Handelsreizigersprobleem

**Concepten:** complexiteit en onhandelbaarheid

**Doel:** Oplossingen vinden voor problemen waarbij het vinden van een optimale oplossing (dus de beste oplossing), onpraktisch is. In dit geval is het vinden van de kortst pad om een aantal locaties te bezoeken en terug te keren naar het startpunt (het handelsreizigersprobleem).

**Leerdoelen:**

Na afloop kun je:

* Beschrijven wat een onhandelbaar probleem is.
* Uitleggen wat een brute force algoritme is, en wat daar de voor- en nadelen van zijn.
* Herkennen of er een efficiënte oplossing voor een probleem bestaat, of dat het onhandelbaar is.
* Een heuristiek toepassen om een oplossing te vinden voor een onhandelbaar probleem.

**Algoritmes en technieken:** Brute force algoritmes, gretige algoritmes, handelsreizigersprobleem, heuristieken.

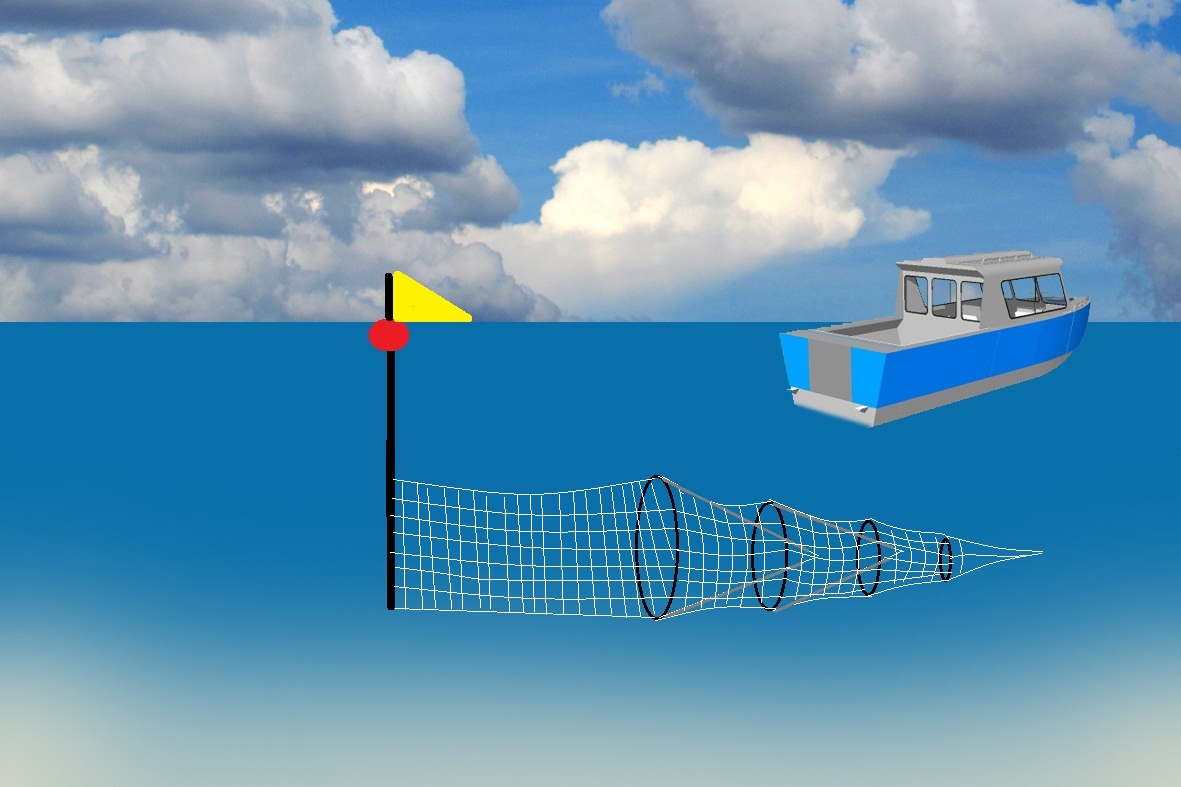
**Praktische context**: Oplossen van het fuikenvissers-probleem (of een andere variant van het handelsreizigersprobleem).

**Voorkennis**: handelsreizigersprobleem (zie werkblad)

**PRAKTISCHE OPDRACHT:**

**Scenario:**

Deze praktische opdracht is gebaseerd op een fictief scenario waarin een fuikenvisser ongeveer 20 fuiken heeft die in het IJsselmeer zijn uitgezet. Fuiken zijn lange rondgebreide netten met steeds kleiner wordende hoepels erin. Om de dag gaat de visser met een boot langs alle fuiken om de vangst eruit te halen.

[[1]](#footnote-1)

De visser wil weten wat de kortste weg is om alle fuiken te controleren en heeft je om hulp gevraagd. Omdat om de paar weken de fuiken verplaatst moeten worden, heeft de visser een voorkeur voor een algemene manier om het probleem op te lossen, in plaats van een oplossing voor een enkele opstelling van de fuiken.

**Voorbereiding**:

1. Om een willekeurige kaart van fuiken te maken pak je 7 munten (voor 7 fuiken) en verspreid je die willekeurig op een velletje A4 papier, bijvoorbeeld door ze op te gooien. Als twee munten op elkaar landen leg je ze naast elkaar zodat ze elkaar raken, maar niet overlappen.
2. Eén voor één haal je de munten weg en zet je een stip op het middelpunt van waar elk muntje lag.
3. Elk stip staat nu voor een fuik dat de visser moet controleren. De linkerbovenhoek van het papier is de thuishaven van de boot. Label deze zo.
4. Maak een foto of kopie van je kaart. Deze heb je later nodig.
5. Maak nu een tweede kaart met 20 munten. Maak hier ook een kopie van.

**Opdracht 1: Inleiding**

Omschrijf de kern van het probleem.

1. Introduceer kort wat het probleem van de fuikenvisser is. Om echt duidelijk te maken waar het over gaat, begin of eindig met:

*“De kern van het probleem waar ik naar kijk is …*”

1. Leg uit hoe het fuikenvisser-probleem vergelijkbaar is met het handelsreizigersprobleem. Omschrijf kort hoe je dat bepaald hebt, en welke dingen bij de fuikenvisser overeenkomen met het Modderige Dorp of de route van de handelsreizigersprobleem.
2. Leg uit waarom informatici zo geïnteresseerd zijn in het probleem van de handelsreiziger.

**Opdracht 2a: Brute force**

1. Voor het probleem bestaat een brute force algoritme. Leg uit wat een brute force betekent. (hint: het principe van het algoritme is eenvoudig, maar het is erg inefficiënt!)
2. Leg uit hoe je een computer kunt gebruiken om met een brute force algoritme tot een optimaal resultaat komt voor 7 fuiken.
3. Leg uit waarom het gebruik van een brute force algoritme niet handig is voor de fuikvisser, die meestal 20 fuiken in het water heeft liggen, om de optimale route te bepalen (één paragraaf is voldoende).

**Opdracht 2b: Menselijke intuïtie**

1. Gebruik je intuïtie om de kortste route tussen de 7 fuiken te vinden die je getekend hebt. Zet een letter of getal bij elk van de fuiken zodat je de volgorde aan kunt geven waarin ze bezocht worden
2. Doe nu hetzelfde met de kaart met 20 fuiken. Let op: stop met de grote kaart na 5 minuten. Je hoeft de oplossing van de grote kaart niet in je PO toe te voegen: De kans is klein dat je een optimale route voor zo’n grote kaart vindt. Het herkennen van de uitdagingen is belangijker dan het vinden van een oplossing..

*Tenzij de fuiken toevallig in een cirkel of ovaal lagen, vond je het waarschijnlijk best lastig om de kortste route te vinden. Een computer zou de kortste route nog moeilijker kunnen vinden; jij kan tenminste gebruik maken van je intuïtie en visuele inzicht. Een computer kan maar twee locaties tegelijk bekijken, terwijl jij er meer dan twee kunt bekijken. Maar zelfs voor jou is het probleem met 20 fuiken en uitdaging! Zelfs als de afstanden tussen elke fuik gemeten zijn en op een kaart getekend, zou het nog steeds een hele uitdaging voor je zijn! Het is duidelijk dat de fuikenvisser niet wil wachten tot jij de optimale oplossing hebt berekend voordat hij op pad kan.*

De vraag is, kun je nog steeds een oplossing bieden die beter is dan het bezoeken van de fuiken in een willekeurige volgorde?

**Opdracht 2c: Heuristiek**

Let uit, aan de hand van minstens één heuristiek (bijvoorbeeld, bij de handelsreizigersprobleem het gretige algoritme van de “Dichtstbijzijnde Buurman”, wat een gretige heuristiek is) hoe deze kan worden gebruikt om een oplossing te vinden voor het fuikvissers-probleem dat jij hebt gemaakt. Leg ook uit wat voor een oplossing door het gretige algoritme gevonden is en waarom dit de fuikenvisser blij maakt.

Dichtstbijzijnde-buur: één van de meest voor de hand liggende heuristieken begint bij de haven in de linkerbovenhoek van de kaart en gaat dan direct naar de dichtstbijzijnde fuik. Vanaf daar bepaalt de visser de dichtstbijzijnde fuik, waar hij nog niet geweest is, en vaart daar rechtstreeks heen. Dit herhaalt de visser totdat hij alle fuiken heeft bezocht. Deze benadering staat bekend als het dichtstbijzijnde-buur algoritme (in het Engels "Nearest Neighbor") en is een voorbeeld van een gretig heuristisch algoritme. Deze neemt altijd de beslissing die op dat moment het beste lijkt, in plaats van op dat moment een minder rendabele keuze te maken om te proberen om zo op termijn meer winst te maken.

Hoewel het in eerste instantie lijkt alsof deze algoritme optimaal is, zal je ontdekken waarom dat niet het geval is.

{HIER OPDRACHT VOOR NEAREST NEIGHBOUR NIET OPTIMAAL TOEVOEGEN}

Gebruik het gretige dichtstbijzijnde-buur algoritme voor zowel je kleine kaart (met 7 fuiken) als je grote kaart (met 20 fuiken) om een ​​oplossing te vinden (het zou niet al te lang duren). Geef de volgorde waarin de fuiken bezocht worden met getalletjes op je kaart weer. Een computer zou veel sneller zijn dan jij, dus je zou een goed idee moeten hebben over hoe de twee algoritmen met elkaar kunnen vergelijken.

**Opdracht 2d: Vergelijken van algoritmen**

Verklaar de resultaten van elk van de algoritmen. Leg uit welk algoritme geschikter is voor het bepalen van een route voor de fuikvisser en waarom. Wat zijn de gevolgen van elke keuze die je overwoog om tot je conclusie te komen? (bijvoorbeeld hoe lang moet de visser op zijn berekening wachten, versus hoeveel tijd en brandstof bespaart de visser bij het varen tussen fuiken). 2 tot 3 alinea's is voldoende.

**Opdracht 3: Je verslag**

Vat nu alles kort en bondig samen in een verslag.

Je zou nu 4 kaarten moeten hebben:

1. kleine kaart met brute force-algoritme,
2. grote kaart met brute force-algoritme,
3. kleine kaart met een gretig heuristisch algoritme, en een
4. grote kaart met een gretig heuristisch algoritme.

Daarnaast heb je inmiddels verschillende toelichtingen bij elk algoritme.

Na je inleiding voeg je de twee brute force kaarten en leg je in het kort uit waarom de grote zo uitdagend was. Leg ook uit waarom dit algoritme voor de fuikvisser niet handig is. Leg vervolgens uit hoe het gretige heuristische algoritme dat je hebt gebruikt werkt. Neem de twee kaarten daarvoor op.

Als je een bekende algoritme voor een andere probleem hebt gebruikt, leg uit wat het verschil is tussen de problemen (bijvoorbeeld: waarom levert alléén het gebruiken van Kruskal geen oplossing voor het TSP probleem)?

Tot slot, leg uit welk van deze twee algoritmes de beste oplossing gaf voor het probleem met de fuiken en de gevolgen van elke keuze. Alles bij elkaar zou je dit in ongeveer 2 tot 3 pagina's moeten kunnen verwoorden (beperk je tot de kern).

**Extra uitdagende opdracht (voor een excellent cijfer):**

Als je echt een hoog cijfer wil, dan moet je verder gaan dan alleen het uitleggen en toepassen van algoritmes. Hiervoor moet je ook zelf verdiepend onderzoek doen naar realistische toepassingen van het handelsreizigersprobleem.

Bespreek het gebruik van de twee algoritmen voor oplossen van echte problemen. Werk voorbeelden uit aan de hand van specifieke kaarten en leg uit hoe de algoritmes voor ieder voorbeeld werken en hoe ze uitpakken.

Evalueer de twee algoritmen in echte wereldproblemen. Dit houdt in dat we de voordelen van kosten ten opzichte van de baten bekijken om betere oplossingen te vinden, voorbeelden geven die laten zien hoe een heuristiek niet de optimale oplossing kan vinden, en potentiële complicaties bekijken die door de basisalgoritmen niet kunnen worden opgelost (bijvoorbeeld verkeersomstandigheden en wegwerkzaamheden).

Analyseer het hebzuchtige heuristische algoritme (bijvoorbeeld gevallen waarin het een echt slechte oplossing geeft en gevallen waarin het de optimale oplossing vindt) en onderzoek hoe effectief het zou zijn in praktijk. Een manier om dit te doen, is door te zoeken op Google maps voor zoiets als supermarkten in een stad (idealiter wil je dat er minstens 20 tot 30 verschijnen), en zorg er dan voor dat de wegen zichtbaar zijn op de kaart, maak er een screenshot van. Gebruik een gretig heuristisch algoritme om de kortste route te bepalen en evalueer hoe effectief dat was. Bespreek je conclusies

Scaffolding informatie voor de docent:

Je zal waarschijnlijk vinden dat sommige delen van het pad zinvol zijn, hoewel het in andere delen wel het geval is

is inefficiënt omdat een bestemming werd "overgeslagen" toen anderen dat enigszins waren

in de buurt ervan werden bezocht, en de kortste pad heuristiek trok het pad weg van dat

bestemming. Welke soorten heuristieken zou u gebruiken om een ​​betere oplossing te krijgen? (B.v. zou je op de een of andere manier de TSP kunnen opdelen in een hoop clusters, waarin je all-in gaat de cluster voordat je naar de volgende cluster gaat?) Misschien wil je een tweede toevoegen kaart met uw andere heuristische ideeën. De kaarten moeten ongeveer een halve pagina beslaan

elk.

Of course, for some situations, intractable problems are a good thing. In particular, most security and cryptography algorithms are based on intractable problems; the codes could be broken, but it would take billions of years and so would be futile. In fact, if anyone ever finds a fast algorithm for solving such problems, a lot of computer security systems would stop being secure overnight! So one of the jobs of computer scientists is to be confident that such solutions don't exist!

**Waar gaat dit eigenlijk over?**

Er bestaan wel efficiënte algoritmes die een goede oplossing geven, maar of deze de beste is valt niet met zekerheid te zeggen.

**Onbehandelbaar**

Het handelsreizigersprobleem is [NP-moeilijk](https://nl.wikipedia.org/wiki/NP-moeilijk). De beslissingsvariant van het probleem (gegeven de onderlinge afstanden tussen de steden, bestaat er een oplossing die korter is dan een gegeven maximum afstand?) is [NP-volledig](https://nl.wikipedia.org/wiki/NP-volledig).[source: https://nl.wikipedia.org/wiki/Handelsreizigersprobleem]

**Vergelijkbare problemen, maar net anders.**

[Chinese postbodeprobleem](https://nl.wikipedia.org/wiki/Chinese_postbodeprobleem). Bij het handelsreizigersprobleem moeten echter alle plaatsen worden aangedaan, terwijl bij het Chinese postbodeprobleem alle wegen tussen de verschillende plaatsen moeten zijn gebruikt. Het Chinese postbodeprobleem is niet NP-moeilijk.

MOGELIJKE TOETSOPGAVE

TOETSOPGAVE Krantenwijk

Een vriend van je heeft een baan als krantenbezorger. Hij heeft veel vaste klanten vaak met verschillende bezorgwensen: Sommigen laten hun krant bij een ander adres bezorgen als ze op vakantie zijn, sommigen krijgen alleen op zaterdag een krant en er zijn adressen met een proefabonnement. Omdat hij per krant wordt betaald en niet per uur vraagt hij jouw hulp voor het bepalen van een optimale route.

1. (2pt) Welk vergelijkbaar probleem hebben we in de les behandeld? Hoe weet je dat?
2. (2pt) Leg uit waarom informatici zo geïnteresseerd zijn in dit probleem.
3. (2pt) Voor het probleem bestaat een brute force algoritme. Wat betekent brute force?
4. (2pt) Noem een voordeel van brute force. Leg uit waarom het gebruik van een brute force algoritme niet handig is om de optimale route te bepalen.

Gegeven zijn alle huizen waar een krant bezorgd moet worden, en alle verbindingen hier tussen. De krantenbezorger is op zoek naar de kortste afstand dat hij moet afleggen.

1. (5pt) Specificeer een (benaderend) algoritme of heuristiek door middel van een ‘hoog niveau’ flowchart (dat wil zeggen, een flowchart zonder veel details).
2. (2pt) Leg uit waarom jouw oplossing handig is voor jouw vriend de krantenbezorger, hoe goed is die, en wat de voor- en nadelen ervan zijn.

1. Bron: https://www.mijnzuiderzee.nl/page/6675/fuikenvisserij [↑](#footnote-ref-1)