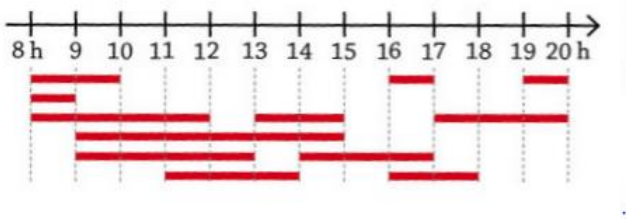
Mini projet 1 p179

- Données :

Le site de l'entreprise Calculus a reçu les **12 demandes de location sur les plages horaires** suivantes :

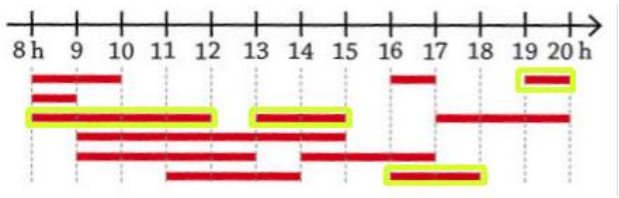


On voit qu'il y a une demande pour la plage horaire`[8, 10]`, une demande pour la plage `[8, 9]`, une demande pour la plage `[8, 12]`, etc.

La liste des demandes est donc :

```
demandes = [[8, 12], [9, 15], [9, 13], [11, 14], [8, 10], [8, 9],\
            [13, 15], [14, 17], [16, 17], [16, 18], [17, 20], [19, 20]]
```

- Contrainte : un seul client à la fois peut être servi parce qu'il n'y a seul cluster disponible.
- Donc tous les clients ne pourront pas être satisfaits puisque les plages horaires se chevauchent.
- Travail à faire : écrire une fonction **`validation`** qui elle-même appelle plusieurs fonctions, et qui à partir de la liste **`demandes`**, produit une liste **`resultat`** par ex. **`resultat`** `= [[8, 12], [13, 15], [16, 18], [19, 20]]` qui correspond aux plages horaires suivantes (entourées en jaune) :

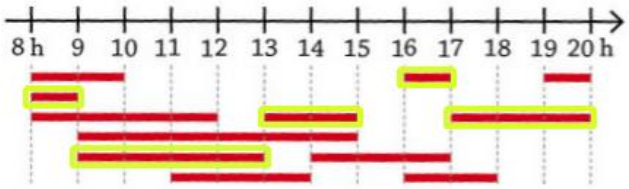


- Dans cet exemple, le résultat de la validation est `[[8, 12], [13, 15], [16, 18], [19, 20]]` et la somme des durées de ces plages horaires est 4 + 2 + 2 + 1 = 9 heures.

On constate qu'aucune plage horaire n'en chevauche une autre, ce qui correspond bien à la contrainte. Cependant, sur les 12 heures disponibles (de 8h à 20h), seulement 9h sont louées aux clients. Donc 3h sont perdues.

Vous utiliserez un algorithme du type **glouton** qui, au départ à 8h valide **la plus courte** plage horaire parmi les demandes. Dès que cette plage est terminée, l'algorithme valide la première plage horaire suivante parmi les demandes. S'il y a plusieurs demandes qui commencent à la même heure alors l'algorithme choisit **la plus courte**.

Avec les 12 demandes de plages horaires reçues, cela donne :



Cela signifie que votre fonction `validation(`**`demandes`**`)` doit renvoyer :
`resultat` `= [[8, 9], [9, 13], [13, 15], [16, 17], [17, 20]].` Ainsi 11h louées.