```
from random import choice
from itertools import combinations
```

algorithm

```
def contract(graph, u, v):
    aux, w = [], f'{u},{v}'
    for x, y in graph:
        x = w if x in [u, v] else x
        y = w if y in [u, v] else y
        if x < y:
            aux.append((x, y))
        elif x > y:
            aux.append((y, x))
    return aux
```

```
def mincut(graph, n):
    components, cost = ['', ''], float('inf')

    # n^2 attempts
    for i in range(n * n):
        aux = graph

        # remove edges one by one
        while len(set(aux)) > 1:
            aux = contract(aux, *choice(aux))

            # min cut so far
            if len(aux) < cost:
                components, cost = aux[0], len(aux)

    return components, cost
```

generate graph

```
# fully connected
nodes_a = [f'A{i}' for i in range(20)]
graph_a = [(u, v) for u, v in combinations(nodes_a, 2)]

# fully connected
nodes_b = [f'B{i}' for i in range(20)]
graph_b = [(u, v) for u, v in combinations(nodes_b, 2)]

# interconnections
graph_c = [(choice(nodes_a), choice(nodes_b)) for i in range(10)]

graph = graph_a + graph_b + graph_c
```

run

```
components, cost = mincut(graph, 40)
print('best cut:', cost)
print('component #1:', components[0])
print('component #2:', components[1])
```

```
best cut: 10
component #1: A0,A3,A4,A7,A9,A16,A19,A2,A1,A5,A12,A18,A17,A8,A15,A13,A6,A10,A11,A14
component #2: B0,B1,B16,B12,B15,B4,B17,B19,B9,B3,B5,B14,B18,B2,B11,B8,B10,B7,B6,B13
```