Complexité des problèmes

III. Les problèmes 2SUM et 3SUM

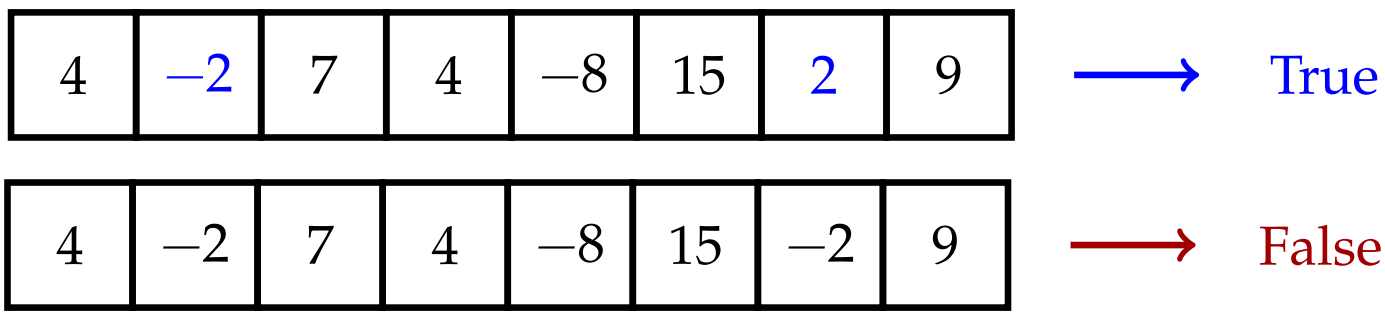
Cyril Nicaud

M1 Informatique, 2014–2015

Problème 2SUM

2SUM

Soit T un tableau contenant n nombres. Le problème `2SUM` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.



Problème 2SUM – modèle d'ordinateur

2SUM

Soit T un tableau contenant n nombres. Le problème `2SUM` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

- On se place dans un modèle où on peut en temps constant:
  - Faire un opération élémentaire sur les nombres : $+$ $\times$ ...
  - Accéder à une case du tableau
  - Faire les opérations élémentaires sur les indices
- On ne précise pas de quel type de "nombres" il s'agit (entiers, réels, ...). On demande juste à ce que le modèle ci-dessus soit respecté

Problème 2SUM – Solution naïve

2SUM

Soit T un tableau contenant n nombres. Le problème `2SUM` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

```
def 2SUM(T):
  for x in T:
    for y in T:
      if x + y == 0:
        return True
  return False
```

Complexité : $\mathcal{O}(n^2)$

Problème 2SUM – Tableau trié

2SUM-Sorted

Soit T un tableau trié contenant n nombres. Le problème `2SUM` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

```
def 2SUMSortedSimple(T):
  for x in T:
    if dichotomie(T,-x):
      return True
  return False
```

Complexité: $\mathcal{O}(n \log n)$

Cela donne une solution en $\mathcal{O}(n \log n)$ pour `2SUM`, en commençant par trier le tableau.

Problème 2SUM – Tableau trié linéaire

2SUM-Sorted

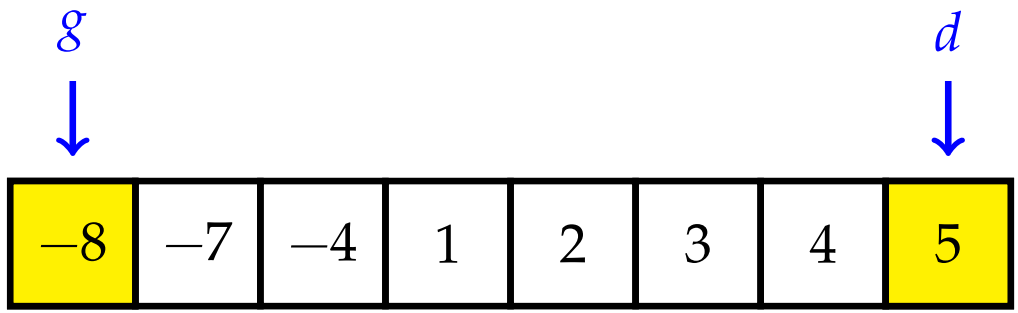
Soit T un tableau trié contenant n nombres. Le problème 2SUM consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

```
def 2SUMSorted(T):
  g, d = 0, len(T)-1
  while g <= d:
    if T[g] + T[d] == 0:
      return True
    if T[g] + T[d] < 0:
      g += 1
    else:
      d -= 1
  return False
```

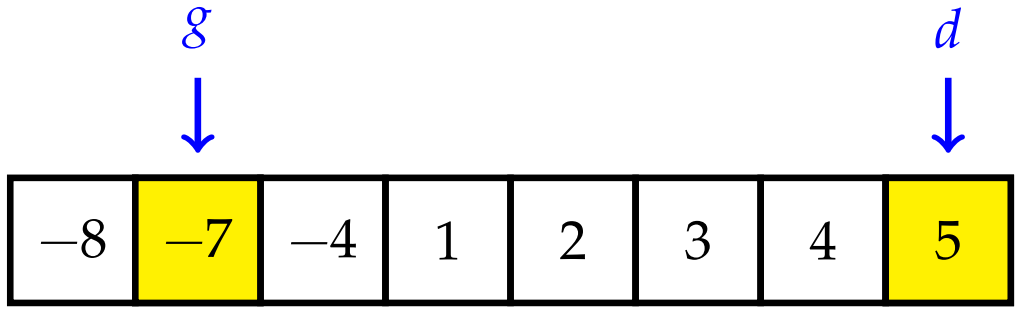
Complexité: $\mathcal{O}(n)$.

On ne peut pas s'en servir pour améliorer 2SUM, puisque l'étape de tri coûte $\mathcal{O}(n \log n)$.

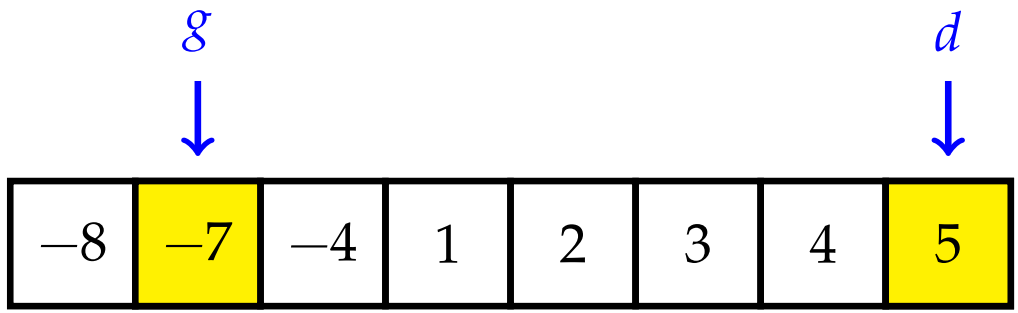
Problème 2SUM Sorted



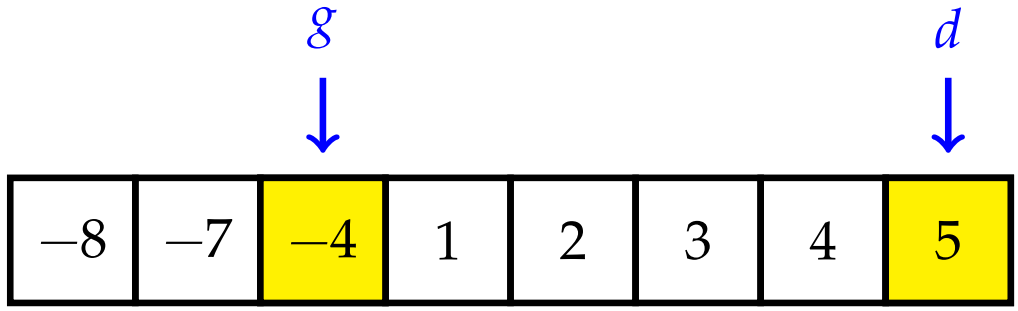
$$T[g] + T[d] < 0 \Rightarrow g \mathrel{+}= 1$$



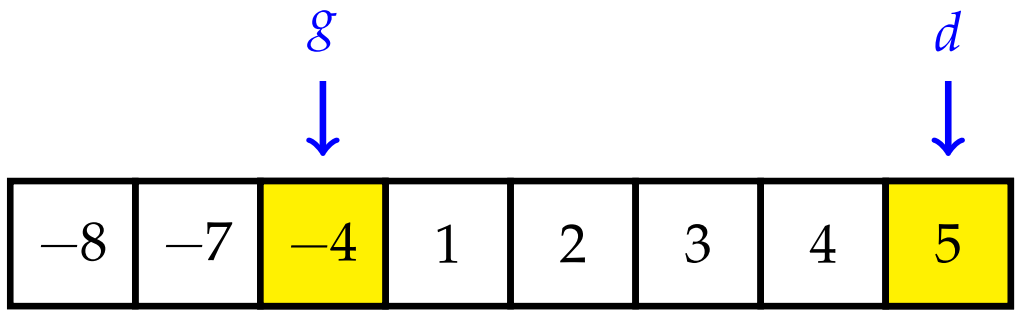
Problème 2SUM Sorted



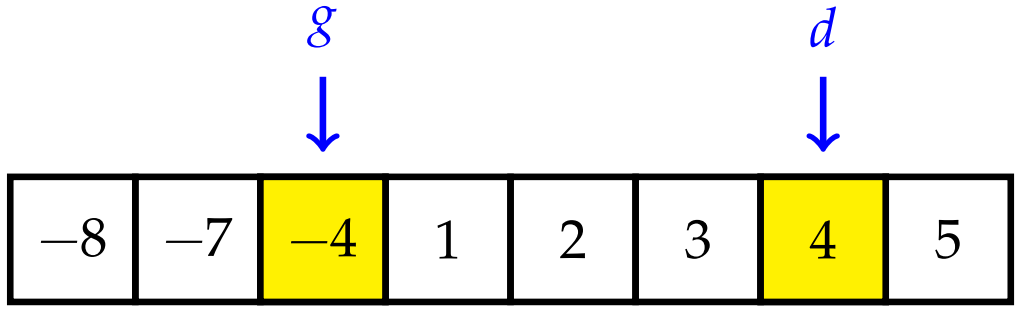
$$T[g] + T[d] < 0 \Rightarrow g \mathrel{+}= 1$$



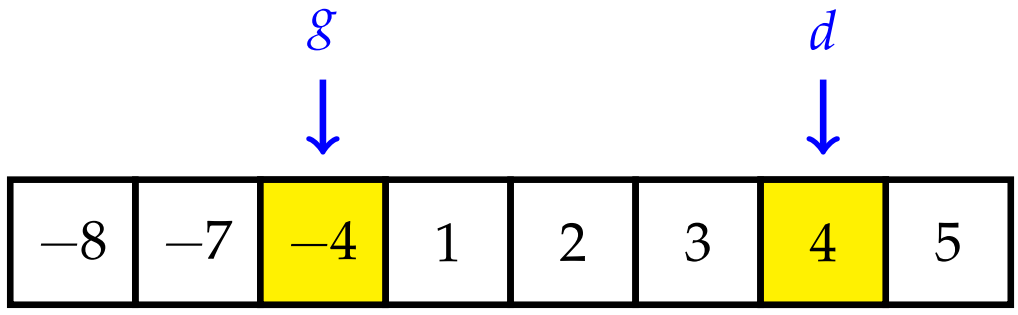
Problème 2SUM Sorted



$$T[g] + T[d] > 0 \Rightarrow d -= 1$$



Problème 2SUM Sorted



- $T[g] + T[d] = 0$
- C'est terminé, le programme retourne `True`

Preuve de correction

```
def 2SUMSorted(T):
  g, d = 0, len(T)-1
  while g <= d:
    if T[g] + T[d] == 0:
      return True
    if T[g] + T[d] < 0:
      g += 1
    else:
      d -= 1
  return False
```

Si l'algorithme retourne True, le résultat est correct.

S'il retourne False, la preuve se fait par récurrence sur la taille du tableau.

Récapitulatif 2SUM

2SUM

Soit T un tableau contenant n nombres. Le problème `2SUM` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

2SUM Sorted

Soit T un tableau trié contenant n nombres. Le problème `2SUMSorted` consiste à indiquer s'il existe x et y dans T tels que $x + y = 0$.

- On peut résoudre `2SUM` en temps $\mathcal{O}(n \log n)$
- On peut résoudre `2SUMSorted` en temps $\mathcal{O}(n)$

Variantes de 2SUM

- La première variante consiste à demander s'il y a $x, y \in T$ tels que $x + y = c$, où c est une constante.
- La seconde variante consiste à avoit deux tableaux S et T et de chercher $x \in S$ et $y \in T$ tels que $x + y = 0$ (ou $x + y = c$).

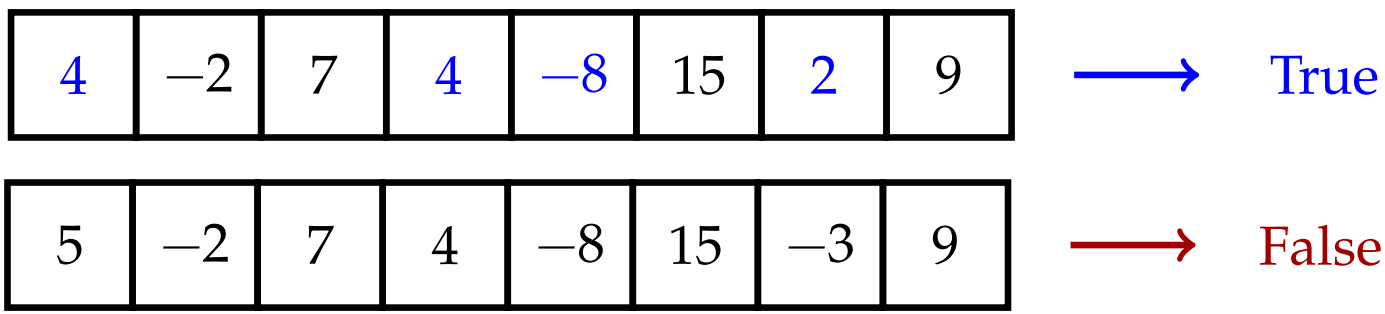
Les résultats sont identiques (à vérifier en TD). Pour les deux variantes, où n est la somme des tailles des deux tableaux dans la seconde variante :

- On peut résoudre `2SUM` en temps $\mathcal{O}(n \log n)$
- On peut résoudre `2SUMSorted` en temps $\mathcal{O}(n)$

3SUM

2SUM

Soit T un tableau contenant n nombres. Le problème `3SUM` consiste à indiquer s'il existe x, y et z dans T tels que $x + y + z = 0$.



Problème 3SUM – Solution naïve

3SUM

Soit T un tableau contenant n nombres. Le problème `3SUM` consiste à indiquer s'il existe x, y et z dans T tels que $x + y + z = 0$.

```
def 3SUM(T):
  for x in T:
    for y in T:
      for z in T:
        if x + y + z == 0:
          return True
  return False
```

Complexité : $\mathcal{O}(n^3)$

Problème 3SUM – Solution avec variante de 2SUM

3SUM

Soit T un tableau contenant n nombres. Le problème `3SUM` consiste à indiquer s'il existe x, y et z dans T tels que $x + y + z = 0$.

```
def 3SUM(T):
  T.sort()
  for z in T:
    if 2SUM(T,-z):
      return True
  return False
```

Ici il s'agit de la première variante de `2SUM` : on cherche si deux éléments de T somment à $-z$.

Complexité : $\mathcal{O}(n^2)$