KI Project B

Kasper Peeters, 5533988
Ian van Uunen, 4166132

1.
   a. In de game state staat alle data die de wereld beschrijft. De game state kijkt welke actions agents kunnen maken en kan een successor genereren. Het houd ook de positie van dingen in de wereld bij, zoals muren, voedsel en spookjes.
   b. Een agent is een spook of pacman. Een agent kan kijken welke legale moves hij kan maken en houd bij waar hij heen gaat, hoe snel hij gaat en waar hij is.
   c. Stack - Pallet of beer crates
      Queue - Supermarket check out line
      Priority queue - Hospital waiting room
   d. Voor de StayEastSearchAgent is het goedkoper om moves aan de oostkant van het boord te maken. Doordat je uniformCostSearch gebruikt zal hij vaker moves naar het oosten maken. Bij de StayWestSearchAgent is het precies andersom. Voor hem is het goedkoper moves naar het westen te maken, dus daar zal hij vaker zitten.

2. Voor Q1 t/m Q4 gebruiken we een generieke structuur die bestaat uit:
   Een lijst voor nodes die al bezocht zijn
   Een queue of stack voor de nodes die we nog willen bezoeken
   En een lijst die we gebruiken om de parent nodes voor alle nodes bij te houden
   Met alleen kleine veranderingen zoals een lijst voor de kosten bij Q3 en Q4 hebben we een goede basis voor onze algoritmen

3.
   a. De oplossing is gegarandeerd compleet. We gebruiken graph search waarbij we niet verder zoeken met eerder gevonden nodes, dus we kunnen nooit in een loop terecht komen en vinden als het mogelijk is uiteindelijk altijd een kloppende oplossing.
   b. De oplossing is niet gegarandeerd optimaal. Je geeft de eerste oplossing die je vindt en het is een depth first search dus je weet niet zeker of er niet een kortere oplossing is.
   c. Pacman bezoekt bijna nooit alle ontdekte nodes. Alleen als in de eerste lijn naar beneden het antwoord is gevonden zal pacman alle ontdekte nodes bezoeken, anders is er altijd een ontdekte node waar pacman niet langs gaat.

4.
   a. De oplossing is gegarandeerd compleet. We gebruiken graph search waarbij we niet verder zoeken met eerder gevonden nodes, dus we kunnen nooit in een loop terecht komen en vinden als het mogelijk is uiteindelijk altijd een kloppende oplossing.
   b. Als je naar een positie op het boord wil is elke stap even duur en omdat je breadth first gebruikt vind je de oplossing die in de minste stappen gevonden wordt, dus als je ergens naartoe wilt lopen is de oplossing altijd een *least cost solution*.
   c. ;

5.

a. De drie soorten UCS agents proberen allemaal met zo'n laag mogelijke cost bij hun target te komen. De normale UCS agent neemt de snelste route. Voor de StayWestSearchAgent is het goedkoper om aan de west kant van het veld te blijven dus hij zal een route nemen die meer daar langs loopt, deze agent doet dit door de kosten om een node in te gaan 2^(x-positie) te laten maken, het is dus goedkoper om door nodes te gaan met een lage x, oftewel de westkant van het scherm. Voor de StayEastSearchAgent is het goedkoper om aan de oostkant te blijven dus hij zal daar langs gaan. Deze agent doet ongeveer hetzelfde als de StayWestSearchAgent alleen zijn de kosten om een node in te gaan ½ ^ (x-positie), dit is exact het tegenoverstelde van de andere agent en daarom blijft deze agent aan de oostkant van het scherm.

8.

a. We gebruiken als heuristic de manhattan distance naar de verste hoek
b. Hierdoor is hij optimistisch maar zal hij met elke stap niet meer dan die stap afnemen
c. 1140 nodes explored

9.

a. Onze food-heuristic is de hoeveelheid food die er over is. We kozen dit, omdat het makkelijk op te zoeken is
b. Je zit altijd op het minimum aantal stappen voor die hoeveelheid food als het tegen elkaar aan lag, dus hij kan niet hoger dan het maximum of lager dan het minimum zijn en is dus sowieso admissable.