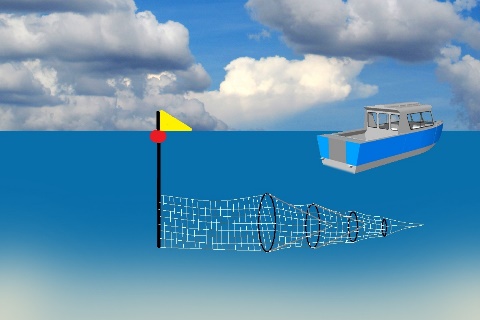
# Praktische Opdracht Onhandelbare problemen

|  |
| --- |
| **Concepten:** Complexiteit en onhandelbare problemen  **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:   * een oplossingsrichting voor een probleem in een bepaalde context ontwikkelen en weergeven als een algoritme; * redeneren over de efficiëntie van een algoritme in relatie tot de invoer; * beschrijven wat een onhandelbaar probleem is; * een aantal standaard graafalgoritmen herkennen en gebruiken; * 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; * de kwaliteit (o.a. juistheid) van een algoritme onderzoeken.   **Algoritmes en technieken:** Brute force algoritmes, gretige algoritmes, handelsreizigersprobleem, heuristieken, herkennen van vergelijkbare problemen.  **Praktische context**: Oplossen van het fuikenvissers-probleem (of een andere variant van het handelsreizigersprobleem).  **Voorkennis**: De stof uit de volgende werkbladen:   1. Minimaal Opspannende Bomen 2. Dijkstra Kortste Pad 3. Onhandelbare problemen 4. Handelsreizigersprobleem |

**Scenario:**

Deze praktische opdracht is gebaseerd op een fictief scenario waarin een fuikenvisser ongeveer 20 fuiken heeft die in het IJsselmeer zijn uitgezet[[1]](#footnote-1). 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.

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.
3. Geef zelf nog een echte wereld voorbeeld van een vergelijkbare probleem.

**Opdracht 2a: Menselijke intuïtie**

1. Gebruik je intuïtie om de kortste route tussen de 7 fuiken te vinden die je getekend hebt. Geef met getallen op je kaart de volgorde aan waarin de fuiken 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 namelijk 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.
3. Evalueer je aanpak.

*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.*

**Opdracht 2b: 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 te komen voor 7 fuiken.
3. Leg uit waarom het gebruik van een brute force algoritme niet handig is voor de fuikvisser, die minstens 20 fuiken in het water heeft liggen, om de optimale route te bepalen (één paragraaf is voldoende).

**Opdracht 2c: Heuristiek**

Je zou de fuikvisser kunnen vertellen om zijn fuiken in een willekeurige volgorde te bezoeken. De vraag is, kun je een oplossing vinden die beter is?

1. Leg uit, aan de hand van minstens één heuristiek (bijvoorbeeld het gretig beste-buur algoritme die je bij de handelsreizigersprobleem hebt gezien), wat een gretige heuristiek is en hoe deze kan worden gebruikt om een oplossing te vinden voor het fuikvissers-probleem.
2. Gebruik het gretig algoritme om een oplossing te vinden voor zowel je kleine kaart met 7 fuiken als je grote kaart met 20 fuiken. Het zou niet al te lang duren. Geef met getallen op je kaart de volgorde aan waarin de fuiken bezocht worden.
3. Evalueer je oplossing. Wat kan je er over de oplossing zelf zeggen? Leg ook uit waarom dit de fuikenvisser wel of niet blij maakt.

**Opdracht 2d: Vergelijken van algoritmen**

Vergelijk de resultaten van beide algoritmen (brute force en de gretig heuristisch algoritme). Een computer zou veel sneller zijn dan jij, dus je zou moeten bedenken hoe je de twee algoritmen met elkaar kunnen vergelijken. Leg uit welk algoritme geschikter is voor het bepalen van een route voor de fuikvisser en waarom. Leg uit wat je overwegingen zijn (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. Hieronder volgen wat aanwijzingen.

Voorbereiding:

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.

Indeling:

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. Voeg de twee kaarten toe bij je uitleg. Als je een bekende algoritme voor een andere probleem hebt gebruikt bij je oplossing, leg uit wat het verschil is tussen de problemen (bijvoorbeeld: waarom levert alléén het gebruiken van Kruskal geen oplossing voor de fuikvisser)?

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 hogere cijfer):**

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

Bespreek het gebruik van de twee algoritmen voor het 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 bij echte wereldproblemen. Dit houdt in dat je de voordelen van de kosten voor het vinden van een betere oplossing afweegt ten opzichte van de baten. Geef voorbeelden die laten zien hoe een heuristiek niet in staat is de optimale oplossing te vinden. Geef aan welke mogelijke moeilijkheden door de basisalgoritmen niet kunnen worden opgelost (bijvoorbeeld verkeersomstandigheden en wegwerkzaamheden).

Analyseer het gretige heuristische algoritme (bijvoorbeeld, geef voorbeelden van gevallen waarin het echt een slechte oplossing geeft en gevallen waarin het de optimale oplossing vindt). Onderzoek hoe effectief het zou zijn in praktijk. Je kunt dit bijvoorbeeld met Google maps doen. Zoek de supermarkten in een stad (het liefst een stad met ongeveer 20 tot 30 supermarken), zorg dat de wegen zichtbaar zijn op de kaart, en maak daar een screenshot van. Gebruik dan een gretig heuristisch algoritme om de kortste route te bepalen en evalueer hoe effectief dat was. Bespreek je conclusies.

EXTRA UITDAGENDE OPDRACHT

De leerling zal waarschijnlijk vinden dat sommige delen van de route zinvol zijn, maar in andere gevallen een ‘onhandige’ keuze is gemaakt omdat een bestemming werd "overgeslagen" toen andere bestemmingen in de buurt ervan werden bezocht. Door het toepassen van de kortste pad heuristiek werd je juist weggetrokken van dat bestemming, waardoor uiteindelijk de route inefficiënt werd.

Bijvoorbeeld, je zou op de een of andere manier de TSP kunnen opdelen in een hoop clusters, waarin je eerst alle bestemmingen in dat cluster bezoekt voordat je naar de volgende cluster gaat.

Welke ideeën heb je om tot een ​​betere oplossing te komen? Licht je heuristiek kort toe en laat het resultaat zien aan de hand van een tweede kaart.

TOELICHTING HELE OPDRACHT:

In sommige situaties is het juist heel goed dat er onhandelbare problemen bestaan. Zo zijn de meeste beveiligings- en cryptografiealgoritmen gebaseerd op dit soort problemen: de codes zouden kunnen worden verbroken, maar het zou miljarden jaren duren en dus nutteloos zijn. Als iemand ooit een snel algoritme vindt voor het oplossen van dit soort problemen, zouden veel computerbeveiligingssystemen niet langer veilig zijn! Een van de taken van computerwetenschappers is om er zeker van te zijn dat zulke oplossingen *niet* bestaan!

Note that the difference between achieved and merit will be in the quality of the explanations (i.e whether or not the marker considers them to be “describing” or “explaining”). Generating the personalised examples is necessary for achieved, because the marker needs to be able to see that the student has done their own work.

It is important that a report discusses both algorithms (brute force and greedy heuristic), because doing the brute force algorithm is necessary to show that it is unhelpful in practice, and then students need to explore approaches that can be used in practice, such as the greedy heuristic algorithm. This allows them to give a solid explanation of TSP based problems and how computer scientists deal with them.

|  |
| --- |
| **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.  **Onhandelbaar**  Het handelsreizigersprobleem is een onhandelbaar probleem. De technische term hiervoor is **NP**. NP staat voor niet-polynomiaal en betekend dat er geen efficiënt oplossing voor is. Voor elke locatie dat toegevoegd wordt, groeit het aantal mogelijkheden namelijk exponentieel (en dus niet polynomiaal).  Ook het bepalen van het minimaal aantal kleuren nodig om een graaf (of kaart) te kleuren is NP. Veel logische puzzels zoals Sudoku, schuifpuzzels en Japanse Puzzels zijn NP.  **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. |

BEOORDELINGSRUBRIC

Cijfers van laag naar hoog:

Omschrijft hoe…

Legt uit dat…

Analyzeeert en weegt tegen elkaar af

Creeert eigen gepersonalizeerde voorbeelden van

Categorieen:

1. Brute force. Hoe toe te passen voor de fuikvisserproblee + uitleg waar

Showing how the brute force algorithm can be applied to the Cray Pot Problem + Explaining why it is not helpful to the crayfisherman

Showing how a greedy heuristic algorithm can be applied to the Cray Pot problem + Explaining what kind of solution the greedy heuristic algorithm has found, and why this is more helpful to the cray fisher

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