

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Informatique

6

Algorithmes gloutons

TD 6-2

Allocation de ressources

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

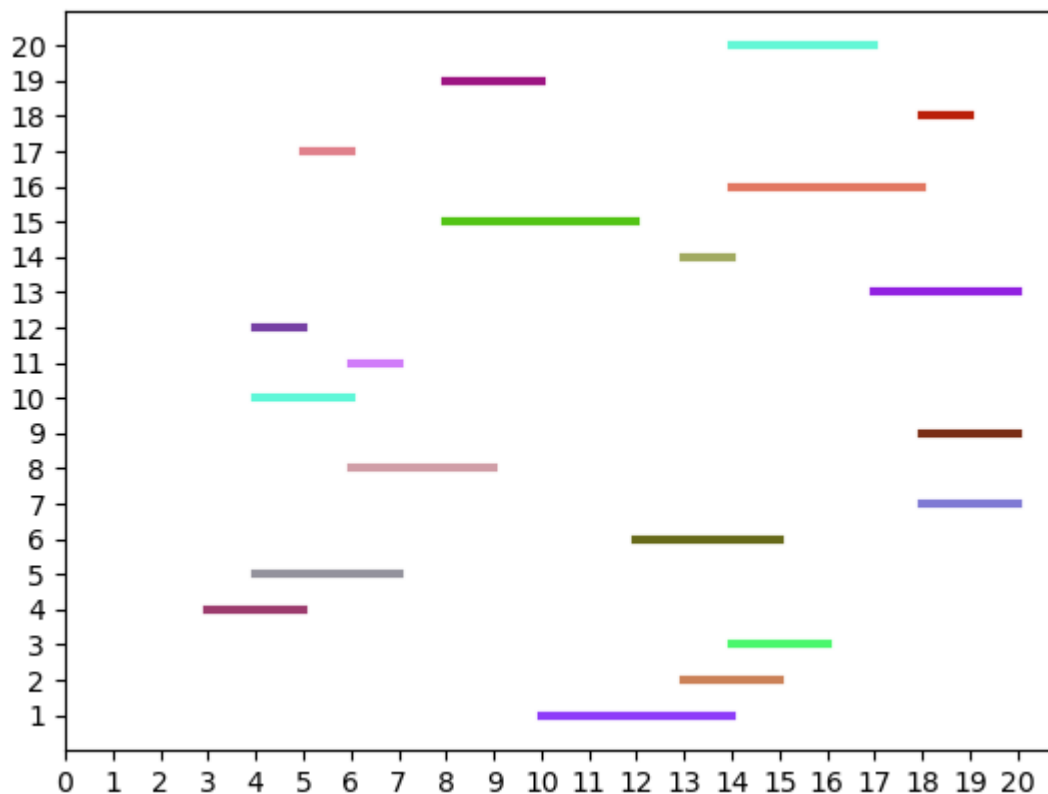
Le principe des algorithmes gloutons consiste à essayer de répondre à une problématique en choisissant, à chaque étape, la meilleure solution « locale », en espérant que la solution finale sera la meilleure des solutions globales.

Exercice 1: Allocation d'une ressource

Contexte

On se place dans le cadre de la réservation d'une salle (ressource) par exemple (adaptable à divers sujets).

Nous avons à disposition plusieurs demandes de réservations R_i , qui précisent donc une date de début D_i et une date de fin F_i . Voici un exemple de 20 demandes (axe des ordonnées) réparties sur 20 jours (axe des abscisses) avec des durées différentes :



Les conditions à respecter sont les suivantes :

- On ne peut pas donner la salle à deux personnes le même jour
- L'objectif de l'optimisation est de répondre au plus grand nombre de demandes

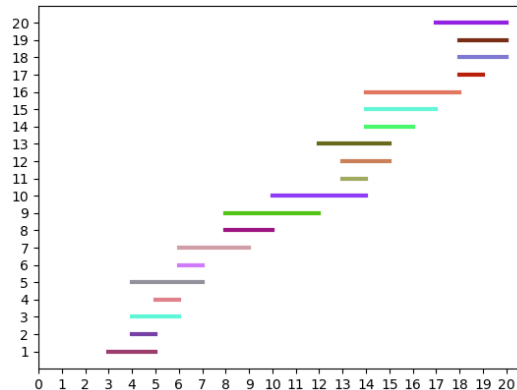
Remarque : Le plus de réservation n'est pas forcément compatible avec la plus grande occupation de la salle. Par exemple, 3 demandes consécutives d'un jour (sur 4 jours au total) seront privilégiées face à une demande de 4 jours...

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Algorithme

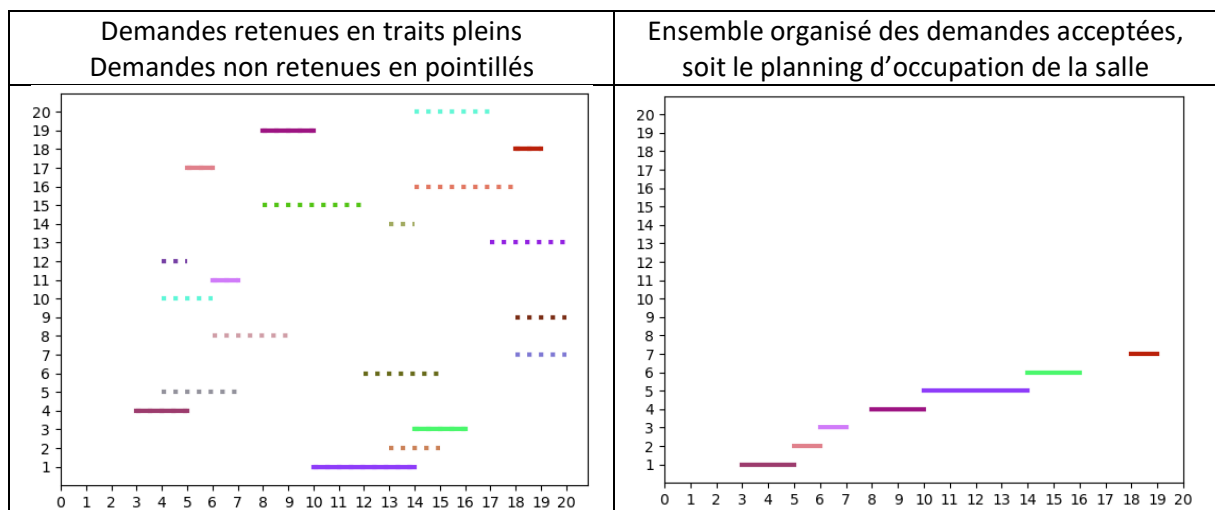
Le principe de l'algorithme glouton associé à ce souhait consiste à :

- Trier les demandes par ordre croissant de fin :



- Parcourir les demandes **triées** en retenant à chaque fois la première réservation compatible ($D_i \geq F_{i-1}$), c'est-à-dire la prochaine réservation possible finissant au plus tôt. S'il y en a plusieurs, on prend la première qui se présente. C'est donc une solution optimale locale qui est recherchée à chaque itération.

On trouve alors la solution suivante :



Remarque : en ne triant les réservations que selon leur date de fin, on choisit la première venue parmi plusieurs réservations potentielles ayant la même fin. Pour le même nombre de réservations, ce choix n'est pas optimal en termes d'occupation de la salle. Il serait intéressant de réaliser un second tri (que nous ne ferons pas) par ordre croissant de début (ie de longueur) des réservations pour toutes celles ayant la même fin, afin de prendre à chaque fois la plus longue.

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Génération des réservations

On souhaite générer une liste de réservations aléatoires comprises entre début (numéro du premier jours inclus de type entier) et Fin (numéro du dernier jour inclus de type entier) de durée max Duree_Max de type entier et telles que chaque réservation soit une liste $[D_i, F_i, i, Couleur]$ avec :

- $D_i \geq Debut$
- $F_i \leq Fin$
- $1 \leq F_i - D_i \leq Duree_Max$
- i est le numéro de réservation, la première étant d'indice $i = 1$
- *Couleur* est une liste de 3 nombres aléatoires dans $[0,1]$ permettant de définir une couleur RGB

Pour rappel, on obtient dans le module random :

- Un entier aléatoire dans $[a,b]$ avec `randint(a,b)`
- Un réel aléatoire dans $[a,b]$ avec `uniform(a,b)`

Question 1: Proposer une fonction `Generation(Debut,Fin,Duree_max,Nb)` permettant de générer une liste de Nb sous-listes de type $[D,F,Indice,Couleur]$

Vérifiez quelques exemples :

```
>>> Generation(0,5,2,5)
[[3, 4, 1, [0.4843192151615079, 0.20766460848757173, 0.5901621536654499]],
 [2, 3, 2, [0.34734071482631157, 0.6904222913788538, 0.10852260069347253]],
 [3, 5, 3, [0.6914586120001831, 0.06256982006673195, 0.9908382206632146]],
 [3, 4, 4, [0.37832039769517056, 0.8142201076732741, 0.6218490665816707]],
 [0, 1, 5, [0.5798197498377815, 0.5876176730550857, 0.9956567596069265]]]

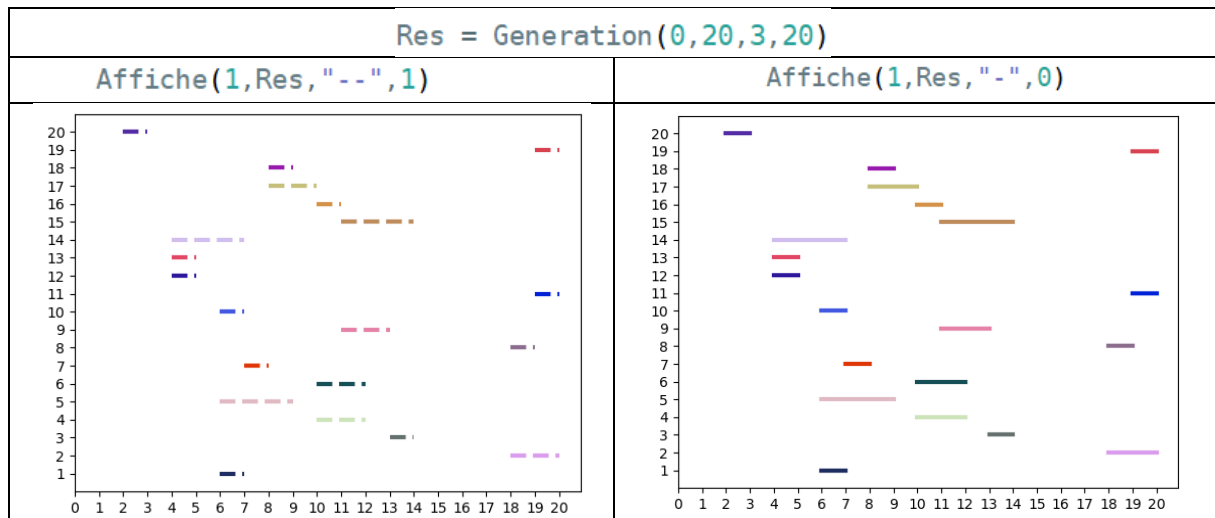
>>> Generation(0,2,2,5)
[[1, 2, 1, [0.0672254354770836, 0.9177783743147795, 0.3559693355504935]],
 [1, 2, 2, [0.562221995856825, 0.9541314855886106, 0.697697354707228]],
 [1, 2, 3, [0.30342223887113795, 0.45732225104455637, 0.14266380040744753]],
 [1, 2, 4, [0.7063868722349047, 0.09027108419734131, 0.48607035383709607]],
 [1, 2, 5, [0.1131446125561788, 0.31562839439704304, 0.6022561045490823]]]

>>> Generation(10,20,3,5)
[[10, 11, 1, [0.15556512978159476, 0.40210214049195714, 0.8175621269626249]],
 [15, 17, 2, [0.050345108117127224, 0.08175886969658375, 0.7364807349382422]],
 [16, 19, 3, [0.7741329800741917, 0.727747682301916, 0.4389707478548195]],
 [17, 19, 4, [0.29361958813498235, 0.7692899280929002, 0.42262797099696237]],
 [10, 12, 5, [0.010285499265123343, 0.5576230869902049, 0.6370744292421786]]]
```

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Affichage

On souhaite créer une fonction **Affiche(fig,LR,Type,Ordre)** telle que :



Cette fonction doit tracer l'ensemble des réservations de la liste LR en argument sur la figure numéro fig avec la couleur associée (triplet contenu dans les réservations), avec les options suivantes :

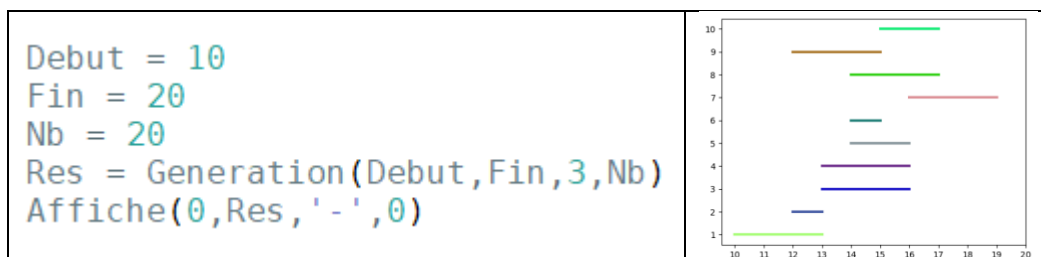
- Type : Chaîne de caractères définissant le type de trait tracé
 - Pointillés : '-'
 - Trait plein : '-'
- Ordre : Les réservations sont placées sur l'axe des ordonnées selon
 - S'il vaut 1, leur ordre d'apparition dans la liste LR
 - S'il vaut 0, leur indice ind

On précise que trois variables globales seront définies par la suite et seront utilisées dans la fonction Affiche afin de créer les graduations sur les axes :

- Nb : Nombre de réservations (Entier) – Avec « yticks », on attend une graduation par réservation
- Debut et Fin : Premier et dernier jour de toutes les réservations (Entiers) – Avec « xticks », on attend une graduation par jour entre début et fin

Question 2: Créer la fonction Affiche comme demandé et vérifier qu'elle fonctionne

Essayez :



A ce stade, on obtient évidemment le même tracé que l'on demande Ordre=0 ou 1.

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Tri de la liste des réservations

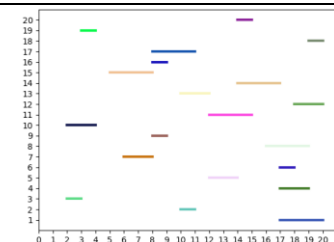
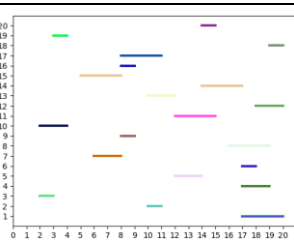
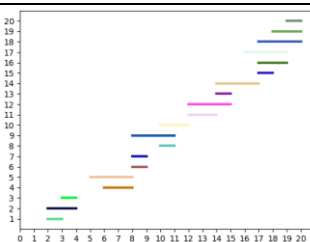
Nous aborderons les algorithmes de tri dans la suite de l'année, alors utilisons directement les tris intégrés au langage Python. Soit le code suivant :

```
L = [[1,3],[1,2],[0,2],[3,3]]
L1 = sorted(L)
L2 = sorted(L, key=lambda x:x[0])
L3 = sorted(L, key=lambda x:x[1])
```

Question 3: Recopiez le code proposé et comprendre ce qu'il réalise

Question 4: Créer la fonction Tri_fin(LR) renvoyant la liste des réservations triées par ordre de fin sans modifier LR

Vous pourrez alors vérifier que vous obtenez des images similaires à celles-ci :

<pre>Res = Generation(0,20,3,20) Res_Tri = Tri_fin(Res)</pre>		
<pre>Affiche(10,Res,'-',0) Affiche(11,Res,'-',1)</pre>	Affiche(12,Res_Tri,'-',0)	Affiche(13,Res_Tri,'-',1)
		
Les réservations ont beau être triées, si on décide de les afficher selon leurs indices dans Res_Tri (à droite), on retrouve leur ordre de création dans Res (à gauche)	Si on affiche les réservations par ordre d'apparition dans Res_Tri, elles sont bien triées par ordre de fin	

Résolution par l'algorithme Glouton

Question 5: Mettre en place une fonction Resolution(LR_Tri) prenant une liste de réservations quelconque LR_Tri triée par la fonction Tri_fin, et renvoyant la liste des réservations choisies par l'algorithme Glouton décrit précédemment

Dernière mise à jour	Informatique	Denis DEFAUCHY
03/02/2022	6 - Algorithmes gloutons	TD 6-2 - Allocation de ressources

Application

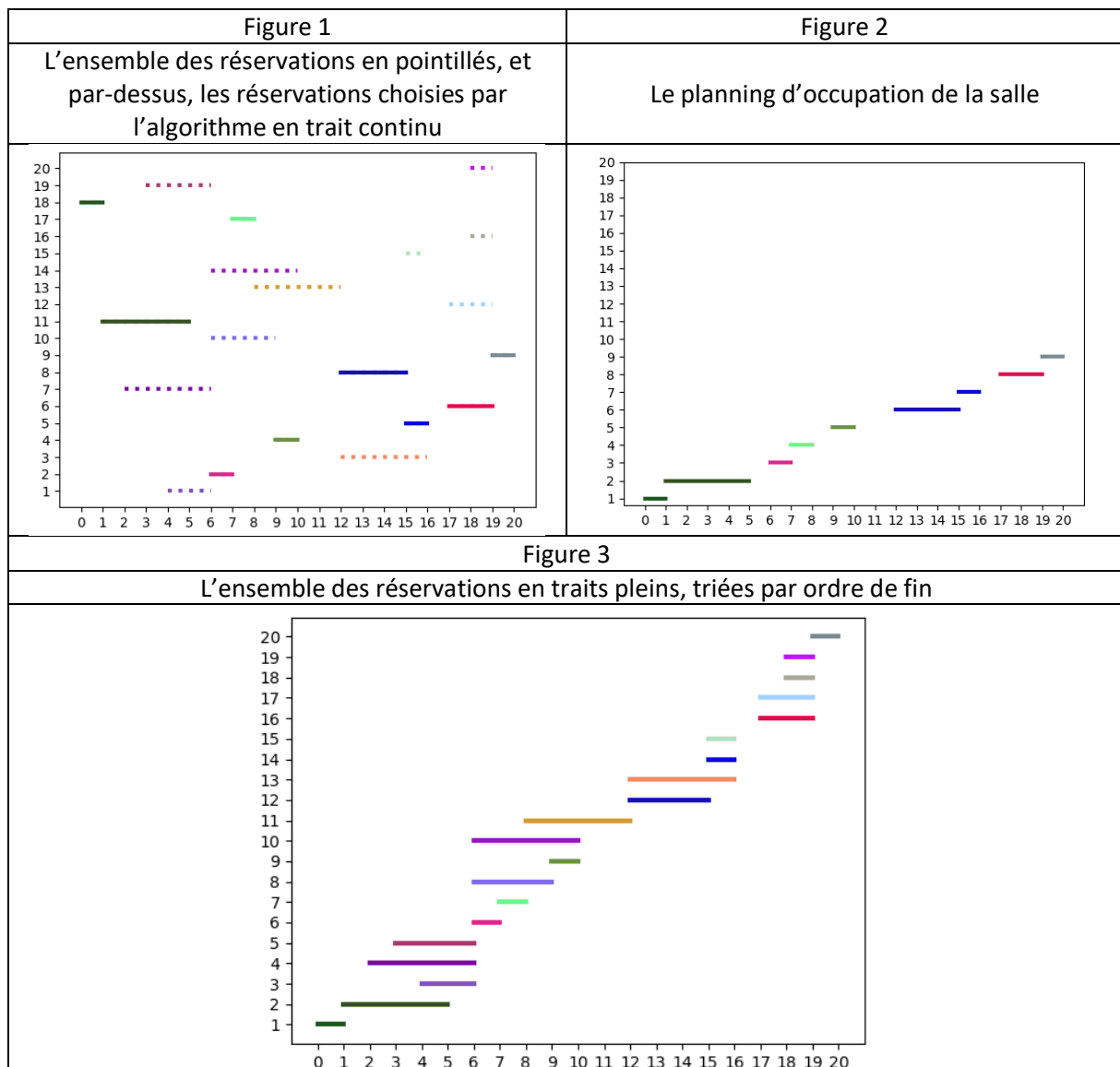
On définira les variables globales suivantes :

```

Debut = 0
Fin = 20
Duree_max = 4
Nb = 20

```

Mettre en place le code permettant, à partir de ces données, de générer les réservations, de déterminer par l'algorithme glouton la solution de distribution des réservations, et d'afficher sur trois figures différentes :



Remarque : compte tenu de notre programmation, l'affichage du planning de la salle (Figure 2) indique Nb réservations alors qu'il y a moins de réservations retenues. Mais, si on définit dans la fonction Affiche, le nombre de réservations comme len(LR), c'est l'image (Figure 1) qui présentera un soucis puisqu'en ajoutant les réservations choisies par l'algorithme, l'axe des ordonnées sera redéfini au nombre retenu... A vous de choisir !