2012 年度 大問 5

hari64boli64 (hari64boli64@gmail.com)

2025 年 4 月 23 日

1　問題

N=1 の場合などには破綻するが、それは無視する。

2　解答

(1)

$$\det A = \begin{cases} 2 & (N \text{ is odd}) \\ 0 & (N \text{ is even}) \end{cases}$$

(2)

連結性より、$M \geq N-1$ であることが必要。
rank の最大値は N なので、$M \leq N$ であることが必要。
$M = N$ のとき、サイクルか木 $+1$ 本の辺であることが必要。
$M = N-1$ のとき、木であることが必要。
これらは共に十分であることは、rank の定義から自明。

(3)

奇数の時 $1/2$
偶数の時 1010 か 0101 のような形になる。

(4)

辺の重みとして正当なのは、$0, 1/2, 1$ のいずれかのみ。
よって、サイクルと単独な辺の和集合となる。

3　例

(2)

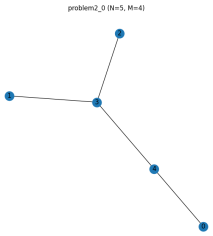

図 1　2-0

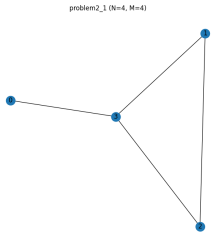

図 2　2-1

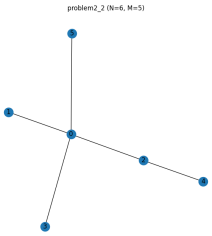

図 3　2-2

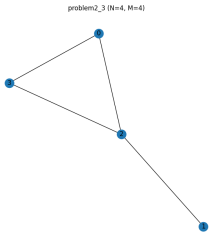

図 4　2-3

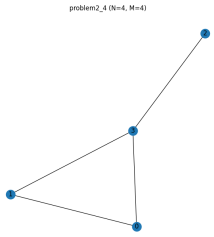

図 5　2-4

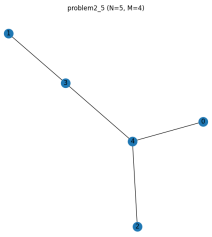

図 6　2-5

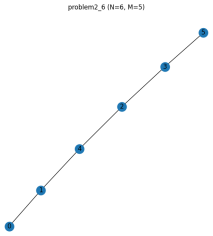

図 7　2-6

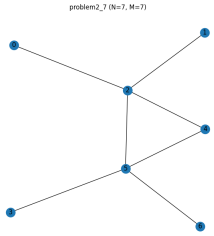

図 8　2-7

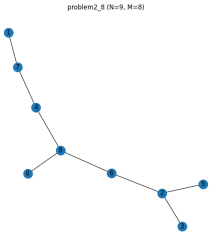

図 9　2-8

(4)

以下の実行例は、辺の重みを $\{0, 1/2, 1\}$ に限っている。なので、正当性の検証にはなっていない。あくまで、そのような制限下での解を列挙したものである。

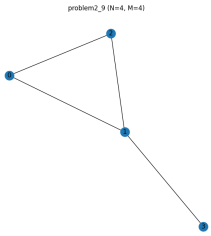

図 10　2-9

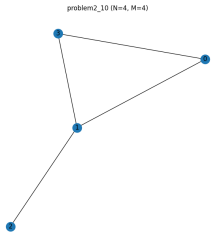

図 11　2-10

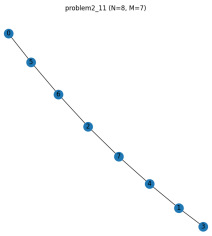

図 12　2-11

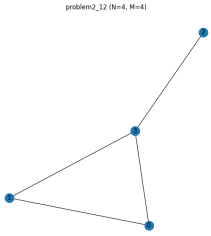

図 13　2-12

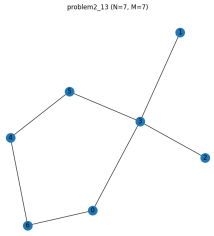

図 14　2-13

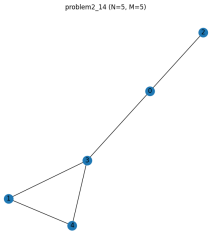

図 15　2-14

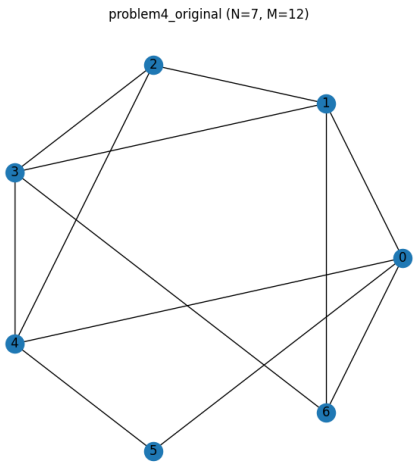

図 16　original

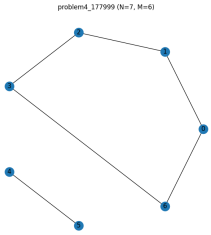

図 17　id=177999

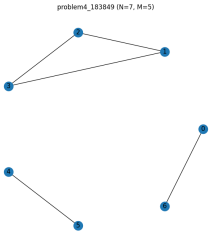

図 18　id=183849

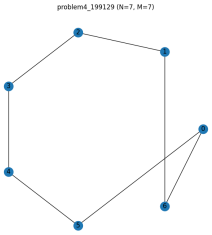

図 19　id=199129

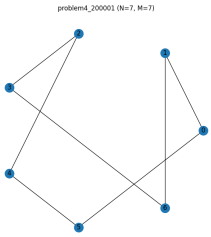

図 20　id=200001

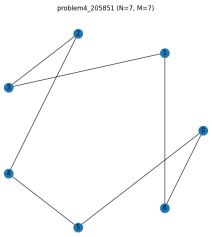

図 21　id=205851

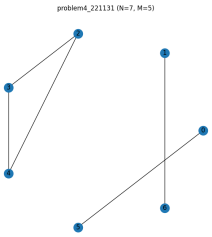

図 22　id=221131

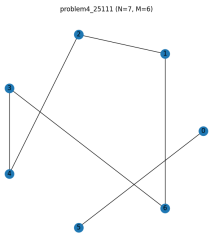

図 23　id=25111

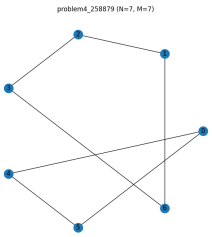

図 24　id=258879

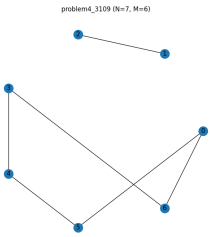

図 25　id=3109

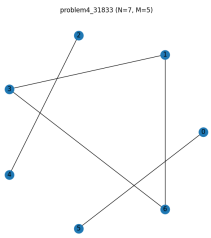

図 26　id=31833

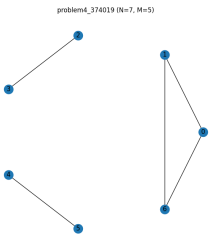

図 27　id=374019

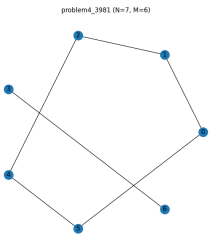

図 28　id=3981

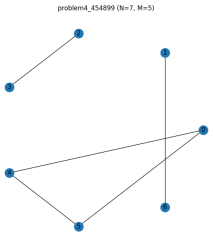

図 29　id=454899

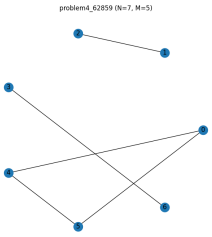

図 30　id=62859

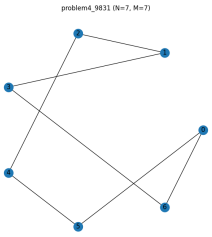

図 31　id=9831

4　おまけ

Listing 1　vis

```python
import os
import random
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from tqdm import tqdm
from typing import List
from itertools import product

os.chdir("情報理工/2012")


def isConnected(adj: List[List[int]]):
    seen = [False] * len(adj)

    def dfs(v):
        seen[v] = True
        for nv in adj[v]:
            if not seen[nv]:
                dfs(nv)

    dfs(0)
    return all(seen)


def makeGraph(N_: int = -1, M_: int = -1, notDupli: bool = False):
    # print("now making...")
    N = random.randint(2, 30) if N_ == -1 else N_
    M = random.randint(1, N * (N - 1) // 2) if M_ == -1 else M_
    adj = [[] for _ in range(N)]
```

```
    A = [[0 for _ in range(M)] for _ in range(N)]
    for j in range(M):
        u, v = random.sample(range(N), 2)
        assert u != v
        adj[u].append(v)
        adj[v].append(u)
        A[v][j] = 1
        A[u][j] = 1
    if not isConnected(adj):
        return makeGraph(N_, M_, notDupli)
    if notDupli and any(len(adj[v]) != len(set(adj[v])) for v in
        range(N)):
        return makeGraph(N_, M_, notDupli)
    return N, M, adj, A


def makeCycle():
    # print("now making...")
    N = random.randint(2, 30)
    M = N
    adj = [[] for _ in range(N)]
    A = [[0 for _ in range(M)] for _ in range(N)]
    for j in range(M):
        u = j
        v = (j + 1) % N
        adj[u].append(v)
        adj[v].append(u)
        A[v][j] = 1
        A[u][j] = 1
    assert isConnected(adj)
    return N, M, adj, A


def visualizeGraph(
    N: int, M: int, adj: List[List[int]], title: str, circular:
        bool = False
):
    g = nx.Graph()
    g.add_nodes_from(range(N))
    for u in range(N):
        for v in adj[u]:
            g.add_edge(u, v)
    plt.figure(figsize=(8, 8))
    plt.title(f"{title} ({N=}, {M=})")
    nx.draw(g, pos=nx.circular_layout(g) if circular else None,
        with_labels=True)
    plt.savefig(f"img_5/{title}.png")
```

```
    plt.close()


def problem1():
    for i in range(100 + 1):
        N, M, adj, A = makeCycle()
        assert N == M
        if i == 100:
            visualizeGraph(N, M, adj, "problem1")
        det = np.linalg.det(A)
        print(f"{N=} {det=}")
        assert det == (2 if N % 2 == 1 else 0)


def problem2():
    cnt = 0
    while True:
        N, M, adj, A = makeGraph()
        if M <= 3:  # trivial case
            continue
        rank = np.linalg.matrix_rank(A)
        if rank == M:
            print(f"{N=} {M=} {rank=}")
            visualizeGraph(N, M, adj, f"problem2_{cnt}")
            cnt += 1
            if cnt >= 15:
                break


def problem4():
    N, M, adj, A = makeGraph(N_=7, M_=12, notDupli=True)
    print(adj)
    A = np.array(A)
    visualizeGraph(N, M, adj, f"problem4_original", circular=True)
    for id, Xs in tqdm(enumerate(product([0, 1 / 2, 1], repeat=M)), total=3**M):
        if np.allclose(A @ np.array(Xs), np.ones(N)):
            print(f"{id=}")
            NVis = N
            MVis = M - Xs.count(0)
            adjVis = [[] for _ in range(NVis)]
            for j, x in enumerate(Xs):
                if x != 0:
                    u, v = np.where(A[:, j] == 1)[0]
                    adjVis[u].append(v)
                    adjVis[v].append(u)
```

```
            visualizeGraph(NVis, MVis, adjVis, f"problem4_{id}",
                circular=True)


def main():
    random.seed(64)
    np.random.seed(64)

    problem1()
    problem2()
    problem4()


if __name__ == "__main__":
    main()
```