PROJET INFORMATIQUE 2020 2021

Présentation générale

Pour l'oral du concours agro veto, un projet informatique doit être présenté. Il est compté dans la note de l'oral de math/info. Cet oral compte coefficient 4 à l'agro et PC Bio, et coefficient 2 à véto.

1. 1 Extrait de la notice 2019-2020 Mathématiques pratiques et Informatique (30 min de préparation et 40 min d'interrogation).

Cette épreuve se déroule en trois phases ; les deux parties de l'interrogation orale auront un poids similaire dans la note.

1. Préparation (30 minutes) : étude d'un problème mathématique relativement ouvert.

2. Présentation et dialogue sur le problème posé en préparation (20 minutes) :

Le traitement peut être de type expérimental dans un premier temps, abordant le problème avec un point de vue plus pratique, plus proche de la réalité des objets mathématiques étudiés, le calcul algébrique n'étant pas l'objectif premier. Le candidat aura été amené à observer et conjecturer, à partir de différentes données (graphiques, tableaux de données numériques, utilisation de logiciels ou de la calculatrice), la rigueur mathématique restant présente. Le sujet est conçu de manière assez ouverte pour que l'oral puisse prendre des directions différentes suivant les réactions du candidat.

L'épreuve évalue les compétences (I = projet informatique, M = mathématiques pratiques) :

– (I,M) Identifier un problème sous différents aspects ;
– (I) Initier des perspectives nouvelles et faire preuve de créativité ;
– (M) Concevoir un modèle à partir d'indications fournies ;
– (I,M) Résoudre un problème complexe :
• (I,M) Conduire des expérimentations dans une démarche exploratoire ;
• (I) S'organiser en fonction d'un but à atteindre, choisir une stratégie ;
• (I) Mettre en œuvre un traitement par étapes, un algorithme ;
• (M) Apporter un regard critique.
– (I,M) Communiquer à l'oral ;
– (I,M) Argumenter, convaincre.

Afin de permettre au candidat de mener une démarche exploratoire, un ordinateur est mis à sa disposition pendant la préparation. Il dispose de logiciels libres et d'usage général, notamment :

– GeoGebra (tracé de courbes, configurations géométriques, etc) ;
– Python (programmation et simulation), distribution Pyzo et Spyder, version 3.x ;
– un tableur.

NB : les questions posées pourront être traitées avec Python, mais les candidats sont libres d'utiliser également le tableur ou Geogebra s'ils le souhaitent.

Un memo d'utilisation du logiciel Python est disponible à la fin du rapport d'épreuve de la session 2019 sur le site Internet : www.concours-agro-veto.net, rubrique « Espace concours »_ « A BCPST »_« se préparer »_« Sujets et rapports » et dans la rubrique « Les commandes Python ».

3. **Présentation de projet (20 minutes) :**

Présentation du projet préparé pendant l'année (7 minutes maximum), puis échange avec l'examinateur autour du projet (13 minutes minimum).

Le candidat présente :

- Les objectifs du projet ;
- Son architecture générale ;
- Les éventuelles bibliothèques et outils logiciels utilisés en plus de Python et de sa bibliothèque standard ;

- Un ou deux extraits de code (correspondant chacun à une fonction) qui lui semblent être importants et intéressants ;
- Éventuellement, les améliorations envisageables, le partage des tâches dans le groupe, les difficultés rencontrées, etc.

L'exposé du projet d'informatique est suivi de questions destinées à évaluer la maîtrise du candidat ainsi que son implication dans le projet.
On attend du candidat qu'il soit capable de répondre précisément à des questions portant sur des extraits du code du projet choisis par l'examinateur. Même si les projets sont réalisés en groupe, chaque candidat doit maîtriser la totalité de son projet : les questions de l'examinateur peuvent porter sur toutes les parties du code.
Un ordinateur est à disposition de l'examinateur pour lui permettre de vérifier, s'il y a lieu, certains points de détail présentés par le candidat.
L'examinateur peut aussi poser des questions plus générales portant sur le programme d'informatique des deux années.
Modalités pratiques : les candidats doivent déposer leurs documents en version électronique sur le site du SCEI entre le jeudi 28 mai et le mercredi 3 juin 2020 en utilisant leurs identifiants de connexion.
Les candidats pourront, dans la période d'ouverture, supprimer s'ils le souhaitent, une version précédente des fichiers déposés précédemment et recharger une nouvelle version.
Les documents à déposer sont :
- Une fiche d'envoi formatée (coordonnées candidat, intitulé du projet) présentée en annexe de ce document, page 47 ;
- Une présentation au format pdf ;
- Le code source du projet au format py ;
- Les éventuels fichiers de données auxiliaires (données numériques, images, etc.).

Si les candidats ont un fichier son (ou une vidéo) généré par leur projet, ils peuvent le joindre mais il n'y a pas de garantie qu'il puisse être lu le jour de l'oral (multitude de formats possibles, etc.).
Le code fourni doit être clair et contenir des commentaires. Il est rappelé aux candidats que le choix judicieux des noms de fonctions et de variables améliore grandement l'intelligibilité du code (et permet d'alléger les commentaires).

Pour faciliter le travail des examinateurs qui pourront être amenés à vérifier l'exécution des programmes avant l'oral, les candidats sont appelés soit à inclure à la fin de leur fichier quelques lignes de script, soit à écrire une fonction sans argument nommée test, permettant de tester leur projet (avec des paramètres bien choisis, pour garantir un temps d'exécution raisonnable). Le choix devra être explicité en début de fichier, dans un commentaire clair et précis, et décrivant les paramètres (si nécessaire) et les résultats renvoyés lors de la compilation du script, ou de l'appel test(). L'ensemble des fichiers ne doit pas dépasser un volume total de 15 Mo.

Le nom de l'établissement de préparation ne doit figurer sur aucun document envoyé.

Le jour de l'oral, les candidats apportent une clé USB contenant les documents déjà déposés sur le site SCEI. Cette clé ne sera utilisée qu'en cas de problème technique.
Par ailleurs, dans chaque établissement concerné, le professeur d'informatique envoie par courriel (stephane.lebrigand@concours-agro-veto.net), ou par courrier au service des concours une liste des travaux de ses élèves, précisant pour chacun s'il est pour la première fois en BCPST 2 ou redoublant, et y porte la mention suivante :
Je soussigné, M……, professeur au Lycée…….atteste que les dossiers de mes élèves correspondent bien à un travail de l'année et, en particulier pour les redoublants, que c'est un nouveau travail.

1.2 Extrait du rapport de concours 2018-2019

Pour la partie présentation, dans le cadre explicité par la notice du concours, il appartient au candidat d'insister sur ce qui fait l'intérêt scientifique et algorithmique de son projet :

intérêt des structures de données utilisées, en regard de la situation étudiée (modélisation) ou des algorithmes utilisés,

solution algorithmique élégante,

qualité des résultats obtenus,

etc.

Pendant l'entretien, les examinateurs cherchent à évaluer la maîtrise du candidat et son implication dans son projet. Cela peut se faire, suivant les cas, en posant des questions d'ordre général sur le contexte algorithmique ou scientifique du projet, ou des questions précises sur le code python présenté par le candidat :

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.

Où dans le code est implémentée telle fonctionnalité ?

etc.

La maîtrise du programme d'informatique enseigné en BCPST est le plus souvent évaluée en demandant aux candidats de traiter au tableau un exercice de programmation. Le plus souvent, cet exercice était basé sur le projet :

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,

rendre une fonction plus générale ou plus efficace,

ré-écrire de manière plus concise une partie du projet traitée maladroitement,

etc.

mais il pouvait aussi être sans rapport direct si celui-ci ne se prêtait pas à ce type de question :

programmer récursivement de la factorielle (si la récursivité a été utilisée dans le projet),

écrire la définition d'une classe très simple (si celle-ci a été utilisée),

écrire une fonction testant une propriété des matrices ou des chaînes de caractère (si de tels objets ont été utilisés dans le projet),

programmer une fonction relative strictement au programme de BCPST (le plus souvent exercice basique sur les listes).

Les candidats étaient, dans l'ensemble, bien préparés à ce type d'exercice.

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.

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.

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.

Remarques sur la présentation

Lors de la phase de présentation du projet, le candidat dispose du diaporama au format PDF 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. Plusieurs candidats n'ont pas déposé correctement leurs fichiers par voie électronique (fichier powerpoint au lieu de pdf, fichier python incomplet, voire absence de fichiers) : ces candidats ont été pénalisés, même s'ils ont pu en général utiliser leur clé de secours.

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 que les candidats 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 ; 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 diapositives 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 . . . à condition de tenir compte de sa réponse.

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,

soient concises ou bien découpées en blocs dans le diaporama, afin qu'elles soient facilement lisibles.

Le jury a apprécié que, dans leurs diapositives, de nombreux candidats utilisent des flèches, des

encadrements, des accolades ou d'autres éléments graphiques permettant d'expliquer facilement leur code.

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 "dans le if ", plutôt que "dans la boucle if ".

Le jury est très satisfait de la qualité globale des présentations des candidats.

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.

Cette année, les projets étaient globalement de bonne qualité et de taille raisonnable. **Le nombre de lignes de code n'est pas une mesure de la qualité d'un projet.** Par exemple, un projet de qualité avec 300 lignes de codes peut se voir attribuer la note maximale. Le jury rappelle que le temps passé à travailler les graphismes et l'interface du projet est beaucoup moins rentable 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 algorithmiquement 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 présentant un grand nombre de cas particuliers dont la traduction informatique est chronophage et ne présente pas ou peu d'intérêt. Elle engendre de nombreuses distinctions de cas, imbrications de if, etc.

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.

Le code doit pouvoir être exécuté sans lever d'erreur.

On peut faire un projet informatique autour de son TIPE, mais ce projet ne doit pas se résumer à un traitement numérique de l'étude expérimentale du TIPE.

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.

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 :

Quelles solutions sont déjà connues ?

Quels algorithmes utilise-t-on habituellement ?

Quelles sont les structures de données pertinentes ?

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, méthode shotgun, arbres phylogénétiques, etc.
- cryptographie : attaque contre Vigenère, etc.

L'analyse de la machine Enigma est technique mais pas algorithmiquement riche.

Les candidats doivent éviter de présenter une méthode cryptographique qu'ils ne comprennent que très partiellement.

- autres algorithmes : problèmes de graphes, labyrinthe, traitement d'image, etc.

Pour ces algorithmes, il est attendu que le candidat cite les sources qu'il a éventuellement utilisées.

Modélisation

Dans ce paragraphe, nous ne parlons pas des sujets décrits ci-dessus, qui adaptent des modèles biologiques à la résolution de problèmes algorithmiques ou mathématiques, mais de sujets dont le but est de modéliser l'évolution d'un système (dynamique de populations, propagation d'une maladie ou d'un feu de forêt, déplacement de poissons ou d'oiseaux, gestion d'un cheptel, . . .). Même si la qualité et la pertinence de la modélisation ne sont pas les principaux éléments évalués, le jury attend du candidat qu'il ait un minimum de bon sens et qu'il y ait dans son projet matière à un développement intéressant d'un point de vue algorithmique. Ainsi, la surenchère de détails et de paramètres n'enrichit pas un modèle, mais empêche souvent l'analyse des résultats obtenus. Enfin, on attend 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 : le traitement algorithmique peut demander un long travail de prise en compte de règles trop nombreuses, mais sans intérêt du point de vue de la programmation (succession d'instructions conditionnelles) ; dans un tel cas, le candidat peut avoir intérêt à simplifier les règles du jeu, pour simplifier la mise en place initiale, et garder ainsi du temps pour programmer et comparer deux intelligences artificielles, même élémentaires (la première pouvant jouer de façon très basique, voire de façon aléatoire), ou encore faire jouer une intelligence artificielle contre une de ses variantes, obtenue en modifiant certains de ses paramètres.

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 quelques lignes de script permettant de tester de façon significative le projet (avec des paramètres bien

choisis, pour garantir un temps d'exécution raisonnable). Cette consigne, qui n'a pas toujours été respectée, sera légèrement modifiée dans la notice 2019.

Le code doit être raisonnablement commenté : un commentaire (docstring) de 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 » _. Dans les chemins de fichiers il est fortement recommandé d'utiliser / et non \, par exemple, on préférera Donnees/exemple.txt à Donnees\exemple.txt.

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("especes.txt") en mettant le fichier Python et le fichier de données dans le même répertoire.

Lorsque plus d'une dizaine de fichiers sont présents, on pourra les regrouper dans un sous-dossier, et utiliser f = open("Donnees/especes.txt").

Ceci fonctionne sans problème sous Spyder, mais demande, sous Pyzo, qu'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, dans le cas où le candidat aura besoin d'écrire une très longue instruction, il pourra placer manuellement un retour à la ligne en utilisant un \.

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.

Notions hors-programme

Les candidats ont le droit d'utiliser des notions hors-programme (comme les dictionnaires ou les classes) 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. Le jury a remarqué cette année une nette amélioration à ce sujet. 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. Le jury note avec satisfaction que les candidats ont, cette année, bien suivi cette recommandation déjà présente dans les rapports précédents.

De nombreux projets nécessitent de parcourir les voisins d'une case d'un tableau bidimensionnel,

ce qui a été en général bien traité : les candidats savent souvent éviter de faire de nombreux cas particuliers.

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 vu attribuer des notes très basses.

Au sujet de l'algorithme de Dijkstra, que le jury sait être un algorithme difficile à appréhender, le jury se contente, quand un candidat ne l'a pas utilisé alors qu'il aurait été bien utile, de lui demander s'il connaît un algorithme vu en cours qui aurait été adapté à son problème, mais sans insister ensuite sur le fonctionnement de cet algorithme ; par contre, si l'algorithme de Dijkstra a été utilisé, le jury pourra vérifier que le candidat comprend l'algorithme et son code Python.

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 parti de cette possibilité, mais cela ne les dispense pas de savoir écrire une boucle while en gérant correctement les conditions de sortie. L'utilisation de boucles while reste un point délicat pour certains candidats (confusions entre if et while).

Plus de candidats que l'année dernière maîtrisent la différence entre une fonction renvoyant une valeur et une fonction modifiant son argument. Le jury s'en réjouit.

Le jury a constaté des progrès sur la manipulation des booléens. Idéalement, on préfère, par exemple, lire return x > 0 plutôt que :

```
if x > 0:
return True
else:
return False
```