```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 29 17:54:20 2021

@author: Tarandeep
"""

#Simplified Code for Kyle

"""
This file contains all distance calculations and serves as the evaluation model


For reference:

Every SKU has the format (r,c) where r is the horizonatl location, and c is the vertical location
Every node has the format (h,v) where h is every horizontal aisle, and v is erery vertical aisle
"""



# imports

from operator import itemgetter

# this method will take the SKUList and convert SKUS to aisle numbers for picks

"""
This is the main function that the other functions depend on
"""
def distanceAlgo(SKUList, aisles):
    SKUComplete = False
    sortedList = sortIntoAisles(SKUList)  # sort SKUS in order
    lastAisle = (sortedList[len(sortedList)-1])[1]
    # this condition will ensure that the picker can leave the warehouse when complete

    if lastAisle % 2 == 1:
        lastAisle += 1

    currentNode = (0, (sortedList[0])[1]-1)
    allNodes = [currentNode]

    while (SKUComplete is False):
        #    if currentNode[0]>lastAisle: #need to fix, should be done through method
        #        break
        if currentNode[0] == 0:
            currentNode = bottomNode(currentNode, sortedList, lastAisle)
        # elif currentNode[0] == 1:
        #     currentNode = middleNode(
        #         currentNode, sortedList, aisles, lastAisle)
        elif currentNode[0] == 1:
            currentNode = topNode(currentNode, sortedList, lastAisle)
        if currentNode == (0, lastAisle):
            SKUComplete = True
        allNodes.append(currentNode)
        # print(currentNode)

    return allNodes



def sortIntoAisles(SKUList):
    # SKU = (2,7) where 2 is aisle, 7 is location
    newSKUList = []
    for SKU in SKUList:
        if SKU[1] % 2 == 0:
            SKUAisle = SKU[1]/2
        else:
            SKUAisle = int(SKU[1]/2)+1
        newSKU = (SKU[0], SKUAisle)
        newSKUList.append(newSKU)

    # https://stackoverflow.com/questions/3121979/how-to-sort-a-list-tuple-of-lists-tuples-by-the-element-at-a
    newSKUList.sort(key=itemgetter(1))
    return newSKUList


def bottomNode(dataTuple, sorted, lastAisle):
    # if at last aisle
    if dataTuple[1] == lastAisle:
        return (dataTuple)

    # if aisle cannot be entered
    if dataTuple[1] % 2 == 0:
        return (dataTuple[0], dataTuple[1]+1)

    for SKU in sorted:
        # only need one sku to move in that direction
        if SKU[1] == dataTuple[1] or SKU[1] == dataTuple[1]+1:
            return (dataTuple[0]+1, dataTuple[1])
    # if none are true, towmotor can move to next aisle
    return (dataTuple[0], dataTuple[1]+1)


def topNode(dataTuple, sorted, lastAisle):
    # if at last aisle
    if dataTuple[1] == lastAisle:
        return (dataTuple[0]-1, dataTuple[1])

    # if aisle cannot be entered
    if dataTuple[1] % 2 == 1:
        return (dataTuple[0], dataTuple[1]+1)
```

```python
    for SKU in sorted:
        # only need one sku to move in that direction
        if SKU[1] == dataTuple[1] or SKU[1] == dataTuple[1]+1:
            return (dataTuple[0]-1, dataTuple[1])

    return (dataTuple[0], dataTuple[1]+1)


#evaluate distance using the allNodes array that is returned from the main function

def distanceCalculation(distanceNodes):
    distance = 0

    # params
    BETWEEN = 144  # travel along aisles
    UPPERVERT = 40  # above/below crossaisle

    prevNode = distanceNodes[0]
    for node in distanceNodes:
        if node == prevNode:
            pass
        # if moving along aisles
        elif node[1] == prevNode[1]:
            distance += BETWEEN
        elif prevNode[0] == 0:
            distance += UPPERVERT
        elif prevNode[0] == 1:
            distance += UPPERVERT
        prevNode = node

    return distance
```