sage code for finding the geo-isomorphism classes of $K\_{2,n}$

This program was written for Version 4.2.1 of the Sage Mathematics Software by W. A. Stein et al. [The Sage Development Team, 2009. http://www.sagemath.org.]

import sage.combinat.permutation as permutation

def symdif(A, B):

#Finds the symmetric difference of two lists A and B.

AbackB = copy(A)

i = len(A)-1

while i >= 0:

d=0

for j in range(len(B)):

if A[i] == B[j]:

d=1

if d==1:

del AbackB[i]

i=i-1

BbackA = copy(B)

k = len(B)-1

while k >= 0:

e=0

for j in range(len(A)):

if B[k] == A[j]:

e=1

if e==1:

del BbackA[k]

k=k-1

return ([AbackB, BbackA])

def SymDif(r, s):

#Finds the symmetric difference E(r o s)\E(r) and E(r)\E(r o s).

#Input must be two permutations.

Er=(r.inverse()).inversions()

ros=Permutation(PermutationGroupElement(list(s))\*PermutationGroupElement(list(r)))

Eros=(ros.inverse()).inversions()

return symdif(Eros, Er)

def Geo\_classify(n):

#Divides the permutations on n letters into geo-isomorphism classes.

#Input is a positive integer n; output is a list of lists of permutations.

#Initialize variables.

G=SymmetricGroup(n)

D=[Permutation(G[i]) for i in range(len(G))]

Geo=[[[]]] #Set of geo-classes.

a=[] #Which permutations of D have already been classified.

j=0 #Indexes the geo-classes.

#The first class is just the identity permutation.

Geo[0][0]=copy(D[0])

a.append(0)

#If necessary, create the next geo-class.

while 0 < len(D) - len(a):

Geo.append([])

j=j+1

#Put in the first permutation that hasn't been classified yet.

m=0

while m in set(a):

m=m+1

Geo[j].append(D[m])

a.append(m)

#Add in those elements of D that are equivalent to

#the starting element in the j-th class.

p=D[m]

Ep=(p.inverse()).inversions()

for i in range(m+1,len(D)):

if not(i in set(a)):

s=D[i]

d=0

if p.number\_of\_inversions() == s.number\_of\_inversions():

for k in range(1,len(D)):

r=D[k]

if sorted(Ep) == sorted(SymDif(r,s)[0]) or sorted(Ep) == sorted(SymDif(r,s)[1]):

d=1

if d == 1:

Geo[j].append(D[i])

a.append(i)

return Geo