TD 4 complexité.

1 _ Transversale et zone vide :

premier algo :

un transversale est un sous-ensemble S' de S tel que pour tout arête $\{x,y\} \in A$, alors $x \in S'$ ou $y \in S'$.

Un transversale est un sous-ensemble de sommets partageant au moins un sommet avec chaque arête du graphe

```
algorithme-det-transversale(G, ℓ)
    T = tableau de booleane de longeur G.n
    i = 1
    Tant que i ≤ K faire
        T[i] = guess(0,1)
        i ← i+1
    pour x ← 1 a G.N
        pour y ← 1 a G.N
        Si (G[x][y] ← 1 and (!T[x] and !T[y]))
            return trouve non
    sinon
            return trouve oui
        si (trouve ← oui)
        pour x.
```

Deuxieme algo :

```
non-det-transversal(G, ℓ)
    t: tableaux de taille (G.size), count = 0
    for i in range(G.size)
        t[i] = guess(0,1)
            count += t[i]
        if count > ℓ
            return false
        For x in range(G.size)
          For y in range(G.size)
            if G(A[x][y] and (not t[x]) and
               (not t[y]))
            return false.
return true.
```

a1 _ b)

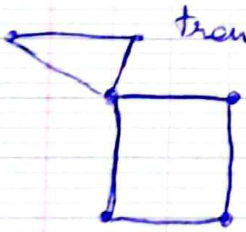
Check _ transversal ($G, \mathcal{E}, T$) artific 1.

count 1 = P
for i in range (G.size)
count += t[i]
if count > K
return false
For (la même chose).

↳ On a un algorithme polynomial ($O(n)$) avec un artificat polynomial : $O(n)$ il verifier qu'on a une seul problem.

Q1_2 : Zone vide (G, K)
return algo tres (G, n - E);

① Zone vide (G, E)
t = Algo tres (G, E)
for i in range (G, N)
Si T[i] alors T[i] = 0
T[i] = 1;



algo de test pour transversal (G, K, X)
C = K
pour x = 1 a G.n faire
if X[x] = 1 alors
c := c - 1.
Si c != 0 alors
retour faux.
pour x := 1 a G.n faire
pour y := x + 1 a G.n faire

si G.A [x][y] et non X[x] et non X[y] alors
retourner faux
retourner vrai.

— supposons que Solution - transversal (G, k) retourne le problème $O(f(x))$

```
Solution - Z.V (G, k)
    retourner Trans (G. G. n - k).
```

① Partition et Somme :

Q) A possède-t-il un sous-ensemble B dont la somme de la taille des ses éléments est exactement égale à la somme de la taille des éléments de A qui ne sont pas dans B ?

```
check - partition (t, s)
    sum0 = 0, sum1 = 0
    for i in range (t. length)
        if S[i] sum1 + = t[i]
            sum0 + = t[i]
        return sum1 = sum0.
```

Question 2.2 : A est une fct t : A → N est un entier K, Donner un algo avec certificat pour somme

```
Somme - certificat (t, K, c)
    check = 0.
    for i in range (t. length)

        if check + = c[i] t[i]
    return check = = K.
```

conclusion

↳ C est un certificat de taille (n), l'algorithme O(n) Donc Somme est dans NP.

2.2.1 : Supposons que sol-sum (t, l) résout le p Somme.

Somme ≤ Partition.

```
Sol_partition (t)
  ↳ count = 0,
  for i in range (t.length)
      count += t[i]
  ~~if~~
  if count % 2 == 1 then (return false
  return sol-sum (t, count/2)
```

~~On a une réduction polynomiale de partition vers somme, si partition est NP-difficile alors somme aussi.~~

2.2.2) Supposon

On a une reduction Polynomial de partition vers somme, si partition est NP-difficile alors somme aussi.