ENSEIRB-MATMECA 1eannée mars 2011

Projet Graphes
Recommandations de rédaction
Rapport algorithmique

Denis Lapoire

Voici quelques recommandations concernant la rédaction du rapport algorithmique du projet graphes. Je vous invite à les lire, à les suivre et à solliciter tout encadrant à leur sujet.

La rédaction d'un rapport n'est pas la simple compilation de résultats algorithmiques obtenus (algorithmes, énoncés et preuves de leurs propriétés) mais un document structuré dans lequel vous mettez en valeur le travail réalisé ainsi que votre maîtrise des notions sousjacentes. Il est fidèle à la vision finale que vous avez du sujet.

Cet exercice est un travail long mais nécessaire. Que cela soit dit dès maintenant : le premier rapport écrit est souvent obscur et nécessite de nombreuses réécritures. Afin de minimiser celles-ci, je vous propose de suivre ces recommandations.

1 Quelques éléments de votre rapport

Ce rapport contiendra notamment :
- des définitions
- des exemples
- des preuves

Exceptée l'écriture des algorithmes et des problèmes, votre expression utilisera des phrases qui contiennent chacune un sujet et un verbe.

1.1 Définitions

1.1.1 Ensemble complet

Toutes les notions mathématiques, problèmes et algorithmes devront être définis. Ainsi, vous devrez notamment rappeler ce qu'est pour vous un graphe, un arc, un chemin, un graphe connexe etc...

Si ces définitions sont exigées, on ne vous demande pas bien-sûr pour autant de redéfinir ce qu'est un entier, un réel ou la fonction $n \mapsto n^2$.

Quelle est la frontière exacte ?

Si l'équipe pédagogique est certaine que vous savez ce qu'est un entier elle est parfois plus dubitative en ce qui concerne par exemple les notations $O(f(n))$ ou $\theta(g(n))$. En conséquence de quoi, vous pouvez par exemple rappeler la définition de ces notations. Si certaines définitions ne sont que des rappels et semblent trop longs, vous pouvez les placer en annexe.

1.1.2 Connaissance de l'existant

Les définitions pourront provenir directement du cours, être inspirées de celui-ci ou à défaut être totalement nouvelles. Nous vous demandons de connaître le cours et de maîtriser toutes les notions qui sont autant d'outils pour rendre votre discours concis et pertinent.

1.1.3 Nommage

Lors de chaque définition (notion ou objet mathématique, problème ou algorithme), vous devez nommer cet objet. Accordez une grande importance à ce choix, il facilite non seulement la compréhension du lecteur mais aussi la votre.

Vous avez une certaine liberté dans le choix de ces noms.

Exemple 1 Ainsi, nous vous demandons de définir précisément ce qu'est un graphe. Si vous utilisez uniquement dans votre rapport des graphes orientés sans boucle, vous pouvez restreindre la notion de graphes à ceux-ci.

1.1.4 Définitions de problèmes

Définir un problème consiste à choisir un nom et à définir selon le format (`Entrée/Sortie`) les spécifications attendues des objets typés mis en relation. Naturellement les définitions mathématiques requises seront définies préalablement.

Exemple 2 Un exemple de définition de problème est :

```
problème ScarabéePair
Entrée : un graphe G connexe
Sortie : un entier pair scarabée de G
```

Préalablement à cette définition, il vous faut définir ce qu'est un graphe (d'un point de vue mathématique), un graphe connexe et ce qu'est un scarabée.

Rappelons qu'en sortie chacun des objets passés en entrée doit apparaître dans les spécifications en sortie ; sinon cela signifierait que l'objet en entrée ne joue aucun rôle et devrait donc être supprimé de l'entrée.

1.1.5 Définitions d'algorithmes

Définir un algorithme consiste à choisir un nom, écrire un prototype et le corps de l'algorithme.

Exemple 3 Un exemple d'algorithme est :

```
fonction scarabée01(G : graphe) : entier

    si estPair(nbSommets(G)) alors
        retourner voilier(poule(G)) ;
    sinon
        retourner 3+max(17,voilier(canard(G))) ;
```

Pour que la définition soit complète, il est nécessaire de définir immédiatement après une telle définition non pas les fonctions auxiliaires utilisées (dans l'exemple `nbSommets`, `voilier`, `poule`, `canard`) mais les problèmes qu'elles résolvent (`Voilier`, `Poule`, `Canard`).

De même qu'on ne vous demande pas de définir ce qu'est un entier, il est inutile de définir le problème `Max` qui retourne le maximum de deux entiers et à fortiori la fonction `max`, ainsi que d'ailleurs l'opération somme '+' !

Quels problèmes (resp. algorithmes) sont à définir et à ne pas définir ? C'est une question de bon sens. Nous y reviendrons dans une prochaine section.

1.2 Exemples et dessins

Un exemple facilite souvent la compréhension d'une définition et sa justification. Aussi, vous devrez illustrer chacune de vos définitions d'un exemple qui est commenté et qui peut être composé d'un dessin.

Lorsque vous définissez la notion de graphe, fournissez un exemple de graphe et dessinez-le !

Lorsque vous définissez un problème, fournissez un ou plusieurs exemples d'entrée-sortie et dessinez les !

Lorsque vous définissez un algorithme, fournissez un ou ou plusieurs exemple d'entrée- sortie et dessinez les ! Il est courant que l'exemple significatif illustrant le problème soit identique à celui du problème associé. Auquel une simple référence suffit.

1.3 Qualité de vos algorithmes

Les propriétés attendues des algorithmes sont :

1. leur simplicité.
2. leur correction

3. leur complexité en temps et en espace

Ces principes sont parfois et souvent contradictoires. La qualité de votre travail réside dans votre capacité à arbitrer des choix et à les justifier.

1.3.1 Simplicité

Si vos problèmes sont simplement et correctement spécifiés, les algorithmes les résolvant pourront être écrits simplement. Sinon, cela sera impossible. En clair, n'essayez pas de résoudre un problème tant que celui-ci est mal spécifié.

1.3.2 Correction

Pour chaque algorithme vous consacrerez une sous-section (commandes latex `\subsection{Correction}`.

Pour rédiger une preuve, la première difficulté est de choisir le niveau de formalisme. Écrire l'ensemble des preuves selon un formalisme irréprochable (par exemple en Logique de Hoare voir cours de F. Herbreteau) nécessiterait un travail très supérieur à celui attendu de vous.

Vos preuves pourront avoir l'aspect suivant :

- absence de preuve (ce n'est pas une preuve !).
  "Nous ne sommes pas arrivés à démontrer la correction par faute de temps mais supposons sa correction".
- Très bref (à utiliser avec précaution)
  "Evident".
- Bref (souvent justifié).
  "Conséquence immédiate des définitions des problèmes X et Y".
- Assez complet (à recommander pour toutes les algorithmes contenant des boucles ou des appels récursifs).
  "Il suffit de considérer les invariants suivants .... qui assurent en sortie de boucle telle propriété. Observant telle autre propriété, on obtient la conclusion."
- Très formel (à faire au moins une fois et au plus deux !).
  On utilise les techniques vues en Analyse d'algorithmes en utilisant son formalisme.

Notons qu'une condition nécessaire à l'existence d'une preuve simple est la simplicité de l'algorithme lui même. Si l'algorithme tient en quelques lignes, la preuve elle aussi. Ainsi, devoir prouver la correction d'un algorithme impose un style algorithmique simple et concis !

1.3.3 Complexité en temps et en espace

Pour chaque algorithme vous consacrerez une sous-section (commandes latex `\subsection{Complexité en temps}`, `\subsection{Complexité en espace}`.

1.3.4 Avertissement

Si la correction d'un problème ne dépend pas des fonctions auxiliaires mais des problèmes qu'elles résolvent et peut ainsi être établie localement sans même connaître ces fonctions auxiliaires, il peut en être autrement de la complexité.

Dans votre rapport, un même problème pourra être résolu par différents algorithmes ayant autant de complexité différentes, elles mêmes dépendant du type abstrait graphe choisi et de l'implémentation associée (matrice, tableau de liste de successeurs et leurs variantes).

Le type abstrait et son implémentation sont des questions de plus bas niveau que la définition mathématique d'un graphe. Elles ne peuvent donc pas être traitées en ouverture de rapport et le seront en cours, en fin de rapport voir en partie annexe (si l'implémentation est classique et ne présente peu ou pas d'originalité).

En conséquence, il faut choisir un nombre très restreint de types abstraits et d'implémentations de ces types. Ce qui permet lors de l'étude de la complexité de conditionner le résultat en fonction de la version de la fonction auxiliaire et du type abstrait choisis.

Exemple 4 Ainsi, on indiquera que la complexité (en temps dans le pire des cas) de `scarabée01` est égale à :

- à $\theta(n^3 \log(n))$ si on utilise `poule01` opérant sur des graphes de type `grapheMatrice`.
- à $O(m\sqrt{m})$ si on utilise `poule02` opérant sur des graphes de type `grapheListeSuccesseurs`.

où n représente le nombre de sommets du graphe et m le nombre d'arcs.

1.3.5 Avertissement

En ce qui concerne les notations de complexité $O(f(n))$ et $\theta(f(n))$, ne les confondez pas. Nous souhaitons que vous énonciez et prouviez les complexités en utilisant la notation $\Theta(f(n))$; à défaut établissez celles en $O(f(n))$ et conjecturez celles en $\theta(f(n))$.

2 La structure de votre rapport

Votre rapport comportera :

1. une introduction.
   Elle pourra rappeler le sujet, présenter la structure du rapport et annoncer quelques points forts de votre travail.
2. un sommaire.

3. un chapitre dédié aux rappels des notations et définitions générales (graphe, arc, chemin ...) souvent utilisées. Des notions utilisées uniquement dans un chapitre ou dans une section sont définis dans ce chapitre ou dans cette section et ne doivent pas être introduits ici.
4. un chapitre dédié à la modélisation du ou des problèmes généraux (si ce chapitre est court, il peut simplement constituer la conclusion du chapitre précédent).

   A cet instant du rapport, aucune notion spécifiquement informatique n'est apparue. Les seuls objets et notions sont mathématiques et logiques. En d'autres termes, ces deux chapitres peuvent être rédigés lors des deux premières semaines de votre projet, et ce définitivement !
5. pour chaque problème significatif, un chapitre dédié à sa résolution.
6. un chapitre dédié à chaque type abstrait graphe utilisé.
7. d'éventuelles annexes.
8. une conclusion.

2.1 L'introduction

Elle peut être consacrée au rappel du sujet, à la présentation des points forts de votre rapport et à sa structure.

2.2 Une progression descendante

Le premier algorithme du rapport doit être le premier par ordre d'importance. Il peut être écrit en quelques lignes en utilisation plusieurs fonctions auxiliaires résolvant des problèmes définis aussitôt mais résolus dans les chapitres suivants. Hiérarchisez par ordre d'importance décroissante ces problèmes. Étudiez au prochain chapitre le plus important et ainsi de suite. L'étude de certains sous-problèmes d'intérêt faible peut être omise ou placée en annexe.

Cette progression descendante a notamment deux avantages :

1. le premier chapitre "algo" est une réponse directe et immédiate à l'introduction. Les connaissances requises pour le lire sont de haut niveau. À cet instant, un graphe est un objet mathématique logique. Il n'est par exemple pas besoin de comprendre comment il est implémenté pour comprendre l'algorithme. Le lecteur ayant lu un minimum de définitions a déjà une vision globale et assez complète de votre travail.
2. vous pouvez commencer à écrire le rapport sans même savoir l'importance que vous accorderez aux problèmes annexes. En d'autre terme, la modification d'une partie de votre algorithme n'entraîne pas la réécriture de votre rapport mais seulement la queue de celui-ci.

2.3 La conclusion

Dans la conclusion, vous devez montrer votre maîtrise du sujet, votre ascendant sur le travail réalisé et la méthode employée.

Insistez naturellement sur les nombreuses qualités de votre réalisation. Vous pouvez aussi évoquer les problèmes rencontrés, les fragilités de votre travail et leurs raisons. Si on vous demandait de refaire ce travail, comment feriez-vous dorénavant ? Quelles erreurs ne commettriez-vous pas ?

Certains d'entre vous se sont précipités sur le net à la recherche de solutions algorithmiques : qu'est ce que cela vous a apporté ?

Concluez en indiquant de nouvelles pistes de réflexion, en posant de nouvelles questions.

3 Evaluation

Tout conseil a un coût. Conséquence de ces recommandations, la moitié de la note d'évaluation de votre rapport sera la somme des évaluations suivants :

1. est-ce que chaque problème :
   (a) est typé : c'est à dire chaque objet présent en entrée ou en sortie est muni d'un type.
   (b) est correctement spécifié (2pts) : c'est à dire est ce que chaque notion présente dans la définition du problème est définie préalablement.
   (c) est correctement illustré (2pts) : c'est à dire associé à un exemple ayant même type.
2. est-ce que chaque algorithme :
   (a) est associé à un problème ayant même signature (2 pts)
   (b) est correctement illustré : voir au-dessus.
   (c) est suivi d'une section correction
   (d) est suivi d'une section complexité en temps
   (e) est suivi d'une section complexité en espace

4 Conclusion

Voici quelques recommandations pour rédiger ce rapport. Un autre conseil est la concision. Votre rapport doit être court : ayez pour objectif une vingtaine de pages.

Ces principes seront parfois dans leur application contradictoires et nécessiteront d'être tempérés. N'hésitez pas à solliciter l'équipe pédagogique. In fine, ce rapport est le votre. Nous sommes confiants en votre culture scientifique pour réaliser ces

nécessaires arbitrages avec pertinence.

Bonne chance !

Denis Lapoire