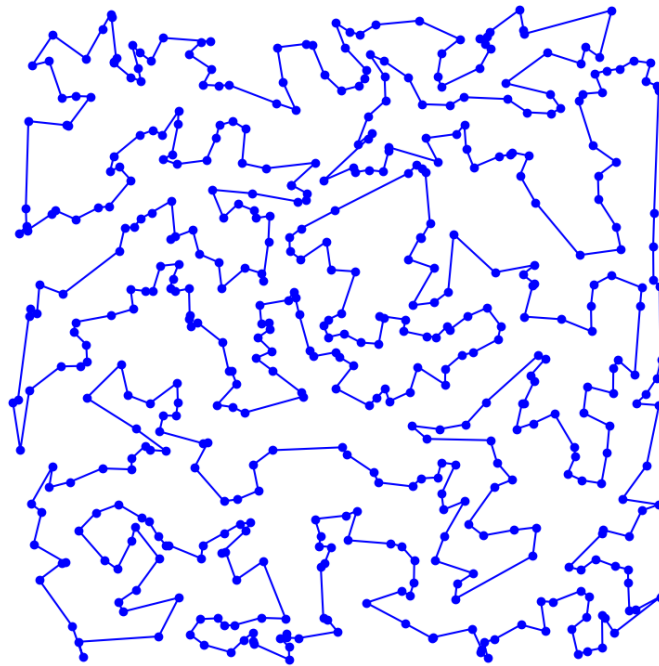


**OPGAVE 1: TSP – ROUTE LANGS 100 STEDEN**

- a) Op Blackboard is het python script `tsp_start.py` te vinden. Breid dit programma uit met een Nearest Neighbor (NN) algoritme, zodat een route wordt berekend langs alle steden. Dit kan in ongeveer 10 regels. Test het programma met 10 steden. Hoeveel procent ligt het resultaat van NN af van de optimale route?
- Tip: als je in `random.seed(n)` dezelfde `n` invult krijg je ook een zelfde verzameling steden.
- b) Hoe lang doet het NN-programma over een route met 500 steden en wat is de totale lengte van de route?
- c) Wanneer je het NN-programma enkele keren probeert zie je dat er altijd wel kruisende wegen (takken) zijn. Hoeveel paren takken zijn er in een route van  $N$  steden?  
Beschrijf op welke manier je kruisende wegen (takken) kan vinden.  
Stel, je maakt een kruising ongedaan, is het dan noodzakelijk om te controleren of de nieuwe route korter is dan de oude?
- d) Maak een programma dat steeds 2 takken neemt en kruisigen detecteert en op basis hiervan de route optimaliseert (2-opt). Hoeveel procent is je 2-opt algoritme beter dan het NN-algoritme bij 500 steden? (Op mijn desktop ongeveer 8% beter in ca. 2 seconden).
- e) Wat is de tijdcomplexiteit van het RC-algoritme?



Resultaat 2-opt met 500 steden

## OPGAVE 2: NEGAMAX - OTHELLO

In het thema 2.3 hebben we al een keer de AI voor het spel Othello moeten implementeren. Dat ging niet in elke groep even goed, en bovendien heeft ook niet elk groepslid dit deel geprogrammeerd. Othello is een spel waarbij je uitstekend Negamax kan toepassen. Want bij het einde van het spel is de score van de winnaar plus de score van de verliezer een constante waarde. Mocht je de spelregels zijn vergeten, op Blackboard is een pdf te vinden met een beschrijving van de spelregels. Misschien is het ook goed om het spel weer eens te spelen, bijvoorbeeld op Android:



Wat hebben we nodig aan datastructuren? In elk geval een representatie van het bord 8x8. Dit kan met een 1- of 2-dimensionale lijst. Verder moeten we de burens van een cel kunnen bepalen en ook de randen van het bord. Ook moeten we een 'bracket' kunnen vinden: gegeven een zwarte/witte positie, wat is de bijbehorende andere positie die een rij witte/zwarte stenen insluit.

Daarnaast zullen we een aantal helper-functies moeten maken:

- het vinden van 'brackets', gegeven een positie, een speler en een richting;
- bijhouden wie de beurt heeft;
- het doen van een zet;
- het vinden van alle brackets en het 'flippen' van de ingesloten stenen van de tegenstander;
- het bijhouden van de score;
- een implementatie van een strategie: het bepalen van de beste zet.

- a) Implementeer een eenvoudige versie van Othello, waarbij de tegenstander willekeurige (geldige) zetten doet. Je hoeft geen GUI te maken, het mag gewoon met een CLI. Op Blackboard is start\_othello.py te vinden die kan je gebruiken (maar dit is niet verplicht).

```

      1 2 3 4 5 6 7 8
1  . . . . . . . .
2  . . . . . . . .
3  . . . @ . . . .
4  . . . @ @ . . .
5  . . . @ o . . .
6  . . . . . . . .
7  . . . . . . . .
8  . . . . . . . .

legal moves: 33, 35, 53
Your move:

```

- b) Implementeer een strategie gebaseerd op Negamax.  
 c) Een belangrijk aspect voor de kwaliteit van de AI is de heuristiek die het bord evalueert. Waarom is het tellen van stenen alleen geen goede evaluatiefunctie? Bedenk en implementeer een betere evaluatiefunctie.  
 d) Wanneer de eis is dat een zet binnen drie seconden moet worden gedaan, tot welke diepte kun je dan gaan met Negamax?  
 e) Verbeter de performance van (b) door pruning toe te passen.

### OPGAVE 3: EXPECTIMAX - 2048

**2048** is een schuifpuzzel(spel) die in 2014 door de - toen 19 jarige - Gabriele Cirulli is geschreven. Gabriele schijnt het spel in een weekend te hebben geschreven in JS en CSS voor de browser. Het werd in 2014 een rage; het spel is nogal verslavend.

Er zijn veel Android/iOS versies, maar het origineel kun je hier (met de pijltjes toetsen) spelen: [2048](#). Wanneer er bij het schuiven twee dezelfde getallen naast of boven elkaar staan worden ze opgeteld en vervangen door een tegel met de nieuwe waarde. De speler wint het spel als hij het getal 2048 kan maken door twee blokjes van 1024 tegen elkaar te schuiven. Als er niet meer kan worden geschoven heeft de speler verloren.



De schuifpuzzel heeft een bord van 4x4, het enige dat de speler kan doen is naar links, rechts, boven of onder schuiven (pijltjes toetsen of swipen). Bij elke beurt zal er een nieuwe tegel met waarde 2 of 4 bijkomen. De tegels (alle vier rijen of kolommen) schuiven – als dat kan - in de gekozen richting: tegels met dezelfde waarde worden opgeteld en veranderen in een nieuwe, en een lege plek wordt opgevuld.

Het doel is de tegels zodanig te schuiven dat er een tegel met de waarde 2048 verschijnt. Na 2048 kan er nog worden doorgespeeld, de maximale score die bereikt kan worden is  $2^{16}$ , maar in dat geval moet er wel op het laatste moment een 4 bijkomen.

De originele sourcecode (JS) kan je hier vinden: [source](#). Het verhaal van Gabriele zelf staat hier: [verhaal](#). Zoals je kunt zien in de sourcecode is de kans op een '2' 90% en de kans op een '4' 10%.

```
68 // Adds a tile in a random position
69 GameManager.prototype.addRandomTile = function () {
70   if (this.grid.cellsAvailable()) {
71     var value = Math.random() < 0.9 ? 2 : 4;
72     var tile = new Tile(this.grid.randomAvailableCell(), value);
73
74     this.grid.insertTile(tile);
75   }
76 };
```

Wanneer je het spel een aantal keer gespeeld hebt, dan begrijp je dat de algemene strategie als volgt is:

- hou het hoogste nummer in een hoek, bijvoorbeeld de linker bovenhoek;
  - zorg ervoor dat hoge nummers in de linker bovenhoek komen;
  - zorg ervoor dat lege cellen niet in de linker bovenhoek komen, dus rechts onder blijven;
  - zorg ervoor dat cellen met gelijke waarden dicht bij elkaar blijven.
- a) Op Blackbord is de file start\_code\_2048.zip te vinden. Deze versie doet random zetten, en zal vrijwel altijd verliezen. Verbeter het spel door AI (Expectimax) toe te voegen. Denk bij de evaluatie van het bord (heuristiek) aan bovengenoemde algemene strategie.
- b) Wat is de maximale diepte waarbij het programma nog een acceptabele performance heeft? Op welke manieren zou je de performance kunnen verbeteren?