Examples of Sagemath Homomorphism Count Error

Raymond Sun

Example 1

H = K55_minus_C10 , G = K_2

Run with Sagemath, count = 0 (incorrect)

```
from standard_hom_count import GraphHomomorphismCounter

H_K55_minus_C10 = graphs.CompleteBipartiteGraph(5,5)
edges_to_remove = [(0,5), (0,6), (1,6), (1,7), (2,7), (2,8), (3,8), (3,9), (4,9), (4,5)]
H_K55_minus_C10.delete_edges(edges_to_remove)
H = H_K55_minus_C10
A = [[0, 1],
     [1, 0]
    ]
G1 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H, G1)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy brute force, count = 2 (correct)

```
import numpy as np
import itertools
def count_homomorphisms(H_adj, G_adj):
    """
    Counts the number of homomorphisms from H to G.

    Parameters:
        H_adj (2D numpy array): Adjacency matrix of graph H (size h x h)
```

```
        G_adj (2D numpy array): Adjacency matrix of graph G (size g x g)

    Returns:
        int: number of homomorphisms from H to G
    """
    h = H_adj.shape[0]
    g = G_adj.shape[0]
    count = 0

    # All functions from V(H) -> V(G)
    for mapping in itertools.product(range(g), repeat=h):
        valid = True
        for i in range(h):
            for j in range(h):
                if H_adj[i, j] == 1 and G_adj[mapping[i], mapping[j]] != 1:
                    valid = False
                    break
            if not valid:
                break
        if valid:
            count += 1
    return count
H = np.array([
    [0,0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,1,0,0,1,1],
    [0,0,0,0,0,1,1,0,0,1],
    [0,0,0,0,0,1,1,1,0,0],
    [0,0,0,0,0,0,1,1,1,0],
    [0,1,1,1,0,0,0,0,0,0],
    [0,0,1,1,1,0,0,0,0,0],
    [1,0,0,1,1,0,0,0,0,0],
    [1,1,0,0,1,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0]
])
G1 = np.array([
  [0, 1],
  [1, 0]
  ])
count_homomorphisms(H, G1)
```

```
2
```

Example 2

H = K55_minus_C10 , G is irregular graph with 4 nodes, adjacency matrix as below:

Run with Sagemath, count = 20 (incorrect)

```
H = H_K55_minus_C10
A = np.array([
    [0, 1, 0, 1],
    [1, 0, 1, 1],
    [0, 1, 0, 0],
    [1, 1, 0, 0],
])
G2 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H, G2)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy brute force, count = 848 (correct)

```
H = np.array([
    [0,0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,1,0,0,1,1],
    [0,0,0,0,0,1,1,0,0,1],
    [0,0,0,0,0,1,1,1,0,0],
    [0,0,0,0,0,0,1,1,1,0],
    [0,1,1,1,0,0,0,0,0,0],
    [0,0,1,1,1,0,0,0,0,0],
    [1,0,0,1,1,0,0,0,0,0],
    [1,1,0,0,1,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0]
])
G2 = np.array([
    [0, 1, 0, 1],
    [1, 0, 1, 1],
    [0, 1, 0, 0],
    [1, 1, 0, 0]
])
count_homomorphisms(H, G2)
```

```
848
```

Example 3

H = K55_minus_C10 , G = complete bipartite graph (3, 3)

Run with Sagemath, count = 0 (incorrect)

```
H = H_K55_minus_C10
A = np.array([
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0]
])
G3 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H, G3)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy backtrack, count = 118098 (correct)

```
def count_homomorphisms_backtrack(H_adj, G_adj):
    h = H_adj.shape[0]
    g = G_adj.shape[0]
    count = 0

    mapping = [-1] * h

    def backtrack(pos):
        nonlocal count
        if pos == h:
            count += 1
            return

        for candidate in range(g):
            # Check edges from pos to previously mapped vertices
            valid = True
            for prev in range(pos):
                if H_adj[pos, prev] == 1 and G_adj[candidate, mapping[prev]] != 1:
                    valid = False
                    break
                if H_adj[prev, pos] == 1 and G_adj[mapping[prev], candidate] != 1:
```

```
                    valid = False
                    break
            if valid:
                mapping[pos] = candidate
                backtrack(pos + 1)
                mapping[pos] = -1

    backtrack(0)
    return count
H = np.array([
    [0,0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,0,1,0,0,1,1],
    [0,0,0,0,0,0,1,1,0,0,1],
    [0,0,0,0,0,0,1,1,1,0,0],
    [0,0,0,0,0,0,0,1,1,1,0],
    [0,1,1,1,0,0,0,0,0,0,0],
    [0,0,1,1,1,0,0,0,0,0,0],
    [1,0,0,1,1,0,0,0,0,0,0],
    [1,1,0,0,1,0,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0]
])
G3 = np.array([
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0]
])
count_homomorphisms_backtrack(H, G3)
```

```
118098
```

Example 4

H = Complete bipartite graph (4, 4), G = complete bipartite graph (3, 3)

Run with Sagemath, count = 13122 (correct)

```
H1 = graphs.CompleteBipartiteGraph(4, 4)
A = np.array([
```

```
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0]
])
G3 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H1, G3)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy backtrack, count = 13122 (correct)

```
H1 = np.array([
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0]
])
G3 = np.array([
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0],
  [0, 1, 0, 1, 0, 1],
  [1, 0, 1, 0, 1, 0]
])
count_homomorphisms_backtrack(H1, G3)
```

```
13122
```

Example 5

H = K55_minus_C10, G is graph with 12 vertices

Run with Sagemath, count = 226927 (incorrect)

```
H = H_K55_minus_C10
A = np.array([
    [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1],
    [0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0],
    [1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1],
    [0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0],
    [0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0],
    [0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]
    ])
G4 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H, G4)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy backtrack, count = 5654308 (correct)

```
H = np.array([
    [0,0,0,0,0,0,0,1,1,1],
    [0,0,0,0,0,1,0,0,1,1],
    [0,0,0,0,0,1,1,0,0,1],
    [0,0,0,0,0,1,1,1,0,0],
    [0,0,0,0,0,0,1,1,1,0],
    [0,1,1,1,0,0,0,0,0,0],
    [0,0,1,1,1,0,0,0,0,0],
    [1,0,0,1,1,0,0,0,0,0],
    [1,1,0,0,1,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0]
])
G4 = np.array([
    [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1],
    [0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0],
    [1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1],
    [0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0],
```

```
    [0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0],
    [0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]
])
count_homomorphisms_backtrack(H, G4)
```

5654308

Example 6

H = Complete bipartite graph (4, 4), G is graph with 12 vertices

Run with Sagemath, count = 180520 (correct)

```
H = graphs.CompleteBipartiteGraph(4, 4)
A = np.array([
    [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1],
    [0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0],
    [1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1],
    [0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0],
    [0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0],
    [0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]
    ])
G4 = Graph(Matrix(ZZ, A), format='adjacency_matrix')
counter = GraphHomomorphismCounter(H, G4)
count = counter.count_homomorphisms()
print(count)
```

Run with numpy backtrack, count = 180520 (correct)

```
H1 = np.array([
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [0, 0, 0, 0, 1, 1, 1, 1],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0],
    [1, 1, 1, 1, 0, 0, 0, 0]
])
G4 = np.array([
    [0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0],
    [1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1],
    [1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1],
    [0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0],
    [1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1],
    [0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0],
    [0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0],
    [0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1],
    [1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0],
    [0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0]
])
count_homomorphisms_backtrack(H1, G4)
```

```
180520
```

Summary

In my tests, Sagemath counts are accurate for H with 8 or less nodes as in example 4 and 6, while they can be only a small fraction or even 0 for H with 9 or more nodes. The backtrack algorithm suggested by ChatGPT is a clear improvement to brute force for medium sized H and G and is accurate as far as I know, but it's slower than Sagemath for large cases like example 5.