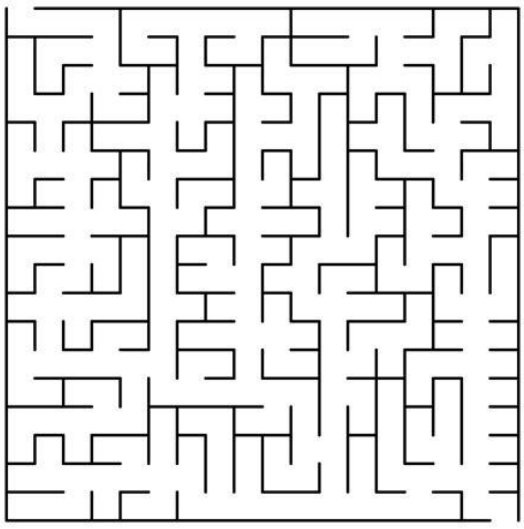
Generarea de labirinte

Introducere

De mii de ani oamenii au fost fascinati de labirinte: au construit structuri asemanatoare, au creat si spus legende cu ele, au facut jocuri care au la baza labirinte si chiar studiaza comportamentul animalelor cu ajutorul lor.

Exista doua tipuri principale de labirinte: cele care au fundatura, si cele unidirectionale. In limba romana nu exista o metoda de a le diferentia direct din numele lor (in engleza exista maze – cu fundatura si labyrinth – unidirectional). Cele ce nu au bifurcatii sunt folosite in general pentru relaxare, meditatie si spiritualitate.



Labirint cu fundatura (Maze)

Labirint unidirectional (Labyrinth)

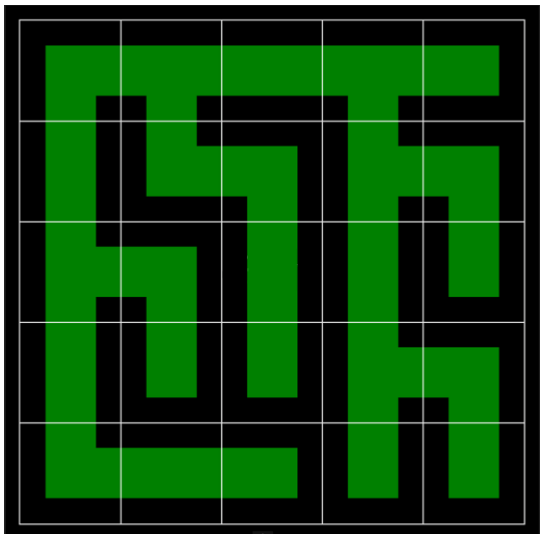
In mitologia greaca, primul labirint a fost construit de arhitectul Dedal si fiul sau Icar pentru a inchide fiorosul Minotaur: o creatura cu corp de om si cap de taur. Legenda spune ca Dedal a creat labirintul intr-o maniera extrem de vicleana, astfel incat nici el, odata ce a fost inchis in propria creatie impreuna cu fiul sau, nu mai stia exact cum sa evadeze.

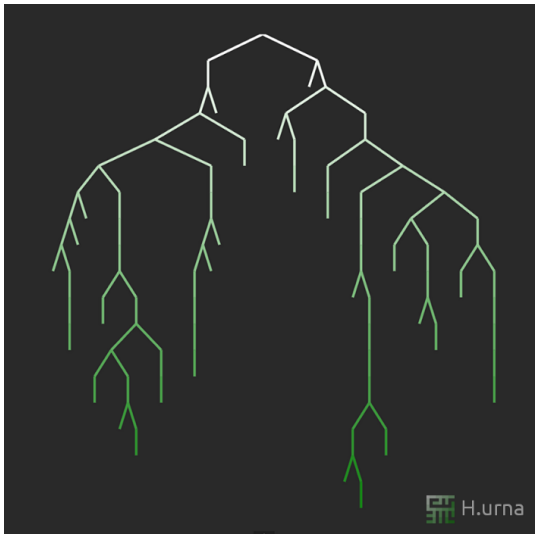
Este interesanta relatia pe care omul o are cu labirintele. Inca de mici copiii au o legatura adanca cu ele, stiind aproape instinctiv ce trebuie facut pentru a le rezolva. Cu ajutorul acestora isi dezvolata capacitatea de a lua o decizie, orientarea, crearea unui plan, memorizarea si in acelasi timp se relaxeaza.

Algoritmi de generare

1. Binary Tree

Este unul dintre putinii algoritmi care poate genera un labirint perfect fara a pastra nici o stare: este un algoritm de generare a labirintului ce nu necesita memorie suplimentara si fără limită la dimensiunea labirintului pe care îl poti genera. Poate construi întregul labirint privind fiecare celulă independent. Acesta este cel mai simplu și mai rapid algoritm posibil.



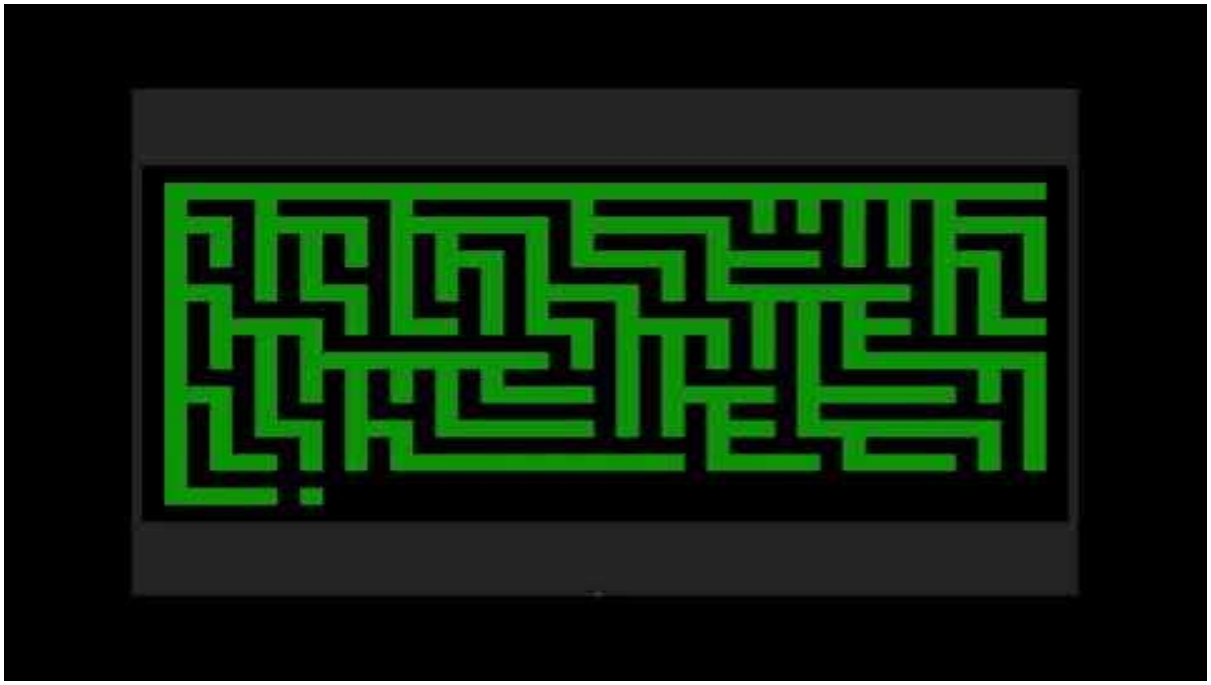


Dupa cum sugereaza si numele, fiecare celula este tratata individual si pentru fiecare decidem daca ne legam de celula din stanga sau de cea de deasupra. Decizia este facuta aleator. In prima figura de mai sus celule sunt parcurse de la stanga la dreapta, de sus in jos.

Daca luam coltul din stanga sus al labirintului si il desemnam ca radacina putem reprezenta structura creata ca un arbore binar. A doua figura de mai sus este pentru un exemplu mai amplu dar idea este aceeasi.

Nu orice arbore binar poate sa fie reprezentat ca un labirint (deoarece exista posibilitatea suprapunerii de noduri si astfel crearea de incaperi fara intrare/iesire), dar orice labirint creat cu acest algoritm poate fi reprezentat ca un arbore binar.

Vizualizarea algoritmului: Maze Generation - Binary Tree



2. Depth First Search (DFS)

Acest algoritm este o metoda random al algoritmului de parcurgere DFS a arborilor. Algoritmul incepe in orice celula si exploreaza cat de mult posibil si dupa se intoarce prin backtracking.

Pentru a avea rezultate cat mai atractive trebuie sa decidem aleator in care parte ne vom mergem, atata timp cat celula in care ne mutam este valida. Odata ce nici un vecin nu mai este valid pentru a progresa (ca si cum am ajuns la o fundatura), ne intoarcem la cea mai recenta celula din stack si aplicam algoritmul in continuare.

Ca si observatie: dupa ce calculam vecinii unei celule si ii validam, ii adaugam in stack si in acelasi timp ii legam de nodul curent (ca si cum i-am fi “vizitat” deja).

Vizualizarea algoritmului: Maze Generation - Depth First Search (DFS) with flooding

3. Kruskal

Se bazeaza pe o varianta random a algoritmului lui Kruskal (paduri de multimi disjuncte). Algoritmul original a lui Kruskal nu “creste” un arbore ci creaza grupari mai mici de arbori care pe urma le leaga facand arbori tot mai mari. Legarea se face folosind cea mai scurta muchie posibila.

Aceasta versiune a algorimului pune muchii ale labirintului random, atata timp cat muchia nu uneste doua subseturi de ziduri ce sunt unite deja. In cele din urma rezultatul este tot un labirint perfect.

Ca si observatie: datorita alegerii aleatoare a muchiilor, acestea trebuie sa fie stocate undeva in prealabil pentru a putea fi alese.

Vizualizare algoritm: Maze Generation - Kruskal's with flooding

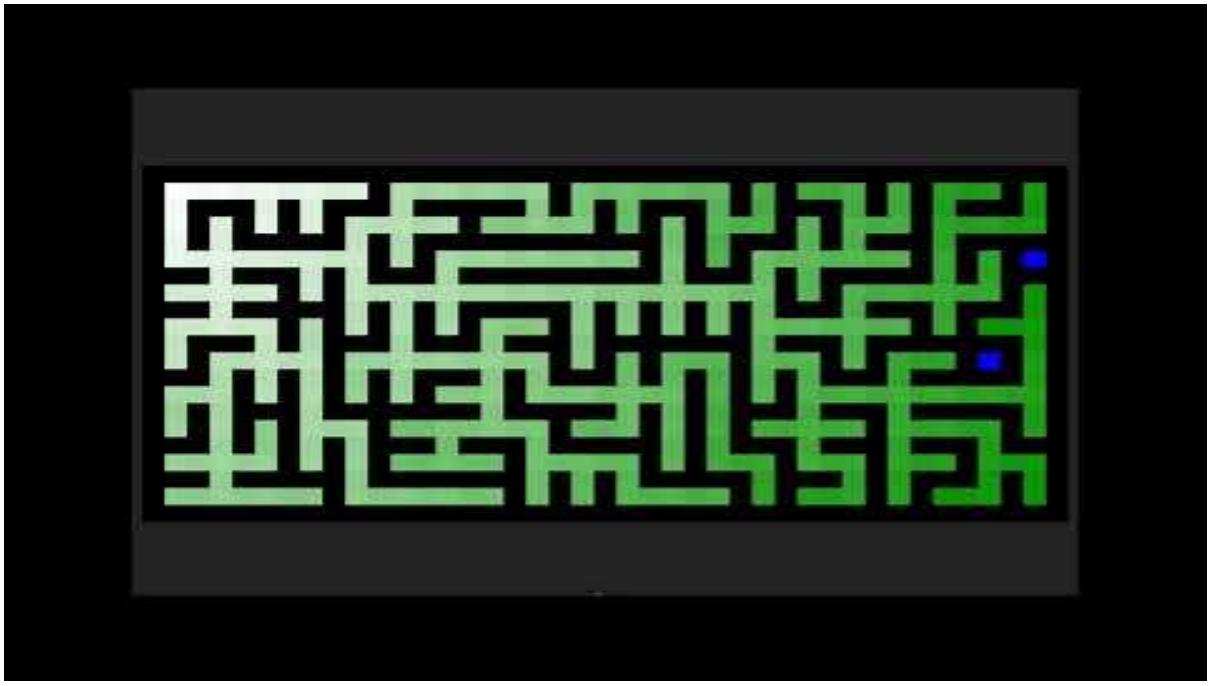
4. Prim

Si acest algoritm se bazeaza pe o modificare a uni algoritm deja existent, si anume algoritmul lui Prim. Algoritmul original a lui Prim calcula care este cel mai bun drum dintr-un punct comparand muchiile care pleaca din acesta.

In aceasta versiune, neavand muchii, putem sa alegem un vecin random pe care sa il lipim de labirint. Pozitia curenta nu conteaza asa ca putem sa avem o lista de vecini posibili, pe care o parcurgem si pentru fiecare decidem daca il lipim sau nu. Exista mai multe metode de validare a algorimului din punct de vedere al vecinilor: putem tine minte din ce celula am venit si il putem lega direct, putem pentru fiecare element din lista de vecini sa decidem de care perete il lipim sau o combinatie dintre cele doua cu cateva validari in plus.

Ca si observatie: o sa fie necesara stocarea vecinilor adiacenti intr-o structura de date suplimentara.

Vizualizarea algoritmului: Maze Generation - Prim's with flooding



5. Recursive division

Aceasta metoda de generare poate fi asemanata cu un algoritm de tipu “divide et impera” datorita naturii ei. In teorie algoritmul ar putea continua la infinit daca zidurile pe care le plasam devin din ce in ce mai mici, astfel labirintul devenind un fractal.

Cum functioneaza algoritmul: labirintul este impartit in doua prin plasarea unui zid, se creaza un drum aleator prin zidul creat, algoritmul se continua pe unul dintre cele doua zone delimitate de zid, cand nu se mai pot crea zone se intoarce prin backtracking la urmatoarea zona accesibila.

Idea de baza este impartirea sarcinii initiale in sarcini mai mici, de aici si analogia cu algoritmii de tipul divide et impera.

Vizualizarea algoritmului: Maze Generation - Recursive Division with flooding

6. Sidewinder

Seamana cu Binary Tree Algorithm prezentat, doar cu cateva modificari. Spre deosebire de Binary Tree, Sidewinder tine cont doar de randul curent din labirint, nu de celula, si prin urmare, precum algoritmul cu care se aseamana, poate fi folosit pentru generarea labiritelor de lungime infinita.

O alta deosebire intre cele doua este ca aceasta metoda de generare construieste un singur drum lung care sufera modificari pe parcurs, pe cand in Binary Rree se creaza mai multe drumuri.

Algoritmul parcurge fiecare linie din labirint, celula cu celula, si decide aleator pentru fiecare in parte daca se uneste cu partea cu celula din dreapta. Daca unirea a avut loc, celula curent devie cea cu care ne-am unit. Daca nu a avut loc legatura, alegem o celula din cele pe care le-am parcurs pe linia curenta si o lipim de celula de deasupra ei.

Vizualizarea algoritmului: [Maze Generation - Sidewinder with flooding]