##### Datastructuren

We gebruiken in de State class een ushort[] om huidige autoposities in op te slaan, in combinatie met een Car class die onveranderende data over de auto’s opslaat. Daarnaast slaan we de gedane stappen op in een Step object, wat eigenlijk een soort dubbel linked list is.

De al geziene State’s slaan we op als uints in een TabooD object, een class die eigenlijk een lockende wrapper is om een normale Dictionary. De uints zijn waarden die wordt berekend uit de ushort[] aan autoposities, en die voor elke unieke autopositie ushort[] uniek is.

Het berekenen van volgende States wordt gedaan door uit een List<State>[] de lijst met de laagste heuristiekwaarde te pakken die bestaat. Die laagste index wordt onder het toevoegen van net berekende volgende State’s opgeslagen, dus het kost O(1) om de laagste index te vinden. We hebben A\* geïmplementeerd, maar de heuristische functie is niet zo geweldig dus hij pakt hard-2, 3 of 4 niet.

De gevonden lijst wordt uit de List<State>[] gehaald en die plek in de array wordt meteen geleegd. De lijst die eruit gehaald is berekend voor al zijn state objecten de volgende States met behulp van een ThreadPool.

##### BottleNecks

Ik denk dat het collision checken van auto’s een bottleneck is omdat dat in O(n^2) gaat.