Dentist algoritme

Onze dentist algoritme werkt met 3 verschillende geprioriteerde lijsten.

- ***InDangerOfPenalty***
- ***CloseToPenalty***
- ***NothingToWory***

De lijsten worden op deze prioriteiten afgewerkt op één uitzondering na die we later behandelen. In deze lijsten worden ook wachttijden bijgehouden zodat deze makkelijk bereikbaar zijn. Om te kijken hoe de lijsten gevuld worden er 2 berekeningen gedaan.

In danger of penalty

Om een penalty te krijgen moet je langer dan 10 ticks hebben gewacht. Het aantal strafpunten komt overeen met de ticks barrière. Wij proberen zo min mogelijk penalty's te krijgen en zetten ze dus de InDangerOfPenalty lijst op een barrière van 10. Alle klanten die dicht bij de 10 staan komen in deze lijst en worden als eerste geholpen.

De lijst wordt geprioriteerd op de korte duur van de behandeling. Door dit te doen vang je minder penalty's, omdat de wacht lijst sneller klein wordt. En hoe minder gasten er wachten. De kleiner de kans op gasten die lang wachten.

CloseToPenalty

De close too penalty lijst wil zeggen dat klanten die in de een 'gele gevaren zone' komen worden eerder geholpen. Om dit te berekenen nemen we een gemiddelde behandel tijd van de klanten en kijken of er na 3 beurten een penalty gegeven kan worden. Is dit zo dan worden ze toegevoegd aan de lijst.

Ook deze lijst wordt geprioriteerd op de duur van een behandeling om zo snel mogelijk zo veel mogelijk patiënten te helpen. Als er meer mensen dichtbij een penalty zitten dan mensen in danger of a penalty dan wordt de volgende klant uit deze lijst gepakt. Dit doen wij omdat in een voorbeeld van 1 klant die al 9 ticks wacht en 3 gasten wachten al 5 ticks. Is het beter om de 3 gasten eerst te helpen omdat de kans op penalty's lager is. Dit hangt ook af van de behandeld tijd maar door het testen van de factoren hebben we kunnen concluderen dat dit redelijk verwaarloosbaar is.

NothingToWorryAbout

Als er geen penaltys gegeven kunnen worden in een korte tijd dan wordt de klant met de korstte duur van een behandeling opgeroepen. Zodoende blijven er zo min mogelijk mensen in de wachtkamer.

Aftermath

Aan de hand van het spelen met de factoren om klanten te prioriteren in verschillende lijsten hebben we gevonden dat de huidige factoren het efficiënts werkten. Om en nog beter algoritme hiervoor te schrijven zou er naar een andere aanpak moeten worden gekeken. Wellicht hebben we onze formule niet goed gedaan in het c# bestand en kan de berekening alsnog sneller.

Aan de andere kant konden we ook één regel code gebruiken en een snellere resultaat krijgen alleen vonden we dat wel vreemd.

```
return room.OrderBy(x => x.Duration).ThenByDescending(x => x.Arrival).First();
```