ALGORITHMIQUE AU LYCÉE
Thème 1 - Probabilités

Après une initiation à l'algorithmique et à la programmation au cours de l'année de seconde, les élèves peuvent utiliser les connaissances acquises (utilisation de boucles et d'instructions conditionnelles) pour répondre à des problèmes mathématiques résolubles au niveau du lycée (ou non !). Le choix des problèmes à proposer dépend évidemment de l'expérience acquise, de la réactivité de la classe et du langage de programmation choisi par l'enseignant.
Nous vous proposons ici un premier article sur le thème des probabilités, deux autres à suivre dans les prochains bulletins porteront sur les thèmes analyse et géométrie.
Le thème des probabilités est très riche en applications et peut donner lieu à de jolis problèmes de modélisation. Il convient néanmoins d'être progressif dans leur présentation et dans ce qui peut être donné à chercher. Nous balayons les programmes de Première et Terminale des filières S ou STAV, en débordant parfois du côté des enseignants, avec des situations à présenter en classe pour *aiguiser l'appétit* des élèves. Ce côté *mettre les maths en situation* est particulièrement riche pédagogiquement et peut donner lieu à de mini-projets à caractère interdisciplinaire.
Les algorithmes écrits en langage naturel ont volontairement des présentations variées.

Exercice 1 : Loi binomiale (niveau STAV)

Un jardinier amateur plante 10 graines de rose. Une étude statistique a montré que la probabilité de germination de chaque graine est de 0,4. Par ailleurs les germinations sont supposées indépendantes.

Question 1 : Soit X la variable aléatoire qui compte le nombre de graines ayant germé sur les 10 graines plantées. Justifier que X suit une loi binomiale dont vous préciserez les paramètres.

Question 2 : On propose l'algorithme suivant :

Entrée
Affecter à n la valeur 10
Affecter à S la valeur 0
Affecter à R la valeur 0
Traitement
Pour k variant de 0 à 2 faire
Affecter à S la valeur $S+\binom{n}{k}0{,}4^k\times 0{,}6^{n-k}$
Fin **Pour**
Affecter à R la valeur $1-S$
Sortie
Afficher R

a) - Tester cet algorithme en remplissant le tableau ci-dessous :

Étape	k	S
Entrée		0
1re exécution de la boucle	0	……..
2e exécution de la boucle	1	……...
3e exécution de la boucle	2	…….

R = ……..

b) - A quoi correspond le résultat renvoyé par cet algorithme ?

Question 3 : Modifier l'algorithme précédent pour calculer la probabilité qu'au moins cinq graines germent si le jardinier plante 15 graines et que la probabilité de germination de chaque graine est de 0,7.

Éléments de solutions

Question 1

Démarche classique (répétition de 10 expériences de Bernoulli indépendantes...), X suit la loi binomiale de paramètres $n = 10$ et $p = 0{,}4$.

Ainsi, pour tout entier k compris entre 0 et 10, on a $P(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$

Question 2

L'algorithme renvoie comme valeur

$$1-\left(\binom{10}{0}0{,}4^0\times 0{,}6^{10}+\binom{10}{1}0{,}4^1\times 0{,}6^9+\binom{10}{2}0{,}4^2\times 0{,}6^8\right)$$

c'est-à-dire $P(X\geq 3)$, la probabilité qu'au moins 3 graines aient germé.

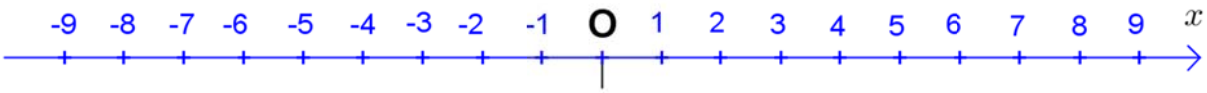
Question 3

Ici, X suit la loi binomiale de paramètres $n = 15$ et $p = 0{,}7$ et on cherche $P(X\geq 5)$.

L'algorithme précédent se modifie alors ainsi :

Entrée	Affecter à n la valeur 15 Affecter à S la valeur 0 Affecter à R la valeur 0
Traitement	**Pour** k variant de 0 à 4 faire Affecter à S la valeur $S+\binom{n}{k}0{,}7^k\times 0{,}3^{n-k}$ Fin **Pour**
Sortie	Affecter à R la valeur $1-S$ Afficher R

Exercice 2 : Une marche aléatoire (niveau Première S)



Un mobile part de l'origine O et effectue, chaque seconde, un saut d'une unité vers la gauche ou vers la droite avec la même probabilité. On note x_t son abscisse à l'instant t (en secondes).

Question 1 : Écrire un algorithme qui demande à l'utilisateur de saisir l'entier t ($t > 1$), et qui calcule et affiche l'abscisse x_t du mobile à l'instant t.

Question 2 : Justifier que la probabilité que le mobile retourne à l'origine est nulle si t est impair.

Question 3 : Écrire un algorithme qui simule **une** marche aléatoire et qui renvoie la valeur de t pour laquelle la particule revient pour la première fois à l'origine.

Question 4 : Modifier l'algorithme précédent de manière à simuler **un nombre** N de marches aléatoires (N saisi par l'utilisateur), et à calculer le temps moyen de premier retour à l'origine pour la particule.

Éléments de solutions

On donne les solutions sous forme de pseudo-code et une implémentation en langage Python. Cet exercice se programme aussi bien sur calculatrices TI 83 ou CASIO 35 + USB.

Question 1

Algorithme en langage naturel		Programme avec Python
Variables :	t est un entier naturel strictement supérieur à 1	from random import *
	i est un entier naturel	x = 0
	x est un entier relatif	t = int(input("saisir t"))
	k est un entier aléatoire entre 1 et 2	i = 0
Initialisation :	**Lire** t	while i < t :
	x prend la valeur 0	k = randint(1 , 2)
	i prend la valeur 0	if k = = 1
Traitement :	**Tant que** i < t	x = x – 1
	k prend aléatoirement la valeur 1 ou 2	else :
	Si $k = 1$	x = x + 1
	Alors x prend la valeur $x - 1$	i = i + 1
	Sinon x prend la valeur $x + 1$	print("x = " , x)
	Fin **Si**	
	i prend la valeur $i + 1$	
	Fin **Tant que**	Attention ! utiliser = = et pas = pour la condition
Sortie :	**Afficher** x	

Programme avec TI 83		Programme avec Casio 35 + USB	
Prompt T	X – 1 → X	? → T	Else X + 1 → X
0 → X	Else	0 → X	IfEnd
0 → I	X + 1 → X	0 → I	I + 1 → I
While I < T	End	While I < T	WhileEnd
entAléat(1 , 2) → K	I + 1 → I	RanInt # (1 , 2) → K	X◢
If K = 1	End	If K = 1	
Then	Disp X	Then X – 1 → X	

Notons qu'en Python, on doit importer dès le départ la bibliothèque random *via la commande* from random import * *(cela permet de se servir de toutes les fonctions prédéfinies de cette bibliothèque).*

La commande randint(*a* , *b*) *renvoie un entier (pseudo-)aléatoire entre a et b inclus. On peut interpréter la marche aléatoire comme étant le résultat (sur t observations) d'une succession de t lancers d'une pièce équilibrée. 1 symbolise* "pile"*: on se déplace vers la gauche d'une unité, et 2 symbolise* "face" *: on se déplace vers la droite d'une unité. Quelques tests effectués par le professeur avec une pièce peuvent être éclairants.*

Question 2

Partant de l'origine, le nombre de déplacements pour y revenir est nécessairement pair (avec 2 déplacements minimum). Un raisonnement par l'absurde amène immédiatement à une contradiction.

Question 3

Ligne	Algorithme en langage naturel	Programme avec Python
	Variables : x est un entier relatif k est un entier aléatoire parmi 1 et 2 t est un entier strictement supérieur à 1	from random import * k = randint(1 , 2) if k = = 1 :
1	**Traitement :** Affecter à k aléatoirement la valeur 1 ou 2	x = – 1
2	**Si** $k = 1$	else :
3	**Alors** Affecter à x la valeur – 1	x = 1
4	**Sinon** Affecter à x la valeur 1	t = 1
5	Fin **Si**	while x ! = 0 :
6	Affecter à t la valeur 1	k = randint(1 , 2)
7	**Tant que** $x \neq 0$	if k = = 1 :
	Affecter à k aléatoirement la valeur 1 ou 2	x = x – 1
	Si $k = 1$	else :
	Alors Affecter à x la valeur $x - 1$	x = x + 1
	Sinon Affecter à x la valeur $x + 1$	t = t + 1
	Fin **Si**	print("temps de premier retour :" , t)
	Affecter à t la valeur $t + 1$	
	Fin **Tant que**	
	Sortie : **Afficher** t	

Programme avec TI 83		Programme avec Casio 35 + USB	
entAléat(1 , 2) → K	entAléat(1 , 2) → K	RanInt # (1 , 2) → K	Then X – 1 → X
If K = 1	If K = 1	If K = 1	Else X + 1 → X
Then	Then	Then – 1 → X	IfEnd
– 1 → X	X – 1 → X	Else 1 → X	T + 1 → T
Else	Else	IfEnd	WhileEnd
1 → X	X + 1 → X	1 → T	T◢
End	End	While X ≠ 0	
1 → T	T+1 → T	RanInt # (1 , 2) → K	
While X ≠ 0	End	If K = 1	
	Disp T		

Remarque : *La particule a pour abscisse initiale 0 ;* ***avant de parler de son retour à l'origine, il est nécessaire qu'elle ait effectué un premier déplacement****. D'où la nécessité des lignes 1 à 6. La variable t prend ainsi la valeur 1 (ligne 7). Les lignes suivantes décrivent le déplacement de la particule, qui va s'arrêter dès qu'elle sera revenue à l'origine pour la première fois (une boucle* ***Tant que*** *est nécessaire).*

Question 4

Il s'agit de répéter le script de la question précédente N fois. On obtient ainsi une structure de boucles imbriquées. On pourra le tester avec $N = 100$.

Algorithme en langage naturel	Programme avec Python
Variables : N, i et t sont des entiers naturels x est un entier relatif k est un entier aléatoire parmi 1 et 2 *temps_total* est un entier naturel **Initialisation :** Lire N Affecter à *temps_total* la valeur à 0 **Traitement :** **Pour** i variant de 1 à N Affecter à x la valeur 0 Affecter à k aléatoirement la valeur 1 ou 2 **Si** $k = 1$ **Alors** Affecter à x la valeur -1 **Sinon** Affecter à x la valeur 1 Fin **Si** Affecter à t la valeur 1 **Tant que** $x \neq 0$ Affecter à k aléatoirement la valeur 1 ou 2 **Si** $k = 1$ Alors Affecter à x la valeur $x - 1$ **Sinon** Affecter à x la valeur $x + 1$ Fin **Si** Affecter à t la valeur $t + 1$ Fin Tant que Affecter à *temps_total* la valeur *temps_total* + t Fin **Pour** **Sortie :** **Afficher** *temps_total*/N	from random import * N = int(input("Nombre d'expériences à effectuer ?")) temps_total = 0 for i in range(N) : x = 0 k = randint(1 , 2) if k = = 1 : x = – 1 else : x = 1 t = 1 while x != 0 : k = randint(1 , 2) if k = = 1 : x = x – 1 else : x = x + 1 t = t + 1 temps_total = temps_total + t print("Temps moyen de premier retour :" , temps_total/N)

Dans les programmes TI ou Casio, la variable *temps_total* est nommée par la lettre Y

Programme avec TI 83		Programme avec Casio 35 + USB	

Prompt N
0 → Y
For(I , 1 , N)
0 → X
entAléat(1 , 2) → K
If K = 1
Then
– 1 → X
Else
1 → X
End
1 → T
	While X ≠ 0
	entAléat(1 , 2) → K
	If K = 1
	Then
	X – 1 → X
	Else
	X + 1 → X
	End
	T + 1 → T
	End
	Y + T → Y
	End
	Disp Y/N
		? → N
		0 → Y
		For 1 → I To N
		0 → X
		RanInt # (1 , 2) → K
		If K = 1
		Then – 1 → X
		Else 1 → X
		IfEnd
		1 → T
		While X ≠ 0
		RanInt # (1 , 2) → K
		If K = 1
			Then X – 1 → X
			Else X + 1 → X
			IfEnd
			T + 1 → T
			WhileEnd
			Y + T → Y
			Next
			Y/N ◢

Exercice 3 : Autre marche aléatoire avec temps d'arrêt (niveau Première S)

Une souris peut se déplacer dans un tunnel muni de deux portes. La première peut s'ouvrir dans les deux sens et la seconde uniquement dans un sens.

On modélise cette situation de la façon suivante : Un mobile se déplace de façon aléatoire sur un axe. À chaque instant, il se trouve en l'un des points O, A, B d'abscisses respectives 0, 1, 2.

- Si à l'instant n, il se trouve en O, alors à l'instant $n+1$, il se trouve obligatoirement en A.
- Si à l'instant n, il se trouve en A, alors à l'instant $n+1$, il se trouve soit en O, soit en B avec la même probabilité.
- Si à l'instant n, il se trouve en B, alors à l'instant $n+1$, il se trouve obligatoirement en B.

On appelle Y la variable aléatoire prenant pour valeur le nombre de déplacements nécessaires au mobile pour atteindre le point B.

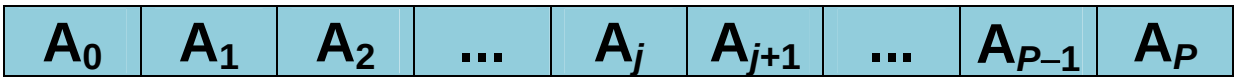
Question 1

1) Écrire un algorithme en langage naturel, que vous implémenterez dans un langage informatique de votre choix demandant à l'utilisateur :
   a) de saisir le nombre n de répétitions de l'expérience aléatoire qui consiste à observer le mobile jusqu'à ce qu'il atteigne le point B,
   b) qui calcule le nombre moyen de déplacements nécessaires pour que le mobile atteigne B au cours de ces n expériences.

2) Vous testerez ce programme avec $n = 20\,000$.

3) En déduire une estimation de l'espérance de Y.

La situation précédente se généralise aisément.
Dans le cas présent, on suppose le tunnel muni de P portes ($p \geq 3$). L'abscisse du mobile varie donc entre 0 (position à l'origine) et P.



Les règles de parcours sont adaptées du cas particulier précédent.
On modélise cette situation de la façon suivante : Un mobile se déplace de façon aléatoire sur un axe. À chaque instant, il se trouve en l'un des points A_0, A_1,…, A_P d'abscisses respectives 0, 1,..., P.

- Si à l'instant n, il est en A_0, alors à l'instant $n+1$, il est obligatoirement en A_1.
- Si à l'instant n, il est en A_j ($1 \leq j \leq P-1$), alors à l'instant $n+1$, il est soit en A_{j-1}, soit en A_{j+1} avec la même probabilité.
- Si à l'instant n, il est en A_P, alors à l'instant $n+1$, il est obligatoirement en A_P.

Question 2

1) Écrire un algorithme en langage naturel, que vous implémenterez dans un langage informatique de votre choix demandant à l'utilisateur :
   a) de saisir le nombre P de portes du tunnel,
   b) de saisir le nombre n de répétitions de l'expérience aléatoire qui consiste à observer le mobile jusqu'à ce qu'il atteigne le point A_P,
   c) qui calcule le nombre de déplacements moyen nécessaires pour que le mobile atteigne A_P au cours de ces n expériences.

2) Compléter le tableau suivant en prenant $n = 20\,000$.

Nombre P de portes	2	3	4	5	6
Nombre moyen de déplacements pour atteindre A_P					

3) En déduire une relation possible liant E(Y) à P.

Éléments de solutions :

Analyse

L'expérience aléatoire consiste à observer le cheminement du mobile jusqu'à ce que ce dernier atteigne le point B. Il faudra bien prendre en compte les données de l'énoncé, à savoir : si le mobile est en O, son abscisse est automatiquement incrémentée de 1 ; s'il est en A son abscisse peut soit augmenter, soit diminuer de 1 de manière équiprobable, et s'il est en B, il y reste. On observera sur un grand nombre n de répétitions de cette expérience le temps de parcours, et on en déduira le temps moyen de parcours pour aller de O à B.

Question 1

On donne d'abord un algorithme modélisant une unique expérience et qui calcule le nombre de déplacements nécessaire pour atteindre B. Notons que nous pouvons simuler ceci *à la main* à l'aide d'une pièce équilibrée avant de passer à la programmation informatique. Il s'agit par ailleurs d'un excellent moyen pratique de concevoir l'algorithme demandé en analysant à chaque étape ce qui se passe : "*J'avance de O en A.*" ; lancer d'une pièce au niveau de A ; "*Si je fais* pile*, j'avance en B sinon je recule en O (et donc je reviens aussitôt en A au coup d'après)...*", "*Arrivé en B, je m'arrête.*".

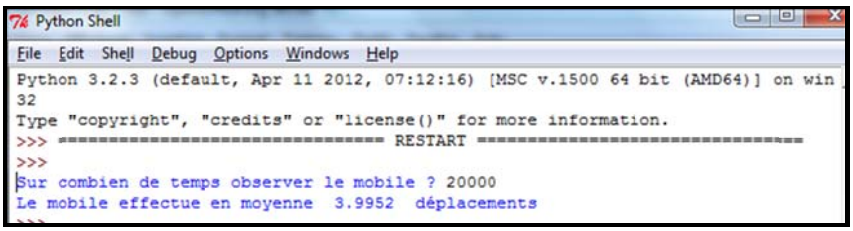
Algorithme pour une marche	**Programme avec Python**
Variables : x est un entier entre 0 et 2 k est un entier aléatoire parmi 1 et 2 y est un entier naturel **Initialisation :** x prend la valeur 0 y prend la valeur 0 **Traitement :** **Tant que** $x < 2$ **Si** $x = 0$ **Alors** x prend la valeur 1 **Sinon** k prend la valeur d'un entier aléatoire parmi 1 et 2 **Si** $k = 1$ Alors x prend la valeur $x + 1$ **Sinon** x prend la valeur $x - 1$ Fin **Si** Fin **Si** y prend la valeur $y + 1$ Fin **Tant que**	from random import * x = y = 0 while x < 2 : if x == 0 : x = 1 else : k = randint(1 , 2) if k = = 1 : x = x + 1 else : x = x – 1 y = y + 1

Programme avec TI 83	Programme avec Casio 35 + USB
0 → X 0 → Y While X < 2 If X = 0 Then 1 → X Else entAléat(1 , 2) → K If K = 1 Then X + 1 → X Else X – 1 → X End End Y + 1 → Y End	0 → X 0 → Y While X < 2 If X = 0 Then 1 → X Else RanInt # (1 , 2) → K If K = 1 Then X + 1 → X Else X – 1 → X IfEnd IfEnd Y + 1 → Y WhileEnd

Pour répéter ceci sur *n* expériences, nous avons besoin de trois variables supplémentaires *i*, *n* et *d*, le nombre de déplacements total, pour calculer le nombre moyen de déplacements par marche.

Algorithme pour *n* marches	Programme avec Python
Variables : *d*, *i*, *n* et *y* sont des entiers naturels *x* est un entier entre 0 et 2 *k* est un entier aléatoire parmi 1 et 2 **Initialisation :** *d* prend la valeur 0 Demander la valeur de *n* *i* prend la valeur 0 **Traitement :** **Tant que** *i* < *n* *x* prend la valeur 0 *y* prend la valeur 0 **Tant que** *x* < 2 **Si** *x* = 0 **Alors** *x* = 1 **Sinon** *k* prend aléatoirement la valeur 1 ou 2 **Si** *k* = 1 **Alors** *x* prend la valeur *x* + 1 **Sinon** *x* prend la valeur *x* – 1 Fin **Si** Fin **Si** *y* prend la valeur *y* + 1 Fin **Tant que** *d* prend la valeur *d* + *y* *i* ← *i*+1 Fin **Tant que** **Sortie :** **Afficher** « temps moyen : » , *d*/*n*	from random import * d = 0 n = int(input("Nombre d'expériences à effectuer ?")) i = 0 while i < n : x = y = 0 while x < 2 : if x = = 0 : x = 1 else : k = randint(1 , 2) if k = = 1 : x = x + 1 else : x = x – 1 y = y + 1 d = d + y i = i + 1 print("temps moyen : " , d/n)

En testant ce programme avec *n* = 20 000, on obtient que le nombre moyen de déplacements nécessaires pour atteindre B est approximativement de 4, ce qui donne une estimation de l'espérance mathématique de *Y*.



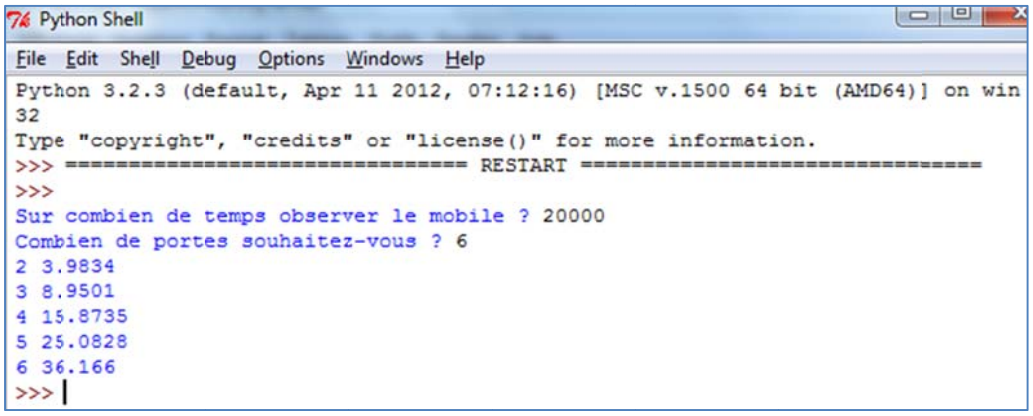
Programme avec TI 83		Programme avec Casio 35 + USB	
0 → D Prompt N 0 → I While I < N 0 → X 0 → Y While X < 2 If X = 0 Then 1 → X Else entAléat(1 , 2) → K If K = 1	Then X + 1 → X Else X – 1 → X End End Y + 1 → Y End D + Y → D I + 1 → I End Disp D/N	0 → D ? → N 0 → I While I < N 0 → X 0 → Y While X < 2 If X = 0 Then 1 → X Else RanInt # (1 , 2) → K If K = 1 Then X + 1 → X Else X – 1 → X	End End Y + 1 → Y End D + Y → D I + 1 → I End D/N◢

Question 2

L'algorithme demandé généralise naturellement le précédent. Il suffit de modifications mineures pour l'écrire à partir du précédent.

Algorithme	Programme avec Python
Variables : *d*, *i*, *n*, *x*, *P* et *y* sont des entiers naturels *k* est un entier aléatoire parmi 1 et 2 **Initialisation :** *d* prend la valeur 0 Saisir *P* Saisir *n* *i* prend la valeur 0 **Traitement :** **Tant que** *i* < *n* *x* prend la valeur 0 *y* prend la valeur 0 **Tant que** *x* < *P* **Si** *x* = 0 **Alors** *x* = 1 **Sinon** *k* prend la valeur d'un entier aléatoire : 1 ou 2 **Si** *k* = 1 **Alors** *x* prend la valeur *x* + 1 **Sinon** *x* prend la valeur *x* – 1 Fin **Si** Fin **Si** y prend la valeur *y* + 1 Fin **Tant que** *d* prend la valeur *d* + *y* *i* prend la valeur *i* + 1 Fin **Tant que** **Sortie :** **Afficher** « temps moyen : » , *d*/*n*	from random import * y = 0 P = int(input("Combien de portes ?")) n = int(input("Nombre d'expériences à effectuer ?")) i = 0 while i < n : x = y = 0 while x < P : if x = = 0 : x = 1 else : k = randint(1 , 2) if k = = 1 : x = x + 1 else : x = x – 1 y = y + 1 d = d + y i = i + 1 print("temps moyen : " , d/n)

En utilisant un programme un peu plus élaboré (mais identique au niveau des performances et du résultat) que celui présenté juste avant, on obtient la série de résultats qui suit :



Le nombre de gauche représente le nombre de portes et celui de droite le temps moyen d'absorption en A_P. Ceci revient au même de tester le programme précédent en affectant à P plusieurs valeurs et en appuyant sur les touches Ctrl + F5 plusieurs fois de suite.

Il semble que l'espérance de Y soit égale à P^2, P désignant le nombre de portes. Autrement dit, le temps moyen de parcours pour aller de O à A_P semble être de P^2 secondes.

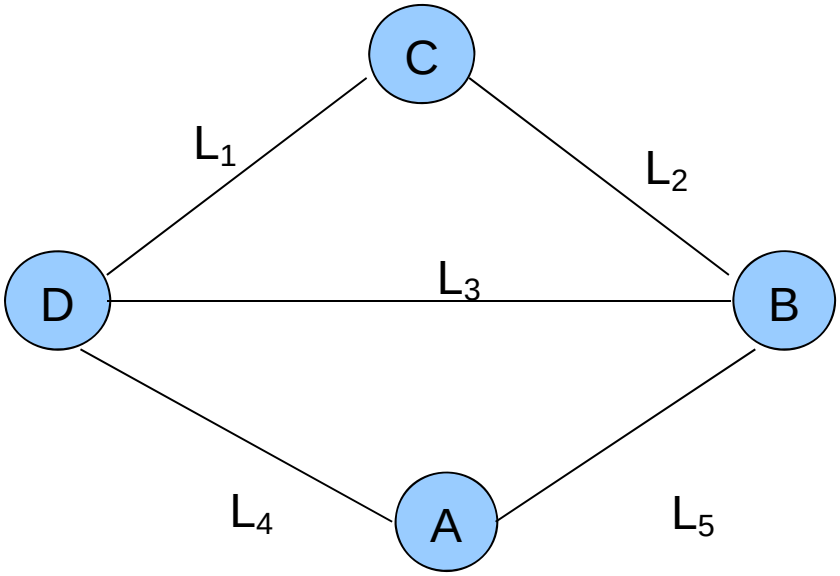
Programme avec TI 83	**Programme avec Casio 35 + USB**
0 → D Prompt P Prompt N 0 → I While I < N 0 → X 0 → Y While X < P If X = 0 Then 1 → X Else entAléat(1 , 2) → K If K = 1 Then X + 1 → X Else X – 1 → X End End Y + 1 → Y End D + Y → D I + 1 → I End Disp D/N	0 → D ? → P ? → N 0 → I While I < N 0 → X 0 → Y While X < P If X = 0 Then 1 → X Else RanInt # (1 , 2) → K If K = 1 Then X + 1 → X Else X – 1 → X End End Y + 1 → Y End D + Y → D I + 1 → I End D/N◢

Exercice 4 : Modélisation d'un réseau d'information (niveau 1ère S)

Remarque :

Cet exercice, issu d'un article de la RMS et un peu plus ardu que les précédents, utilise la notion de liste, qui doit donc avoir été travaillée précédemment. L'algorithme reste néanmoins relativement court, ce qui peut le placer dans le cadre d'un TP d'une durée d'une heure trente à deux heures.

On considère un réseau de sommets ABCD (que l'on peut se représenter comme un réseau d'ordinateurs par exemple) reliés par des liaisons L_1 à L_5 comme représenté sur le schéma suivant.



Ce réseau a pour objet de transmettre de l'information de A vers D. Ces informations transitent par n'importe quel chemin en fonctionnement : par exemple $L_5L_2L_1$ est un chemin possible et il fonctionne si et seulement si les trois liaisons L_5, L_2 et L_1 fonctionnent.
Chaque liaison a la même probabilité p de fonctionnement durant une unité de temps. Durant chaque unité de temps, on considère qu'une liaison ne change pas d'état : ou bien elle reste en fonctionnement ou bien elle est en panne.

Les cinq liaisons sont indépendantes les unes des autres en ce qui concerne leurs pannes.

On dit que le réseau fonctionne durant une unité de temps si l'un des chemins possibles entre A et D a permis le passage de l'information entre ces points.

Créer un algorithme modélisant le fonctionnement du réseau sur n unités de temps que l'on implémentera informatiquement. En déduire une estimation de la probabilité de fonctionnement $f(p)$ du réseau en fonction de p. Pour ceci, on complétera le tableau suivant, en donnant à n la valeur 20 000.

p	$f(p)$
0	
0,1	
0,2	
0,3	

p	$f(p)$
0,4	
0,5	
0,6	
0,7	

p	$f(p)$
0,8	
0,9	
1	

Éléments de solutions

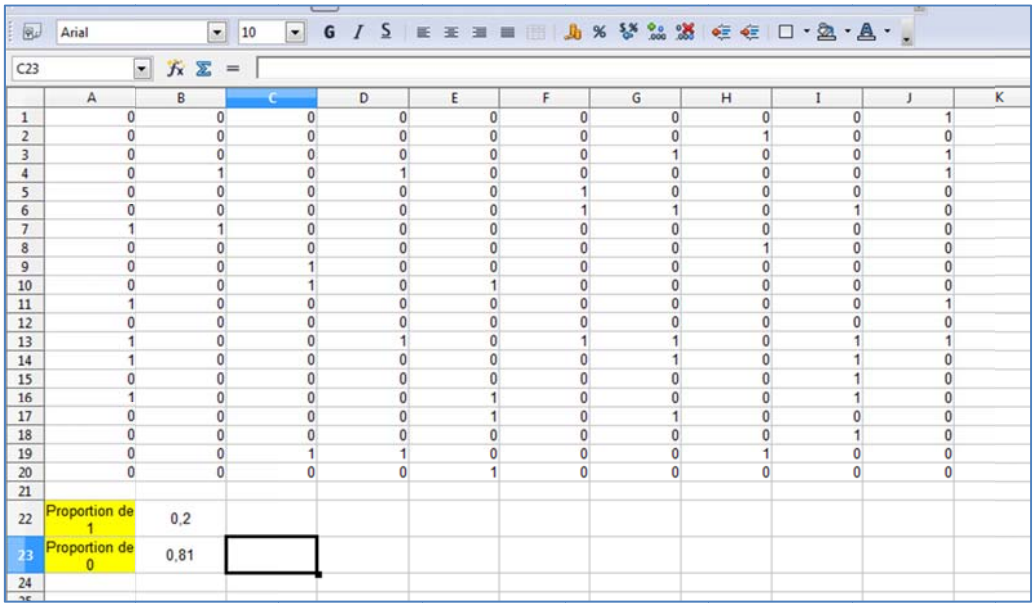
Analyse

Le but de l'algorithme est de modéliser le fonctionnement du réseau. Les différents chemins menant de A à D sont : L_4, L_5L_3 et $L_5L_2L_1$. Il s'ensuit d'après les hypothèses du texte que le circuit fonctionne au cours d'une unité de temps si et seulement si l'un au moins de ces chemins est opérationnel (on n'oriente pas les chemins)

- On va être amené à construire une matrice ligne de 5 éléments, chacun d'entre eux représentant l'état de la liaison : 1 pour en état de marche et 0 sinon. Par exemple [1 , 0 , 0 , 1 , 0] signifie que les chemins L_1 et L_4 sont en état de marche, mais pas L_2, ni L_3, ni L_5.
- Un problème se pose alors : Comment doit-on compléter aléatoirement la matrice ligne avec "1" et "0"? À ce moment, on doit prendre en compte les données du texte : chaque liaison a la même probabilité *p* de fonctionnement durant une unité de temps. L'idée est de procéder comme on le ferait dans un tableur pour générer des nombres en proportion déterminée à l'avance en utilisant une formule du type **`=SI(ALEA()<=0,2;1;0)`** qui génère des "1" en proportion 0,2.

Remarque

Pour une première utilisation de cette formule avec les élèves, on peut vérifier sa pertinence. Ici elle a été copiée dans 200 cellules et on vérifie qu'elle génère des "1" en proportion 0,2 et des "0" en proportion 0,8.



	A	B	C	D	E	F	G	H	I	J	K
1	0	0	0	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	0	1	0	0	
3	0	0	0	0	0	0	1	0	0	1	
4	0	1	0	1	0	0	0	0	0	1	
5	0	0	0	0	0	1	0	0	0	0	
6	0	0	0	0	0	1	1	0	1	0	
7	1	1	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	1	0	0	
9	0	0	1	0	0	0	0	0	0	0	
10	0	0	1	0	1	0	0	0	0	0	
11	1	0	0	0	0	0	0	0	0	1	
12	0	0	0	0	0	0	0	0	0	0	
13	1	0	0	1	0	1	1	0	1	1	
14	1	0	0	0	0	0	1	0	1	0	
15	0	0	0	0	0	0	0	0	1	0	
16	1	0	0	0	1	0	0	0	1	0	
17	0	0	0	0	1	0	1	0	0	0	
18	0	0	0	0	0	0	0	0	1	0	
19	0	0	1	1	0	0	0	1	0	0	
20	0	0	0	0	1	0	0	0	0	0	
21											
22	Proportion de 1	0,2									
23	Proportion de 0	0,81									
24											

On a calculé les proportions de "1" et de "0" dans les cellules C22 et C23. Bien entendu, plus le nombre de cellules est grand plus les fréquences calculées se rapprochent de ce que l'on attend en théorie (on peut éventuellement parler de la loi des grands nombres).

- On simulera le fait qu'un segment fonctionne avec une probabilité *p* en générant un réel aléatoire entre 0 et 1. Si ce nombre est inférieur ou égal à *p*, on renvoie 1 et 0 sinon. On répète cette opération pour chacun des 5 segments. Il n'est pas inintéressant d'insister sur le fait qu'il faut considérer un nombre important de matrices lignes pour *estimer* la probabilité de fonctionnement du réseau.
- Enfin, le réseau fonctionne si et seulement si le segment L_4 **OU** les segments L_5 et L_3 **OU** les segments L_5, L_2 et L_1 fonctionnent. Ceci est à la base de l'algorithme.

Dans l'algorithme suivant, t est la variable qui va compter le nombre de fois où la transmission de l'information de A à D a été effectuée sur une unité de temps.

Algorithme en langage naturel

Variables : N, i, j et t sont des entiers naturels
p est un nombre réel entre 0 et 1
k est un nombre réel aléatoire entre 0 et 1
L est une matrice ligne

Initialisation : Lire N
Lire p
Affecter à t la valeur 0

Traitement : **Pour** i variant de 1 à N
  Pour j variant de 1 à 5
    Affecter à k un nombre réel aléatoire entre 0 et 1
    Si $k \leq p$
      Alors Affecter la valeur 1 au j-ème élément de la liste L
      Sinon Affecter la valeur 0 au j-ème élément de la liste L
    Fin **Si**
  Fin **Pour**
  Si L = [.. , .. , .. , 1 , ..] OU L = [.. , .. , 1 , .. , 1] OU L = [1 , 1 , .. , .. , 1]
    Alors Affecter à t la valeur $t + 1$
  Fin **Si**
Pour

Sortie : **Afficher** t/N

Programme avec Python

```
from random import *
p = float(input("Entrez la probabilité de fonctionnement de chaque segment "))
n = int(input("Sur combien d'unités de temps voulez-vous observer le réseau ? "))
test_bon = 0
for i in range(n) :                          #vecliaisons est la matrice ligne d'état du réseau
    vecliaisons = [ ]                        #on initialise vecliaisons comme une liste vide
    for j in range (5) :                     #on va définir l'état des 5 branches du réseau
        k = random( )                        #on génère un réel aléatoire entre 0 et 1
        if k <= p :
            vecliaisons.append(1)            #si k ≤ p on ajoute 1 à la liste vecliaisons
        else :
            vecliaisons.append(0)            #sinon, on ajoute 0
    if vecliaisons[3] == 1 or (vecliaisons[4] == 1 and vecliaisons[2] == 1)
                    or (vecliaisons[4] == 1 and vecliaisons[1] == 1 and vecliaisons[0] == 1) :
        test_bon = test_bon + 1
print("La fréquence de fonctionnement du réseau par unité de temps est de " , test_bon/n)
```

Remarque :

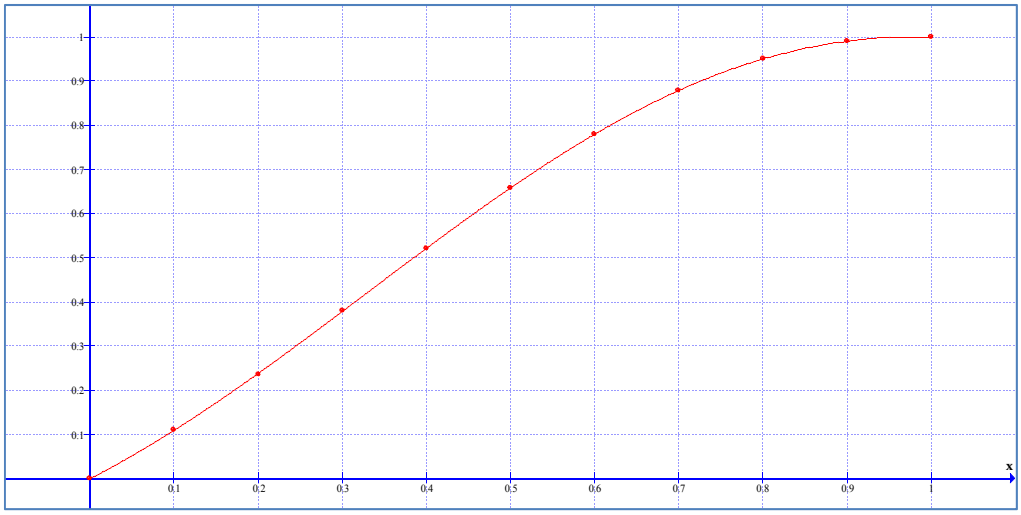
En Python le premier élément d'une liste se note LISTE (0), *le second* LISTE(1), *etc. l'instance « append » permet d'ajouter un élément à la fin d'une liste.*
En testant ce script avec n = 20 000 et en donnant à p des valeurs de 0 à 1 avec un pas de 0,1, on peut compléter le tableau demandé. Bien entendu, les résultats varient à chaque fois que l'on saisit une même valeur de p (fluctuation...) sauf pour p = 0 et p = 1 !!

p	*f* (*p*)
0	0
0,1	0,110 95
0,2	0,236 65
0,3	0,380 55

p	*f* (*p*)
0,4	0,521 00
0,5	0,656 65
0,6	0,780 15
0,7	0,878 35

p	*f* (*p*)
0,8	0,950 2
0,9	0,989 15
1	1

En utilisant la bibliothèque *numpy* (à installer) qui dispose d'un grapheur, ou en reportant ces données dans n'importe quel grapheur, on obtient un nuage de points, à partir duquel on peut tracer une courbe d'ajustement.



Remarques

- *L'expression exacte de f en fonction de p n'est pas au programme de Première S (petit exercice pour les enseignants !). La solution est donnée à l'adresse suivante :* http://univ-irem.fr/videos/PackageTutoPython *(modélisation d'un réseau d'information Parties 1 et 2).*
- *En langage TI ou Casio, le premier élément d'une liste se note* LISTE(1)*, le second* LISTE(2)*, etc. D'autre part, il est inutile d'initialiser une liste vide au départ du programme.*

Programme avec TI 83

```
Prompt N
Prompt P
0 → T
For(I , 1 , N)
For(J , 1 , 5)
NbrAléat → K
If K ≤ P
Then
1 → L1(J)
Else
0 → L1(J)
End
End
If L1(4) = 1
  ou L1(3) = 1
     et L1(5) = 1
  ou L1(1) = 1
     et L1(2) = 1
     et L1(5) = 1
Then
T + 1 → T
End
End
Disp T/N
```

Programme avec Casio 35 + USB

```
? → N
? → P
0 → T
For 1 → I To N
For 1 → J To 5
Ran # → K
If K ≤ P
Then 1 → List 1[J]
Else 0 → List 1[J]
IfEnd
Next
If List 1[4] = 1
  Or List 1[3] = 1
     And List 1[5] = 1
  Or List 1[1] = 1
     And List 1[2] = 1
     And List 1[5] = 1
ThenT + 1 → T
IfEnd
Next
T/N◢
```

Remarque : Le fonctionnement des calculatrices CASIO et TI rend prioritaire le "ET " sur le "OU", ce qui n'est pas la règle en logique où ils ont le même niveau de priorité.