0.1 Rappels

Types de tris vus :
- Tri par insertion dichotomique
- Tri arborescent, éventuellement reprendre le concept au tableau
- Améliorations tri arborescent (à revoir dans Froidevaux)
- Quicksort
- Eventuellement tris particuliers, mais probablement pas

A priori Pascal va distribuer les algos en cours. Il faut changer les TDs.

.1 Tris efficaces

1. **Retour sur le tri à bulles :** calculer la complexité du tri à bulles en y intégrant des calculs précis de probabilité (nombre de comparaisons et nombre d'échanges)
2. On considère le tableau $[13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21]$. Donner les traces d'exécution du tri de ce tableau par les trois alggorithmes vus en cours :
   (a) Tri par insertion dichotomique
   (b) Tri arborescent (donner les différentes versions de l'arbre ainsi que sa représentation sous forme de tableau)
   (c) Tri par tas
   (d) Tri rapide
3. **Tri par insertion dichotomique :**
   (a) Écrire une fonction *recherche_dichotomique* qui prend en argument un tableau d'entiers ordonnés t, un indice de départ d, un indice de fin f et un entier x à rechercher et qui retourne l'indice i de la case contenant x si x figure dans le tableau entre les indices d et f, et *null* sinon
   (b) Donner une version du tri par insertion qui utilise cette recherche dichotomique
4. **Tri arborescent :**
   (a) Proposer un algorithme qui stocke dans un tableau l'arbre de tri d'un tableau d'entiers
   – entrée=un tableau d'entiers de longueur N à trier.
   – sortie=un tableau d'entiers de longueur $2N - 1$ qui contient l'arbre de tri du tableau donné en entrée.
   Le tableau correspondant à l'arbre de tri contient d'abord les feuilles de l'arbre, puis les noeuds de hauteur 1 etc.
   (b) Écrire une fonction qui prend en argument un tableau d'entiers (contenant un arbre de tri) et qui renvoie un tableau contenant les entiers triés.
5. **Tri par tas :**
   (a) Proposer un algorithme qui crée un tas à partir d'une liste d'entiers
   – entrée=un tableau d'entiers de longueur N à trier.
   – sortie=un tableau d'entiers de longueur N qui contient un tas formé des entiers de la liste d'entrée
   (b) Écrire une fonction qui prend en argument un tableau d'entiers et qui renvoie un tableau contenant les entiers triés en utilisant la méthode du tri par tas.
6. **Drapeau tricolore :** on se donne un tableau de n éléments qui ont chacun une couleur : vert, blanc ou rouge. On veut obtenir un tableau avec trois zones : à gauche tous les éléments verts, au milieu tous les éléments blancs, à droite tous les rouges. Écrivez une procédure qui réalise cette organisation du tableau en ne testant qu'une seule fois la couleur de chaque élément.

.2 Correction

1. – Le nombre de comparaisons est le même que pour la sélection ordinaire : $Max_C(n) = Moy_C(n) = \frac{n(n-1)}{2} = \Theta(n^2)$
   En effet : pour toute liste de taille n on effectue $n-1$ comparaisons pour trouver le minimum, soit $\sum_{i=0}^{n} i = \frac{n(n-1)}{2}$
   – Le nombre d'échanges est au pire en $n-1$ au premier parcours, $n-2$ au deuxième etc. Cela arrive si le tableau est trié en ordre décroissant. Donc : $Max_E(n) = \sum_{i=0}^{n-1}(n-i) = \sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$
   Le nombre d'échanges en moyenne est également d'ordre n^2. Voir la feuille de Froidevaux et al.
2. Dérouler au tableau
   – insertion : facile
   – Arbo std. : faire l'arbre. Expliquer en quoi il est redondant par rapport le tas (cf. plus tard).
   – Heapsort : OK. On initialise avec la construction du premier tas (mentionner le gain de mémoire par rapport à l'arbre), ensuite on opère les shift en fin de tas. cf. feuille Froidevaux.
3. Appliquer l'algo de cours : on crée le tableau de la bonne taille. On initialise les feuilles (début tableau), et puis :

```
function niveau(L,M):
 for i=L to M-1:
  if t[i]<t[i+1]:
   t[M+1+i/2]=t[i];
  else:
   t[M+1+i/2]=t[i+1];
 niveau(M+1,(M-L+1)/2);
```

   Ensuite algo de cours, assez simple.
4. Le principe ici est de mémoriser dans des variables les deux indices suivants :
   – v l'indice de la première case du tableau dont on n'est pas sûr qu'elle contienne une boule verte ;
   – r l'indice de la dernière case du tableau dont on n'est pas sûr qu'elle contienne une boule rouge ;
   Ensuite, il ne reste qu'à parcourir le tableau, échanger le contenu de la case courante d'indice i avec :
   – le contenu de la case d'indice v si la case courante contient une boule verte ;
   – le contenu de la case d'indice r si la case courante contient une boule rouge ;
   – augmenter i si la case courante contient une boule blanche (et vérifier la condition finale à ce moment)
   et mettre à jour les indices i, v et r. Le tri est effectué quand l'indice i dépasse r