Taak verdeling RAM -> Algoritmen & heuristieken

Yessin Radouane:
Ik zal het tweede base line algoritme maken. De eerste baseline was erg slecht en volledig random. Ik zal een algoritme maken die dus een random batterij kiest voor elk huis en dan een pad vindt d.m.v. een keuze te maken tussen de mogelijke paths die niet het meeste afstand creëert tussen de huidige positie van de kabel en het huis.
Eventueel ook een algoritme die een random batterij kiest voor elk huis en daarbij een greedy path kiest. Dit houdt dus in dat er een randomise_v2 en greedy_random.py script gemaakt moet worden.
Daarnaast zal ik functionaliteiten in het main programma maken om sneller en efficiënter verschillende algoritmen aan te roepen. Hierbij hoort dus een class AlgoRunner() die een algoritme naam als argument in de init functie neemt zodat het weet dat dit algoritme gerund moet worden. Zo hoef je dus niet elke keer de main te veranderen om een algo aan te roepen.

Mercedez van der Wal:
Ik zal een algoritme uitwerken die op zoek gaat naar de kortste route naar een batterij. Op basis van Euclidische afstand wordt gekeken welke batterij het meest dichtbij is en daarna wordt er een heuristiek gebruikt die kijkt naar de dichtstbijzijnde kabel in de buurt. Aan de hand hiervan wordt een kabel gelegd. Het leggen van de kabel wordt ook gedaan door een heuristiek, namelijk de meest eenvoudige/snelste route.

Functionaliteiten:
- Bereken dichtstbijzijnde batterij
- Zoek dichtstbijzijnde kabel
- Vergelijk bovenste twee om kabels te kunnen leggen of verbinden

Functies:
- calculate_distance(coordinate1, coordinate2)
- find_closest_battery():
- find_closest_cable():
- compare_results():
- create_cable():

Met behulp van deze functionaliteiten kunnen wij resultaten genereren en vergelijken hoe optimaliserend dit algoritme werkt.

Rembrand Ruppert:
Ik heb en zal een algoritme schrijven dat een ‘heatmap’ maakt van de dichtheid van huizen. Dit algortime heb ik al een simpele basis van gemaakt, maar doet niks anders als van individuele huizen de dichtheid op een grid bepalen. Als dit werkt kan ik vervolgens een algoritme schrijven dat een grove indeling van subdistricts zal maken, wat wij als team kunnen gebruiken om de hoeveelheid kabels te minimaliseren en om beter te voldoen aan de batterij capaciteit restrictie, gezien een deel huizen binnen een subdistrict alleen op 1 batterij aan kunnen sluiten als het goed werkt i.p.v. alle huizen op 1 batterij. Ook ben ik verantwoordelijk geweest voor het visualiseren van de districts door het gebruik te maken van matplotlib.pyplot. Dit is nog een erg slordige code, dus naast de heatmap zal ik bezig zijn met deze code net en simpel te navigeren maken. In de visualisatie kunnen kabels die gelegd zijn door algoritmen vrij simpel getekend worden, hoe deze ook liggen.