""" verision #2 wherein I'm adjusting all the indices in the for loops to try and make them compatible

with how python starts numbering 0 to n-1 for n numbers"""

import numpy as np #for matrix operations

import math

def knapsack(items, c, k):

#sort based on price

sorted\_items = np.sort(items,order='price')

prices=sorted\_items['price']

for i in range(len(prices)):

prices[i]=int(math.ceil(prices[i]))

#discard all items with weight larger than wmax

remove=[]

for j in range(k-1,len(sorted\_items)): #k to current n

wmax = sum(prices[0:k]) + prices[j] #note: upper bound k is NOT inclusive, goes to k-1

if wmax > c:

remove.append(j)

filt\_items=np.delete(sorted\_items,range(max(remove)+1,len(sorted\_items)),0)

prices2=filt\_items['price']

n=len(filt\_items) #number of items post-removal of ones that are too big

#calculate weightsum of the first k items of the sorted list

v=int(sum(prices2[0:k+1:1]))

#initialize g as an n by c matrix (?), sigma as...nothing?

g = np.zeros([n,c])

#sigma=np.zeros([n,1])

sigma = np.zeros([c,1])

#copied the balcardssp algorithm more-or-less verbatim, no clue what's going on tho:

for weightsum in range(v+1,c):

g[k,weightsum]=0

g[k,v]=k+1

for weightsum in range(int(v+prices2[k+1]),c):

#sigma.insert(weightsum,1)

sigma[weightsum]=1

for weightsum in range (c+1,int(c+prices2[k])):

temp\_keep=[]

for j in range(n):

if prices2[j]<weightsum-c:

temp\_keep.append(j)

#sigma.insert(weightsum,max(temp\_keep)+1)

sigma[weightsum]=max(temp\_keep)+1

for b in range(k+1,n):

for weightsum in range(v,c):

g[b,weightsum]=g[b-1,weightsum]

for weightsum in range(v,int(c+prices2[k]-prices2[b])):

weightsum2=int(weightsum+prices2[b])

#print g.shape #TROUBLESHOOTING ONLY

#print sigma.shape #TROUBLESHOOTING ONLY

if g[b-1,weightsum]>sigma[weightsum2]:

for h in range(int(sigma[weightsum2]),int(g[b-1,weightsum]-1)):

weightsum3=int(weightsum2-prices2[h])

g[b,weightsum3]=max(g[b,weightsum3],h)

#sigma.insert(weightsum2,g[b-1,weightsum])

sigma[weightsum2]=g[b-1,weightsum]

np.savetxt("test\_results\_1.csv", g, delimiter=",")

np.savetxt("test\_results\_2.csv", sigma, delimiter=",")

#now apply the knapsack function to imported data

csv = np.genfromtxt ('dummy\_list\_subset2.csv', delimiter=",", dtype=None, names=['name',

'club','pos','price','value'])

knapsack(csv,70,10) #alter values here during testing