

# 2012 年度 大問 5

hari64boli64 (hari64boli64@gmail.com)

2023 年 6 月 13 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)

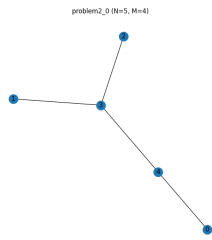


図 1 2-0

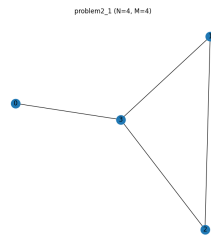


図 2 2-1

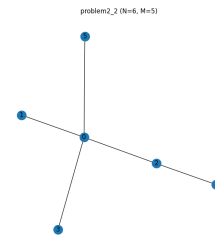


図 3 2-2

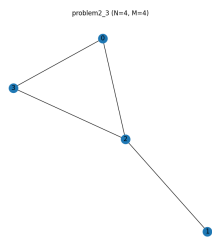


図 4 2-3

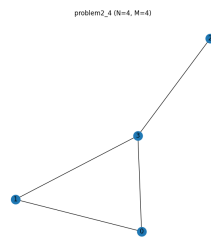


図 5 2-4

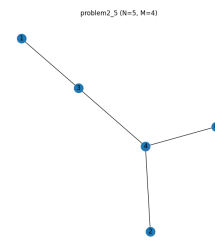


図 6 2-5

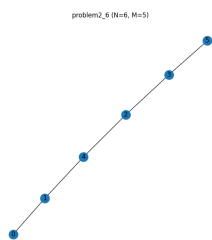


図 7 2-6

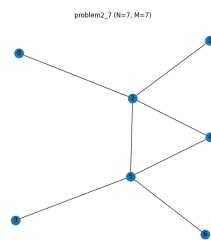


図 8 2-7

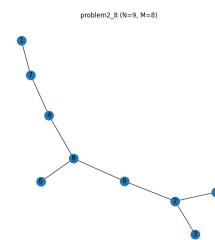


図 9 2-8

(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。

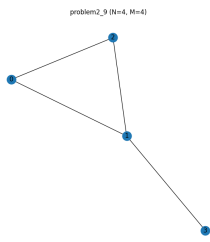


图 10 2-9

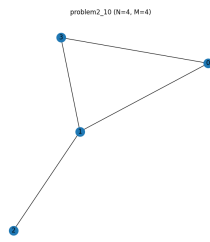


图 11 2-10

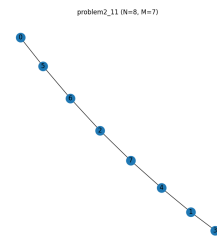


图 12 2-11

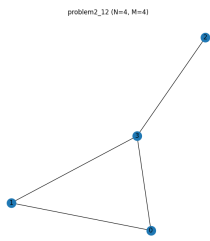


图 13 2-12

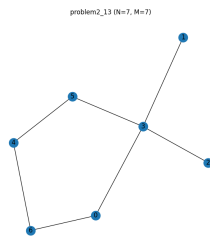


图 14 2-13

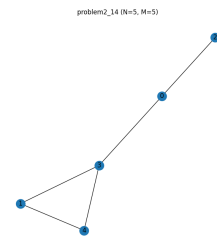


图 15 2-14

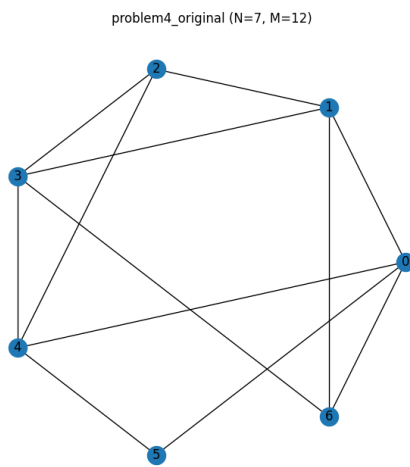
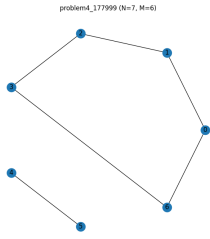
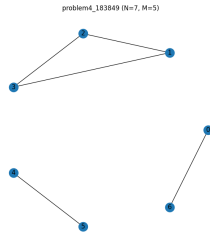


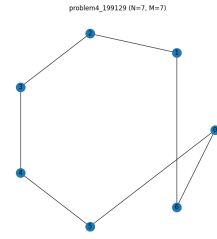
图 16 original



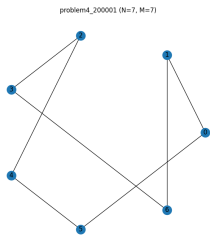
☒ 17 id=177999



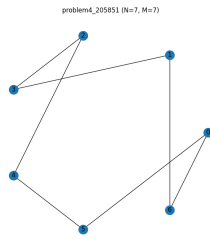
☒ 18 id=183849



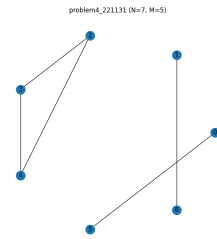
☒ 19 id=199129



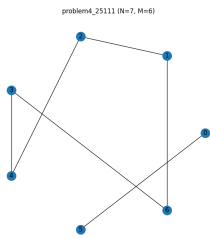
☒ 20 id=200001



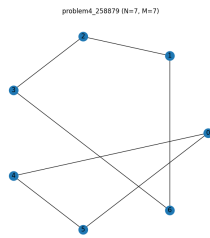
☒ 21 id=205851



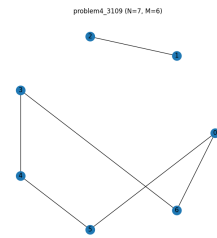
☒ 22 id=221131



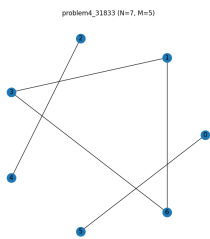
☒ 23 id=25111



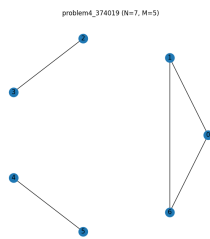
☒ 24 id=258879



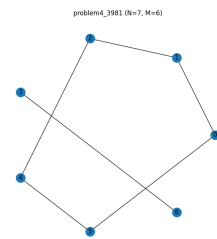
☒ 25 id=3109



☒ 26 id=31833



☒ 27 id=374019



☒ 28 id=3981

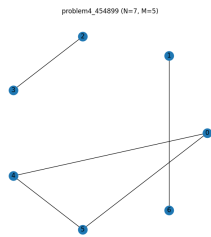


図 29 id=454899

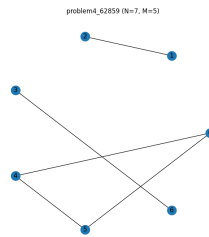


図 30 id=62859

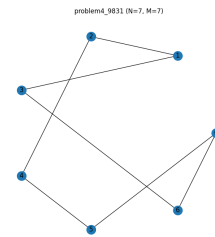


図 31 id=9831

## 4 おまけ

Listing 1 vis

```

1 import os
2 import random
3 import numpy as np
4 import networkx as nx
5 import matplotlib.pyplot as plt
6 from tqdm import tqdm
7 from typing import List
8 from itertools import product
9
10 os.chdir("情報理工/2012")
11
12
13 def isConnected(adj: List[List[int]]):
14     seen = [False] * len(adj)
15
16     def dfs(v):
17         seen[v] = True
18         for nv in adj[v]:
19             if not seen[nv]:
20                 dfs(nv)
21
22     dfs(0)
23     return all(seen)
24
25
26 def makeGraph(N_: int = -1, M_: int = -1, notDupli: bool = False)
27 :
28     # print("now making...")
29     N = random.randint(2, 30) if N_ == -1 else N_
30     M = random.randint(1, N * (N - 1) // 2) if M_ == -1 else M_
31     adj = [[] for _ in range(N)]

```

```

31     A = [[0 for _ in range(M)] for _ in range(N)]
32     for j in range(M):
33         u, v = random.sample(range(N), 2)
34         assert u != v
35         adj[u].append(v)
36         adj[v].append(u)
37         A[v][j] = 1
38         A[u][j] = 1
39     if not isConnected(adj):
40         return makeGraph(N_, M_, notDupli)
41     if notDupli and any(len(adj[v]) != len(set(adj[v]))) for v in
42         range(N)):
43         return makeGraph(N_, M_, notDupli)
44     return N, M, adj, A
45
46 def makeCycle():
47     # print("now making...")
48     N = random.randint(2, 30)
49     M = N
50     adj = [[] for _ in range(N)]
51     A = [[0 for _ in range(M)] for _ in range(N)]
52     for j in range(M):
53         u = j
54         v = (j + 1) % N
55         adj[u].append(v)
56         adj[v].append(u)
57         A[v][j] = 1
58         A[u][j] = 1
59     assert isConnected(adj)
60     return N, M, adj, A
61
62
63 def visualizeGraph(
64     N: int, M: int, adj: List[List[int]], title: str, circular:
65     bool = False
66 ):
67     g = nx.Graph()
68     g.add_nodes_from(range(N))
69     for u in range(N):
70         for v in adj[u]:
71             g.add_edge(u, v)
72     plt.figure(figsize=(8, 8))
73     plt.title(f"{title}_␣({N=},␣{M=})")
74     nx.draw(g, pos=nx.circular_layout(g) if circular else None,
75             with_labels=True)
76     plt.savefig(f"img_5/{title}.png")

```

```

75     plt.close()
76
77
78     def problem1():
79         for i in range(100 + 1):
80             N, M, adj, A = makeCycle()
81             assert N == M
82             if i == 100:
83                 visualizeGraph(N, M, adj, "problem1")
84                 det = np.linalg.det(A)
85                 print(f"{N=}_{det=}")
86                 assert det == (2 if N % 2 == 1 else 0)
87
88
89     def problem2():
90         cnt = 0
91         while True:
92             N, M, adj, A = makeGraph()
93             if M <= 3: # trivial case
94                 continue
95             rank = np.linalg.matrix_rank(A)
96             if rank == M:
97                 print(f"{N=}_{M=}_{rank=}")
98                 visualizeGraph(N, M, adj, f"problem2_{cnt}")
99                 cnt += 1
100                if cnt >= 15:
101                    break
102
103
104     def problem4():
105         N, M, adj, A = makeGraph(N_=7, M_=12, notDupli=True)
106         print(adj)
107         A = np.array(A)
108         visualizeGraph(N, M, adj, f"problem4_original", circular=True)
109         for id, Xs in tqdm(enumerate(product([0, 1 / 2, 1], repeat=M)), total=3**M):
110             if np.allclose(A @ np.array(Xs), np.ones(N)):
111                 print(f"{id=}")
112                 NVis = N
113                 MVis = M - Xs.count(0)
114                 adjVis = [[] for _ in range(NVis)]
115                 for j, x in enumerate(Xs):
116                     if x != 0:
117                         u, v = np.where(A[:, j] == 1)[0]
118                         adjVis[u].append(v)
119                         adjVis[v].append(u)

```

```
120         visualizeGraph(NVis, MVis, adjVis, f"problem4_{id}",
121                           circular=True)
122
123     def main():
124         random.seed(64)
125         np.random.seed(64)
126
127         problem1()
128         problem2()
129         problem4()
130
131
132     if __name__ == "__main__":
133         main()
```