Datastructuren en algoritmiek - Opdracht 2

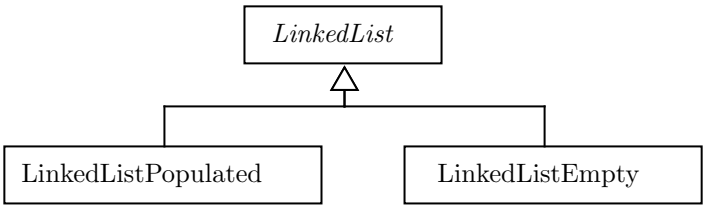
February 2, 2026

Gefeliciteerd! Vanaf vandaag kun je je vaardigheden die je dit semester opgedaan hebt aanwenden in een onderzoek naar de milieuvriendelijkheid van het Nederlandse wagenpark. Een onderwerp waar je je mogelijk al eens bij stil gestaan hebt, of je misschien ook wel eens zorgen over gemaakt hebt omdat het een relevante invloed kan hebben op jouw toekomst. Vanwege de beperkte tijd is natuurlijk ook het onderzoek beperkt en helpen we je een flink stuk op weg.

In lijn met dit semester kiezen we voor een data gerichte aanpak en stellen we de onderzoeksvraag: "Hoe groot is het Nederlandse wagenpark?" Een vraag die we met een schatting zullen beantwoorden, namelijk met het aantal uniek geregistreerde kentekens.

In deze opdracht implementeer je zelf een '(Singly) linked list' datastructuur[1], een aantal nuttige operaties hierop en een algoritme om de lijst te sorteren. De opgaven zullen je stap voor stap naar het eindresultaat begeleiden. Tenslotte pas je het sorteeralgoritme toe op een kleine lijst (als test) en een grote lijst met kentekens (die je download bij de overheid). De opgaven dienen geimplementeerd te worden in Python en je kunt zelf je favoriete ontwikkelomgeving kiezen.

1 Maak een eigen implementatie van een 'Singly linked list'. Er zijn meerdere implementaties denkbaar, maar we adviseren om de list als abstracte class te implementeren met twee concrete subclasses. Een voor lege en een voor niet-lege lijsten.



2 Schrijf een (recursieve) methode 'toString', die een lijst omzet naar een string waarin de inhoud wordt opgesomt gescheiden door een spatie. (Een overtollige spatie aan het einde is niet erg.) Een lijst met elementen '4' en '7' bijvoorbeeld, zou omgezet worden naar "4 7 ". Schrijf voorbeeldcode met lijsten van 0, 1 en 2 elementen.

3 Schrijf een (recursieve) methode 'addFirst(value), die een nieuwe lijst retourneert met de gegeven waarde vooraan en daarna alle elementen uit de oorspronkelijke lijst. Een aanroep van 'addFirst(5)' op een lijst met elementen '4' en '7' bijvoorbeeld levert een nieuwe lijst op met elementen '5', '4' en '7'. Schrijf voorbeeldcode die aannemelijk maakt dat de methode correct werkt.

4 Schrijf een (recursieve) methode 'remove(value), die een nieuwe lijst retourneert met alle elementen uit de oorspronkelijke lijst, behalve *het eerste voorkomen van* de gegeven waarde. Een aanroep van 'remove(4)' op een lijst met elementen '5', '4', '7' en '4' bijvoorbeeld levert een nieuwe lijst op met elementen '5', '7' en '4'. Een aanroep van 'remove(4)' op deze laatste lijst levert een nieuwe lijst op met elementen '5' en '7'. Schrijf voorbeeldcode die aannemelijk maakt dat de methode correct werkt.

5 We zijn een eind opgeschoten! Het eenvoudige sorteeralgoritme dat we gaan schrijven werkt (ongeveer) als volgt. Vind het kleinste element in de oorspronkelijke lijst en verwijder dit. Voeg het toe aan een nieuwe lijst en herhaal alle stappen tot de oorspronkelijke lijst leeg is. Omdat alle elementen in volgorde van klein naar groot aan de nieuwe lijst zijn toegevoegd is de nieuwe lijst gesorteerd.

Er is nog een kleine hulpmethode nodig. Schrijf een (recursieve) methode 'smallest()' die de kleinste waarde van een lijst retourneert. Een aanroep van 'smallest()' op een lijst met elementen '5', '4' en '7' bijvoorbeeld geeft '4' als

[1] https://www.geeksforgeeks.org/singly-linked-list-tutorial/

antwoord. Schrijf voorbeeldcode die aannemelijk maakt dat de methode correct werkt.

6 Alle ingrediënten zijn aanwezig voor het eenvoudige sorteeralgoritme! Schrijf een (recursieve) methode 'sortSimple(), die een nieuwe lijst retourneert met alle elementen uit de oorspronkelijke lijst gesorteerd van klein naar groot. Een aanroep van 'sortSimple()' op een lijst met elementen '5', '4', '7' en '4' bijvoorbeeld levert een nieuwe lijst op met elementen '4', '4', '5' en '7'. Schrijf voorbeeldcode die aannemelijk maakt dat de methode correct werkt.
Mocht een recursieve methode te lastig zijn mag een niet-recursieve methode ook. Het kan dan handig zijn om gebruik te maken van een hulpmethode 'largest()' in plaats van 'smallest()'.

7 Download de bestanden 'Read.py' en 'kentekens1000.txt' van Brightspace. Gebruik de code in 'Read.py' om het bestand met kentekens in te lezen in een lijst (of schrijf hiervoor je eigen code). Gebruik de 'sortSimple()' methode om ze te sorteren en 'toString()' om ze uit te printen. Zijn ze netjes gesorteerd?

8 Schrijf een (recursieve) methode 'uniq()'die, indien aangeroepen op een *gesorteerde* lijst, retourneert hoeveel unieke elementen de lijst bevat. Een aanroep van 'uniq()' op een lijst met elementen '4', '4', '5' en '7' bijvoorbeeld geeft '3' als antwoord. Schrijf voorbeeldcode die aannemelijk maakt dat de methode correct werkt.

9 Gebruik de methode 'uniq()' om te bepalen hoeveel unieke kentekens het bestand 'kentekens1000.txt' bevat. Hoeveel zijn dit er?

10 Ga naar de Open Data webpagina van de overheid[2] en download (onder tabblad 'Databronnen') het 'Text/Csv' bestand met de kentekenregistraties in Nederland. Verwijder de eerste regel uit het bestand. Gebruik de code uit de vorige opdracht om te bepalen hoeveel unieke kentekens dit bestand bevat. Lukt dit? Geef een verklaring waarin je laat zien ook de stof uit de hoorcolleges te beheersen.

[2] https://data.overheid.nl/dataset/6bbef6af-8f1a-47d0-b696-7325361cfe19