Quick Algorithmique et Modélisation

Quelques éléments de correction

Jean-Marc.Vincent@imag.fr

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

20 mars 2017

Attention : ceci est un corrigé, il y a bien entendu d'autres solutions, d'autres approches tout aussi valides. La rédaction a été détaillée pour aider à la compréhension, un tel niveau de détail n'était pas attendu sur la copie.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Corrigé Q1 2017

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 1 : Vocabulaire

On souhaite analyser la richesse de vocabulaire d'un auteur littéraire. On dispose pour cela d'un texte et on souhaite compter le nombre de mots différents utilisés par l'auteur.
On suppose que le texte est segmenté en mots et que l'on accède au $k^{\text{ième}}$ mot par `texte[k]`. On notera n la taille du texte en nombre de mots.

Remarques préliminaires

Faire un dessin

- on travaille avec des éléments qui sont des mots
- il faut un opérateur de test d'égalité entre les mots
- un algorithme naïf effectuerait des comparaisons entre tous les mots, ce qui entrainerait un coût en $\mathcal{O}(n^2)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 1 : Vocabulaire

On souhaite analyser la richesse de vocabulaire d'un auteur littéraire. On dispose pour cela d'un texte et on souhaite compter le nombre de mots différents utilisés par l'auteur.
On suppose que le texte est segmenté en mots et que l'on accède au $k^{\text{ième}}$ mot par `texte[k]`. On notera n la taille du texte en nombre de mots.

Remarques préliminaires

Faire un dessin

- on travaille avec des éléments qui sont des mots
- il faut un opérateur de test d'égalité entre les mots
- un algorithme naïf effectuerait des comparaisons entre tous les mots, ce qui entrainerait un coût en $\mathcal{O}(n^2)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 1 : Vocabulaire

On souhaite analyser la richesse de vocabulaire d'un auteur littéraire. On dispose pour cela d'un texte et on souhaite compter le nombre de mots différents utilisés par l'auteur.
On suppose que le texte est segmenté en mots et que l'on accède au $k^{\text{ième}}$ mot par `texte[k]`. On notera n la taille du texte en nombre de mots.

Remarques préliminaires

Faire un dessin

- on travaille avec des éléments qui sont des mots
- il faut un opérateur de test d'égalité entre les mots
- un algorithme naïf effectuerait des comparaisons entre tous les mots, ce qui entrainerait un coût en $\mathcal{O}(n^2)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 1 : Vocabulaire

On souhaite analyser la richesse de vocabulaire d'un auteur littéraire. On dispose pour cela d'un texte et on souhaite compter le nombre de mots différents utilisés par l'auteur.
On suppose que le texte est segmenté en mots et que l'on accède au $k^{\text{ième}}$ mot par `texte[k]`. On notera n la taille du texte en nombre de mots.

Remarques préliminaires

Faire un dessin

- on travaille avec des éléments qui sont des mots
- il faut un opérateur de test d'égalité entre les mots
- un algorithme naïf effectuerait des comparaisons entre tous les mots, ce qui entrainerait un coût en $\mathcal{O}(n^2)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.1 : structure de donnée

Proposer (et justifier) une structure de donnée appropriée pour ce type de problème.

L'espace des possibilités pour les mots de la langue française étant très grand (en se limitant à 25 lettres par mot, on a une borne supérieure de 26^{25})[1] donc le stockage des mots dans un tableau d'adressage direct est exclu. On peut donc utiliser une table de hashage T avec chaînage qui permet, avec une taille mémoire m "raisonnable" (par exemple $m = n$) de tester à moindre coût le fait d'avoir déjà rencontré un mot, si la fonction de hashage h est uniforme (faible probabilité de collision).[2]

attention

[1] C'est une borne très très large...

[2] Si le texte est "À la recherche du temps perdu" de M. Proust, le choix $m = n$ reste-t-il raisonnable ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.1 : structure de donnée

Proposer (et justifier) une structure de donnée appropriée pour ce type de problème.

L'espace des possibilités pour les mots de la langue française étant très grand (en se limitant à 25 lettres par mot, on a une borne supérieure de 26^{25})[1] donc le stockage des mots dans un tableau d'adressage direct est exclu. On peut donc utiliser une table de hashage T avec chaînage qui permet, avec une taille mémoire m "raisonnable" (par exemple $m = n$) de tester à moindre coût le fait d'avoir déjà rencontré un mot, si la fonction de hashage h est uniforme (faible probabilité de collision).[2]

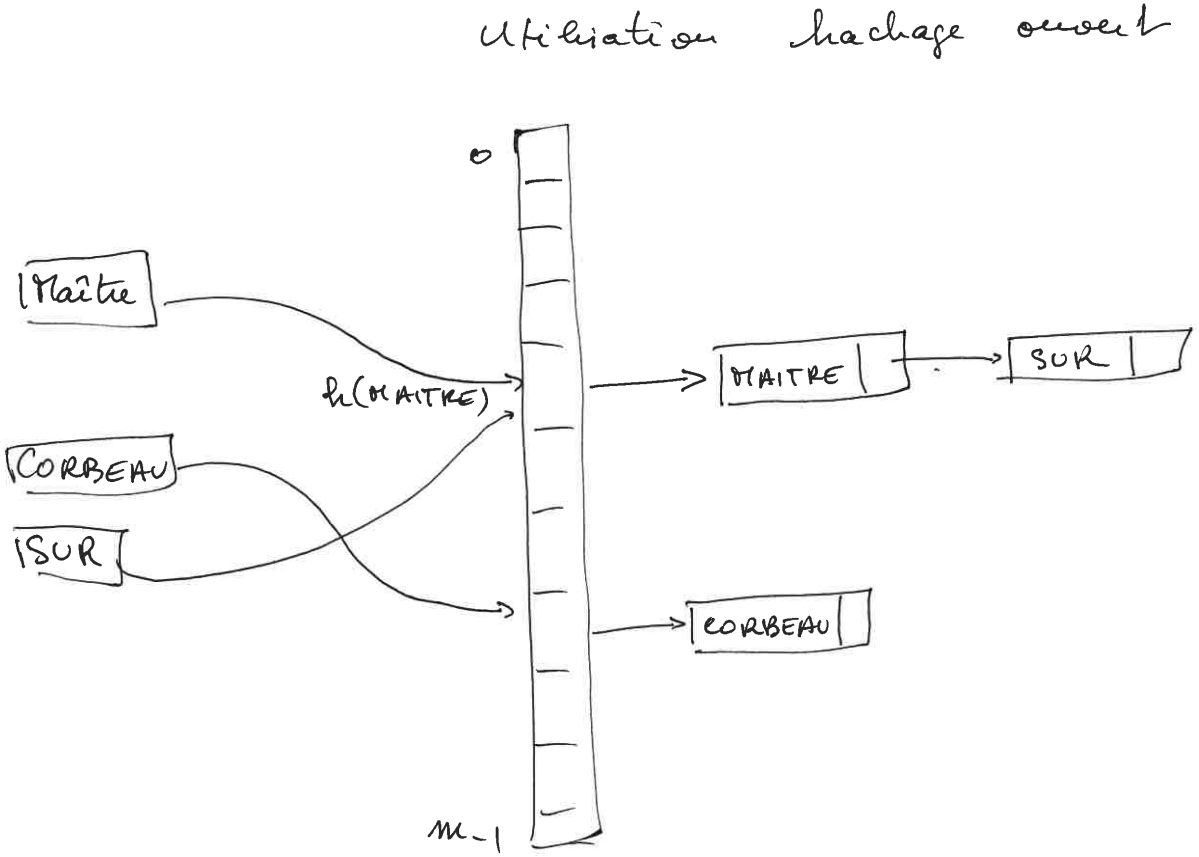
attention

[1] C'est une borne très très large...

[2] Si le texte est "À la recherche du temps perdu" de M. Proust, le choix $m = n$ reste-t-il raisonnable ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Un dessin


Utilisation hachage ouvert
0
Maître
CORBEAU
SUR
h(MAITRE)
MAITRE
SUR
CORBEAU
m-1

Question 1.2 : Algorithme en $\mathcal{O}(n)$

Proposer un algorithme de coût moyen $\mathcal{O}(n)$.

On utilise les fonctions d'accès suivantes à la table de hashage :

- `Créer` (T,m,h) *// crée une table de hashage vide T de taille m et utilisant la fonction de hashage h*
- `Insérer` (T,w) *// insère le mot w en tête de la liste T[h(w)]*
- `Rechercher` (T,w) *// renvoie Vrai si le mot w est présent dans la liste T[h(w)], Faux sinon*

La complexité moyenne de Insérer est en $\mathcal{O}(1)$ (insertion en tête de liste). La complexité moyenne de Rechercher est en $\mathcal{O}(1+\alpha)$ où $\alpha = n/m$ est le taux de remplissage. Ici si l'on prend $m = \mathcal{O}(n)$ alors la recherche est aussi en $\mathcal{O}(1)$ (compromis temps/mémoire).

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

```
RichesseVocabulaire(m,h)
  Data: texte[] texte de taille n dans lequel on souhaite compter les mots différents (tableau de chaînes de caractères)
  Output: Le nombre de mots différents dans le texte

  compteur_mots← 0  // entier servant à compter les mots différents
  Créer(T,m,h)
  for i = 1 to n
  |  mot← texte[i]
  |  if Not(Rechercher(T,mot))  // mot absent du chaînage en T[h(mot)]
  |  |  compteur_mots ← compteur_mots+1
  |  |  Insérer(T,mot)
  return compteur_mots
```

Coût : à chaque itération de la boucle `for`, on appelle les fonctions `Rechercher` et `Insérer` qui sont (dans le cas $m = \mathcal{O}(n)$) de complexité moyenne $\mathcal{O}(1)$. En conséquence la complexité de l'algorithme est en $\mathcal{O}(n)$.

Remarque : Dans l'implémentation Java, que certains d'entre vous ont utilisée, la taille de la table de hachage est optionnelle : sa valeur par défaut est 11 et est adaptative (agrandissement de la table quand le taux de remplissage dépasse 75% par défaut)[3]. La fonction de hashage (méthode `hashCode`) est également optionnelle car il en existe une par défaut dans la classe `Object`. La fonction `Rechercher` est équivalente à `containsKey` et `Insérer` à `put`.

[3] http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Hashtable.html

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Corrigé Q1 2017

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 2 : Jeu de Dames

Dans le jeu de dames, une case noire d'un damier $n \times n$ porte soit un pion ou une dame blanc, soit un pion ou une dame noire, soit ne porte pas de pion. L'objectif est d'écrire un algorithme qui énumère toutes les configurations possibles de jeu (sans prise en compte d'autres contraintes que le nombre maximum de pions/dame blancs ou noir (au plus 20 de chaque dans un jeu de dames classique).
On représente le damier par un ensemble de cases, numérotées de 1 à N et on code l'état de la case par l'une des 3 valeurs : 1 =blanc, -1=noir, 0 s'il n'y a pas de pion ou de dame (pour l'instant seule compte la couleur). On suppose qu'il y a au maximum K pions/dames blancs et K pions/dames noirs.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

2.1 Ordre de grandeur du nombre de configurations

Donner, en fonction de K et de N un majorant et un minorant du nombre de configurations possibles.

Une configuration est un vecteur de N nombres compris entre -1 et 1 (Dans le cas du jeu de dames, on ne joue que sur les cases noires, $N = n^2/2$ est le nombre de cases noires).

- *Un premier majorant :* Le nombre maximal de configurations pour ces vecteurs est donc 3^N. *// 3 façons de choisir la valeur de chaque coordonnée*
- *Un meilleur majorant :* En réalité le nombre de pions n'est pas constant mais au cours d'une partie varie entre K et 0. En suivant le même raisonnement pour chaque valeur des nombres de pions noirs et blancs (i, j), le nombre de configurations possibles est donc au maximum

$$\sum_{i=0}^{K} \sum_{j=0}^{K} \binom{N}{i} \binom{N-i}{j} \tag{1}$$

Ces quantités sont des majorants puisque certaines de ces configurations sont interdites en raison des règles du jeu. Si les règles n'empêchaient aucune configuration, la quantité (1) serait alors le nombre exact de possibilités.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

2.1 Ordre de grandeur du nombre de configurations (2)

Pour les minorants, on continue d'"ignorer" ces règles du jeu (qui ne sont pas rappelées dans le sujet). Le but est de minorer la quantité (1). On peut le minorer plus ou moins précisément par :

- $\binom{N}{2K}$ *// choix de 2K cases occupées, sans se soucier de la couleur des pions*
- $\binom{N}{\frac{N}{3}}\binom{N-\frac{N}{3}}{\frac{N}{3}}$ *// le terme maximal dans la somme (1), lorsque* $K \geqslant \frac{N}{3}$.

L'objectif de cette question est de se rendre compte de la taille de l'espace de configurations à explorer lorsque l'on souhaite étudier une partie de dames. Même si l'on a une taille inférieure à 3^N, les minorants restent très grands $\binom{N}{K} \geqslant \left(\frac{N}{K}\right)^K$.
Par suite, si l'on souhaite explorer l'espace des configurations, il faudra s'appuyer surun algorithme d'énumération bien adapté.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 2.2 : Énumération

Proposer un algorithme qui énumère toutes les configurations possibles du jeu.

Remarques préliminaires

Énumération des éléments d'un ensemble ⇒ approche récursive, décomposition de l'ensemble en sous-ensembles disjoints

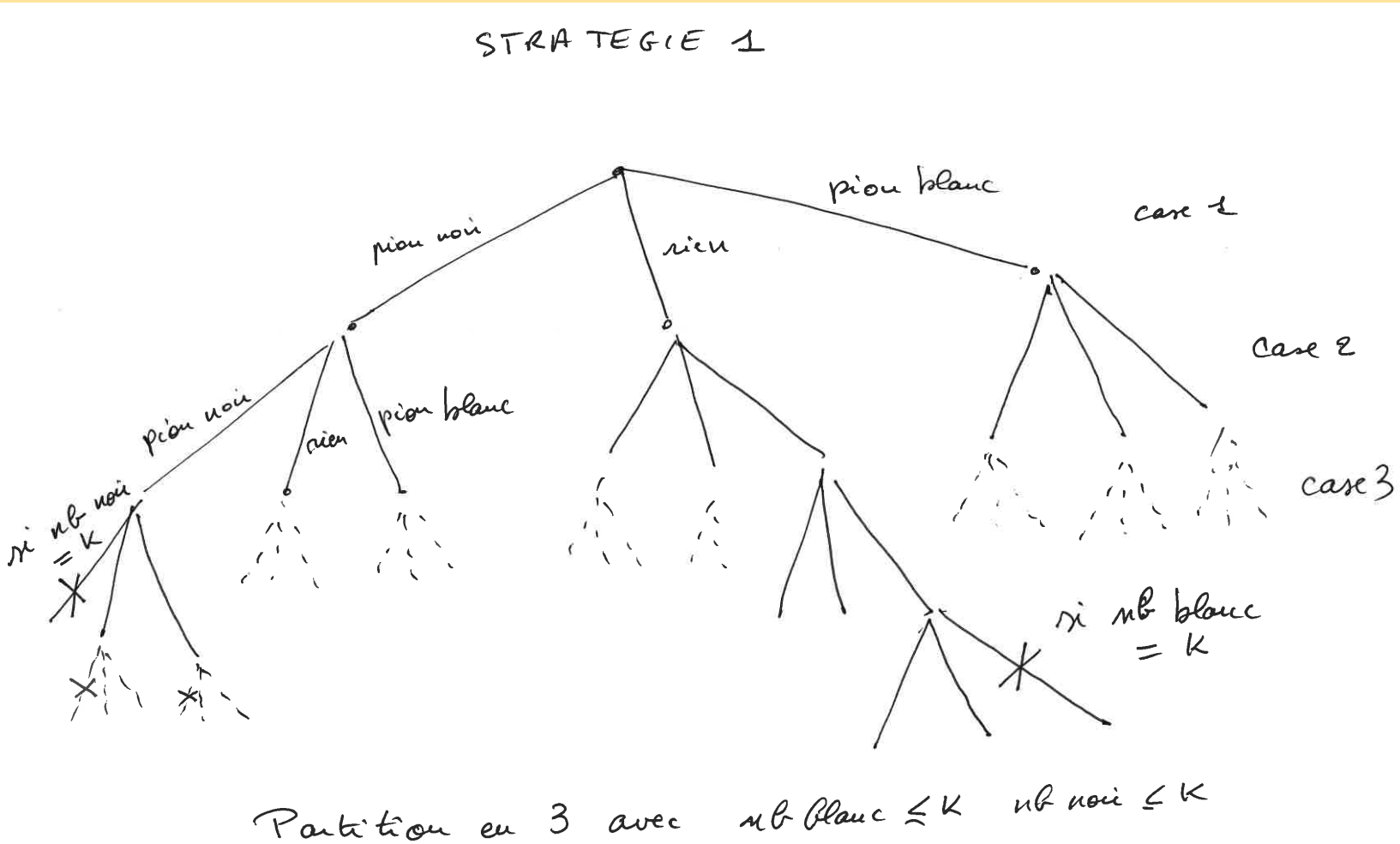
⇒ structure de l'algorithme d'énumération des parties d'un ensemble.

- **Stratégie 1** : adapter l'algorithme aux données
- **Stratégie 2** : adapter les données à l'algorithme

Il faut donc trouver une décomposition correspondante de l'espace des configurations.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Un dessin


STRATEGIE 1
pion noir
rien
pion blanc
case 1
case 2
case 3
pion noir
rien
pion blanc
si nb noir = K
si nb blanc = K
Partition en 3 avec nb Blanc ≤ K nb noir ≤ K

Stratégie 1 : récursivement comme pour l'énumération des parties d'un ensemble.
Une configuration : tableau T de taille N prenant des valeurs dans $\{-1, 0, 1\}$.
`Set` (T,ind,val) : modifie la case `ind` du tableau T à la valeur `val`
`Traiter` (T) : traitement du tableau.

`EnumConfig` (P, ind, pb, pn, N, K)
Data: ind, pb et pn (cases déjà remplies, pn pions noirs posés, pb pions blancs posés, passés par *valeur*
Data: T tableau-damier en construction, passé par *référence*
Output: traite l'ensemble des configurations dont les ind premières cases sont données par les ind premières valeurs de T sur lesquelles on a déjà placé pb pions blancs et pn pions noirs
Invariant Le préfixe T[1,...,ind] contient pn cases à -1, pb cases à 1 et $ind - pn - pb$ cases à 0 avec $pn \leqslant K$ et $pb \leqslant K$

```
if ind == N  Traiter (T)                    // on a affecté les N cases

else
    EnumConfig (Set (T,ind+1,0), ind+1, pb, pn, N, K)
    // on laisse vide la case ind+1
    if (pn< K)  EnumConfig (Set (T,ind+1,-1), ind+1, pb,pn+1, N, K)
    // on positionne un pion noir sur la case ind+1
    if (pb< K)  EnumConfig (Set (T,ind+1,1), ind+1, pb+1,pn, N, K)
    // on positionne un pion blanc sur la case ind+1
```

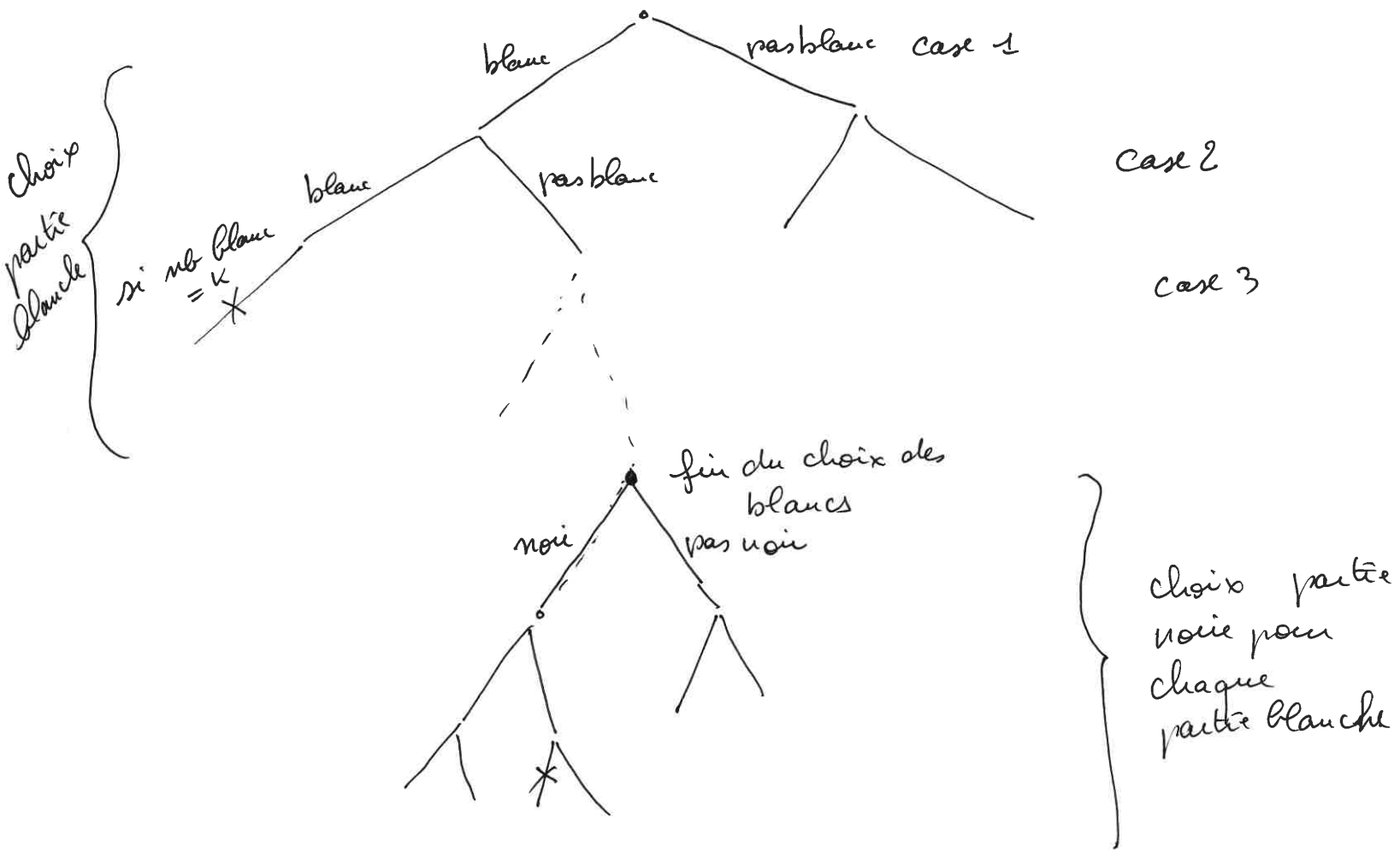
UFR IM²AG
UNIVERSITÉ Grenoble Alpes

L'appel principal se fait à partir d'un tableau T à 0

```
ExploreConfig (N,K)
  Data: Nb de pions K, Taille de damier N
  Output: traitement des configurations

  Plateau : tableau de N entiers contenant des 0 ;   // création de variable
  EnumConfig (Plateau,0,0,0,N,K);      // 0 cases déjà vues, 0 pion déjà
  placé, N cases à considérer, K pions possibles de chaque couleur
```

Un dessin


STRATEGIE 2
blanc
pas blanc
case 1
case 2
case 3
choix partie blanche
si nb blanc = k
blanc
pas blanc
fin du choix des blancs
noir
pas noir
choix partie noire pour chaque partie blanche

Pour la stratégie 2, on utilise tel quel l'algorithme d'énumération des parties de taille $\leqslant K$.

```
procédure énumérer_visiter_parties (X, Y)
    Data: X et Y ensembles disjoints d'éléments
    Output: Une et une seule visite de chaque partie de X ∪ Y de la forme p ∪ Y
            avec p ⊂ X
    if X = ∅
        Visiter (Y)
    else
        x = Choisir (X)
        énumérer_visiter_parties (X\{x}, Y)
        énumérer_visiter_parties (X\{x}, Y ∪ {x})
```

- Comment se fait le passage de paramètres ?
- Quel est l'appel initial ?
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Pour la stratégie 2, on utilise tel quel l'algorithme d'énumération des parties de taille $\leqslant K$.

```
procédure énumérer_visiter_parties (X, Y)
    Data: X et Y ensembles disjoints d'éléments
    Output: Une et une seule visite de chaque partie de X ∪ Y de la forme p ∪ Y
            avec p ⊂ X et de cardinal ⩽ K
    if X = ∅
        Visiter (Y)
    else
        x = Choisir (X)
        énumérer_visiter_parties (X\{x}, Y)
        if |Y| < K
            énumérer_visiter_parties (X\{x}, Y ∪ {x})
```

- Comment se fait le passage de paramètres ?
- Quel est l'appel initial ?
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Pour la stratégie 2, on utilise tel quel l'algorithme d'énumération des parties de taille $\leqslant K$.

```
procédure énumérer_visiter_parties (X, Y)
    Data: X et Y ensembles disjoints d'éléments
    Output: Une et une seule visite de chaque partie de X ∪ Y de la forme p ∪ Y
            avec p ⊂ X et de cardinal ⩽ K
    if X = ∅
        énumérer_visiter_parties (X0 \ Y, ∅)
    else
        x = Choisir (X)
        énumérer_visiter_parties (X \ {x}, Y)
        if |Y| < K
            énumérer_visiter_parties (X \ {x}, Y ∪ {x})
```

- Comment se fait le passage de paramètres ?
- Quel est l'appel initial ?
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ?

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Pour la stratégie 2, on utilise tel quel l'algorithme d'énumération des parties de taille $\leqslant K$.

```
procédure énumérer_visiter_parties (X,Y)
    Data: X et Y ensembles disjoints d'éléments
    Output: Une et une seule visite de chaque partie de X ∪ Y de la forme p ∪ Y
            avec p ⊂ X  et de cardinal ⩽ K
    if X = ∅
        énumérer_visiter_parties (X_0 \ Y, ∅)
    else
        x = Choisir (X)
        énumérer_visiter_parties (X\{x}, Y)
        if |Y| < K
            énumérer_visiter_parties (X\{x}, Y ∪ {x})
```

- Comment se fait le passage de paramètres ? passage par valeurs
- Quel est l'appel initial ? `énumérer_visiter_parties` $(X,\emptyset)$
- Que fait l'algorithme ?
- Quel est le coût de cet algorithme ? à calculer, c'est la formule vue avant

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

QuestionPion ou dame

Modifier votre algorithme pour prendre en compte le type d'occupation d'une case (pion ou dame).

Le problème est identique au problème précédent, sauf que l'on fait 5 conditions de branchement pour la stratégie 1 et on remplace dans la stratégie 2 l'énumération de chacune des parties produites la blanche et la noire, la sous-partie trouvée correspondant aux dames.

Corrigé Q1 2017

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 3 : Tri par dénombrement

extrait de l'ouvrage de Cormen et al, Algorithmique (Dunod 2012)

Le ***tri par dénombrement*** suppose que chacun des n éléments de l'entrée est un entier de l'intervalle 0 à k, k étant un certain nombre entier. Lorsque $k = O(n)$, le tri s'exécute en un temps $\Theta(n)$.

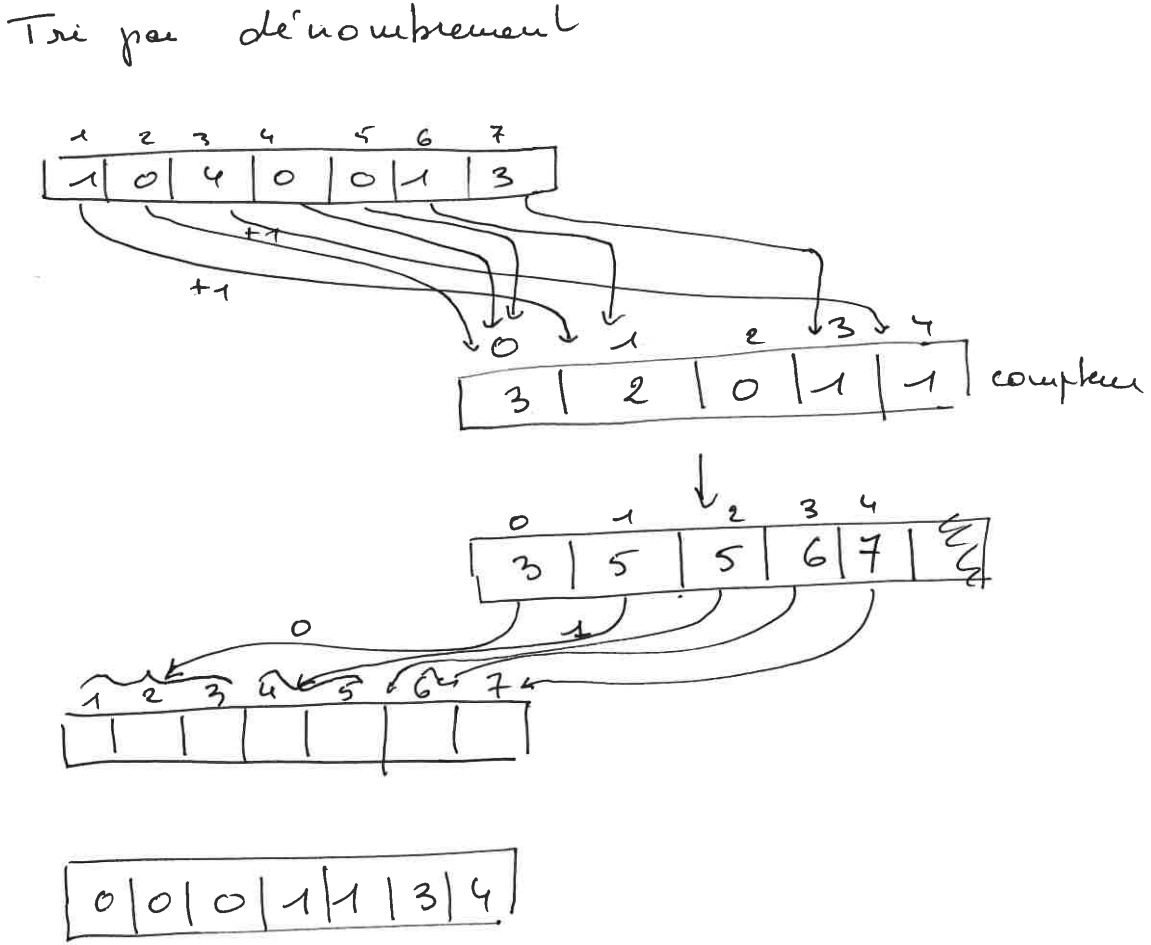
Le principe du tri par dénombrement est de déterminer, pour chaque élément x de l'entrée, le nombre d'éléments inférieurs à x. Cette information peut servir à placer l'élément x directement à sa position dans le tableau de sortie. Par exemple, s'il existe 17 éléments inférieurs à x, alors x se trouvera en sortie à la position 18. Ce schéma doit être légèrement modifié pour gérer la situation dans laquelle plusieurs éléments ont la même valeur, puisqu'on ne veut pas tous les placer à la même position.

Dans le code du tri par dénombrement, on suppose que l'entrée est un tableau $A[1\,..\,n]$ et donc que $longueur[A] = n$. Nous avons besoin de deux autres tableaux : le tableau $B[1\,..\,n]$ contient la sortie triée et le tableau $C[0\,..\,k]$ sert d'espace de stockage temporaire.

```
TRI-DÉNOMBREMENT(A, B, k)
 1  pour i ← 0 à k
 2      faire C[i] ← 0
 3  pour j ← 1 à longueur[A]
 4      faire C[A[j]] ← C[A[j]] + 1
 5  ▷ C[i] contient maintenant le nombre d'éléments égaux à i.
 6  pour i ← 1 à k
 7      faire C[i] ← C[i] + C[i − 1]
 8  ▷ C[i] contient maintenant le nombre d'éléments inférieurs ou égaux à i.
 9  pour j ← longueur[A] jusqu'à 1
10      faire B[C[A[j]]] ← A[j]
11           C[A[j]] ← C[A[j]] − 1
```

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Un dessin


Tri par dénombrement
1 2 3 4 5 6 7
1 0 4 0 0 1 3
+1
+1
0 1 2 3 4
3 2 0 1 1
compteur
0 1 2 3 4
3 5 5 6 7
0
1
1 2 3 4 5 6 7
0 0 0 1 1 3 4

Question 3.1 : se faire la main

Exécuter cet algorithme sur le tableau A suivant :

A :

2	5	3	0	2	3	0	3

On précisera en particulier la valeur du tableau C à la ligne 6 et à la ligne 9. On illustrera également le remplissage du tableau B durant l'exécution de la boucle de la ligne 10.

Attention aux indices. C contient le nombre d'occurrences de chaque valeur permise dans A. C n'a donc besoin que de 6 cases (le tableau A ne contient que des nombres entre 0 et 5). À la ligne 6 (au début de la ligne 6, i.e. après les 2 premières boucles) on a donc

C :

0	1	2	3	4	5
2	0	2	3	0	1

Ensuite, C est modifié par la 3^e boucle (lignes 6-7) pour cumuler ces valeurs:

C :

0	1	2	3	4	5
2	2	4	7	7	8

Puis B et C seront modifiés dans la dernière boucle (lignes 9-11) au fur et à mesure que j décroît :

B :

1	2	3	4	5	6	7	8
0	0	2	2	3	3	3	5
j=4	j=7	j=1	j=5	j=3	j=6	j=8	j=2

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 3.2 : Compter

Calculer le coût de cet algorithme.

Dans chaque boucle on a $\mathcal{O}(1)$ affectations et/ou additions. Les accès mémoires se font en $\mathcal{O}(1)$ (accès direct aux cases du tableau). Donc le coût de l'algorithme est linéaire en fonction de la longueur des boucles:

$$\mathcal{O}(k) + \mathcal{O}(n) + \mathcal{O}(k) + \mathcal{O}(n) = \mathcal{O}\left(\max(n, k)\right)$$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 3.3 : Analyse

Analyser ce coût et comparer avec les différents algorithmes de tri déjà rencontrés dans votre formation. Qu'en pensez-vous ?

On a montré que les algorithmes Quicksort et Merge sort avaient un coût de $\mathcal{O}(n\log(n))$ opérations de comparaisons d'éléments.
Ici on utilise un information supplémentaire sur les *valeurs* des entrées, elles sont comprises entre 0 et k. Si k est petit devant n, e.g. $k = \mathcal{O}(n)$, le nombre d'opérations est linéaire en n ! On ne fait que 2 passages sur le tableau de taille n.
C'est un tri linéaire donc beaucoup plus efficace que les autres algorithmes rencontrés.
Si $k >> n$, c'est moins clair (si $k = \mathcal{O}(n^2)$ par exemple, ce n'est pas l'algorithme le plus performant). Il faut donc disposer de cette information *a priori* afin de décider de la pertinence de cet algorithme.
Ce n'est pas en contradiction avec la complexité du problème du tri car ici seuls les accès aux données compte et il n'y a pas de comparaisons (implicite dans l'indice du tableau).
De plus l'algorithme proposé ici est stable puisqu'il préserve l'ordre initial dans le tableau. Il peut donc être très utile si l'on souhaite trier selon 2 critères. Par exemple on dispose d'un tableau trié des étudiants d'informatique et on souhaite construire la liste triée des étudiants par groupe. Il y a 3 groupes (clé de tri) et le tri se fait en temps linéaire !

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Corrigé Q1 2017

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Principes pour la conception d'algorithmes

1. Déterminer les entrées et les sorties (spécification)
2. Trouver la structure de donnée adaptée pour ce problème
   - Ne pas hésiter à pré-traiter les entrées pour les mettre sous une forme adéquate
3. Essayer de réduire le problème à un problème connu
   - Tri, recherche, chemin dans un graphe,...
   - Vérifier si une solution existe déjà (livres, internet,...)
4. Décider de la manière d'approcher le problème : itératif/récursif/mix
   - Dépend de la manière de penser, de la facilité à trouver des invariants, de la manière de décomposer le problème en sous problèmes,...
5. **Écrire** l'algorithme
6. Faire tourner l'algorithme sur des exemples simples et des exemples "limite".
7. Donner les invariants de l'algorithme et faire la preuve
8. Évaluer la complexité de l'algorithme

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Principes pour l'expression des algorithmes

Ma position personnelle (que certains de mes collègues ne partagent pas):

- Un algorithme est toujours accompagné de schémas (ou dessins) et de texte représentation de l'état des variables,
- utiliser les conventions
  (i,j sont des indices, x un réel, n un entier par exemple une taille de tableau,. . .).
- utiliser des identificateurs explicites, une variable un usage
  noms de variables, de fonctions,
- la forme est importante
  l'indentation doit permettre de comprendre la structure de l'algorithme
- mettre des commentaires dans le code et accompagner l'algorithme par une explication en français
- ne mettre que l'information importante : minimiser l'encre
- être cohérent
  utiliser le même formalisme pour toutes les présentations d'algorithmes

de manière plus générale

favoriser tout ce qui aide à la compréhension et éliminer ce qui peut perturber la lecture

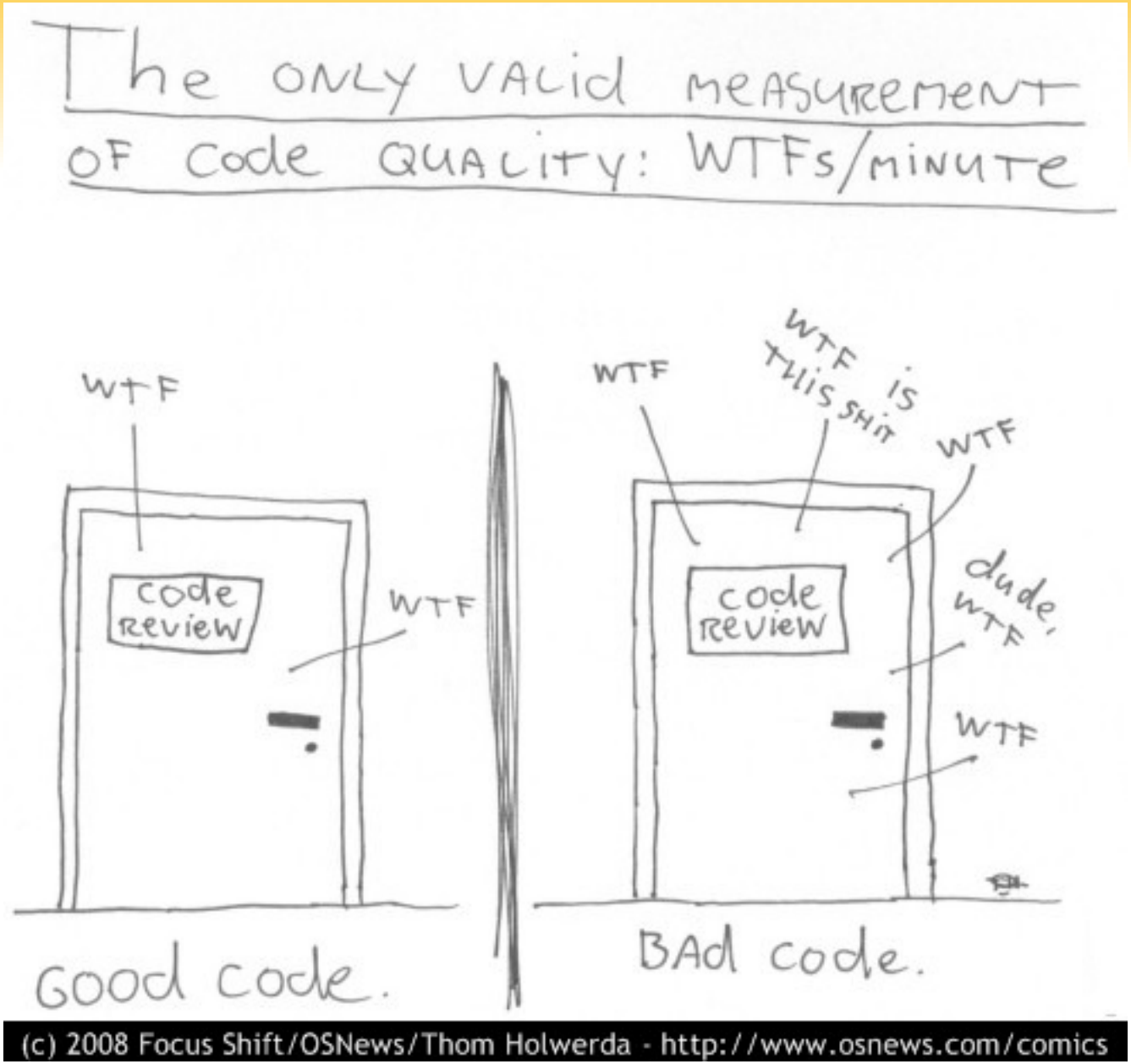
UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Conseils pour un examen

- au préalable : quels sont les grands chapitres du cours au programme de l'examen ? (identifier les gros blocs, les exercices types, les "grands résultats")
- lire le sujet en entier (pour voir où l'on va)
- rédiger les réponses, ainsi s'il y a une erreur sur l'algorithme vous aurez expliqué votre méthode
- rédiger les preuves correctement hypothèses/déroulement logique des arguments/conclusion
- un algorithme sans explication c'est comme une Ferrari sans roue
- soigner la présentation et la rédaction

Bref, entrainez-vous...

UFR IM²AG
UNIVERSITÉ Grenoble Alpes


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics