Un problème de rangement ? Une solution : notre algorithme !

Année 2016-2017

Élèves :
ROBIN Manon
LE MEUDEC Juliette
1S3

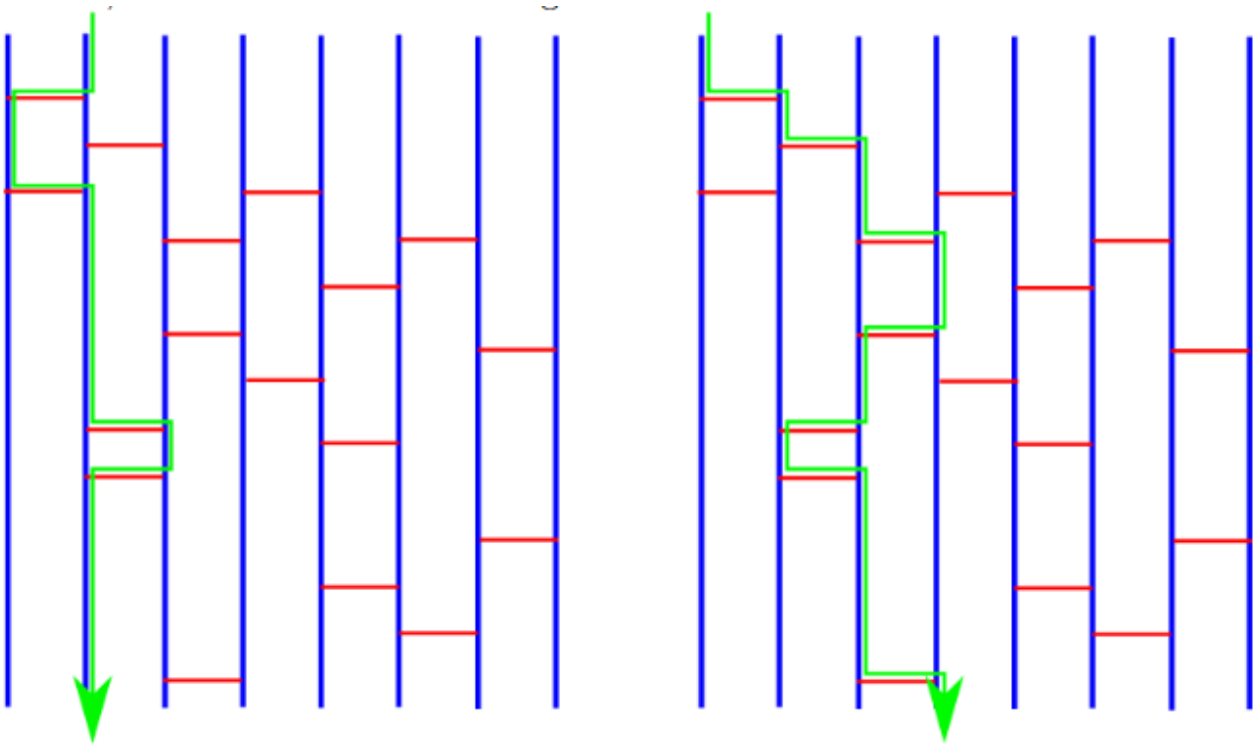
Encadrants : M Tisserand, Mme Caret, Mme Balliot

Chercheur : Vincent Guirardel, Université de Rennes 1

Établissement : Lycée Victor et Hélène Basch Rennes (Bretagne)

I) Présentation du sujet :

On souhaite ranger un paquet de 32 cartes grâce à des échelles. Chaque carte est placée au sommet d'une échelle (donc 32 cartes → 32 échelles), elles sont placées dans un ordre aléatoire. Le but étant que les cartes arrivent rangées dans l'ordre croissant en bas de ces dernières.
Les cartes doivent donc pouvoir se déplacer !
Pour ce faire, nous allons utiliser des barreaux qui seront placés entre les différentes échelles.
Chaque carte descend le long de son échelle et lorsque qu'elle arrive au niveau d'un barreau, elle l'« emprunte », elle change donc d'échelle.



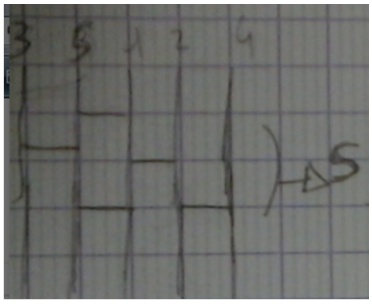
Plusieurs questions nous ont été proposées sur ce sujet :
« Quel est le nombre minimum de barreaux pour ranger 32 cartes ? Pour ranger un nombre n de cartes ? »
« Peut-on ranger toutes les cartes? »
« Est-ce que 2 cartes peuvent arriver en bas de la même échelle ? »

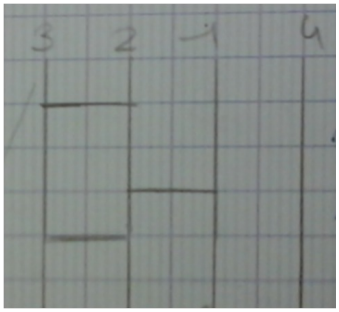
Nous avons choisi de chercher à savoir s'il était possible de ranger n'importe quel nombre de cartes, peu importe le nombre de barreaux qui sera utilisé. C'était de notre point de vue la question la plus intéressante, celle qui nous inspirait le plus et était différente de celles choisies par nos camarades de classe.

II) Annonce des résultats :

A l'aide d'un algorithme réalisé sur Algobox, nous avons réussi à prouver qu'il était possible de ranger n'importe quel nombre n de cartes. Pour rendre plus compréhensible cet algorithme, nous avons réalisé et filmé une vidéo mettant en scène son procédé.

III) Introduction au sujet :





Pour commencer, nous avons élaboré plusieurs schémas dans l'optique de bien comprendre le fonctionnement des échelles et des barreaux. Nous espérions trouver un modèle de symétrie ou une loi qui pourrait s'appliquer dans chaque cas.
Ce sujet est compliqué et surtout très large, nous étions donc un peu perdues et ne savions plus vers quelle voie nous tourner.

Notre professeure nous a alors montré des vidéos de danses roumaines qui mettaient en scène des algorithmes dans le but de nous faire découvrir une nouvelle perspective des mathématiques.

Ces danses nous ont beaucoup inspirées, principalement celle qui consistait à comparer des nombres 2 à 2. Nous avons alors directement fait le rapprochement avec notre sujet.
Cette technique s'appelle le « tri à bulles » et nous nous en sommes inspirées pour réaliser notre algorithme de tri.

IV) Le fonctionnement de l'algorithme :

Notre algorithme a été programmé via le logiciel Algobox.
Son mécanisme n'est pas bien compliqué, il compare les nombres 2 à 2 et se charge d'intervertir les nombres de place s'ils ne sont pas déjà dans l'ordre croissant.

1) Explication du code
Nos différentes variables :

```
VARIABLES
  B EST_DU_TYPE NOMBRE
  L EST_DU_TYPE LISTE
  n EST_DU_TYPE NOMBRE
  i EST_DU_TYPE NOMBRE
  C EST_DU_TYPE NOMBRE
  NB EST_DU_TYPE NOMBRE
  ECH EST_DU_TYPE NOMBRE
  SEG EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  //n est le nombre de cartes numériques à ranger
  //i est un compteur pour note boucle
  //B=1 on avance, B=0 rien ne se passe
  //C est la valeur intermédiaire
  //L est la liste
  //NB est le compteur de barreaux
```

```
B PREND_LA_VALEUR 1
NB PREND_LA_VALEUR 0
LIRE n
ECH PREND_LA_VALEUR 0
SEG PREND_LA_VALEUR 39
POUR i ALLANT_DE 1 A n
  DEBUT_POUR
  LIRE L[i]
  FIN_POUR
TANT_QUE (B=1) FAIRE
  DEBUT_TANT_QUE
  B PREND_LA_VALEUR 0
  POUR i ALLANT_DE 1 A n-1
    DEBUT_POUR
    SI (L[i]>L[i+1]) ALORS
      DEBUT_SI
      C PREND_LA_VALEUR L[i+1]
      L[i+1] PREND_LA_VALEUR L[i]
      L[i] PREND_LA_VALEUR C
      B PREND_LA_VALEUR 1
      NB PREND_LA_VALEUR NB+1
      TRACER_SEGMENT (i,SEG)->(i+1,SEG)
      SEG PREND_LA_VALEUR SEG-1
      FIN_SI
    FIN_POUR
  FIN_TANT_QUE
POUR ECH ALLANT_DE 1 A n
  DEBUT_POUR
  TRACER_SEGMENT (ECH,40)->(ECH,SEG-1)
  FIN_POUR
POUR i ALLANT_DE 1 A n
  DEBUT_POUR
  AFFICHER L[i]
  FIN_POUR
AFFICHER "On a utilisé "
AFFICHER NB
AFFICHER " barreaux."
```

lecture de la liste initiale à ranger

le travail de tri !

affichage des barreaux horizontaux

Affichage des barreaux verticaux (les échelles)

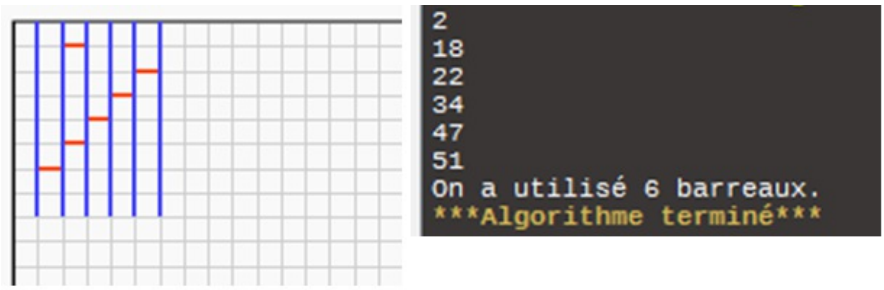
Affichage de la liste rangée

La variable B va permettre à l'algorithme de savoir s'il doit continuer à comparer les cartes ou bien s'il peut s'arrêter. En effet au démarrage B prend la valeur 0, et à chaque déplacement de cartes, B va prendre la valeur 1. Tant que B=1, l'algorithme continue. Il va ainsi traiter toutes les cartes dans l'ordre et les comparer deux à deux. Cela correspond à la variable « i ».

Si L(i), (un nombre de la liste) est plus grand que L(i+1) (le nombre placé à sa droite), alors on va chercher à échanger les 2 cartes de place et pour se faire, on va faire intervenir une variable que l'on appelle C. Cette variable va servir de valeur intermédiaire, elle va prendre la valeur de L(i+1) pour pouvoir la conserver pendant que L(i+1) soit la carte de droite prendra la valeur L(i) (la carte de gauche). La valeur L(i), soit la carte qui était placée à gauche, va prendre la valeur de L(i+1) qui a été conservée dans la variable C.
Les 2 cartes ont échangé leur place, et elles sont donc bien dans l'ordre croissant.
Une fois cette opération réalisée, la variable B va prendre la valeur 1 ce qui permettra à l'algorithme de continuer à fonctionner. La variable NB, le nombre de barreaux, va prendre la valeur qu'il avait auparavant à laquelle on ajoute 1, car l'opération que nous venons de réaliser a permis de tracer un barreau.

Nous avons également programmé une partie de l'algorithme en incluant 2 nouvelles variables (ECH et SEG) de sorte qu'il affiche le dessin correspondant aux barreaux et aux échelles pour pouvoir mieux visualiser les opérations réalisées précédemment.

2) Test
On teste cet algorithme avec la liste initiale 18, 34, 22, 47, 51, 2.



A la fin, l'algorithme affiche bien la liste des valeurs des cartes rangées dans l'ordre croissant et le nombre de barreaux utilisés pour classer les nombres.

3) Preuve de programme :
Montrons que le programme s'arrête toujours et qu'à la sortie la liste est bien rangée.
On s'intéresse à la taille t de la sous-liste non-rangée. Au départ t=n, n étant défini au lancement de l'algorithme.

18	***34***	22	47	51	2
18	***34***	***22***	47	51	2
18	22	***34***	***47***	51	2
18	22	34	***47***	***51***	2
18	22	34	47	***51***	***2***
18	22	34	47	2	***51***

Liste initiale non-rangée (taille de la liste : 6)

Le plus grand nombre est rangé à sa place à la fin des boucles « Pour... » présentes dans la première boucle « Tant que... ». La taille de la liste non rangée diminue donc de 1 à la fin de la première boucle « Tant que... ».

La sous-liste non rangée (taille de la liste : 5)

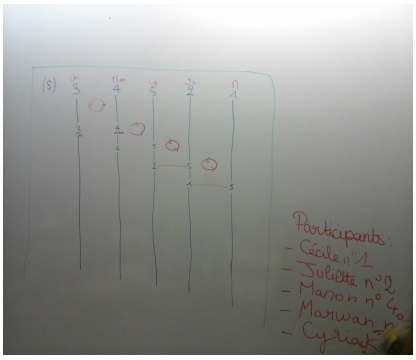
On a donc une suite de nombres entiers strictement décroissante (la suite des tailles de la liste non rangée) qui finira donc forcément par 0.

La variable B va conserver la valeur 0. Cela mettra donc un terme à la boucle « Tant que (B==1) » et par conséquent à l'algorithme.
A un moment donc la liste non rangée est de taille nulle, ce qui signifie que la liste finale est rangée.

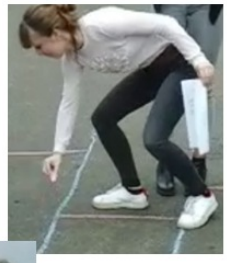
V) <u>La vidéo :</u>

Un algorithme n'est pas toujours facile à comprendre, et encore moins à imaginer. C'est pourquoi nous avons décidé de réaliser une vidéo qui correspond au mécanisme de l'algorithme. Nous étions 6 élèves à participer, chacun ayant un numéro et placés dans un ordre aléatoire. Nous avons choisi la même liste que celle testée précédemment avec notre programme sur Algobox.

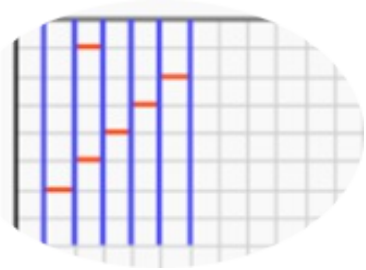
Le choix des participants :



La réalisation :



Le résultat :



Idem !

VI) Pour aller plus loin :

Lorsque Vincent Guirardel (le mathématicien qui nous a proposé les problèmes) est venu voir l'avancement de nos projets, il nous a proposé, à la suite de notre algorithme, d'essayer de répondre à une autre question.
Il s'agissait de savoir si l'on pouvait toujours ranger n'importe quel nombre de cartes mais en utilisant cette fois-ci, non pas des barreaux mais des « cordes ». Une « corde » signifie qu'au lieu de ne pouvoir échanger une carte qu'avec ses deux voisins (la carte située à sa gauche et la carte située à sa droite) elle peut aussi se déplacer de deux colonnes ou plus, c'est-à-dire que l'on peut comparer deux cartes non voisines ensemble.

Nous avons seulement eu le temps d'élaborer quelques idées ; nous avons cependant presque mis au point un nouvel algorithme.
Des cartes sont donc placées aléatoirement au sommet du nombre équivalent d'échelles. Ce que notre algorithme va faire, c'est comparer les cartes entres elles, d'abord les 2 premières, en mettant dans une nouvelle variable, la plus petite des deux cartes. Ensuite, il va comparer la valeur de la 3^e carte avec la valeur présente dans la variable puis conserver la plus petite des 2. Soit la variable restera la même dans le cas où la 3^e carte est plus grande, soit la valeur de la variable sera écrasée et remplacée par celle de la 3^e carte dans le cas où cette dernière est plus petite.
Ce procédé sera reproduit tout le long de la première ligne. Une fois que le plus petit nombre de la liste de cartes est dans la variable, il sera placé sur l'échelle la plus à gauche à l'aide d'une « corde ». Cette partie du dessin sera isolée pour le restant de l'algorithme et le même schéma se reproduit avec les cartes restantes jusqu'à ce que les cartes soient triées dans l'ordre croissant.