które potrafią robić wiele różnych rzeczy: odtwarzać filmy, prognozować pogodę, pomagać w nauce języków obcych czy matematyki. Ale czy wiesz, że można zaprogramować komputer tak, aby tworzył labirynty? W dodatku takie, które zawierają tajne informacje!

10+

DOWIESZ SIĘ

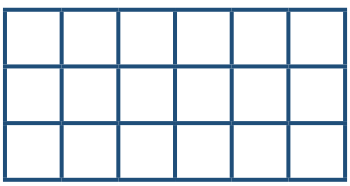
- Co to jest labirynt doskonały.
- W jaki sposób generować labirynty z pomocą komputera.
- Co to jest steganografia.

POTRZEBNA WIEDZA

- Co to jest labirynt 🙂.
- Podstawy matematyki.

Według mitologii greckiej, w czasach, gdy nie było jeszcze komputerów, pewien słynny wynalazca Dedal zaprojektował labirynt, aby ukryć w nim potwora Minotaura. Nie było to łatwe zadanie, ponieważ taki labirynt musiał być odpowiednio duży i skomplikowany. Na szczęście w dzisiejszych czasach może nam w tym pomóc komputer. Co prawda nie potrzebujemy już labiryntów do ukrycia Minotaura, ale przydają się one w tworzeniu gier komputerowych i ukrywaniu tajnych informacji.

W jaki sposób komputer tworzy labirynty i ukrywa w nich tajne informacje? Istnieje wiele sposobów: jedne są lepsze z punktu widzenia programisty, a inne gorsze. Przypuśćmy, że chcemy policzyć, z ilu kostek składa się czekolada, na przykład taka jak na Ilustracji 1.



Ilustracja 1. Schematyczny rysunek czekolady

Możemy to zrobić, zliczając kawałki po kolei: 1, 2, 3, … , 10, 11, 12, … , 17, 18. Jednak znamy szybszy sposób: wystarczy policzyć, ile kostek jest w jednym rzędzie, a następnie pomnożyć wynik przez liczbę rzędów. Oba sposoby dają dobrą odpowiedź, ale drugi jest szybszy. Najważniejszą cechą algorytmów jest czas, w jakim dochodzą one do oczekiwanego rozwiązania: im jest on krótszy, tym lepiej. Programiści wolą efektywne algorytmy, czyli takie, które znajdują rozwiązanie szybko, co jest korzystne zarówno dla ludzi (nie chcemy długo czekać na wynik), jak i ze względu na komputery (mniejsze zużycie energii).

Pokażemy przykładowo jeden z prostych i efektywnych sposobów, w jaki komputer może stworzyć labirynt, znany jako algorytm Ellera. Na początku nasz labirynt to rząd komórek, jak na Ilustracji 2. Zauważ, że każda komórka ma swój numer.

1	2	3	4	5	6

Ilustracja 2. Początek

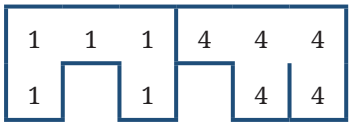
Następnie komputer losowo łączy niektóre sąsiednie komórki poprzez usunięcie pionowej ściany. Pionową ścianę można usunąć tylko wtedy, gdy po obu jej stronach znajdują się różne liczby (Ilustracja 3).

1	1	1	4	4	4

Ilustracja 3. Pierwszy wiersz

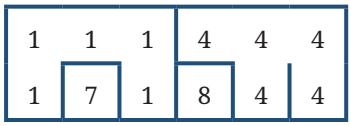
Gdy usuwamy pionową ścianę pomiędzy dwoma sąsiednimi komórkami w jednym wierszu, to te komórki należą teraz do jednego obszaru. Chcemy, aby miejsca należące do jednego obszaru były tak samo oznaczone, więc niektóre komórki muszą zmienić numer.

W kolejnym kroku każdy z obszarów musi urosnąć o co najmniej jedną komórkę w dół. W naszym przykładzie obszar jedynek urósł o dwie komórki, podobnie obszar czwórek. Zauważ, że gdy obszary rosną w dół, nowe komórki otoczone są trzema ściankami: wszystkimi oprócz górnej (Ilustracja 4).



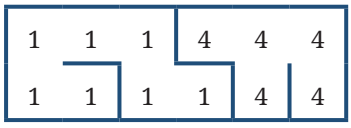
Ilustracja 4. Pierwszy wiersz rozrasta się

W pustych miejscach w nowym wierszu komputer tworzy nowe komórki, każda otoczona czterema ściankami. Nie są połączone z żadnym dotychczasowym obszarem, dlatego otrzymują własne numery (Ilustracja 5).



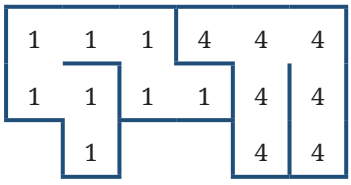
Ilustracja 5. Nowe komórki w wierszu drugim

W drugim wierszu komputer znów losowo usuwa niektóre pionowe ścianki pomiędzy różnymi obszarami, na przykład tak jak na Ilustracji 6.



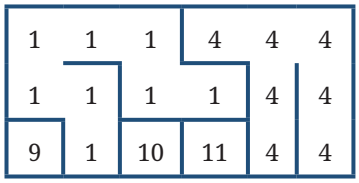
Ilustracja 6. Usunięte ścianki w wierszu drugim

Znów każdy z obszarów rośnie o co najmniej jedną komórkę w dół (Ilustracja 7).



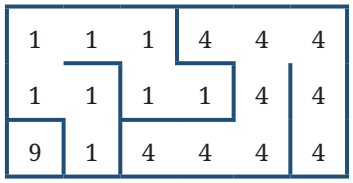
Ilustracja 7. Drugi wiersz rozrasta się

Tworzą się nowe komórki (Ilustracja 8).



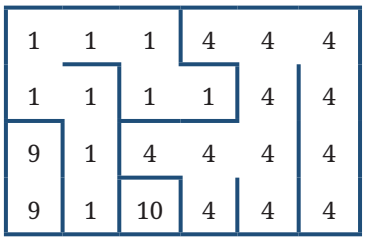
Ilustracja 8. Nowe komórki w wierszu trzecim

Niektóre losowo ulegają scaleniu (Ilustracja 9).



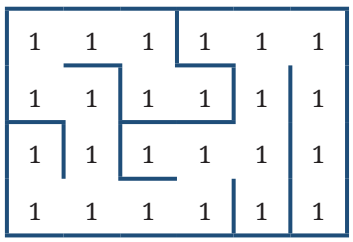
Ilustracja 9. Usunięte ścianki w wierszu trzecim

Obszary rozrastają się (Ilustracja 10).



Ilustracja 10. Trzeci wiersz rozrasta się

Algorytm mógłby wymyślić dowolnie dużo kolejnych wierszy labiryntu, ale my będziemy kończyć. To będzie nasz ostatni rząd. Musimy usunąć w nim ściany, które po obu stronach mają inne liczby. W ten sposób w całym labiryncie będzie na koniec tylko jedna liczba (Ilustracja 11).



Ilustracja 11. Końcowy efekt działania algorytmu

Taki sposób rozwiązywania problemu przez komputer nazywamy *algorytmem iteracyjnym*. Oznacza to, że komputer powtarza te same kroki określoną ilość razy (aż do spełnienia pewnego warunku – w naszym przypadku jest to wygenerowanie całego labiryntu o zadanej wielkości).

Takie małe labirynty zapewne moglibyśmy tworzyć bez pomocy komputera. Jednak co w przypadku, gdyby taki labirynt składał się z dziesiątek czy setek wierszy? Z pewnością nie byłoby to łatwe zadanie dla człowieka, natomiast dla komputera jest to kwestia tylko kilku sekund.

Generowanie takich labiryntów można zobaczyć w Internecie tutaj:

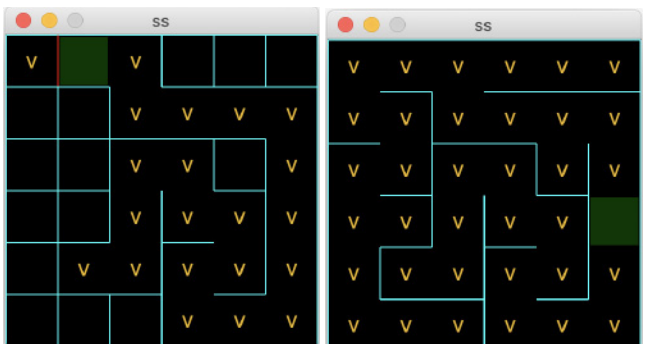
Link do repozytorium:

» https://github.com/mazedemo/labirynty

Dodatkowo, jeśli masz możliwość uruchomienia programu w języku Python, to pod adresem podanym wyżej znajdziesz również programy: *Maze* i *Maze2*.

Program *Maze* pozwala na tworzenie labiryntów w opisany wcześniej sposób o dowolnych wymiarach. W ten sposób możesz na żywo zobaczyć, jak komputer generuje taki labirynt.

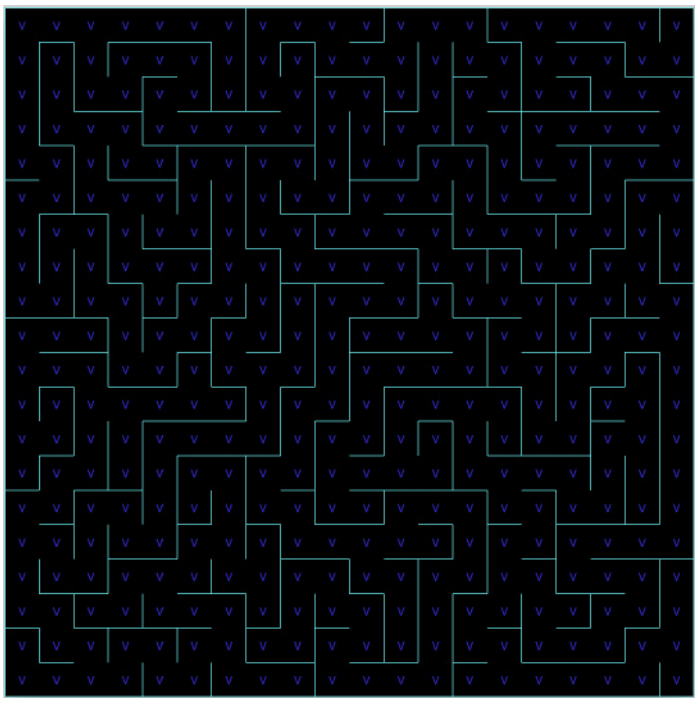
Opisany sposób generowania nie jest jedyny. Istnieje wiele innych sposobów na wykonanie tego zadania. Takim przykładem jest na przykład algorytm o nazwie *Recursive Backtracker*. Jest to prosty algorytm rekursywny (czyli dzielący problem na mniejsze części i wywołujący sam siebie dla tych mniejszych części). Sposób działania został przedstawiony na Ilustracji 12. Przypomina on nieco Pac Mana, który stopniowo „zjada" nieodwiedzone miejsca na planszy, przy okazji generując labirynt.



Ilustracja 12. Budowanie labiryntu przez algorytm „Recursive Backtracker"

'en sposób możesz zobaczyć w katalogu *Maze2* we wspo-ınianym linku.

Możesz spróbować rozwiązać labirynt wygenerowany przez ten algorytm. Zauważ, że sam/a możesz określić, gdzie ma być wejście, a gdzie wyjście – Ilustracja 13.



Ilustracja 13. Labirynt doskonały wygenerowany przez komputer

Wspomnieliśmy wcześniej o tajemniczym terminie „steganografia". Otóż polega ona na ukrywaniu informacji w plikach komputerowych (ale nie tylko), na przykład w plikach obrazu, ale ukrytą wiadomość może też zawierać wymyślony przez komputer labirynt. Taki labirynt ma rozwiązanie, które nie jest przypadkowe. Ścieżka jest tak wyznaczona, że kierunek kolejnych jej punktów ma pewne znaczenie – zawiera bit informacji. Jak zapewne wiesz, komputer posługuje się językiem zer i jedynek. Przy użyciu tych dwóch liczb może zakodować dowolną informację. Możemy w ten sposób stworzyć labirynt, w którym idąc ścieżką z jednego punktu do drugiego, brak ściany po lewej stronie komórki ozna-

CIEKAWOSTKA

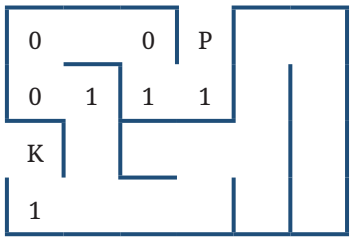
Większość labiryntów, które znamy z gier bądź czasopism, to tak zwane „labirynty doskonałe". Wyróżniają się one tym, że z jednego miejsca w labiryncie możemy się dostać do drugiego tylko jedną ścieżką.

WARTO WIEDZIEĆ

Steganografia jest dziedziną sztuki i nauki umieszczania ukrytych wiadomości w taki sposób, że nikt – poza nadawcą i odbiorcą zamierzonym – nie wie o ich istnieniu.

cza 0, a brak ściany po prawej stronie komórki oznacza 1 (jeśli żadnej ściany nie brakuje lub brakuje jednocześnie ściany prawej i lewej, to taka komórka nie ukrywa informacji). Podążając taką ścieżką (i znając sposób ukrycia informacji), można odczytać zakodowaną informację. Dla komputera będzie to ciąg zer i jedynek, natomiast my możemy zamienić go na litery, na przykład przy użyciu kodu ASCII (zobacz https://pl.wikipedia.org/wiki/ASCII). W ten sposób dla obserwatora z zewnątrz będzie to najzwyklejszy labirynt, który można rozwiązać, ale dla osób wtajemniczonych ścieżka rozwiązania będzie zawierała tajną informację.

Na przykład, ścieżka wyznaczona w labiryncie widocznym na Ilustracji 14, wiodąca od komórki początkowej oznaczonej literą P do komórki końcowej K, koduje ciąg 1100011, w kodzie ASCII oznaczający literę „c”.



Ilustracja 14. Ukryty ciąg: 1100011 (litera „c” w kodzie ASCII)

Michał Gleba

Na co dzień studiuje informatykę na Politechnice Gdańskiej, w wolnym czasie lubi tworzyć gry w środowisku Unity oraz grać w szachy.

MICHALGLEBA1@GMAIL.COM

Maryia Mankevich

Studentka studiów magisterskich na kierunku informatyka na Politechnice Gdańskiej. Interesuje się sztuczną inteligencją, algorytmami oraz złożonością obliczeniową. Zdobyła brązowy medal na Międzynarodowej Olimpiadzie Matematycznej.

MANKEWI4.MASHA@GMAIL.COM

Grzegorz Godycki Ćwirko

Jest programistą aplikacji iOS, a także pasjonatem nowych technologii. W szczególności zainteresowany tematyką z elektromobilności, a zawodowo tematami dotyczącymi zarządzaniem jakością kodu w języku Swift.

GODYCKI@NTHSOFTWARE.PL

Joanna Raczek

Pracuje ze studentami Politechniki Gdańskiej, zajmuje się algorytmami i matematyką dyskretną.

JOANNA.RACZEK@PG.EDU.PL

ZAPAMIĘTAJ

- Czym wyróżnia się labirynt doskonały.
- Przykładowe rodzaje algorytmów to algorytmy iteracyjne (powtarzają te same kroki) oraz rekursywne (wywołujące same siebie).
- Na czym polega steganografia.

ĆWICZ W DOMU

- Algorytmu generowania labiryntów możesz użyć, aby utworzyć planszę – labirynt na potrzeby programowania w języku Scratch.
- Gdy twoje umiejętności programowania z biegiem czasu będą coraz lepsze, możesz wykorzystać taki algorytm do tworzenia własnych gier planszowych.