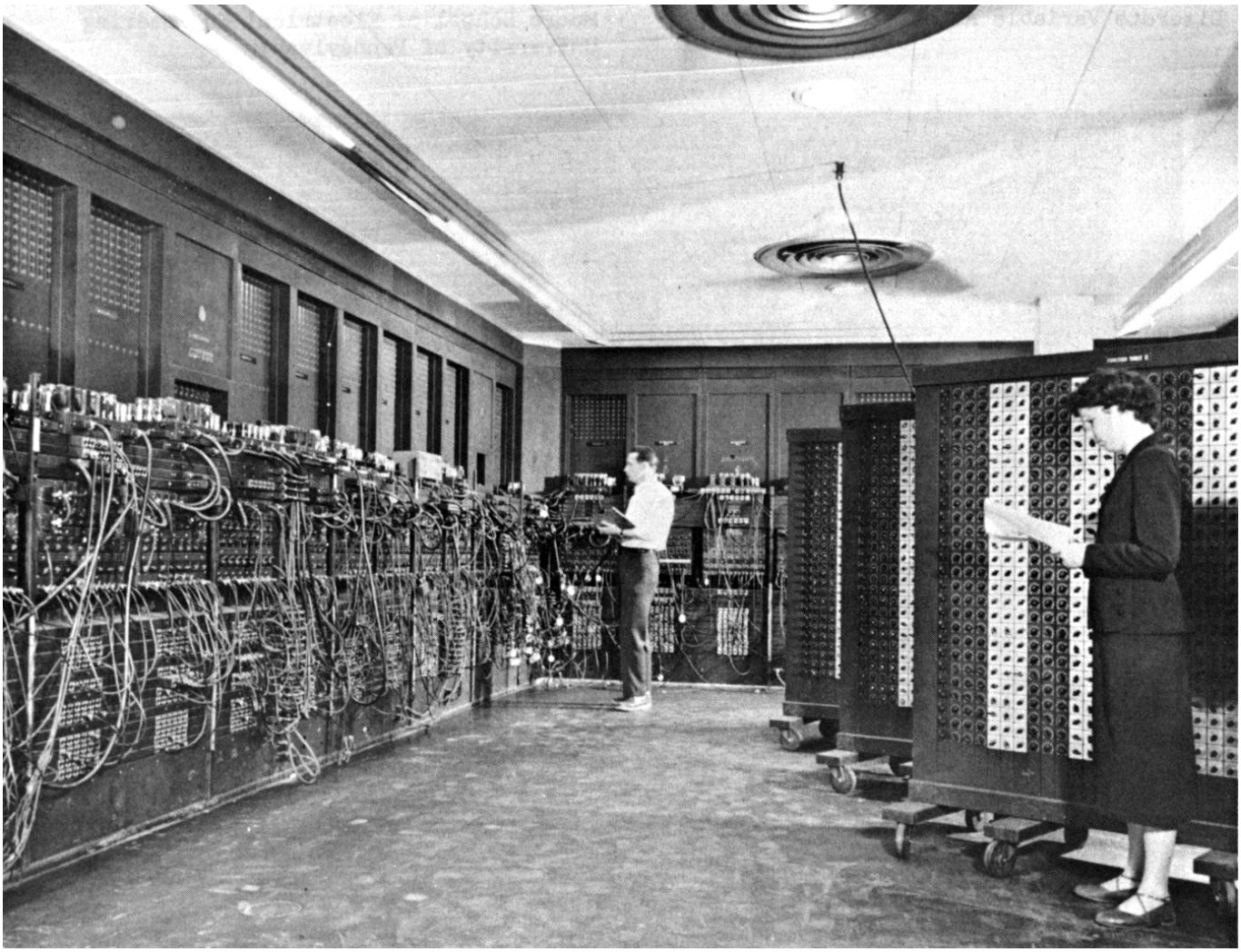
Algorithmes de tri

Plan du T.P.



Betty Holberton et l'ENIAC

La création de la première routine de tri sur ordinateur est attribuée à Betty Holberton [1], durant la seconde guerre mondiale.

L'ENIAC [2] est alors le premier ordinateur entièrement électronique, développé par l'armée américaine pour les besoins du laboratoire de recherche en balistique. Betty Holberton fut rapidement choisie pour être l'une des six programmeuses de l'ENIAC. Elle y développe la première routine de tri et la première application logicielle.

1 Trier

Activité 1 : Des cartes . . .

1. Les élèves étant en îlots de quatre, proposer un jeu de cartes à chaque îlots.
2. Demander de constituer 4 jeux de cartes : les jeux de cœur, pique, carreau, trèfle.
3. Chaque joueur prend un jeu et le tri.

Synthèse :
- Comparer sa méthode de tri et celles employées par les voisins.
- Y-a-t-il plusieurs façons d'ordonner ces objets ?
- "Valet-Dame-Roi-As" **ou** "1-2-3 . . . " ?
- Relation d'ordre.

Activité 2 : Pourquoi trier ?

1. On distribue aux élèves un listing de noms avec des noms et des dates de naissance. Un, plusieurs ou la moitié de la classe possède le listing trié par ordre alphabétique, les autres possède une version non ordonnée. Les élèves ne savent pas que certains possèdent des versions différentes.
2. On demande la date de naissance de M MARTIN. Les élèves qui ont le listng trié répondent tout de suite, les autres restent dubitatifs.
3. On recommence l'expérience avec MMe BRETON. Même constat.
4. On demande aux élèves qui ont répondu tout de suite ce qui leur a permis de le faire. –> Le listing est donné trié alphabétiquement donc recherche aisé.
5. On demande aux élèves quelle est la personne la plus jeune. –> Il faut trier à nouveau toute le liste.

Synthèse :
Chercher un élément dans une liste est effectué plus rapidement si celle-ci est triée.
Les données qu'on étudie peuvent déjà être donnée sous forme triée, ou non...

Activité 3 : Trier les données deux à deux

1. On distribue à chaque élève un papier sur lequel est écrit un nombre de 6 ou 7 chiffres écrits uniquement égaux à 3 ou 4, pas nécessairement écrits dans la même police et avec le même espacement.
2. On demande aux élèves de s'aligner en ordre croissant. Comportement attendu : les nombres sont comparés deux à deux puis rangés en fonction du résultat de la comparaison.

Synthèse :
Quand on ne peut pas disposer des facultés humaines (vision large, prise en compte de plusieurs nombres en même temps), on est obligé d'opérer en traitant les données deux par deux.

(2). https://fr.wikipedia.org/wiki/Betty_Holberton
(2). https://fr.wikipedia.org/wiki/ENIAC

Activité 4 : Verbalisation

La vidéo *Algorithmes de tri* [3], de la chaîne Youtube **Infotéo** [4], offre de très belles séquences :

1. Commençons par regarder la vidéo collectivement **jusqu'à 1 min 20 s** :

2. **Questionnement** : comment définir un algorithme de tri ?

   On pourra se remémorer :

   - le jeune enfant triant des boites dans la vidéo,
   - les listings de données en tables de l'activité 1,
   - trier les habitants des États-Unis en 1890, par ordre alphabétique, ou encore par âges croissants...

3. **Procédé mnémotechnique** : quelle phrase simple proposer pour définir un algorithme de tri ?

Synthèse :

Un **algorithme de tri** [5] est une suite d'opérations qui permet de ranger une collection d'objets selon un certain ordre.

(3). Algorithmes de tris : https://www.youtube.com/watch?v=ra79TDfotno
(4). Infotéo : https://www.youtube.com/channel/UCxghCEo9w2hvbS5F2ut4rqQ
(5). https://fr.wikipedia.org/wiki/Algorithme_de_tri

2 Le Tri par Sélection

Activité 5 :

1. Regardons dans la vidéo la partie sur le tri par sélection **à partir de 2 min 50 s** :

2. Décrire les éléments suivants rencontrés dans la vidéo :

- Ensemble des 3 carrés verts : .....
- Ensemble des 4 carrés gris : .....
- Le carré gris foncé : .....

Synthèse :

- Nous recherchons le minimum du tableau pour l'échanger avec le premier élément. Puis nous recherchons le minimum suivant, pour l'échanger avec le second élément du tableau ...
- On peut se représenter deux sous-tableaux :
le tableau déjà trié, et le tableau restant à trier, où nous recherchons le minimum pour l'échanger.

Activité 6 : Jeu de cartes, à quatre

Algorithme 1 RÈGLE DU JEU

Entrée(s) 4 Joueurs

Prendre **un jeu** de cartes
Constituer **4 ensembles de cartes** :
- les cartes de **cœur**
- les cartes de **pique**
- les cartes de **carreau**
- les cartes de **trèfle**

Chacun des **Joueurs** :
- prend un ensemble au hasard
- le trie **par sélection**
- aide ses voisins qui auraient des difficultés

Exercice 1 : listes de nombres, papier-crayon

Nous nous représentons facilement, avec les cartes, la liste triée, et la liste non triée. Pour une liste de nombres, sur papier, nous pourrons représenter la séparation entre ces deux listes, à l'aide d'un séparateur vertical « | ».

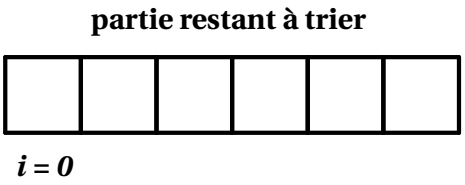
1. Observons le tri par sélection la liste ci-contre (au départ, rien n'est trié donc le séparateur vertical se trouve à gauche).
2. Trier par sélection, papier-crayon, les listes suivantes :
   - [9, 6, 7, 5, 8]
   - [3, 5, 1, 9, 2]
   - [11, 5, 17, 16, 4, 9, 5]

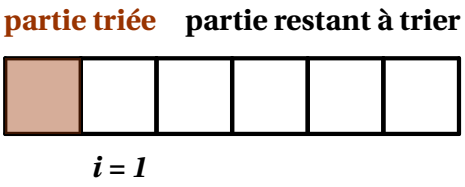
| 4 1 5 2 6
| 1 4 5 2 6
1 | 4 5 2 6
1 | 2 5 4 6
1 2 | 5 4 6
1 2 | 4 5 6
1 2 4 | 5 6
1 2 4 5 6 |

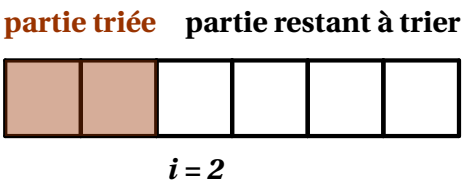
3 Algorithme de tri par sélection

Représentation imagée de l'algorithme

- Une **première boucle**, de compteur ***i***, balaye la liste et indique la position de la partie non triée.

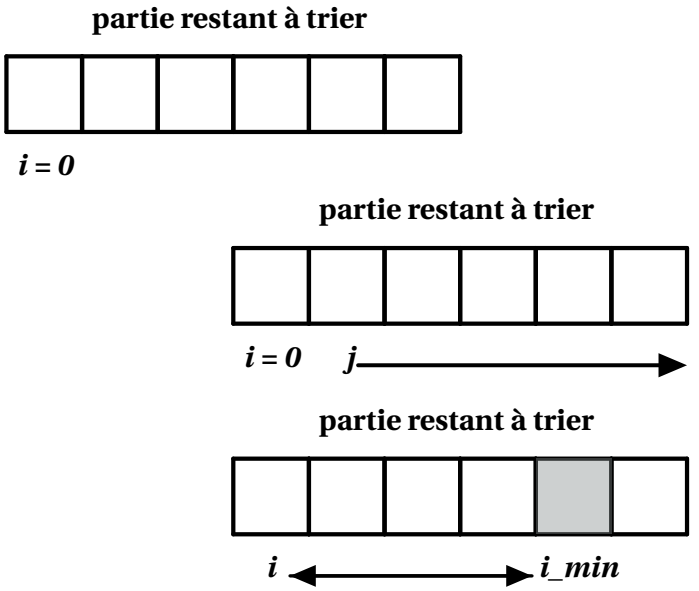


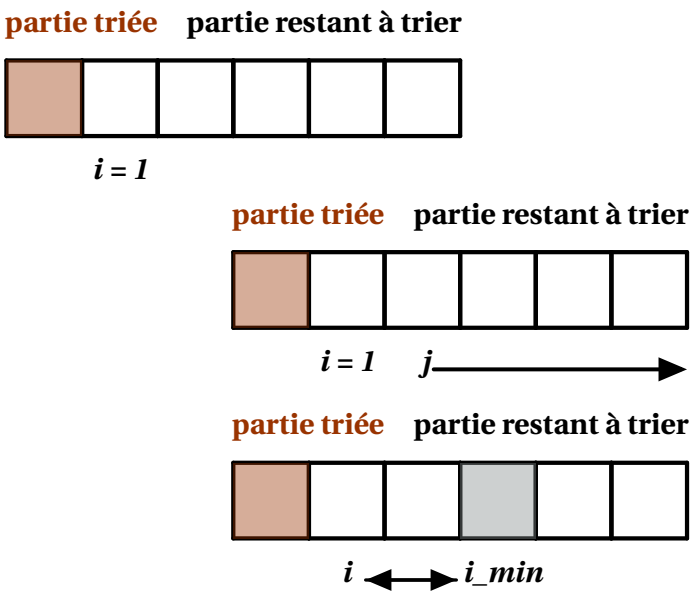


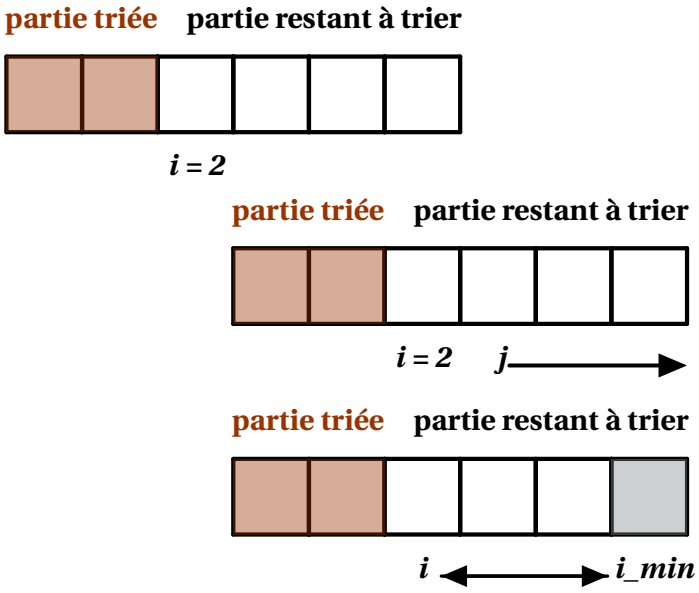


- Une **seconde boucle**, imbriquée dans la première, et de compteur ***j***, parcourt la partie non triée à la recherche de son minimum.

  Une fois l'indice ***i_min*** du minimum de la partie non triée trouvé, il ne reste plus qu'à échanger les éléments d'indices i et i_min.


partie restant à trier
i = 0
partie restant à trier
i = 0
j
partie restant à trier
i
i_min


partie triée
partie restant à trier
i = 1
partie triée
partie restant à trier
i = 1
j
partie triée
partie restant à trier
i
i_min


partie triée
partie restant à trier
i = 2
partie triée
partie restant à trier
i = 2
j
partie triée
partie restant à trier
i
i_min

Exercice 2 : représentation graphique de l'algorithme

Représenter sur feuille les étapes suivantes de l'algorithme : $i = 3$, $i = 4$, ...

Ou encore, imaginer et dessiner votre propre représentation graphique !

Exercice 3 : l'algorithme de tri par sélection

Compléter l'algorithme ci-dessous de tri par sélection :

Algorithme 2 LE TRI PAR SÉLECTION

```
1:  fonction tri_selection(T)
2:      n = longueur(T)
3:      pour i allant de ... à ... faire
4:          i_min = i
5:          pour j allant de ... à ... faire
6:              si ..... alors
7:                  i_min = j
8:          si ..... alors
9:              echanger(T, i, i_min)
10:     retourner T
```

4 Implémentation sous Python

Exercice 4 : fonction auxiliaire

Nous remarquons que lorsqu'un minimum a été trouvé dans la liste restant à trier, nous allons devoir l'échanger avec le premier élément de cette même liste.

1. Écrire une fonction `echanger(T, i, j)` permutant les éléments d'indices i et j dans un tableau T.
2. Tester cette fonction avec différents tableaux T pour s'assurer de son bon fonctionnement.

Exercice 5 : fonction de tri par sélection

Implémenter sous Python l'algorithme 2 de tri par sélection complété dans l'exercice 3.

```
1  def echanger(T, i, j) :
2      """ permute les elements d'indices i et j de T """
3
4  def tri_selection(T) :
5      """ trie T par selection """
```

5 Terminaison et correction de l'algorithme

Exercice 6 :

1. Pourquoi est-on sûr que l'algorithme se termine ?
2. Chercher la phrase vraie parmi les quatre phrases ci-dessous.
   Pour un tableau de taille n, avec le tri par sélection,
   a. Au bout de k étapes, les éléments k et $(k+1)$ sont triés et à leur place définitive.
   b. Au bout de k étapes, les k premiers éléments sont triés et à leur place définitive.
   c. Au bout de k étapes, les $k+1$ premiers éléments sont triés et à leur place définitive.
   d. Au bout de k étapes, les $n-k$ derniers éléments sont triés et à leur place définitive.
3. Réécrire cette phrase dans le cas où $k = n$.
   Que nous dit-elle ?

Corrigé :

1. *l'algorithme ne contient des boucles "for" donc il n'a qu'un nombre fini d'itérations, il se termine quand i est à longueur du tableau-1 et j à n*
2. *Pour un tableau de taille n, avec le tri par sélection, ii u bout de k étapes, les k premiers éléments sont triés et à leur place définitive.*
3. *Au bout de n étapes, les n premiers éléments sont triés et à leur place définitive.*
   Ce qui nous indique que l'algorithme aboutit à la fin à une liste triée. On dit que l'algorithme est correct.

6 Complexité de l'algorithme de tri par sélection

Objectif :
réaliser une mesure de la durée d'exécution de l'algorithme de tri par sélection vus à la question précédente pour des tableaux de tailles différentes contenant des nombres aléatoires.

Mesure du temps d'exécution du programme sélection

Exercice 7 : Écriture du programme

1. Dans l'éditeur Python Thonny, réaliser une liste de nombres aléatoires compris entre 1 à n pour n=100.(creer_une_liste_a_trier.py)
2. Recopier votre code Python tri par sélection en intégrant la mesure de temps d'exécution utilisant la fonction time(). (mesure du temps.py)
3. Relever le temps d'exécution.

```
import time

heure_depart = time.time()
# Insérer ici le code dont le temps d'exécution est à évaluer
heure_fin = time.time()

temps_execution = heure_fin - heure_depart
```

Evolution des temps d'execution pour des listes de longueurs croissantes

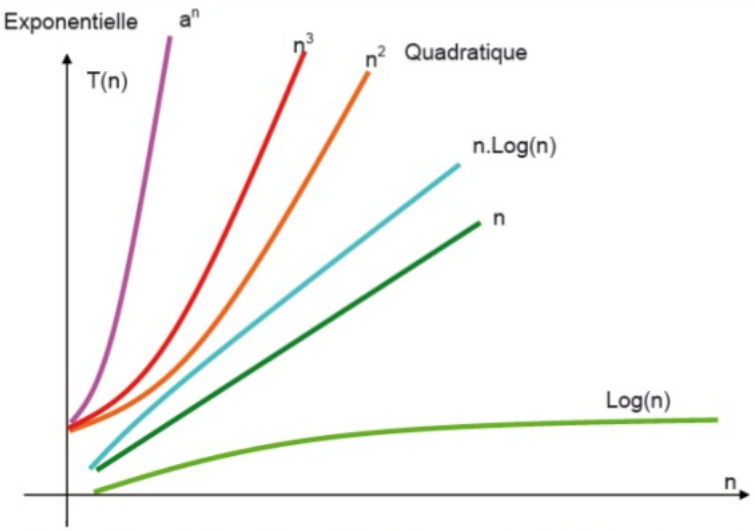
Exercice 8 : Représentation graphique

1. Réaliser une liste des temps d'execution pour des listes de longueurs croissantes contenant 1000 à 15000 éléments de pas 1000. (liste_taille_croissante.py)

```
# Création d'une liste aléatoire de longueur n,
# constituées de nombres compris entre 0 et n.

def creer_liste(n) :
    l = [random.randrange(0, n) for i in range(n)]
    return l
```

2. Réaliser une représentation graphique présentant les différentes valeurs de votre liste des temps d'exécution pour des listes de longueurs croissantes.
3. Comparer l'allure de la courbe obtenue avec celles des fonctions linéaire, carré et cube représentées ci-dessous.



Recherche par calcul du modèle de complexité

Exercice 9 :

1. Comparer les temps d'exécution pour une liste contenant 100 éléments aléatoires et une liste contenant le double d'éléments.
2. Comparer les temps obtenus aux modèles de complexité présentés sur le graphique ci-dessus et en déduire la complexité de la fonction de tri par sélection.

Corrigé

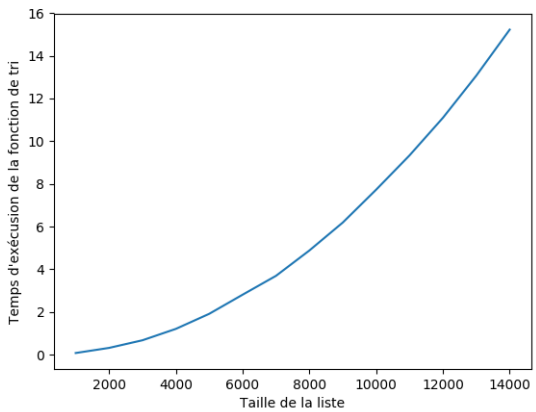
```python
import time
import random
import matplotlib.pyplot as plt
import numpy


# Code par selection
def tri_selection(T):
    t0 = time.time()
    nb = len(T)
    for i in range(0,nb):
        plus_petit = i
        for j in range(i+1,nb) :
            if T[j] < T[plus_petit] :
                plus_petit = j
            if min is not i :
                temp = T[i]
                T[i] = T[plus_petit]
                T[plus_petit] = temp
    return time.time() - t0

# Création d'une liste aléatoire de longueur n constitués de nombre compris entre 0 et n
def creer_une_liste_a_trier(n):
    l = [random.randrange(0, n) for i in range(n)]
    return l


# creation_liste_des_temps_execution():
listeTaille = range(1000,15000,1000)
listeTemps = []
for taille in listeTaille:
    L=creer_une_liste_a_trier(taille)
    listeTemps.append(tri_selection(L))

# représentation graphique de la fonction de sélection
plt.ylabel('Temps d\'exécution de la fonction de tri par sélection')
plt.xlabel("Taille de la liste")
plt.plot(listeTaille,listeTemps)
plt.show()
```



Exercice 10 :
Rajouter un compteur dans le programme "tri_selection" pour compter le nombre d'affectation et de comparaison qu'il réalise.

Corrigé

```
from random import randint


def echanger(L,i,j):
    aux=L[i]
    L[i]=L[j]
    L[j]=aux



def triSelection(L):
    n=len(L)
    cpt=1
    for i in range(n):
        iMin=i
        cpt+=1
        for j in range(i,n):
            cpt+=1
            if L[j]<L[iMin]:
                iMin=j
                cpt+=1
            cpt+=1
            if i!=iMin:
                echanger(L,i,iMin)
                cpt+=3
    return((L,cpt))
```