Implementarea algoritmului in Javascript

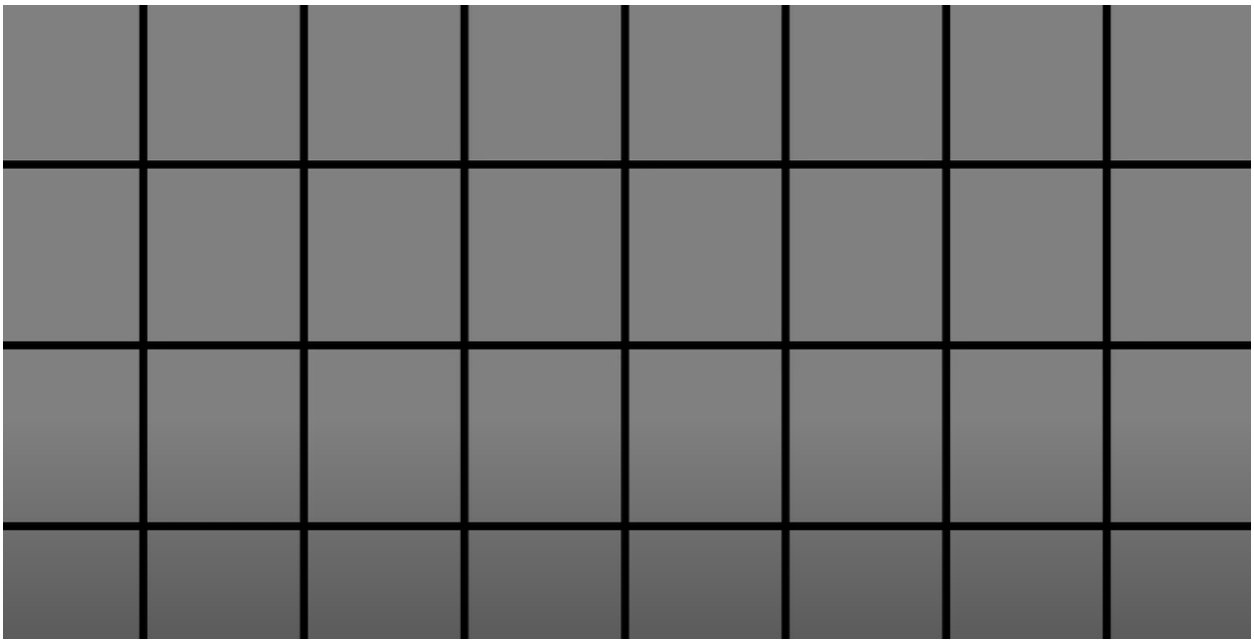
O sa incep prin a vorbii despre cautarea in adancime (Depth-first search). Algorimul de cautare in adancime este o modalitate de a parcurge graful de noduri adesea asociat cu traversarea a unui arbore. Exemplu pe care il gasiti in cartile de informatica, ar fi traversarea unui arbore binar sau ceva de genul. Algoritmul de baza incepe de la un nod marcat drept vizitat si apoi pentru fiecare vecin conectat la acest nod apeleaza recursiv aceasta functie.

Pentru soarecele nostru din labirint o sa folosesc cautarea in adancime cu algoritm recursiv de urmarire inversa (Depth-first search with recursiv backtracking). Veti observa imediat ca este aproape identic cu ce am prezentat mai sus, deoarce se bazeaza foarte mult pe cautarea in adancime. Astfel incat incep cu un nod vizitat sau marcat si apoi pentru fiecare vecin care nu a fost vizitat inca, aleg unul la intamplare, asta daca nu exista un perete generat intre nodul parinte si cel care urmeaza sa fie vizitat. Fata de cautarea in adancime sunt cateva diferente, dar o sa va prezint si codul imediat.

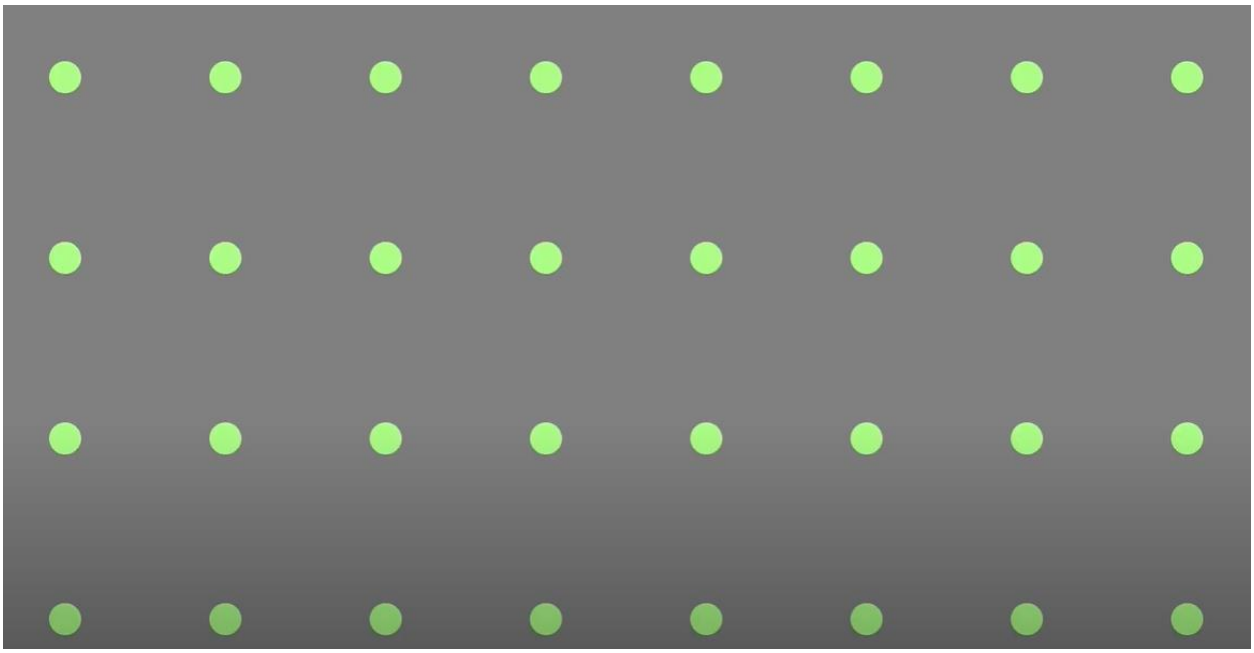
```
_ProcessNode(nodeKey) {
  this._visited[nodeKey] = true;

  const node = this._nodes[nodeKey];

  const neighbours = [...node.potentialEdges];
  while (neighbours.length > 0) {
    const ki = RouletteSelect(neighbours);

    if (!(ki in this._visited)) {
      const adjNode = this._nodes[ki];

      node.edges.push(ki);
      adjNode.edges.push(nodeKey);
      this._ProcessNode(ki);
    }
  }
}
```

Am facut aceasta aplicatie de vizualizare a generari labirintului, prin care putem parcurge si vedea cum labirintul s-a generat pas cu pas.

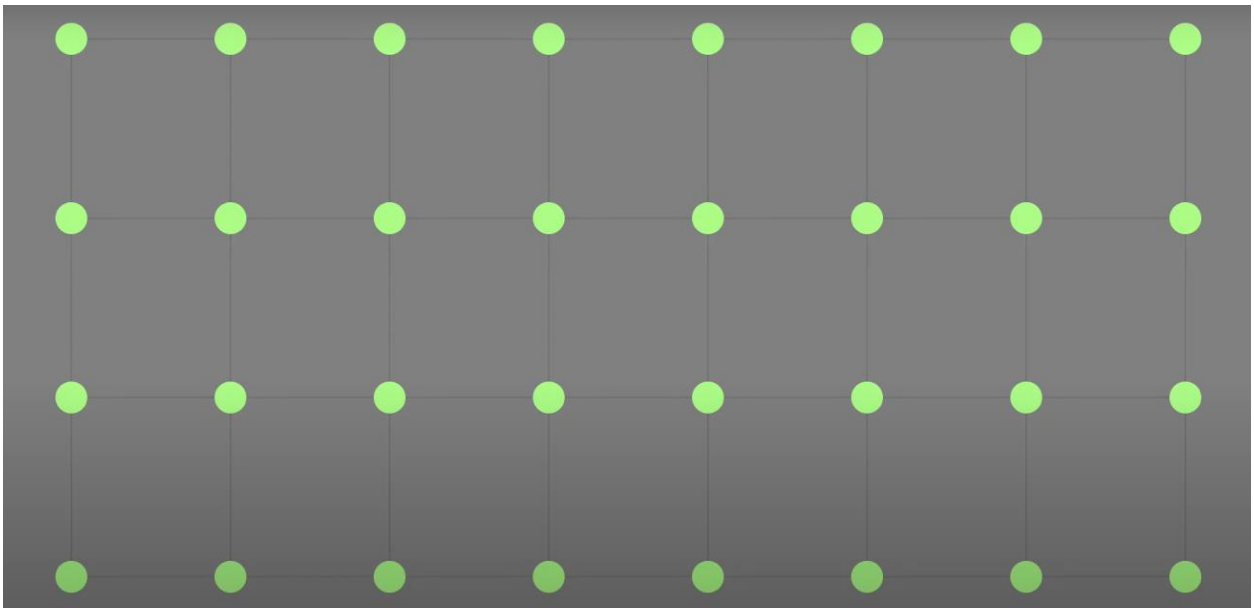
Avem 4 celule pe verticala si 8 pe orizontala. Fiecare celula are pereti in jurul sau in acest punct.



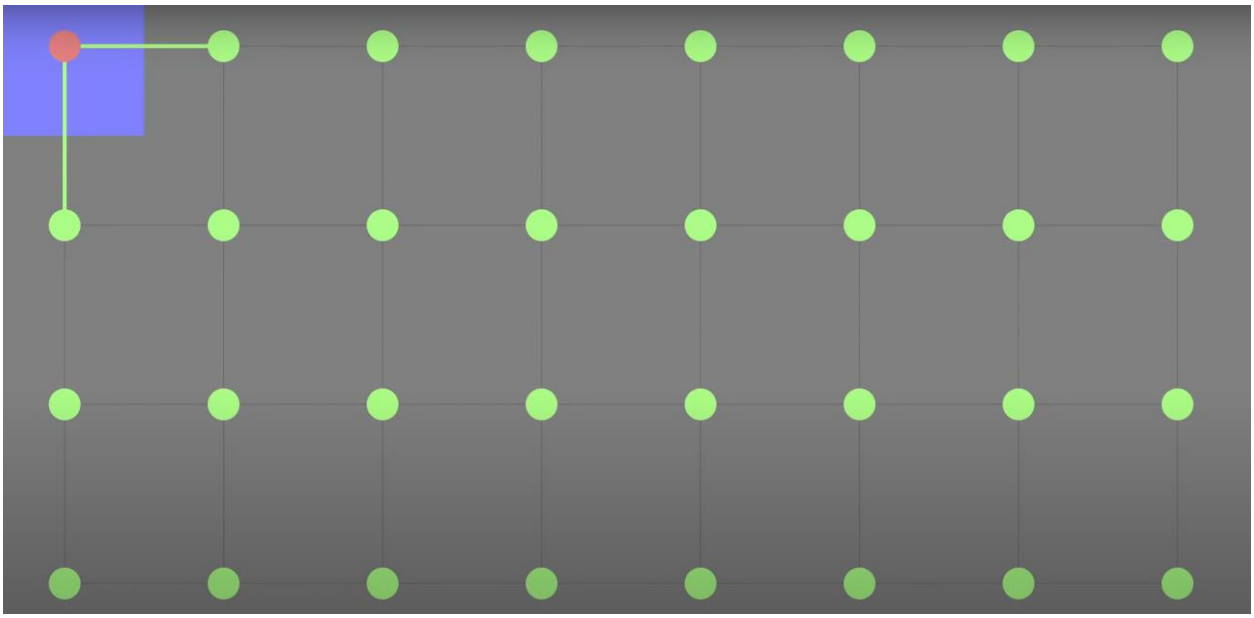
Apoi o sa anulez peretii dintre celule, pentru a se genera marginile dintre celule.



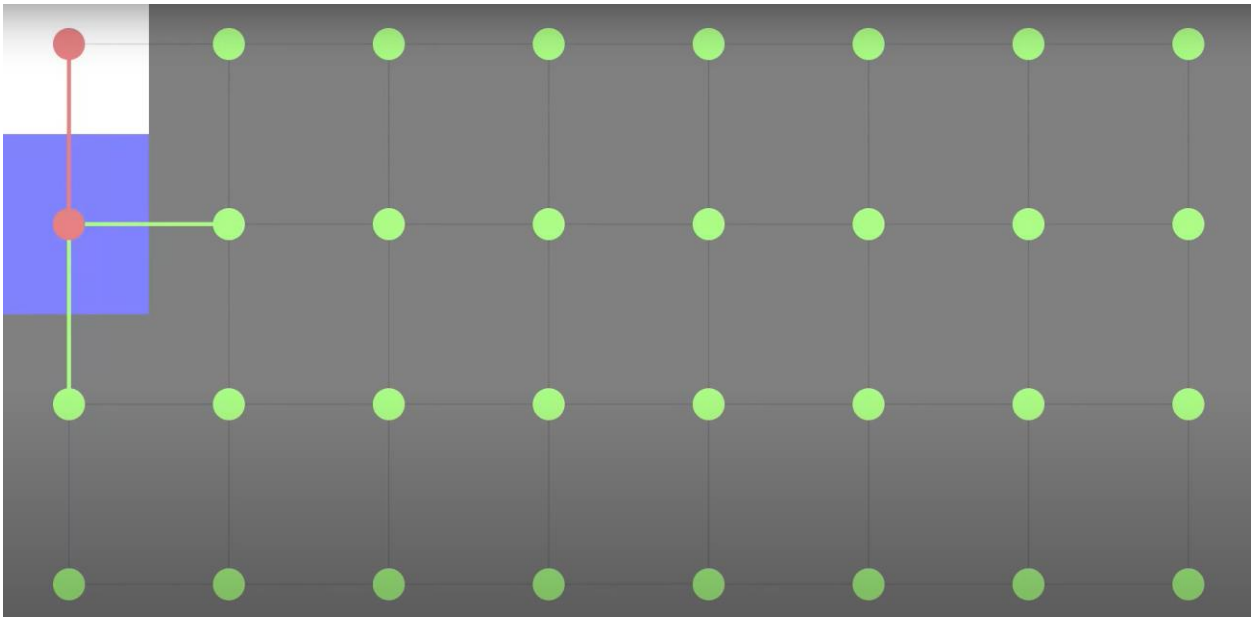
In punctul acesta avem cate un nod in fiecare celula, deci automat avem si toate marginile potentiale vizibile. Aceste marigni nu sunt reale le-am facut doar pentru a vizualiza si intelege mai bine codul. Acum sunt vizualizati doar posibilii vecini pentru fiecare celula si pe masura ce ii parcurgem, o sa completam mariginile si o sa vedeti cum acestea se leaga de pereti.



Asa ca o sa incep prin a preciza ca patratul albastru reprezinta nodul vizitat in acest moment. Cercul rosu inseamna ca acel nod a fost deja vizitat, iar linile verzi care leaga cercurile intre ele repezinta posibilii vecini are o sa fie vizitati. Deci dupa cum inaintam in labirint se alege un vecin are nu a fost vizitat.

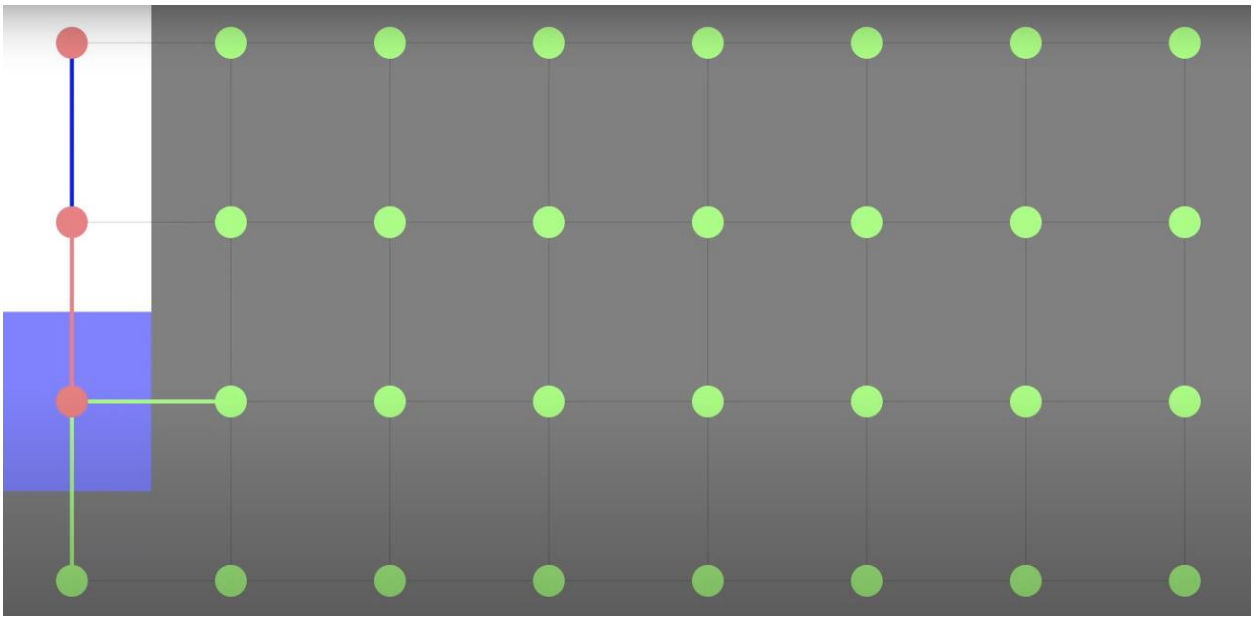


Puteti vedea ca celula anterioara pe care am fost devine rosie, ceea ce inseama ca a fost vizitata si nu ne mai putem intoarce acolo, si linia dintre ele este de asemenea rosie pentru noua pozitie. Avem trei celule care se conecteaza, cea de sus care a fost citita, cea de sub si cea din dreapta.

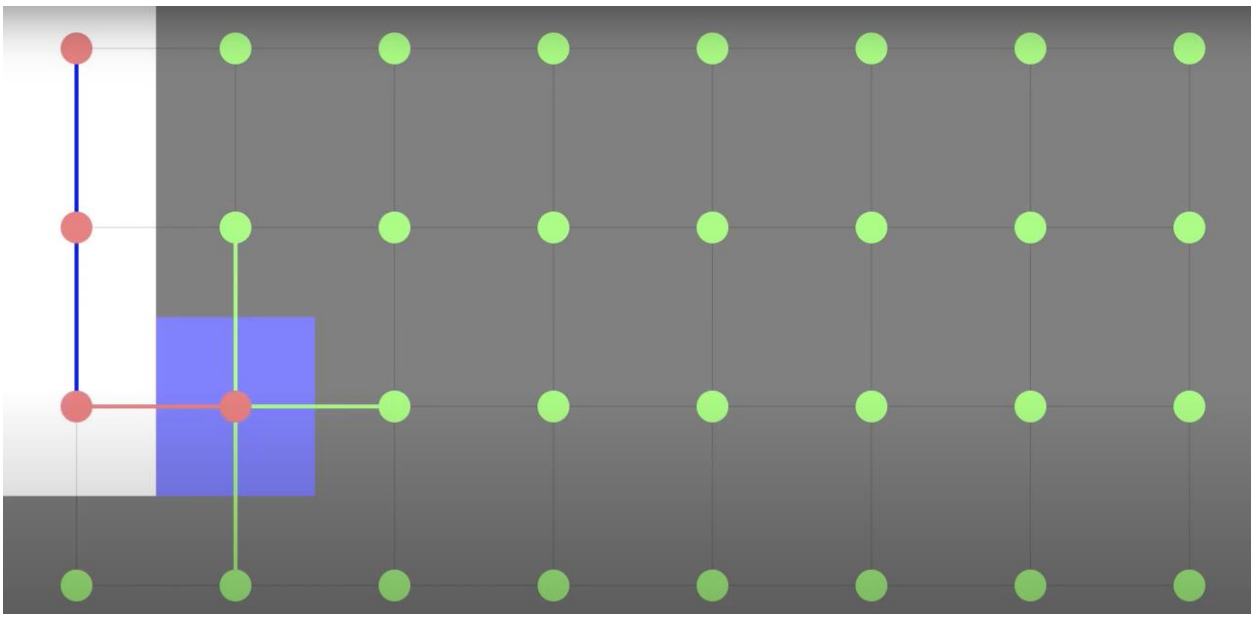


Tot asa o sa avansam.

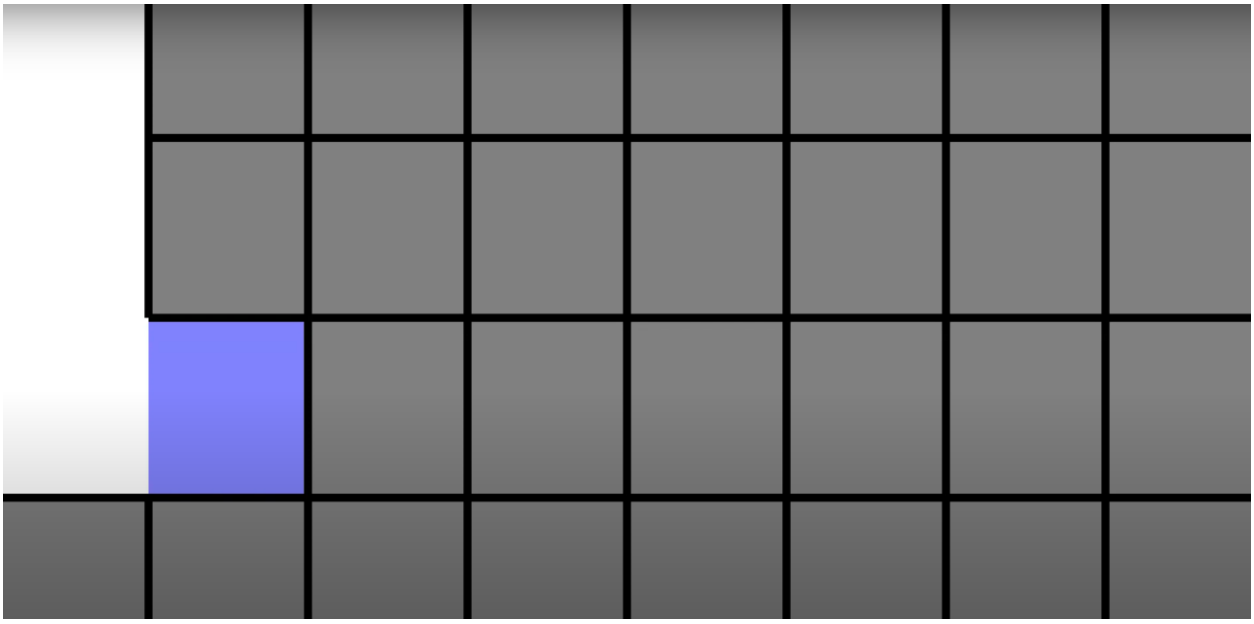
Acum am colorat liniile dintre nodurile rosii, si asta pentru a indica ca exista o margine intre ele. La celula activa actuala, cea din patratul albastru, celul de deasupra are un nod rosu, iar celula din dreapta si cea de jos au ambele linii verzi intre ele. Deci putem merge in dreapta sau in jos.



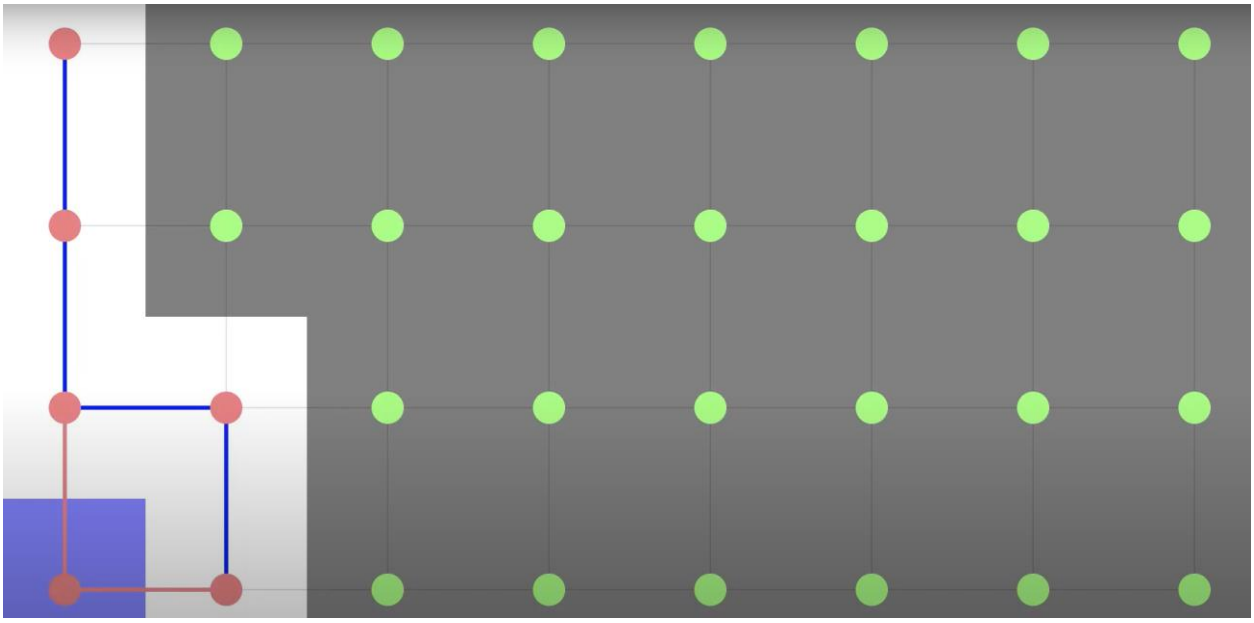
A mers la dreapta ceea ce a devenit noua pozitie. Avem 4 optiuni acum pe care le putem folosi, la stanga, dreapta, sus, jos, cu exceptia faptului ca nodul din stanga a fost deja vizitat, astfel incat acesta este blocat.



Acum o sa ascund din nou nodurile si voi arata peretii. Dupa cum puteti vedea ca peretii se continua acolo unde nu sunt sunt vizitate restul de noduri, astfel acolo unde nu exista linii intre un nod exista un perete.

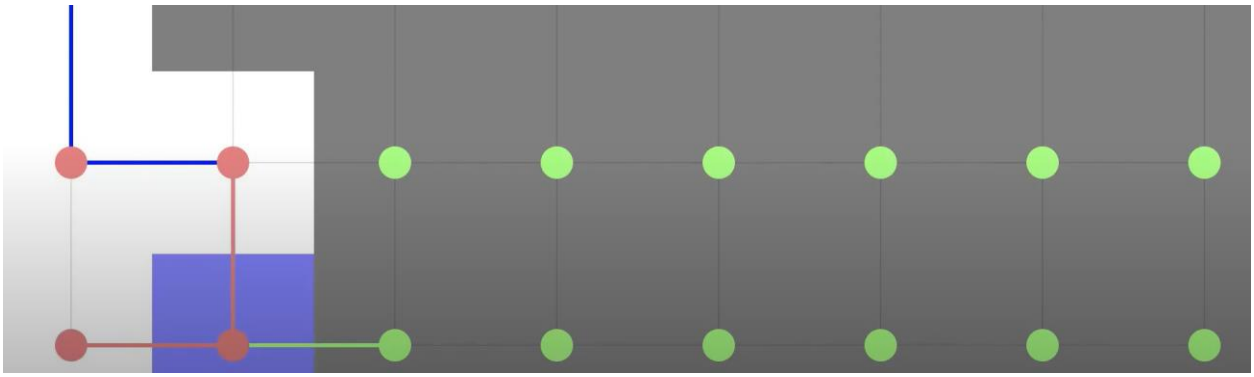


Acum o sa continuam sa mai facem inca cateva mutari.



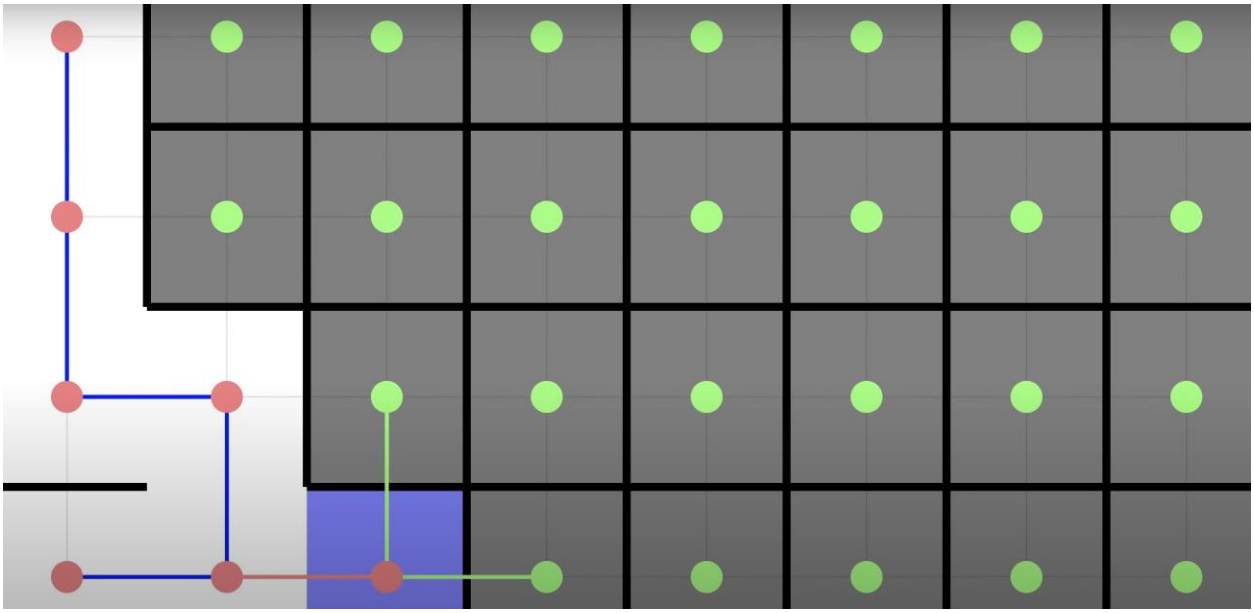
Acum a ajuns in coltul din stanga jos si nu exista noduri in jurul sau care sa fie vizitate. Daca va amintiti de partea de pseudocod, cand ajungeti in aceasta situatie si nu mai exista vecini care nu au fost inca vizitati, este nevoie sa faca

backtrack. Deci pe masura ce avansam din nou, se intoarce intr-un nod deja vizitat.

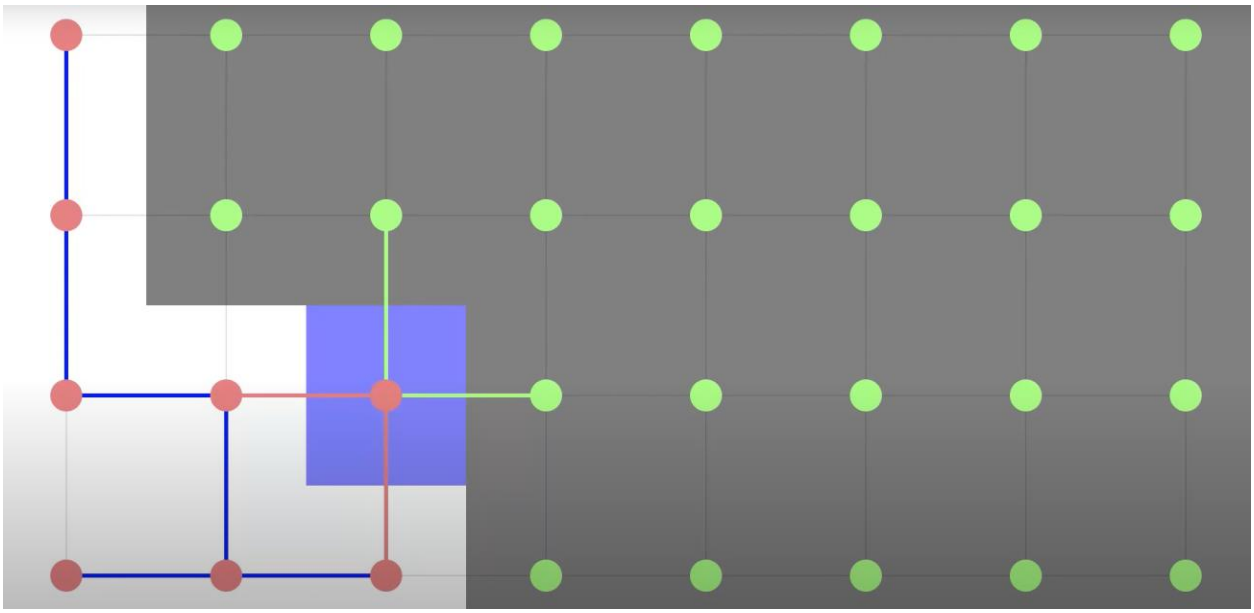


Puteti vedea ca exista posibiliatea de a merge la dreapta, dar nu sus pentru ca am vizitat deja acel nod. Algoritmul va prioritiza nodurile care nu au fost inca vizitate.

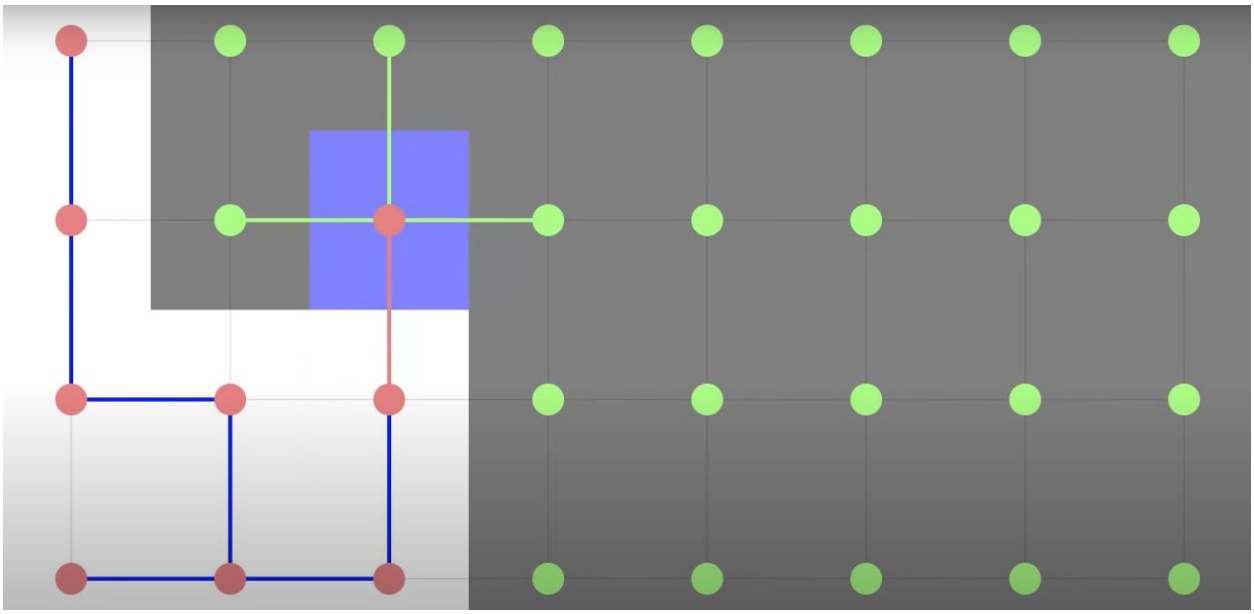
Sa facem inca o miscare. Dupa cum observati exista noduri libere in dreapta si deasupra nodului curent. Acum am activat si vizualizarea peretilor pentru a se observa cum incet, incet incepe sa arate ca un labirint.



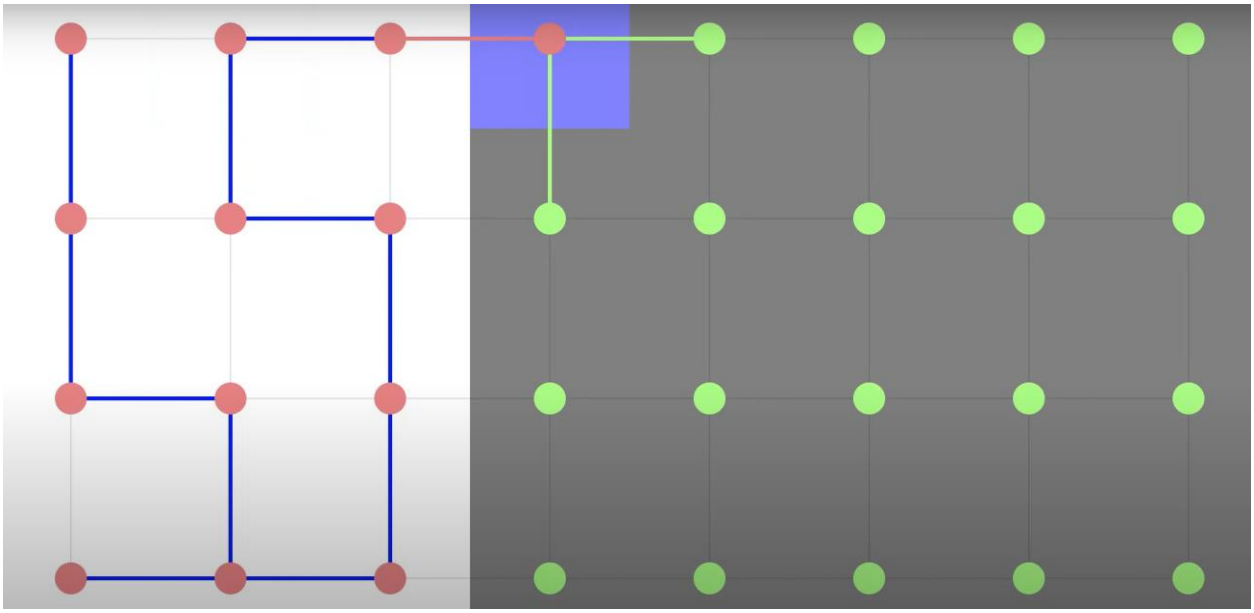
O sa mai facem cateva miscari. Aici putem merge in dreapta sau in sus.



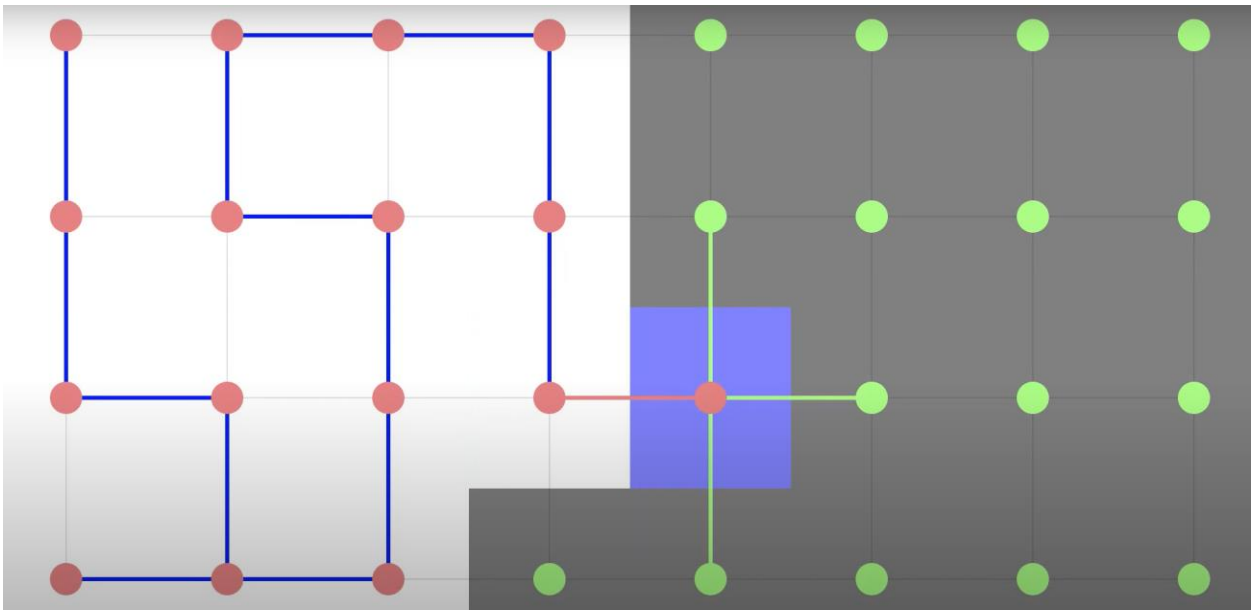
A urcat iar sus si aici poate face 3 miscari posibile.



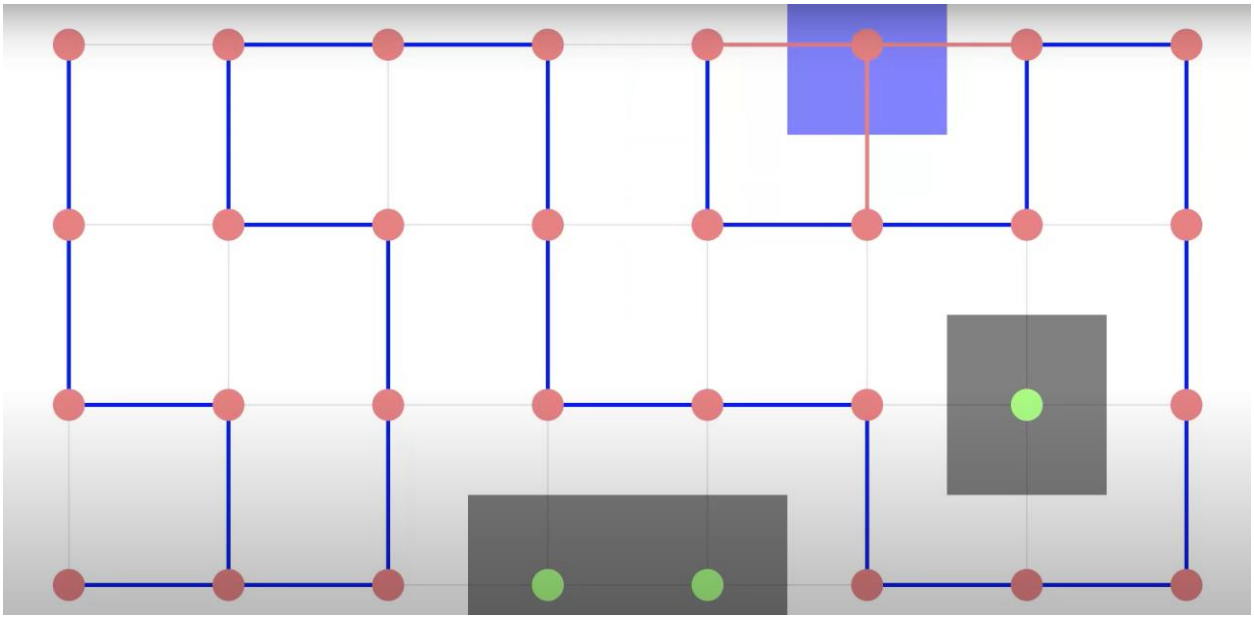
Apoi o sa continui cu cateva miscari in plus.



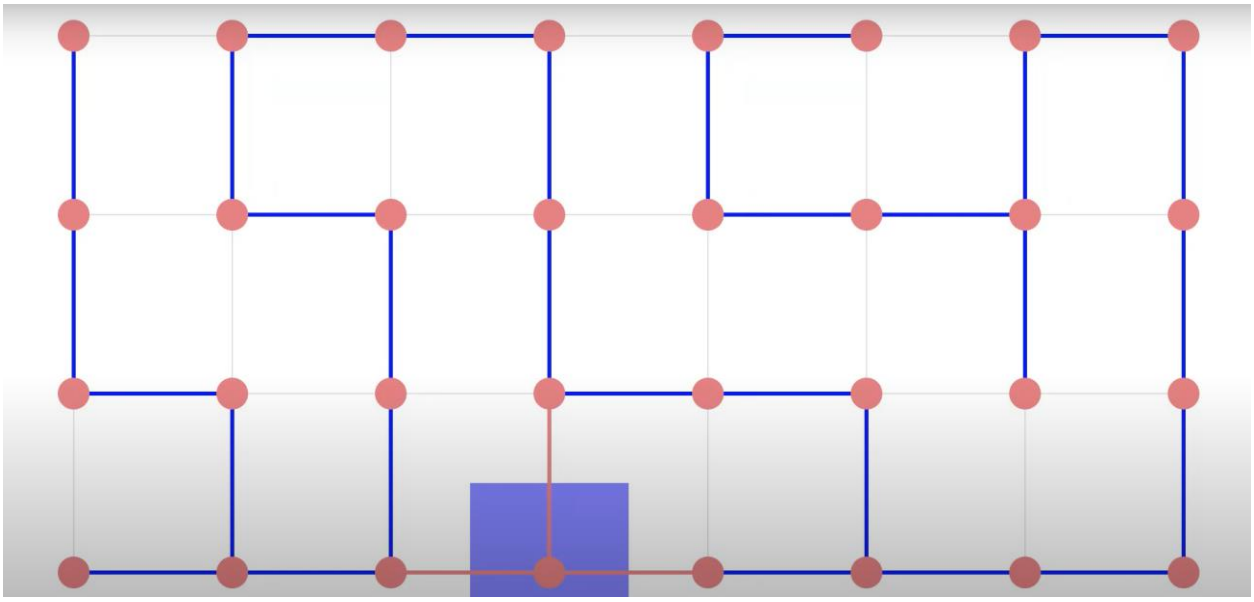
Dupa cum puteti vedea ca tot ceea ce face este sa aleaga intre nodurile unde este o cale posibila, dar nu este posibil sa vizitati un nod rosu daca unul verde este in preajma.



Acum s-au epuizat aproape toate nodurile, nu mai are vecini nevizitati in jurul sau, asa ca incepe backtrackul si trebuie sa ia aceasi ruta inapoi. De fiecare data va verifica daca vecinii pe langa care trece a fost sau nu vizitat.



In acest moment fiecare nod a fost deja vizitat.



Dupa cum puteti vedea avem un labirint complet pe care il poate parcurge soarecele nostru. Linile albaste este exact drumul pe care poate sa il traverseza soarcele. In spatele acestei vizualizari se genereaza doar un graf. Linile pot fi traseate oriunde, doarce in puntul de pornire nu exista, ceea ce exista sunt nodurile si marginele dintre ele.