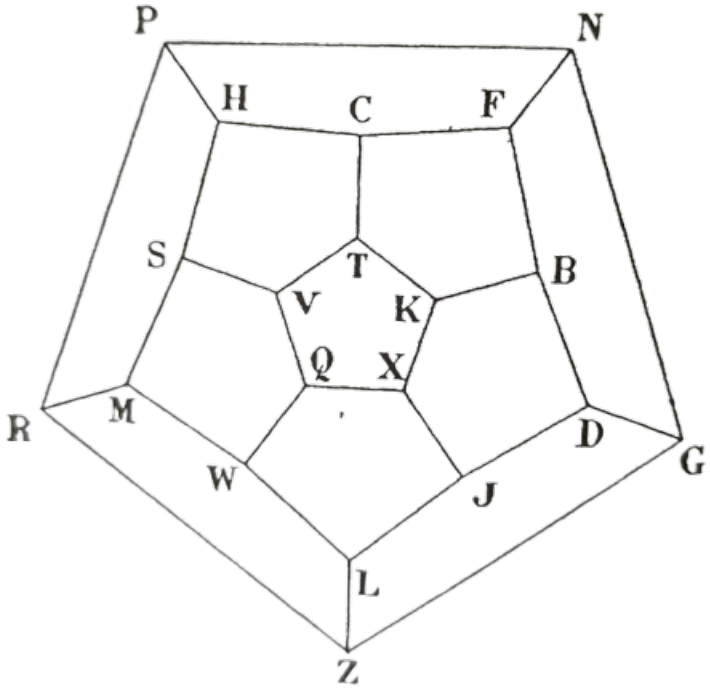



```
b 0  [d f k] = [3 4 8]
c 1  [f h t] = [4 6 16]
d 2  [b g j] = [1 5 7]
f 3  [b c n] = [1 2 11]
g 4  [d n z] = [3 11 20]
h 5  [c p s] = [2 12 15]
j 6  [d l x] = [3 9 19]
k 7  [b j t] = [1 7 16]
l 8  [j w z] = [7 18 20]
m 9  [r s w] = [14 15 18]
n 10 [f g p] = [4 5 12]
p 11 [h n r] = [6 11 14]
q 12 [v w x] = [17 18 19]
r 13 [m p z] = [10 12 20]
s 14 [h m v] = [6 10 17]
t 15 [c k v] = [2 8 17]
v 16 [q s t] = [13 15 16]
w 17 [l m q] = [9 10 13]
x 18 [j k q] = [7 8 13]
z 19 [g l r] = [5 9 14]
```

```
     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 1 [ 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 2 [ 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
 3 [ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 4 [ 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 5 [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]
 6 [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0]
 7 [ 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
 8 [ 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
 9 [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1]
10 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0]
11 [ 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
12 [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]
13 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0]
14 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1]
15 [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
```

```
16 [ 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
17 [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0]
18 [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
19 [ 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
20 [ 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]



# This code is contributed by divyesh072019.

import time

def isSafe(v, graph, path, pos):
    if graph[path[pos - 1]][v] == 0:
        return False
    for i in range(pos):
        if path[i] == v:
            return False
    return True

hasCycle = False

def hamCycle(graph):
    global hasCycle
    hasCycle = False
    path = []
    path.append(0)
    visited = [False]*(len(graph))
    for i in range(len(visited)):
        visited[i] = False
    visited[0] = True
    FindHamCycle(graph, 1, path, visited)
    if hasCycle:
        print("No Hamiltonian Cycle" + "possible ")
        return

def FindHamCycle(graph, pos, path, visited):
    if pos == len(graph):
        if graph[path[-1]][path[0]] != 0:
            path.append(0)
            for i in range(len(path)):
                print(path[i], end = " ")
            print()
            path.pop()
            hasCycle = True
        return
    for v in range(len(graph)):
        if isSafe(v, graph, path, pos) and not visited[v]:
            path.append(v)
            visited[v] = True
            FindHamCycle(graph, pos + 1, path, visited)
            visited[v] = False
            path.pop()

tic = time.time()
graph = [[ 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [ 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
         [ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [ 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
         [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
         [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
         [ 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
         [ 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
         [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],
```

```
          [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0],
          [ 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
          [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
          [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
          [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1],
          [ 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0],
          [ 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
          [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0],
          [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0],
          [ 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
          [ 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]]
hamCycle(graph)
tac = time.time()
print('\n',tac-tic,' s.')
```

```
0  2  4  10  3  1  5  11  13  19  8  6  18  12  17  9  14  16  15  7  0
0  2  4  10  3  1  15  16  12  17  9  14  5  11  13  19  8  6  18  7  0
0  2  4  10  11  5  14  16  12  17  9  13  19  8  6  18  7  15  1  3  0
0  2  4  10  11  13  19  8  6  18  7  15  16  12  17  9  14  5  1  3  0
0  2  4  19  8  6  18  7  15  1  5  14  16  12  17  9  13  11  10  3  0
0  2  4  19  8  6  18  12  17  9  13  11  10  3  1  5  14  16  15  7  0
0  2  4  19  13  9  14  5  11  10  3  1  15  16  12  17  8  6  18  7  0
0  2  4  19  13  9  14  16  12  17  8  6  18  7  15  1  5  11  10  3  0
0  2  4  19  13  9  17  8  6  18  12  16  14  5  11  10  3  1  15  7  0
0  2  4  19  13  11  10  3  1  5  14  9  17  8  6  18  12  16  15  7  0
0  2  6  8  17  9  13  19  4  10  11  5  14  16  12  18  7  15  1  3  0
0  2  6  8  17  9  14  5  11  13  19  4  10  3  1  15  16  12  18  7  0
0  2  6  8  17  9  14  16  12  18  7  15  1  5  11  13  19  4  10  3  0
0  2  6  8  17  12  18  7  15  16  14  9  13  19  4  10  11  5  1  3  0
0  2  6  8  19  4  10  3  1  15  16  14  5  11  13  9  17  12  18  7  0
0  2  6  8  19  4  10  11  13  9  17  12  18  7  15  16  14  5  1  3  0
0  2  6  18  7  15  1  5  11  13  9  14  16  12  17  8  19  4  10  3  0
0  2  6  18  7  15  16  12  17  8  19  4  10  11  13  9  14  5  1  3  0
0  2  6  18  12  16  14  5  11  13  9  17  8  19  4  10  3  1  15  7  0
0  2  6  18  12  17  8  19  4  10  3  1  5  11  13  9  14  16  15  7  0
0  3  1  5  11  10  4  2  6  18  12  17  8  19  13  9  14  16  15  7  0
0  3  1  5  14  9  17  8  19  13  11  10  4  2  6  18  12  16  15  7  0
0  3  1  15  16  12  17  8  19  13  9  14  5  11  10  4  2  6  18  7  0
0  3  1  15  16  14  5  11  10  4  2  6  8  19  13  9  17  12  18  7  0
0  3  10  4  2  6  8  19  13  11  5  1  15  16  14  9  17  12  18  7  0
0  3  10  4  2  6  18  12  16  14  9  17  8  19  13  11  5  1  15  7  0
0  3  10  11  5  1  15  16  14  9  13  19  4  2  6  8  17  12  18  7  0
0  3  10  11  13  9  14  5  1  15  16  12  17  8  19  4  2  6  18  7  0
0  3  10  11  13  9  17  8  19  4  2  6  18  12  16  14  5  1  15  7  0
0  3  10  11  13  19  4  2  6  8  17  9  14  5  1  15  16  12  18  7  0
0  7  15  1  3  10  4  19  8  17  9  13  11  5  14  16  12  18  6  2  0
0  7  15  1  3  10  11  5  14  16  12  18  6  8  17  9  13  19  4  2  0
0  7  15  1  5  11  13  19  8  17  9  14  16  12  18  6  2  4  10  3  0
0  7  15  1  5  14  16  12  18  6  2  4  19  8  17  9  13  11  10  3  0
0  7  15  16  12  18  6  2  4  10  11  13  19  8  17  9  14  5  1  3  0
0  7  15  16  12  18  6  8  17  9  14  5  1  3  10  11  13  19  4  2  0
0  7  15  16  14  5  1  3  10  11  13  9  17  12  18  6  8  19  4  2  0
0  7  15  16  14  9  13  11  5  1  3  10  4  19  8  17  12  18  6  2  0
0  7  15  16  14  9  13  19  8  17  12  18  6  2  4  10  11  5  1  3  0
0  7  15  16  14  9  17  12  18  6  8  19  13  11  5  1  3  10  4  2  0

0.0317387580871582  s.
```

Après conversion nombres en lettres, voici les 40 solutions trouvées par le programme de l’internaute divyesh072019. On dessine les 6 premières.

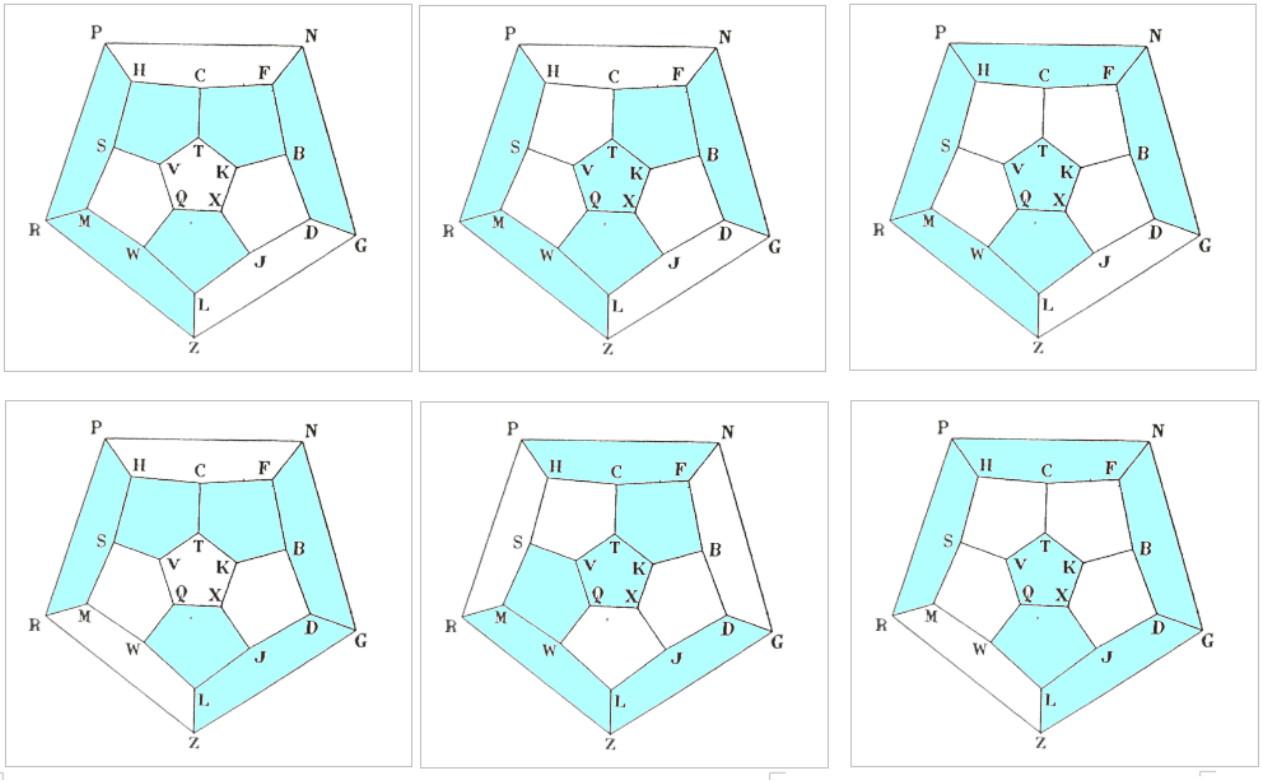
1) B D G N F C H P R Z L J X Q W M S V T K B
2) B D G N F C T V Q W M S H P R Z L J X K B
3) B D G N P H S V Q W M R Z L J X K T C F B

4) B D G N P R Z L J X K T V Q W M S H C F B
5) B D G Z L J X K T C H S V Q W M R P N F B
6) B D G Z L J X Q W M R P N F C H S V T K B
7) B D G Z R M S H P N F C T V Q W L J X K B
8) B D G Z R M S V Q W L J X K T C H P N F B
9) B D G Z R M W L J X Q V S H P N F C T K B
10) B D G Z R P N F C H S M W L J X Q V T K B
11) B D J L W M R Z G N P H S V Q X K T C F B
12) B D J L W M S H P R Z G N F C T V Q X K B
13) B D J L W M S V Q X K T C H P R Z G N F B
14) B D J L W Q X K T V S M R Z G N P H C F B
15) B D J L Z G N F C T V S H P R M W Q X K B
16) B D J L Z G N P R M W Q X K T V S H C F B
17) B D J X K T C H P R M S V Q W L Z G N F B
18) B D J X K T V Q W L Z G N P R M S H C F B
19) B D J X Q V S H P R M W L Z G N F C T K B
20) B D J X Q W L Z G N F C H P R M S V T K B
21) B F C H P N G D J X Q W L Z R M S V T K B
22) B F C H S M W L Z R P N G D J X Q V T K B
23) B F C T V Q W L Z R M S H P N G D J X K B
24) B F C T V S H P N G D J L Z R M W Q X K B
25) B F N G D J L Z R P H C T V S M W Q X K B
26) B F N G D J X Q V S M W L Z R P H C T K B
27) B F N P H C T V S M R Z G D J L W Q X K B
28) B F N P R M S H C T V Q W L Z G D J X K B
29) B F N P R M W L Z G D J X Q V S H C T K B
30) B F N P R Z G D J L W M S H C T V Q X K B
31) B K T C F N G Z L W M R P H S V Q X J D B
32) B K T C F N P H S V Q X J L W M R Z G D B
33) B K T C H P R Z L W M S V Q X J D G N F B
34) B K T C H S V Q X J D G Z L W M R P N F B
35) B K T V Q X J D G N P R Z L W M S H C F B
36) B K T V Q X J L W M S H C F N P R Z G D B
37) B K T V S H C F N P R M W Q X J L Z G D B
38) B K T V S M R P H C F N G Z L W Q X J D B
39) B K T V S M R Z L W Q X J D G N P H C F B
40) B K T V S M W Q X J L Z R P H C F N G D B

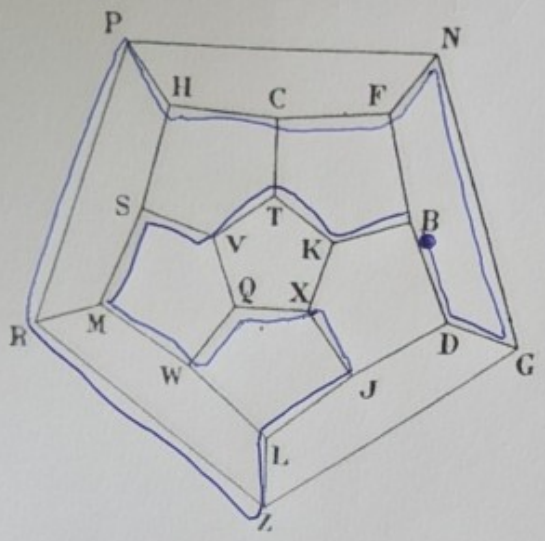
Pourquoi le programme de calcul des cycles hamiltoniens trouve-t-il 40 solutions ?
Il faut penser à la façon dont Hermary coupe le dodécaèdre en deux demi-dodécaèdre et imaginer qu'on colorie l'un des deux demi-dodécaèdres en noir. Alors ce demi-dodécaèdre contient ou pas le pentagone central, ce qui fait deux possibilités combinatoires, on voit que dans la couronne centrale des pentagones de taille moyenne, le demi-dodécaèdre contient parfois 1, parfois 2, parfois 3 mais jamais 4 pentagones centraux, et une fois ça fixé, il ne lui reste pas de choix pour le nombre de pentagones à colorier sur la couronne externe, puisque ce nombre est déterminé (il est égal au complémentaire à 6 du nombre de pentagones coloriés jusque-là). On a donc 2x3 = 6 possibilités. Je n'arrive pas à retomber sur mes pattes car j'en trouve 8.
C'est sans compter que le programme a considéré comme différentes des cycles égaux à rotation d'un cinquième de tour près. D'où les 40 possibilités.

Il vaut mieux utiliser le raisonnement fourni dans le livre Théorie des graphes Problèmes, théorèmes et algorithmes de Olivier Cogis et Claudine Robert : la méthode de hamilton de substitution des lettres amène au chemin
1 = gggdddgdgdgggdddgdgd
Aucun facteur strict de ce mot n'est égal à 1 et à permutation cyclique des lettres, ce mot amène 20 autres chemins possibles et à permutation de d et de g, on multiplie ça par 2, et ça donne 40.

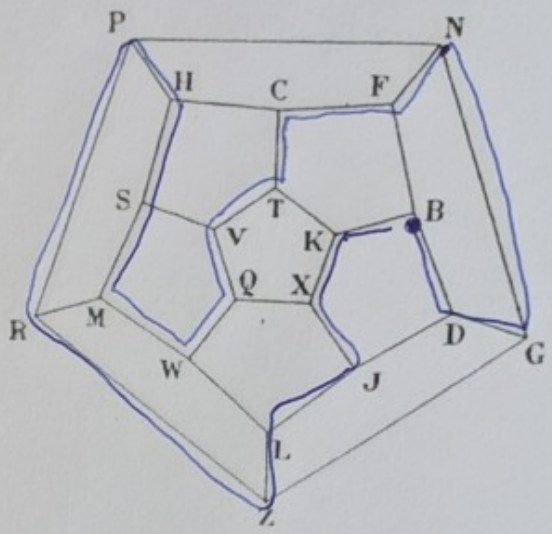
(j’aurais pensé que c’était plutôt à ordre de lecture dans un sens ou dans l’autre que par interversion d ↔ g, je ne sais pas si c’est équivalent). Bizarrement, pour le pentagone de Penrose, je trouve 52 possibilités.



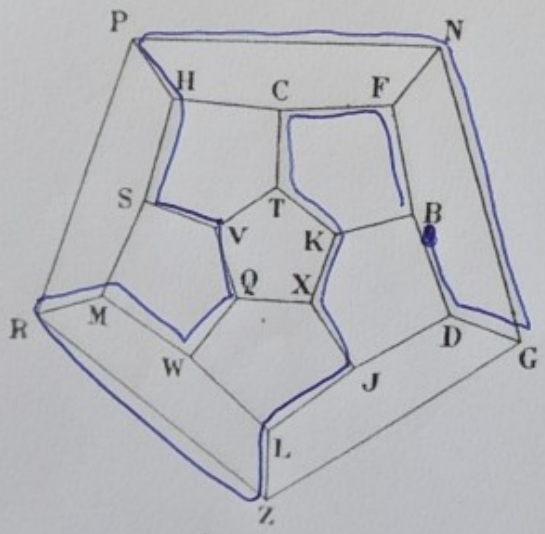
1


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

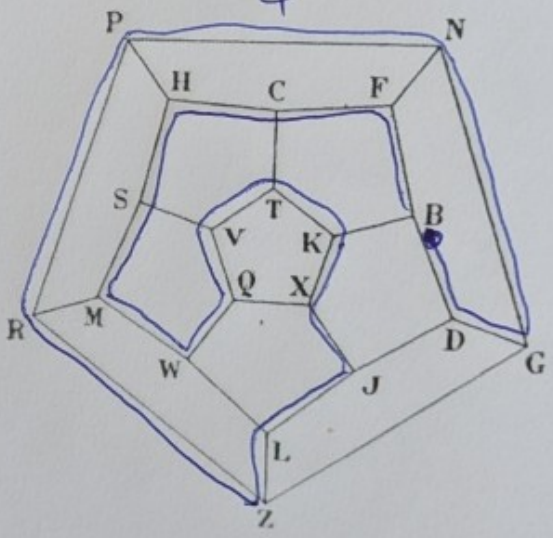
2


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

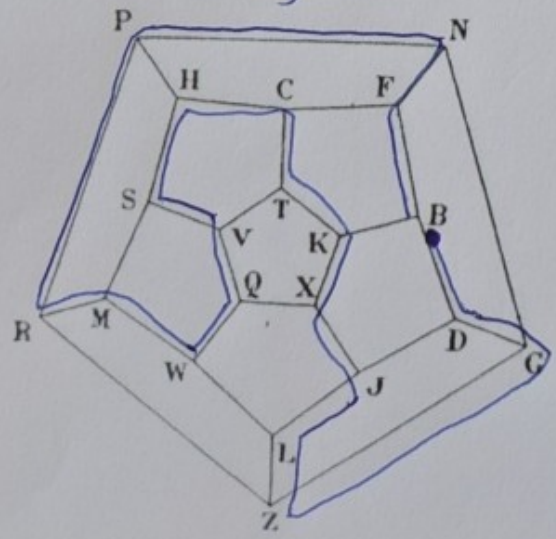
3


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

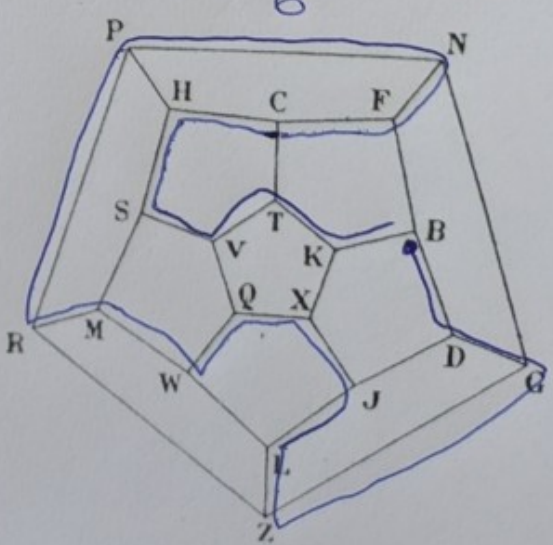
4


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

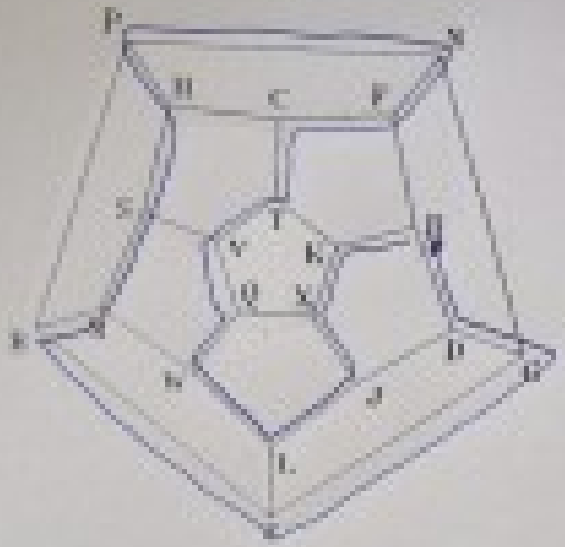
5


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

6


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
D
G
W
J
L
Z

7


P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

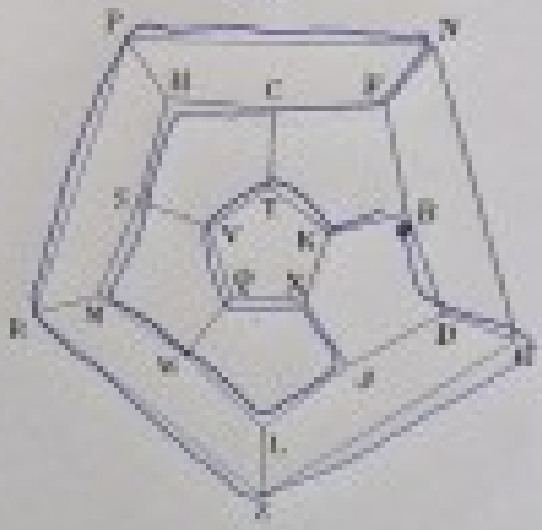
8


P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

9


P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

10


P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

11

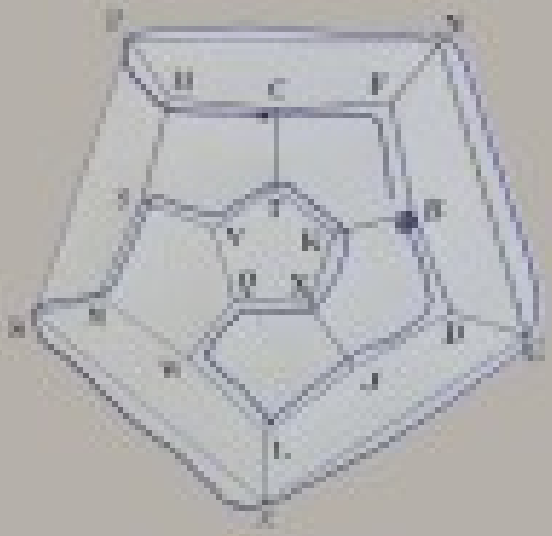

P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

12


P
N
H
C
F
S
T
V
K
Q
X
R
M
W
J
D
G
L

A3

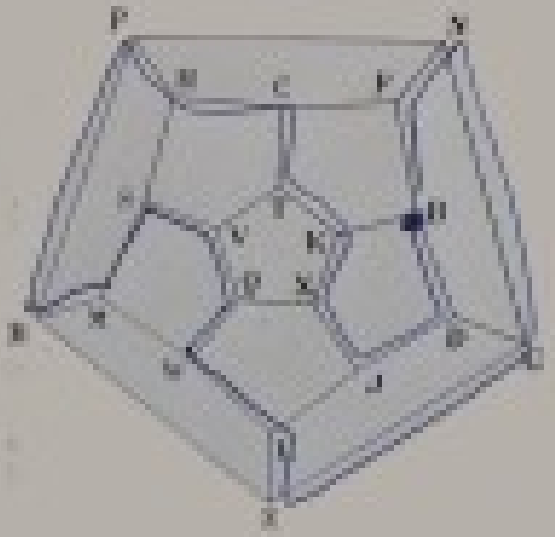
A4



A5

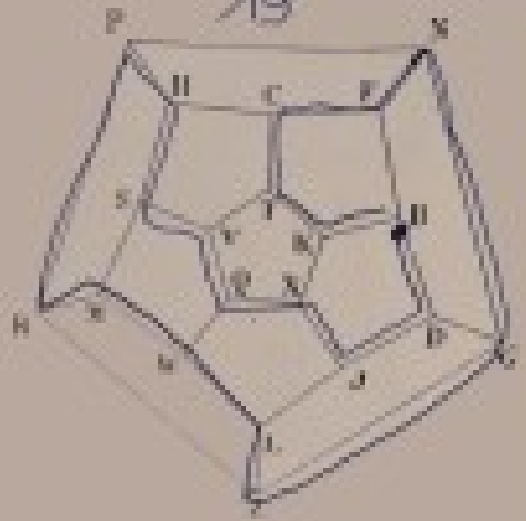
A6

A7

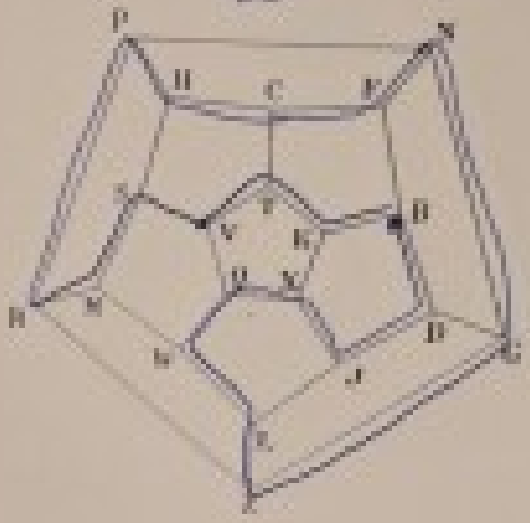


A8

19



20

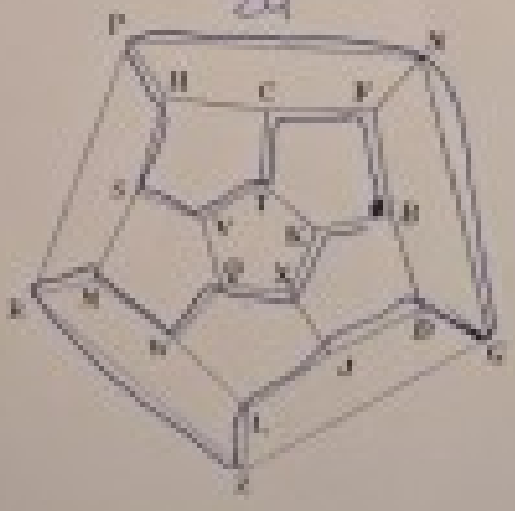


21

22

23

24



25

26

27

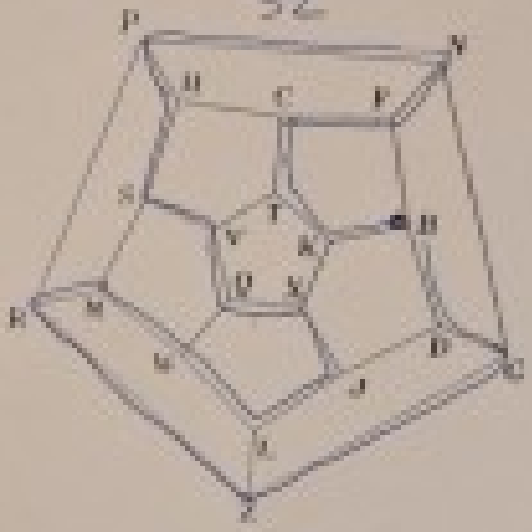
28

29

30

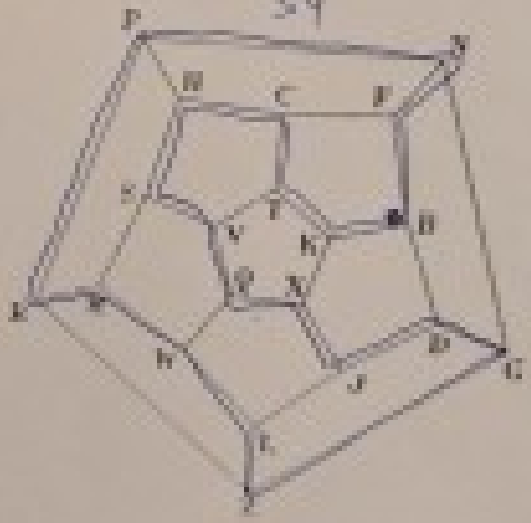
31

32



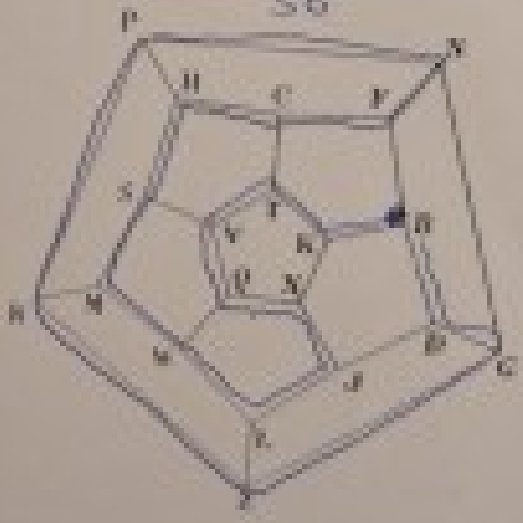
33

34

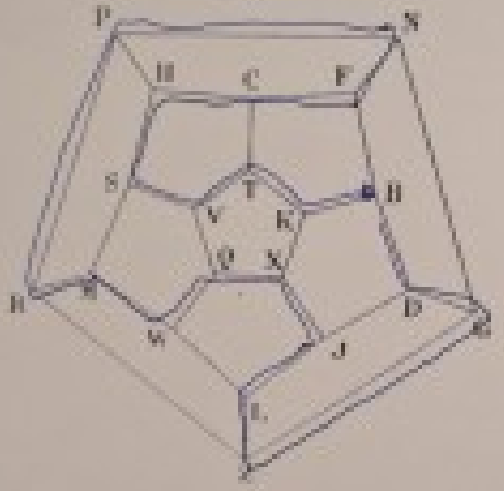


35

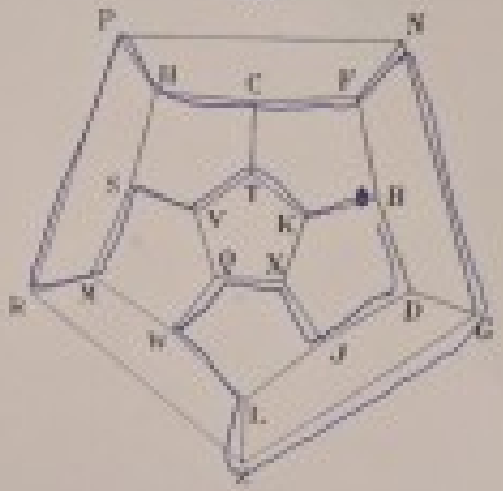
36



37


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
W
D
G
J
L

38


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
W
D
G
J
L

39


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
W
D
G
J
L

40


P
N
H
C
F
S
T
B
V
K
Q
X
R
M
W
D
G
J
L