Épreuve de mathématiques pratiques et informatique
Rapport du jury – Session 2017

1 Modalités de l'épreuve

Un sujet d'oral est composé d'un unique exercice imposé au candidat, portant sur le programme de mathématiques et d'informatique de BCPST. Dans la mesure du possible, les sujets ont été choisis pour couvrir l'ensemble du programme des deux années (BCPST1 et BCPST2).

Les candidats disposent de 30 minutes de préparation dans une salle dédiée, où ils ont accès à un ordinateur sur lesquels sont installés les logiciels Pyzo (Python 3), Excel et Geogebra. **À partir de la session 2018, sera proposé également sur les ordinateurs le logiciel Spyder, les candidats pouvant utiliser pour leurs programmes en Python au choix les logiciels Pyzo ou Spyder**.

Les candidats disposent d'une clé USB fournie par le concours au début de leur préparation, sur laquelle ils peuvent enregistrer leur travail.

À l'issue de la préparation, ils sont accompagnés dans une salle d'interrogation (5 minutes au maximum sont prévues pour la transition), où ils disposent d'environ 18-20 minutes pour exposer le résultat de leur travail de préparation et dialoguer avec l'examinateur. Dans cette salle de passage, ils disposent également d'un ordinateur connecté à un vidéoprojecteur, sur lequel ils peuvent exécuter les programmes sauvegardés sur la clé USB fournie.

À l'issue de ce premier temps d'interrogation, les candidats sont accompagnés vers une deuxième salle, dans la mesure du possible voisine de la précédente, où ils présentent le résultat de leur projet informatique durant 18-20 minutes devant un second examinateur. Dans cette deuxième salle, les candidats disposent d'un ordinateur avec vidéoprojecteur sur lequel est affiché le dossier qu'ils ont au préalable envoyé au concours au début du mois de juin.

2 Éléments statistiques

Un résumé statistique est fourni ci-dessous, pour chacune des parties de l'épreuve ainsi que pour la note finale (chaque partie était notée sur 10). On notera que la corrélation entre la note d'un candidat sur la partie Mathématiques pratiques et sa note sur la partie Projet informatique reste comme les années précédentes assez faible ($r = 0{,}38$, $r^2 = 0{,}15$). En particulier, d'assez nombreux candidats ayant de grosses difficultés en mathématiques ont pu montré des compétences très intéressantes en informatique.

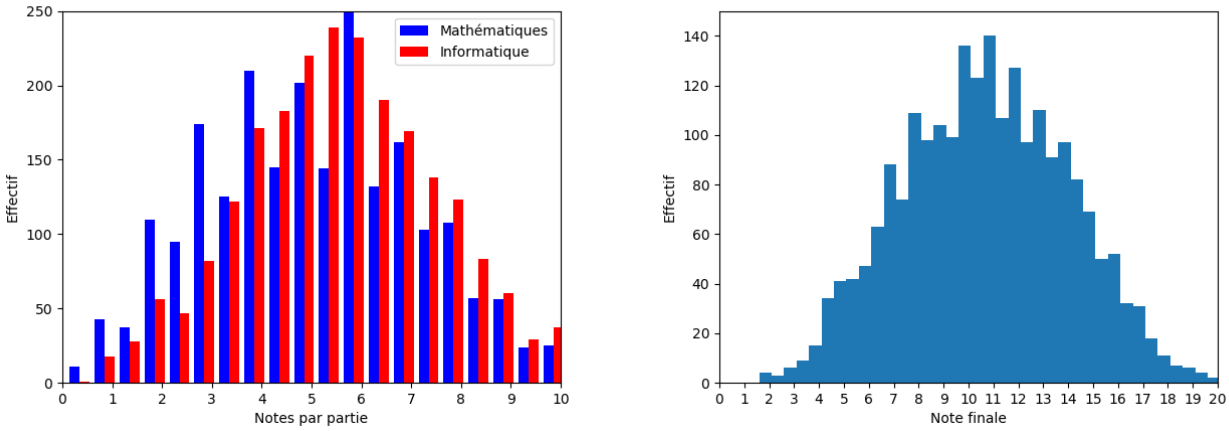
Les distributions des notes sont semblables aux années précédentes concernant chacune des parties, mais les notes globales obtenues sont un peu moins dispersées, les notes extrêmes sur 20 ayant été faibles (très peu de candidats ont eu une note maximale (ou minimale) dans les deux épreuves).

Partie	Moyenne	Médiane	Écart-type	$0 \leqslant x < 2$	$2 \leqslant x < 4$	$4 \leqslant x < 6$	$6 \leqslant x < 8$	$8 \leqslant x \leqslant 10$
Mathématiques	5,16	5	2,09	9%	27%	34%	23%	7%
Informatique	5,66	5,5	1,91	5%	19%	39%	28%	9%

Résumé statistique par partie de l'épreuve

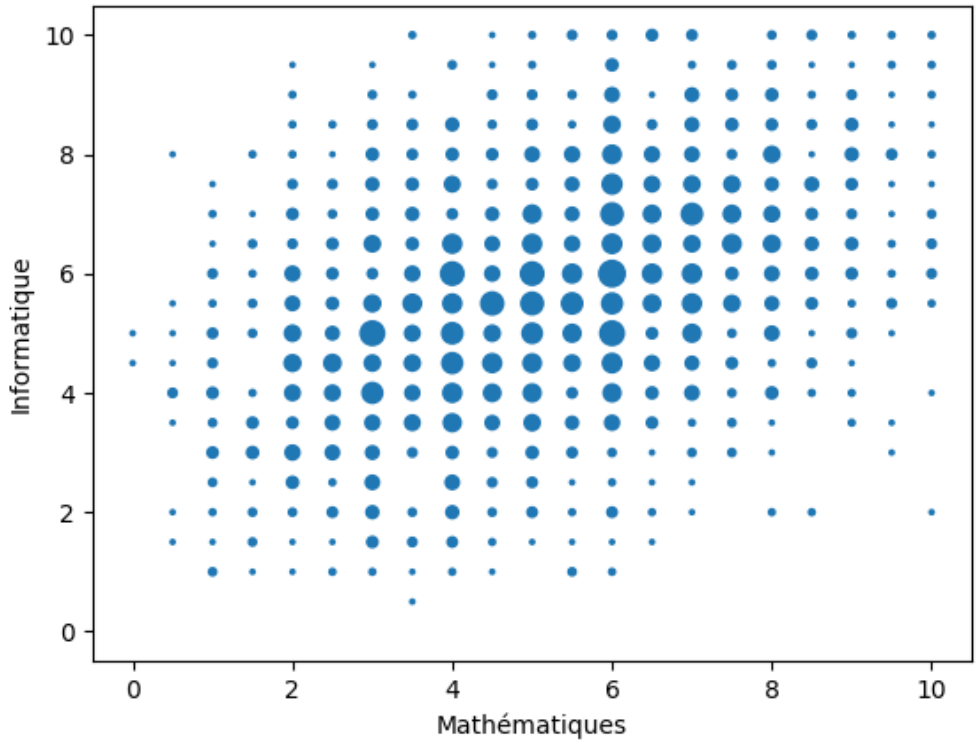
	Moyenne	Médiane	Écart-type	$0 \leqslant x < 4$	$4 \leqslant x < 8$	$8 \leqslant x < 12$	$12 \leqslant x < 16$	$16 \leqslant x \leqslant 20$
Note finale	10,81	11	3,32	2%	22%	42%	29%	5%

Résumé statistique général



Histogrammes des notes

Dans le graphique suivant, la taille du disque de centre (x, y) est proportionnelle au nombre de candidats ayant obtenu la note x à la partie Mathématiques pratiques et y à la partie Informatique. Comme dit plus haut, les notes sont assez faiblement corrélées ; ce graphique est encore très similaire à celui des deux années précédentes.



3 Partie Mathématiques Pratiques

3.1 Ce qui est attendu des candidats en Mathématiques Pratiques

Dans un but d'équité, tous les candidats convoqués à une même heure sont examinés sur le même sujet, quel que soit leur jury d'interrogation. Les examinateurs connaissent les énoncés sur lesquels portent l'interrogation, il est donc inutile **et même déconseillé** de relire l'énoncé.

Pendant la période d'interrogation, un dialogue entre le candidat et l'examinateur s'instaure. L'oral est un échange, n'est pas un écrit, et en aucun cas n'est une conférence du candidat face à un jury muet. Les questions du jury visent à évaluer les compétences du candidat, et cherchent avec bienveillance à l'aider pour compléter son argumentation ou se rendre compte d'une erreur. Elles ne constituent en aucun cas des pièges tendus. Le jury attend de la bonne volonté et une attitude ouverte pour répondre aux questions.

Il n'est pas indispensable d'avoir traité la totalité de l'exercice pour obtenir une excellente note. Il est préférable d'avoir mené un raisonnement rigoureux et argumenté, reposant sur des connaissances solides, plutôt que d'avoir donné tous les résultats (même justes), trop vite et sans explication réelle.

Au début de l'interrogation, le candidat est en général invité à préciser la liste des questions qu'il a eu le temps d'aborder pendant la préparation. Cette question n'a pas pour but d'évaluer a priori le candidat, mais permet de gérer au mieux le déroulement de l'oral, et éviter que certaines questions préparées par le candidat n'aient pas le temps d'être exposées.

Nous demandons aux candidats, à partir de la session 2018, de démarrer systématiquement leur interrogation par l'énoncé de la liste des questions qu'ils ont au moins partiellement résolue. L'examinateur le demandait habituellement jusqu'en 2017, nous attendrons des candidats que cette démarche soit spontanée de leur part à partir de 2018, et préparée avant l'entrée dans la salle d'interrogation, ceci afin de ne pas perdre trop de temps sur la durée de l'oral.

Chaque sujet comporte au moins une question d'informatique et les examinateurs interrogent systématiquement sur au moins une de ces questions, qu'elle ait été préparée ou non par le candidat en amont. Le cas échéant, le jury peut poser une question élémentaire pour vérifier les compétences du candidat en informatique.

À partir de la session 2018, chaque sujet commencera par une brève question de cours, pas nécessairement liée au thème de l'exercice qui suivra. Les seules questions de cours qui pourront être posées seront des énoncés de définitions ou de théorèmes, à l'exclusion de toute démonstration. Le but est simplement de vérifier brièvement la connaissance d'un point du cours, et ne doit pas donner lieu à un développement trop long. Si cela est pertinent, le candidat pourra se contenter d'une réponse orale.

Exemples :

- Définition d'une valeur propre d'une matrice carrée A.
- Formule des probabilités totales.
- Espérance et variance d'une variable aléatoire de loi exponentielle de paramètre λ
- Convergence et somme d'une série géométrique.
- etc.

La durée de l'interrogation orale doit impérativement être respectée. Lorsque l'examinateur annonce la fin du temps réglementaire, il est attendu que le candidat efface efficacement le tableau et range rapidement ses affaires, afin de ne pas pénaliser le candidat suivant.

3.2 Remarques générales sur la forme de l'oral Mathématiques Pratiques

- Pour cette troisième année où cet oral est mis en place, nous avons encore une fois fortement apprécié que les candidats soient bien préparés à l'épreuve. Nous tenons à féliciter les préparateurs qui ont manifestement tenu compte des remarques des rapports 2015 et 2016 dans la préparation de leurs étudiants.
- Le niveau général des candidats est plutôt bon, notamment en probabilités. Peu de candidats sont en grande difficultés face aux notions du programme. La connaissance du cours était moyennement satisfaisante, ce qui justifie les propositions d'évolutions pour 2018 (introduction d'une question de cours). Certains candidats montrent toujours de grosses lacunes en calcul et en analyse (calculs de dérivées et primitives, connaissance des fonctions usuelles,...), mais cette année, ce sont les exercices d'algèbre linéaire (même classiques) qui ont été ceux qui ont été le moins réussis par les candidats.

- L'énoncé des définitions et des théorèmes au programme doit être précis, notamment sur le rappel des bonnes hypothèses. Le jury se réserve le droit de poser une question de cours à tout moment de l'interrogation s'il le juge nécessaire, pour s'assurer de la bonne compréhension du candidat.
- Le jury tient à encourager les candidats à émettre un avis critique sur leurs résultats lorsqu'ils sont clairement incohérents (tableau de variations en désaccord avec les limites, obtention d'une probabilité supérieure à 1, etc.)
- Faire l'impasse sur une ou plusieurs parties du programme de mathématiques ou d'informatique peut avoir de lourdes conséquences sur la prestation du candidat et peut être très pénalisant pour son évaluation.
- Les candidats doivent pouvoir s'adapter à des sujets d'oraux parfois originaux, de longueur très diverse, notamment ceux où il est attendu un peu de modélisation. Les examinateurs connaissent les sujets et leur difficulté relative, et adaptent leurs attentes et la notation en conséquence. Certains candidats sortent de leur préparation en ayant traité très peu de questions, et commencent leur oral en étant déstabilisés. L'examinateur trouvera toujours le moyen d'évaluer lors de la discussion les connaissances du candidat en le guidant dans la résolution ou en lui posant des question supplémentaires.

3.3 Remarques générales sur le contenu

Algèbre. En 2017 encore, la plupart des exercices proposés en algèbre linéaire et bilinéaire était élémentaire et leur résolution a été décevante par un trop grand nombre de candidats. La cause de ces échecs était souvent due à une connaissance trop approximative du cours. Nous espérons que les futurs candidats seront tenir compte de nos remarques pour les sessions futures.

Probabilités. Les probabilités discrètes étaient globalement mieux maîtrisées par les candidats que lors des années précédentes, malgré des erreurs récurrentes lors de la manipulation des séries, par exemple lors de l'étude de l'existence ou du calcul d'une espérance.
Les produits de convolution restent mal maîtrisés par les candidats, même dans des cas simples, la formule étant pourtant systématiquement rappelée par les énoncés.
Encore trop souvent, les candidats ont du mal à distinguer les notions d'événements et de variables aléatoires.
Enfin, le théorème central limite reste souvent méconnu des candidats.

Analyse. Les candidats perdent souvent leur temps d'interrogation dans la présentation de leurs méthodes et calculs en analyse dans la présentation de leur oral. Lorsque le candidat le juge pertinent, il peut proposer au jury de donner directement le résultat d'une étape calculatoire (ceci étant vrai également pour la résolution d'un système, le calcul des racines d'un trinôme, ...) le jury se réservant le droit de demander des explications complémentaires.

Informatique. Les candidats montrent toujours une nette amélioration chaque année en informatique, témoin d'un bon accompagnement par leurs enseignants pour acquérir les compétences du langage Python. En particulier, la simulation des variables aléatoires usuelles est bien maîtrisée.
L'algorithme de dichotomie, pourtant redonné dans chaque sujet l'utilisant (énoncé dans un cadre neutre et général) est souvent mal compris et utilisé de manière incorrecte par les candidats.
Les tracés de courbes en Python sont peu corrects dans l'ensemble. Nous ne pouvons que conseiller les futurs candidats de savoir parfois utiliser le logiciel Geogebra qui, sans avoir besoin d'être expert dans son usage, permet également de tracer rapidement des courbes.
Signalons que les sujets conseillent parfois d'utiliser des bibliothèques. Ce ne sont que des indications, les candidats peuvent très bien utiliser un autre logiciel plutôt que celui proposé.

3.4 Conclusion

Le niveau des candidats est très hétérogène, avec tous les intermédiaires entre le candidat brillant et le candidat n'ayant quasiment aucune connaissance ni compétence. L'examinateur s'attache à poser des questions adaptées pour évaluer le niveau du candidat de manière exacte.
Le ressenti de certains candidats n'est pas toujours en adéquation avec la note obtenue. Tous les jurys s'efforcent d'être aimables et bienveillants, mais n'en ont pas moins pour mission d'évaluer et de classer les candidats. Les examinateurs sont tenus à un devoir de réserve quant à l'évaluation du candidat, ce qui peut parfois être

interprété à tord par certains candidats pour de la froideur ou de l'indifférence. Un candidat peut ressortir déçu de son interrogation et peut néanmoins obtenir une note tout à fait satisfaisante, et inversement.
L'ensemble des examinateurs a une fois de plus apprécié l'attitude respectueuse et courtoise de tous les candidats, ce qui a facilité le bon déroulement des oraux de cette session 2017.

4 Partie Informatique

4.1 Modalités de l'épreuve

La partie de l'épreuve consacrée au projet informatique durait vingt minutes, dans un format qui a légèrement évolué cette année. Les candidats disposaient de sept minutes (au maximum) pour présenter leur projet, suivies d'un entretien. Le jury est très satisfait de cette évolution, qui a permis de mieux juger de la maîtrise du projet par le candidat, mais également d'évaluer de manière plus générale ses compétences en informatique, conformément au programme d'informatique des classes de BCPST.

Pour la partie présentation, il appartient au candidat de faire comprendre à l'examinateur ce qui fait l'intérêt algorithmique de son projet :

- solution algorithmique élégante,
- modélisation pertinente,
- qualité des résultats obtenus dans un problème d'optimisation,
- etc.

Pour le premier point de l'entretien, le candidat doit être capable de répondre précisément aux questions qu'on peut lui poser sur le code :

- quels sont les arguments de cette fonction ?
- que renvoie-t-elle si on l'appelle sur telle valeur ?
- comment pourrait-on ajouter telle fonctionnalité ?
- justifier et/ou décrire les structures de données,
- etc.

Pour évaluer le deuxième point, les examinateurs ont pu demander aux candidats de traiter au tableau un exercice de programmation. Le plus souvent, cet exercice était basé sur le projet :

- rendre une fonction plus générale ou plus efficace,
- ré-écrire de manière plus concise une partie du projet traitée maladroitement,
- écrire une fonction parcourant une structure de données du projet,
- écrire une fonction à l'aide des fonctions du projet,
- reprogrammer dans un cas plus simple une fonction du projet,
- etc.

Si le projet ne se prêtait pas à ce type de question, l'exercice pouvait être sans rapport direct avec le sujet. Les candidats présentant un projet ambitieux et apparemment bien maîtrisé mais incapables d'écrire au tableau une fonction élémentaire (déterminer le maximum d'une liste, compter le nombre d'occurrences d'une valeur dans une structure de données par exemple) ont été fortement pénalisés.

4.2 Remarques générales

Nous reprenons ici une grande partie des points évoqués dans les rapports précédents. Les attentes du jury sont maintenant globalement comprises et peu de projets très faibles ont été présentés cette année.

- La très grande majorité des candidats semble s'être investie dans leur projet, et avoir ce faisant acquis des compétences qui leur seront utiles tant dans la suite de leurs études que dans leur vie professionnelle.
- La maîtrise générale des projets était globalement bonne et le format très contraint de la présentation n'a posé de problème qu'à un très petit nombre de candidat.
- Les candidats qui mettent en place une modélisation doivent veiller à faire la différence entre la situation initiale (souvent liée à la biologie) et la modélisation informatique ; ainsi, on ne peut apporter une réponse de nature biologique à une question de nature algorithmique.
- Les candidats veilleront à garder un esprit critique quant à la portée des programmes qu'ils ont réalisés. En particulier, il est de bon aloi de se demander si les objectifs que l'on s'était fixés initialement sont remplis ou non.

- Les présentations orales, malgré quelques défauts récurrents évoqués plus bas, étaient globalement satisfaisantes et illustrées de schémas clairs et bien choisis.
- Aucun problème majeur d'organisation n'a été constaté cette année.

4.3 Remarques sur la présentation

Lors de la phase de présentation du projet, le candidat dispose du diaporama qu'il a déposé préalablement par voie électronique et qui sert de support à sa présentation. Le dossier déposé par le candidat est directement mis à disposition sur l'ordinateur utilisé par le candidat mais il est rappelé que les candidats doivent venir avec une clé USB de secours contenant leurs fichiers en cas de problème informatique. L'usage de notes est proscrit.

Une bonne présentation doit être synthétique. Nous attirons l'attention des candidats sur le fait que la présentation des structures de données et algorithmes utilisés est cruciale. Le jury apprécie les candidats qui présentent l'enjeu de leur projet, les hypothèses qui ont été faites dans la modélisation choisie (le cas échéant), l'architecture générale du projet et une analyse des résultats et des perspectives ultérieures.

Le jury a apprécié que les candidats n'aient pas passé trop de temps à présenter des concepts non-informatiques (biologiques, physiques, mathématiques, etc.). Les candidats ont généralement pris soin d'expliciter leurs modèles, et/ou les règles des jeux étudiés.

Nous rappelons aux candidats que l'examinateur n'est pas censé interrompre le candidat durant cette phase, et que si les examinateurs ont pris connaissance du projet du candidat, ils n'ont pas passé une année à travailler sur celui-ci ; il faut donc éviter de se retrouver dans une situation où l'examinateur ne voit pas où le candidat veut en venir. Pour les mêmes raisons, les présentations contenant de trop nombreuses pages ou présentées en récitant un texte à toute vitesse pour tenir dans les sept minutes imposées sont à proscrire.

Pour un jeu, demander à l'examinateur s'il connaît déjà les règles ou s'il souhaite qu'on les lui explique rapidement semble une idée raisonnable.

Il est fortement recommandé de commenter une ou deux fonctions présentant un intérêt algorithmique durant cette phase de présentation. Il est donc souhaitable que ces fonctions fassent apparaître des algorithmes non triviaux. Les candidats prendront garde à ne pas cacher les difficultés dans des fonctions auxiliaires non présentées.

Il est par contre inutile de lister l'ensemble des fonctions présentes dans le programme (y compris dans un diagramme d'appels qui sera en général illisible).

Une conclusion est évidemment la bienvenue, dans des proportions raisonnables : présenter les résultats de sa modélisation durant la moitié de l'oral est donc à éviter.

Le jury apprécie une bonne maîtrise du vocabulaire. Nous rappelons à ce titre que la structure `if` n'est pas une « boucle » : en effet, elle ne permet pas d'effectuer de répétitions. Le jury préfère entendre "dans l'instruction `if`", voire même "dans le `if`", plutôt que "dans la boucle `if`".

4.4 Qualité des projets

Le thème du projet est complètement libre, ce qui permet en règle générale aux candidats de choisir un sujet qui les motive. Il est important de choisir un projet dont la difficulté soit en adéquation avec le niveau du candidat. Le nombre de candidats excellents et la qualité de leurs projets n'ont pas évolué de manière significative.

Le jury rappelle que le temps passé à travailler les graphismes et l'interface du projet est beaucoup moins productif d'un point de vue comptable que celui passé à améliorer les aspects algorithmiques. Une interface graphique via l'utilisation d'un module (tkinter, pygame, etc.) est bienvenue seulement si elle vient en complément d'un projet algorithmiquent intéressant mais ne peut pas s'y substituer.

Les données d'entrée d'un programme doivent être passées en argument (ou via un fichier) et non demandées à l'utilisateur par des `input`, dont l'utilisation doit être réservée à des interactions avec l'utilisateur au cours du programme (par exemple pour un jeu).

Nous présentons ici quelques écueils à éviter :

- certains jeux de société ont des règles assez compliquées dont la traduction informatique ne présente que peu d'intérêt mais demande du temps. Programmer un Monopoly, par exemple, demande de traiter toute une série de cas particuliers et de cartes aux effets variés mais ne permet pas vraiment de mettre en valeur les compétences du candidat ;
- de manière plus générale, le code ne devrait pas faire apparaître de longues disjonctions de cas (quatre ou plus). Si de telles séries de `if...elif...` semblent nécessaires au candidat, c'est sans doute qu'il n'a pas suffisamment réfléchi à la structure du code ;

- lorsque le programme traite un grand nombre de données, il est pertinent de les stocker dans un fichier annexe.

Certains projets se prêtent bien à une analyse statistique des résultats : en général, cette analyse a été faite par les candidats.

Globalement, on a noté un effort des candidats pour éviter le code très répétitif (quatre fonctions différentes pour se déplacer dans les quatre directions par exemple) : ceux n'ayant pas fourni cet effort ont naturellement été pénalisés.

Il est pertinent pour les candidats de s'aider de références bibliographiques. On peut également s'attendre à ce qu'un candidat qui étudie un problème classique se soit un peu renseigné, ne serait-ce qu'en sollicitant un moteur de recherche ou son encyclopédie préférée : la question traitée admet-elle une réponse connue ou conjecturale ? d'où vient ce problème et de quelle époque date-t-il ? etc

4.5 Thèmes des projets

Cette année, les principaux thèmes abordés ont été les suivants :

- Algorithmes classiques
  - algorithmes inspirés de la biologie appliqués à des problèmes classiques : algorithmes de fourmis, algorithmes génétiques, réseaux de neurones, etc. Nous insistons sur le fait que ces algorithmes ont pour but de résoudre un problème informatique et non de modéliser des comportements réels, même s'ils s'en inspirent.
  - algorithmes pour la biologie : alignement de séquences, arbres phylogénétiques, etc.
  - cryptographie. L'analyse de la machine Enigma est technique mais pas algorithmiquement riche. Les méthodes de chiffrement trop basiques (César, Vigenère, etc.) ne sont souvent pas assez consistantes.
  - autres algorithmes : problèmes de graphes, labyrinthe, traitement d'image, méthodes numériques, etc.
- Modélisation. Il est possible d'obtenir une note moyenne mais difficile de briller sur ce type de projet. La qualité et la pertinence de la modélisation ne sont pas les principaux éléments évalués. La surenchère de détails et de paramètres n'enrichit pas un modèle, mais empêche souvent l'analyse des résultats obtenus. On attend d'autre part des candidats un minimum de lucidité sur la portée et les limites de leurs modèles.
- Jeux : jeux de plateau, jeux logiques (kakuro, binaro, logimages, logipix, etc.), jeux vidéos. C'est un domaine riche et varié. Les candidats doivent cependant faire attention au choix du jeu. Nous conseillons d'éviter les jeux ayant un trop grand nombre de règles entraînant trop de distinctions de cas (`if .. elif .. elif .. else ..`). Lorsque la programmation du jeu n'est pas algorithmiquement consistante, le projet devrait être complété par une ou plusieurs intelligences artificielles. Il est souvent intéressant de comparer plusieurs intelligences artificielles entre elles (a minima comparer une IA subtile avec une IA jouant aléatoirement). Il est souvent plus intéressant de ne pas se contenter d'automatiser une stratégie humaine.

4.6 Style et lisibilité du code

La lisibilité du code s'est améliorée, mais les conseils des rapports précédents restent d'actualité :

- Comme demandé dans la notice du concours, le fichier python principal doit contenir la définition d'une fonction ne prenant pas d'argument et permettant de tester de façon significative le projet (avec des paramètres par défaut bien choisis, en particulier pour que le temps de calcul reste raisonnable).
- Le code doit être raisonnablement commenté : une ou deux lignes au début de chaque fonction expliquant ce qu'elle prend en argument, ce qu'elle fait et ce qu'elle renvoie est très appréciable.
- Dans les noms de variables, de fonctions et de fichiers, il faut éviter tous les caractères dits spéciaux, c'est-à-dire se restreindre aux lettres non accentuées (minuscules et majuscules), aux chiffres et au "tiret bas" `_`.
- Éviter d'utiliser des chemins absolus : si l'examinateur souhaite tester le programme sur sa machine, il y a peu de chance qu'un appel `f = open("D:/MonProjetInfo/Donnees/especes.txt")` fonctionne et on écrira à la place `f = open("Donnees/especes.txt")`. Ces chemins relatifs ne fonctionnent sous Pyzo que si l'on utilise la commande *Execute file as script* (raccourci clavier `Ctrl-Maj-E`).
- La longueur des lignes doit être raisonnable (idéalement, pas plus de 80 colonnes). Pour faciliter cela, on pourra, dans le menu *View* de Pyzo, choisir une *Location of long line indicator* à 80 colonnes et désactiver l'option *Wrap long lines*. Au besoin, on peut placer manuellement un retour à la ligne en utilisant un `\` :
  `x = nom_de_variable_excessivement_long + \`

`encore_une_variable_dont_le_nom_est_tres_long`

- Si l'on s'intéresse, par exemple, à l'évolution d'une population vivant dans un environnement en deux dimensions représenté par une matrice carrée, les fonctions devront être écrites pour des matrices de taille n "abstraite", ou bien en définissant au début du programme la constante n, ou bien en récupérant en début de fonction la taille de la matrice. On ne devrait ainsi jamais trouver une ligne telle que `for i in range(50)` dans le code d'une telle fonction.
  De manière plus générale, on essaiera toujours d'écrire un code le plus générique possible : un projet ne fonctionnant que sur un exemple précis ne présente le plus souvent que peu d'intérêt.
- De même, plutôt que d'écrire `if M[i][j] == 2` avec éventuellement un commentaire précisant "on regarde si la cellule est vivante", on préférera définir `VIVANTE = 2` au début du programme et écrire ensuite `if M[i][j] == VIVANTE`.

4.7 Notions hors-programme

Les candidats ont le droit d'utiliser des notions hors-programme (ou à la limite du programme comme les fonctions récursives). Dans certains projets, leur emploi peut être judicieux, mais un candidat utilisant une telle notion doit savoir qu'il s'expose à des questions (élémentaires) portant dessus et être capable de justifier son utilisation.

4.8 Remarques spécifiques de programmation et d'algorithmique

Nous reprenons ici les remarques faites les années passées.

- Le code doit absolument être découpé en fonctions, et il faut réfléchir soigneusement au découpage le plus judicieux. Déjà rares l'année dernière, les candidats n'utilisant pas de fonction (ou mettant tout leur code dans une seule fonction, ce qui revient au même) étaient exceptionnels cette année : ils ne peuvent obtenir une note correcte.
- De nombreux projets nécessitent de parcourir les voisins d'une case d'un tableau bidimensionnel, ce qui a encore donné lieu à du code extrêmement maladroit.
- Il est fortement déconseillé d'utiliser un algorithme que l'on n'est pas capable d'expliquer correctement lorsque l'examinateur le demande. Si le projet est riche par ailleurs, le candidat peut éventuellement préciser qu'il s'est contenté de reprendre une correction de Travaux Dirigés ou que son professeur l'a aidé. Les projets très pauvres à l'exception d'une fonction relativement compliquée et non maîtrisée se sont vus attribués des notes très basses.
- L'algorithme de Dijkstra n'a d'intérêt que si le graphe que l'on considère est pondéré. Pour la recherche d'un plus court chemin dans un graphe non pondéré, on utilisera un parcours en largeur.
- Il est souvent possible en Python d'utiliser une boucle `for` là où une boucle `while` serait nécessaire dans certains langages (à l'aide typiquement d'un return dans le corps de la boucle). Les candidats sont encouragés à tirer partie de cette possibilité, mais cela ne les dispense pas de savoir écrire une boucle `while` en gérant correctement les conditions de sortie. Sur ce point, on ne constate que très peu d'amélioration cette année.
- Très peu de candidats maîtrisent la différence entre une fonction renvoyant une valeur et une fonction modifiant son argument. Le jury a ainsi vu de très nombreuses fonctions modifiant un argument (souvent une matrice) et renvoyant inutilement la matrice modifiée.
- Très peu de candidats utilisent correctement les booléens. Idéalement, on aimerait lire plus souvent
  `return x > 0`
  plutôt que
  ```
  if x >0:
      return True
  else:
      return False
  ```
  (par exemple), mais le jury est conscient que ce n'est pas évident conceptuellement.