DOSSIER de validation du Bloc 5

Diplôme Universitaire Enseigner la Spécialité Numérique et Sciences Informatiques

Bloc 5 – Algorithmique avancée – « Présenter oralement une démarche algorithmique »

A l'intention de Madame THÖNI Anne, responsable DU ESNSI - UCO

Session : novembre 2019

N° étudiant : 191097

N° INE Oaipnt08u31

Faculté d'éducation - UCO

Université Bretagne Loire - Arradon

Année Universitaire : 2019-2020

UCO ANGERS
UNIVERSITÉ CATHOLIQUE DE L'OUEST

3 place André Leroy | BP 10808- 49008 Angers cedex 01

CHARTE DE NON PLAGIAT

Protection de la propriété intellectuelle

Tout travail universitaire doit être réalisé dans le respect intégral de la propriété intellectuelle d'autrui. Pour tout travail personnel, ou collectif, pour lequel le candidat est autorisé à utiliser des documents (textes, images, musiques, films etc.), celui-ci devra très précisément signaler le crédit (référence complète du texte cité, de l'image ou de la bande-son utilisés, sources internet incluses) à la fois dans le corps du texte et dans la bibliographie. Il est précisé que l'UCO dispose d'un logiciel anti-plagiat dans lms.uco.fr, aussi est-il demandé à tout étudiant de remettre à ses enseignants un double de ses travaux lourds sur support informatique.

Cf. « Prévention des fraudes à l'attention des étudiants »

Je soussigné(e), ……., étudiant(e) en
…… m'engage à respecter cette charte.

Fait à ……..……, le……..

Signature (pour la version imprimée) :

Table des matières

Introduction

Ce dossier a pour objectif de contextualiser le problème d'algorithmique classique de niveau Première, sur lequel s'est basé notre dossier de validation du bloc 2 de la formation DU ESNSI et de présenter une nouvelle alternative de résolution de ce problème au niveau Terminale du lycée général dans la spécialité Numérique et Science Informatique.

En binôme avec Loïc KERINEC, pour valider le bloc 2, nous avions imaginé une situation d'évaluation mixte (écrite et pratique) de fin de première pour mesurer les capacités des élèves à reconnaitre un corpus de constructions élémentaires en documentant un programme et à implémenter en Python les algorithmes classiques pour améliorer une application logicielle de découpage optimisé de tubes profilés d'aluminium.
De façon idéale, les élèves devraient répondre en produisant une solution itérative de type « gloutonne » et pour la correction de cette évaluation nous présenterons une première solution récursive en prévision du programme de terminale.

La nouvelle solution alternative décrite dans la suite implémente également un algorithme glouton de façon récursive pour résoudre la fonction de découpage optimisé. Elle devrait aussi faciliter l'intégration de fonctionnalités complémentaires pour mieux satisfaire aux besoins de l'application. En parallèle de ces développements fonctionnels, nous pourrons mettre en évidence l'intérêt potentiel d'une programmation dynamique dans ce cas d'étude.

C'est cette démarche didactique que j'exposerai durant l'exercice de présentation orale individuelle lors de cette dernière semaine de formation portant sur l'algorithmique avancée.

1- Problème posé

1.1- Contexte :

Il s'agit d'une évaluation de fin de première en Numérique et Sciences Informatiques prévue pour le lycée Saint Sébastien de Landerneau et le lycée Notre Dame du Mur de Morlaix.

1.2- Prérequis :

L'ensemble des algorithmes classiques a été abordé ;
Les notions de modularisation, documentation de programmes et de tests sont connues.

1.3- Enoncé du besoin :

Problématique : "Pouvez vous optimiser la découpe en modifiant l'algorithme proposé ?:
Votre entreprise produit des **"billets"** de profilés d'aluminium en **tronçons standardisés** de 0,6 ; 0,8 ; 1,1 et 2,3m. Les barres à tronçonner ont une longueur initiale de 3m. Le nombre de "billets" à fournir chaque jour dépend de la commande passée le soir précédent et la répartition de ces découpes est présente dans un **fichier de commande** d'extension csv.
Le **but recherché est d'optimiser la découpe** en minimisant le nombre de "chutes" devant être refondues par la suite (dispositif coûteux).

Figure 1 les profilés d'aluminium (ManoMano.fr)

En effet, tout tronçon inférieur à 0,6m sera considéré comme un rebus à refondre par la suite.
Nous vous fournissons un algorithme "solution" qui permet de réduire la consommation journalière de barres en faisant en sorte d'avoir le moins de chutes possibles.
Pourtant cet algorithme n'est pas optimal et nous vous proposons d'y **apporter des modifications** pour tenter de l'optimiser, cela constitue le défi que nous vous fixons lors de cette évaluation.

1.4- Questionnement de la partie 1 de l'évaluation :

On donne une solution algorithmique de l'optimisation sous forme de script python (annexe 1 : solution1.ipynb)

1- **Charger** le fichier solution1.ipynb dans votre NoteBook et **tester** le script en ayant au préalable modifié à partir de la commande présente dans le fichier commande_1.csv de manière à respecter les commandes suivantes :

- 3 pièces de 0,6m
- 2 pièces de 0,8m
- 3 pièces de 1,1m
- 1 pièce de 2,3m

2- **Relever** les constructions élémentaires présentes dans le script proposé

3- **Commenter** le script de la fonction proposée (calc_chute) en **décrivant les préconditions** sur les arguments et les **post conditions** sur les résultats.

4- **Montrer** qu'une des boucles de la fonction "tri_longueurs_demandees" **se termine bien** en identifiant un **variant** de boucle.

5- **Estimer** la complexité de la fonction "tri_longueurs_demandees" comportant deux structures imbriquées, un tableau de complexités de référence étant donné ci-dessous

constant	$\mathcal{O}(1)$	• opérations élémentaires : affectation, comparaison, ...
logarithmique	$\mathcal{O}(\log n)$	• recherche dichotomique
linéaire	$\mathcal{O}(n)$	• recherche séquentielle • recherche du max (ou min) • calculer la longueur d'une liste
quasi-linéaire	$\mathcal{O}(n \log n)$	• tri fusion
quadratique	$\mathcal{O}(n^2)$	• tri à bulle, tri par insertion • parcours d'une matrice $n \times n$ • recherche de motif dans une chaîne de caractères
polynomial	$\mathcal{O}(n^k)$	avec k fixé $(k = 3$: cubique)
exponentiel	$\mathcal{O}(k^n)$	• Fibonacci récursif

Figure 2 : PIVOTEAU (algorithmes et complexité)

1.5- Questionnement de la partie 2 de l'évaluation :

On donne une solution algorithmique de l'optimisation sous forme de script python (annexe 1 : solution1.ipynb)

Proposer une modification de la fonction "calc_chute" (obligatoire)ainsi que de la fonction "tri_longueurs_demandees" (si nécessaire) du script précédent de manière à obtenir un résultat optimisé en termes de barres consommées afin de réduire la matière perdue. La solution devra être choisie parmi les algorithmes classiques vus au cours de l'année.

2- Solution proposée en correction fin de Première

Une des solutions à notre problématique est basée sur un algorithme de type glouton (annexe 2 : correction.ipynb). Elle permet de rendre pertinent la comparaison entre les 3 solutions (solution de départ donnée à l'élève, solution trouvée par l'élève, la solution proposée lors de la correction).

Cette solution est récursive ce qui permet de faire une ouverture sur le programme de terminale…

3- Nouvelle solution alternative de niveau Terminale

3.1- Contexte :

Nous sommes maintenant rendus en terminale. Bon nombre de nos élèves de première ont choisi de poursuivre la spécialité NSI. ;)
Nous venons de leur faire découvrir les fonctions récursives par des applications classiques telles que l'implémentation d'une fonction factorielle ou le tracé d'une figure fractale avec le module turtle (ipyturtle dans jupyter notebook).
C'est l'occasion de revenir sur notre problème d'évaluation de fin de première et de proposer une nouvelle solution (annexe 3 : alternative.ipynb) qui va nous permettre de revoir la spécification, la documentation et la mise au point d'une fonction ainsi que les notions de complexité et d'algorithme glouton tout en analysant le fonctionnement d'un programme récursif.

3.2- Didactique :

Le choix didactique fait ici est de se concentrer sur la seule fonction de découpe :

- Elle renvoie le total des chutes et des barres consommées pour réaliser une commande complète ;
- Et elle prend en entrée l'ensemble des paramètres utiles au réglage initial de la scie d'une part et, d'autre part, ceux de la répartition des pièces de la commande du jour à usiner :
    - Paramètres de sciage :
        - r = restachou ;
        - L = longueur initiale de la barre (ce programme permet de reprendre une barre incomplète) ;
        - l = longueur du tronçon initiale (lui attribuer une valeur supérieure à 0mm permet de redresser une barre dont le bout serait émoussé) ;
        - n = nombre de barres consommées (initialisé à 0) ;
    - Paramètres de la commande :
        - a = nombre de pièce de 0,6m ;
        - b = nombre de pièce de 0,8m ;
        - c = nombre de pièce de 1,1m ;
        - d = nombre de pièce de 2,3m ;

Aussi ce choix de nommage permet de ne pas trop « alourdir » la lecture du code.
De même, c'est volontairement que ce script n'utilise pas pour l'instant de types construits : c'est une évolution que l'on demandera aux élèves d'intégrer au programme dans la suite des activités.

3.2- Activités :

A partir du script de la fonction « decoupe » fourni, les élèves vont :

1. Analyser le programme pour reconnaitre un fonctionnement récursif ;
2. Spécifier et documenter la fonction ;
3. Utiliser des jeux de tests et en définir de nouveaux ;
4. Intégrer d'autres fonctionnalités et utiliser des variables de types construits :
    - Adapter le programme à des outils de coupe d'épaisseurs différentes ;
    - Permettre d'autres dimensions des pièces à découper ;
    - Redéfinir le paramétrage d'entrée.
5. Dessiner un plan du débit pour chaque commande en expérimentant avec différents modules graphiques : ipyturtle, ipythonblocks, la fonction SVG() de iPython.display, Pillow, …, pygraphviz ou tout autre module permettant de tracer des graphes. Cette activité sera menée en miniprojet, avec une restitution orale devant le groupe, chaque équipe utilisera une bibliothèque différente à partir de sa documentation.
6. Observer la répétition de certains motifs, mettre en évidence l'inutilité de certains appels récursifs qui conduisent à refaire des calculs déjà exécutés et découvrir l'intérêt de développer une nouvelle version en programmation dynamique…
7. Produire un bilan d'analyse critique en termes de complexité, coût mémoire, optimisation de la découpe, maintenabilité, évolutivité…

Le sujet pourra donc être repris après avoir fait l'étude de la programmation dynamique sur des cas classiques tels que le rendu de monnaie par exemple.

Conclusion

Pour conclure, puisqu'il faut conclure et terminer cette séquence de cinq semaines de formation en présentiel au DU ESNSI, j'énumère ici un méli-mélo de quelques réflexions personnelles.

Comme je l'ai déjà écrit par ailleurs, je confirme ici que l'obligation de validation à chaque semaine a ceci de bon qu'elle oblige à se mettre en production par avance et donc à s'interroger sur nos futures pratiques.

Ainsi, je n'aurais certainement pas revisité de moi-même le problème algorithmique de l'évaluation de fin de première que nous avions prévu pour le bloc 2, et encore moins imaginé qu'il pourrait offrir autant de pistes d'exploitation pour l'application du programme de terminale.

Ce qui va me manquer principalement, c'est la possibilité de rencontrer physiquement régulièrement les collègues car dans notre spécialité plus que dans d'autres nous serons relativement isolés dans nos établissements respectifs. Les travaux en équipe et l'évaluation par les pairs sont donc l'occasion de confronter nos points de vue, de nouer des relations, et également de s'inspirer des solutions proposées par d'autres.

Ainsi, j'imagine donc que, d'ici mon passage à l'oral, je risque de faire évoluer ce dossier. En tous cas je l'espère, tout au moins pour les domaines de la programmation dynamique et des autres algorithmes avancés que je ne maitrise pas suffisamment encore et pour lesquels j'attends les conseils avisés d'un expert(Hix) lors de cette dernière semaine.

Je me répète encore ici, mais cette rencontre est aussi l'occasion d'être conforté. Ainsi, je suis particulièrement satisfait de constater que mes choix techniques d'utiliser les outils basés sur Git et plus particulièrement Jupyter sont toujours plébiscités. Je reste néanmoins critique car si les Jupyter Notebook offrent de nombreux avantages, ils ont aussi des limites et bien des défauts à l'usage…

Depuis le début de la formation j'ai découvert, grâce à nos échanges, d'autres outils que j'utilise quotidiennement tels que :

- L'application Carnets qui permet de lire, modifier et même créer des jupyter notebook sur un iPad en local (c'est l'équipement personnel de tous les élèves de seconde et de première dans mon lycée) ;
- L'application drawio avec laquelle je crée en vectoriel toutes les figures schématiques que j'intègre dans mes cours que je rédige systématiquement dans des carnets jupyter.

Voilà, cette formation va bientôt se terminer et il me reste encore beaucoup d'interrogations. Merci d'avance pour les réponses que j'obtiendrai dans la semaine qui vient.

Bibliographie

https://www.supinfo.com/cours/2ADS/chapitres/05-programmation-dynamique

Annexes

Les annexes de ce dossier sont au format jupyter notebook.

Elles sont rendues visibles sur le web par ce lien :

https://nbviewer.jupyter.org/github/ericECmorlaix/DU-ESNSI_B5/tree/master/

Aparté

Quant à la difficulté d'enseigner des notions avancées d'algorithmiques et plus généralement d'informatique à des élèves de lycée :

- **Une piste à éviter** : tout comme je ne vais pas leur apprendre le Chinois (ou le Coréen) pour leur expliquer l'informatique, je dois m'interdire de m'appuyer sur un formalisme (notation) de Mathématiques qu'ils ne possèdent pas déjà (= niveau moyen de fin de seconde) ;
- **Une piste potentielle** : cette citation attribuée à Alan KAY