Quick Algorithmique et Modélisation Q1

Quelques éléments de correction

Jean-Marc.Vincent@imag.fr

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

23 mars 2016

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 2016

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice : Recherche du pic

On dispose d'un tableau T de taille n, d'éléments comparables ayant un pic, c'est à dire qu'il existe $p \in \{1, \ldots, n\}$ tel que

$$T[1] < T[2] < \cdots < T[p] > T[p+1] > \cdots > T[n].$$

Remarques préliminaires

Faire un dessin

- les éléments sont distincts (pas d'égalité possible), les indices vont de 1 à n
- les contraintes sur le tableau sont fortes, (un seul pic, croissant puis décroissant...)
- il faut faire attention aux deux cas extrêmes (le pic est sur un bord), ici il faut compléter la spécification. Par exemple le pic sera en 1 si le tableau est décroissant, en n s'il est croissant.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice : Recherche du pic

On dispose d'un tableau T de taille n, d'éléments comparables ayant un pic, c'est à dire qu'il existe $p \in \{1, \ldots, n\}$ tel que

$$T[1] < T[2] < \cdots < T[p] > T[p+1] > \cdots > T[n].$$

Remarques préliminaires

Faire un dessin

- les éléments sont distincts (pas d'égalité possible), les indices vont de 1 à n
- les contraintes sur le tableau sont fortes, (un seul pic, croissant puis décroissant...)
- il faut faire attention aux deux cas extrêmes (le pic est sur un bord), ici il faut compléter la spécification. Par exemple le pic sera en 1 si le tableau est décroissant, en n s'il est croissant.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

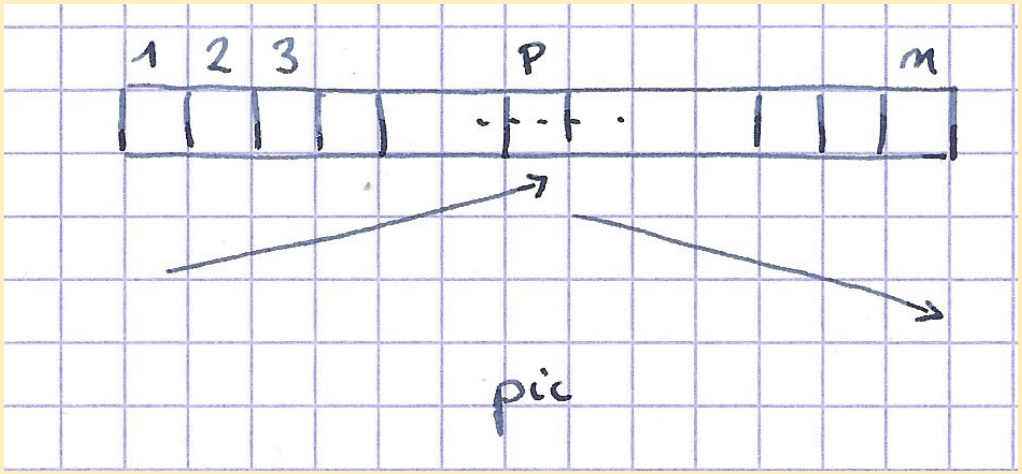
Exercice : Recherche du pic

On dispose d'un tableau T de taille n, d'éléments comparables ayant un pic, c'est à dire qu'il existe $p \in \{1, \ldots, n\}$ tel que

$$T[1] < T[2] < \cdots < T[p] > T[p+1] > \cdots > T[n].$$

Remarques préliminaires

Faire un dessin



- les éléments sont distincts (pas d'égalité possible), les indices vont de 1 à n
- les contraintes sur le tableau sont fortes, (un seul pic, croissant puis décroissant...)
- il faut faire attention aux deux cas extrêmes (le pic est sur un bord), ici il faut compléter la spécification. Par exemple le pic sera en 1 si le tableau est décroissant, en n s'il est croissant.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

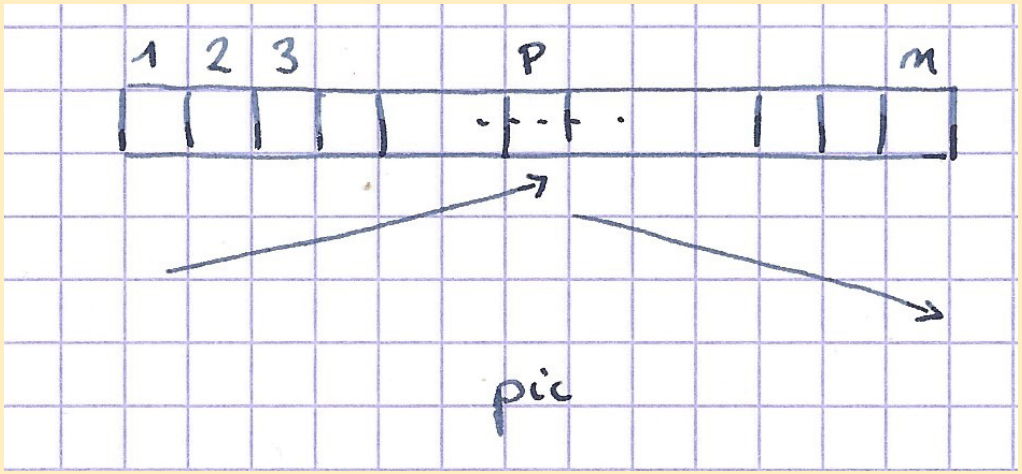
Exercice : Recherche du pic

On dispose d'un tableau T de taille n, d'éléments comparables ayant un pic, c'est à dire qu'il existe $p \in \{1, \ldots, n\}$ tel que

$$T[1] < T[2] < \cdots < T[p] > T[p+1] > \cdots > T[n].$$

Remarques préliminaires

Faire un dessin



- les éléments sont distincts (pas d'égalité possible), les indices vont de 1 à n
- les contraintes sur le tableau sont fortes, (un seul pic, croissant puis décroissant...)
- il faut faire attention aux deux cas extrêmes (le pic est sur un bord), ici il faut compléter la spécification. Par exemple le pic sera en 1 si le tableau est décroissant, en n s'il est croissant.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.1 : algorithme naïf

Proposer un algorithme naïf pour calculer la position du pic et calculer sa complexité.

Pour trouver le pic, on peut parcourir le tableau et s'arrêter lorsque l'on observe la rupture de croissance (on est certain qu'il n'y en a qu'une seule).

Pic(T, n)
Data: Un tableau T de taille n ayant un pic
Output: La position du pic

$i = 1$
while ($i < n$) **etpuis** ($T[i] < T[i+1]$)
  i=i+1
Return i

Le coût de l'algorithme est en $\mathcal{O}(n)$ puisque c'est un parcours de tableau de taille n.

attention aux bornes et aux indices, beaucoup de solutions compliquées, des for avec des breaks, des return dans des conditions,...

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.1 : algorithme naïf

Proposer un algorithme naïf pour calculer la position du pic et calculer sa complexité.

Pour trouver le pic, on peut parcourir le tableau et s'arrêter lorsque l'on observe la rupture de croissance (on est certain qu'il n'y en a qu'une seule).

Pic(T, n)
Data: Un tableau T de taille n ayant un pic
Output: La position du pic

$i = 1$
while ($i < n$) **etpuis** ($T[i] < T[i+1]$)
  i=i+1
Return i

Le coût de l'algorithme est en $\mathcal{O}(n)$ puisque c'est un parcours de tableau de taille n.

attention aux bornes et aux indices, beaucoup de solutions compliquées, des for avec des breaks, des return dans des conditions,...

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.2 : diviser pour régner

Proposer un algorithme de type diviser pour régner pour calculer la position du pic et calculer sa complexité.

On va d'abord étendre la fonction pic à un sous-intervalle du tableau. On s'inspire donc de la recherche dichotomique en coupant le tableau en 2 et en identifiant la partie du tableau ou se trouve le pic.

Faire un dessin

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.2 : diviser pour régner

Proposer un algorithme de type diviser pour régner pour calculer la position du pic et calculer sa complexité.

On va d'abord étendre la fonction pic à un sous-intervalle du tableau. On s'inspire donc de la recherche dichotomique en coupant le tableau en 2 et en identifiant la partie du tableau ou se trouve le pic.

Faire un dessin

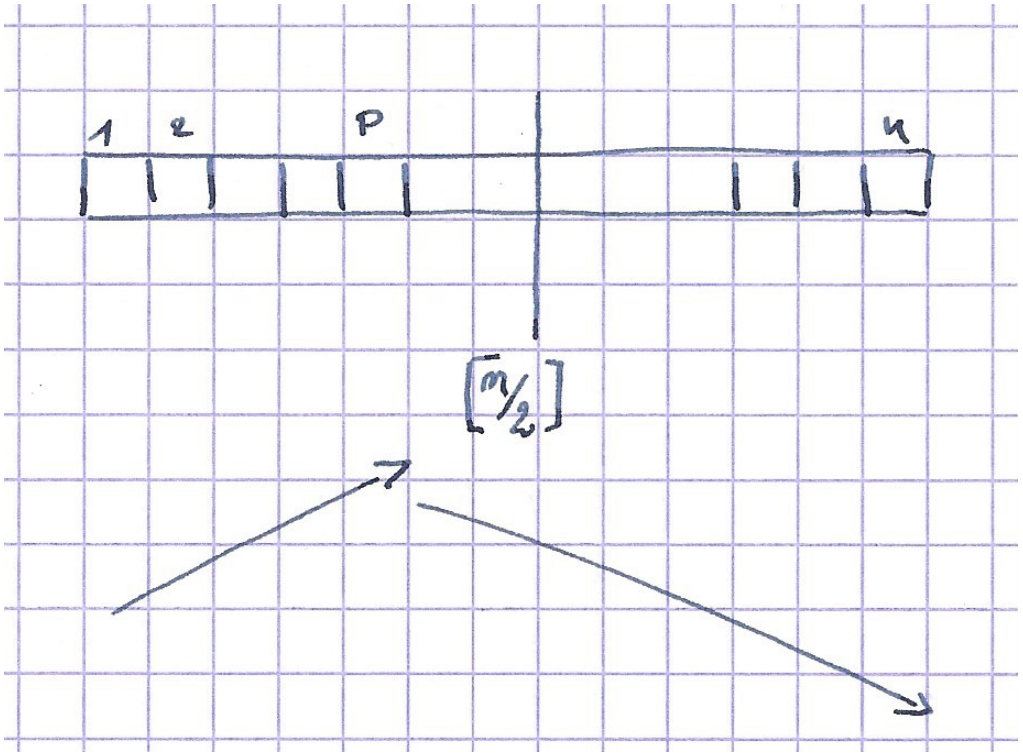
UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Question 1.2 : diviser pour régner

Proposer un algorithme de type diviser pour régner pour calculer la position du pic et calculer sa complexité.

On va d'abord étendre la fonction pic à un sous-intervalle du tableau. On s'inspire donc de la recherche dichotomique en coupant le tableau en 2 et en identifiant la partie du tableau ou se trouve le pic.

Faire un dessin

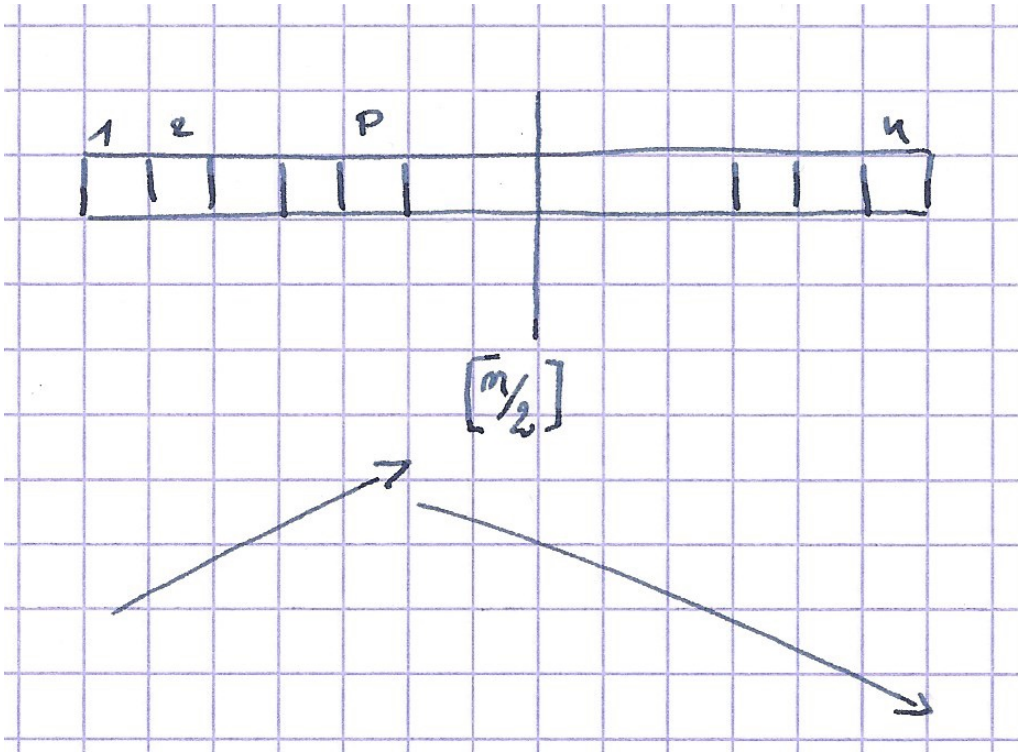


Question 1.2 : diviser pour régner

Proposer un algorithme de type diviser pour régner pour calculer la position du pic et calculer sa complexité.

On va d'abord étendre la fonction pic à un sous-intervalle du tableau. On s'inspire donc de la recherche dichotomique en coupant le tableau en 2 et en identifiant la partie du tableau ou se trouve le pic.

Faire un dessin



Pic(T, i, j)

Data: Un sous-tableau T ayant un pic entre l'indice i et l'indice j, on a $i \leqslant j$
Output: La position du pic

if $i = j$ // cas de base
  Return i
else
  if $T[\frac{i+j}{2}] < T[\frac{i+j}{2} + 1]$
    // le pic est à droite du milieu
    // Ici $\frac{i+j}{2}$ est une division entière donc $i \leqslant \frac{i+j}{2} < \frac{i+j}{2} + 1 \leqslant j$
    Return Pic($T, \frac{i+j}{2} + 1, j$)
  else
    // le pic est à droite du milieu
    Return Pic($T, i, \frac{i+j}{2}$)

L'appel initial : Pic($T, 1, n$)
Le coût de l'algorithme est identique au coût d'une recherche par dichotomie dans un tableau trié donc en $\mathcal{O}(\log n)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Corrigé Q1 2016

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 2 : Opérations sur les ensembles

On se donne deux tableaux T et U de tailles n et m, on note $N = \max\{m, n\}$. On veut produire un tableau V contenant tous les éléments apparaissant dans T mais pas dans U.

Proposer un algorithme de coût $\mathcal{O}(N^2)$ pour effectuer cette opération.

Pour effectuer cette opération, on devra examiner tous les éléments de T et rechercher s'ils sont dans U. Une première solution serait de parcourir T (boucle for) et rechercher si l'élément est dans U (boucle for), l'algorithme est en $\mathcal{O}(N^2)$ ($n \times m$ tests)

Delta(T, U, V)
Data: Deux tableaux T et U d'éléments de même nature, T de taille n, U de taille m
Output: un tableau V (de taille n) d'éléments de T qui ne sont pas dans U

```
k = 1
for i = 1 to n
    b = (T[i] == U[1])
    for j = 2 to m
        b = b ∨ (T[i] == U[j])
    if not(b)
        V[k] = T[i]
        k = k + 1
```

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 2 : Opérations sur les ensembles

La boucle interne de l'algorithme précédent consiste à rechercher si un élément est dans U. On peut donc réécrire l'algorithme :

```
Delta(T, U, V)
    Data: Deux tableaux T et U d'éléments de même nature, T de taille n, U de taille
          m
    Output: un tableau V (de taille n) d'éléments de T qui ne sont pas dans U

    k = 1
    for i = 1 to n
    |   if not(Appartient (T[i], U))
    |   |   V[k] = T[i]
    |   |   k = k + 1
```

Tout dépend donc du coût de la fonction `Appartient`.
Dans le cas précédent, `Appartient` était codé comme une recherche séquentielle.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 2 : Opérations sur les ensembles

On peut améliorer l'opération précédente en commençant par trier les tableaux, calculer la complexité d'une telle solution ?

Si le tableau U est trié, ce qui est un prétraitement de l'algorithme de recherche. Alors la recherche peut se faire par dichotomie, donc en $\mathcal{O}(\log m)$. Comme on fait n recherches on obtient un coût hors prétraitement de $\mathcal{O}(N \log N)$. Le prétraitement, consistant en un tri a une complexité en $\mathcal{O}(N \log N)$ (on prendra un algorithme de tri qui va bien tel que le heapsort qui trie en place en $\mathcal{O}(N \log N)$). Le coût global de l'algorithme est en $\mathcal{O}(N \log N)$ ce qui est une réduction significative de la complexité de l'approche naïve.
Une autre solution consiste à trier les deux tableaux et à parcourir simultanément les deux tableaux (comme pour la fusion de liste triée) Le coût est alors de $\mathcal{O}(N \log N)$ pour faire les 2 tris et de $n + m$ pour parcourir simultanément les 2 tableaux. Ce qui fait du $\mathcal{O}(N \log N)$

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Exercice 2 : Opérations sur les ensembles

Peut-on calculer le tableau V en coût $\mathcal{O}(N)$? Si oui, écrire l'algorithme.

Ici encore on demande un saut de complexité, il faut donc trouver une structure adaptée.Une bonne table de hachage fera l'affaire.
On se donne donc une table de hachage H de taille largement supérieure à m, associée à une fonction de hachage h. On suppose un hachage ouvert avec résolution des collusions par chaînage. La recherche d'un élément dans H est en moyenne de $\mathcal{O}(1)$. Et donc le coût de Delta est $\mathcal{O}(n)$ plus le coût du prétraitement (insertion des éléments dans la table qui est en $\mathcal{O}(m)$) donc finalement le calcul de V est en $\mathcal{O}(N)$
La fonction `Appartient` s'écrit donc

```
Appartient (x, H)
    Data: x un élément et une table de hachage H
    Output: un booléen indiquant si l'élément x est dans H

    if H[h(x)]
    |   // Il y a collision, l'élément x est peut-être dans la
    |      table
    |   // on parcourt la liste des éléments hachés sur la case
    |      H[h(x)]
    |   Return (x est l'un des éléments chaîné en H[h(x)])
    else
    |   // l'élément x n'est pas dans la table
    |   Return Faux
```

Corrigé Q1 2016

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Problème : Élément de rang k

On se donne un tableau T comprenant n éléments distincts. On appelle élément de rang k du tableau T, le $k$ième plus grand élément du tableau T (l'élément de rang 1 est donc le plus petit élément du tableau tandis que l'élément de rang n est le plus grand élément du tableau).

Écrire un algorithme de complexité $\mathcal{O}(n)$ qui calcule les éléments de rang 1 et de rang n.

exercice de L1

Écrire un algorithme naïf permettant de calculer l'élément de rang k, calculer la complexité de votre algorithme et, si ce n'est pas le cas, modifier votre algorithme pour obtenir une complexité $\mathcal{O}(n \log n)$ dans le pire cas.

On effectue un prétraitement en triant le tableau, l'élément de rang k est alors l'élément à l'indice k du tableau trié. Le coût de l'algorithme est donc le coût du tri plus le coût d'un accès (en $\mathcal{O}(1)$), soit $\mathcal{O}(n \log n)$ avec un algorithme de tri adéquat.

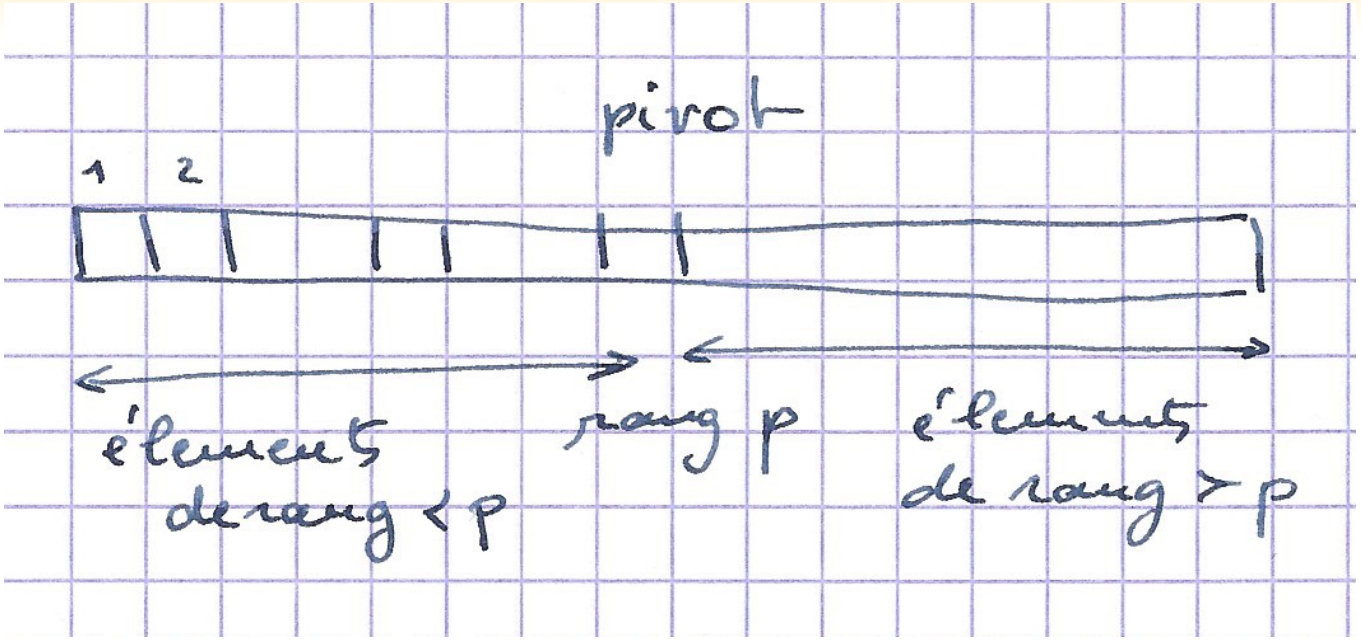
UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Problème : Élément de rang k

On rappelle l'algorithme *quicksort* randomisé:
- Choisir un élément au hasard (uniformément) parmi $T[1] \dots T[n]$ comme pivot
- Partitionner les éléments en plaçant les éléments plus petit que le pivot en début du tableau et les éléments plus grand que le pivot en fin de tableau.
- Appeler récursivement *quicksort* sur les deux sous-tableaux.

En s'inspirant de l'algorithme de *quicksort*, écrire un algorithme `QuickSelect`(T,k) qui calcule l'élément de rang k (**indication:** après avoir partitionné le tableau en deux, il facile de tester si l'élément de rang k est plus petit ou plus grand que le pivot.)

UFR IM²AG
UNIVERSITÉ Grenoble Alpes



Lorsque l'on réalise le partitionnement, on calcule le rang du pivot, donc on sait si l'élément de rang k est dans le sous-tableau de gauche ou de droite et on fait l'appel récursif sur le sous-tableau concerné en cherchant l'indice correspondant (tout se passe dans les paramètres.

Montrer que (il est fortement suggéré de faire des dessins)

$$C(n,k) = (n-1) + \frac{1}{n}\sum_{i=1}^{k-1} C(n-i,k-i) + \frac{1}{n}\sum_{i=k+1}^{n} C(i,k)$$

On notera $M(n)$ et on admettra (question bonus) que

$$M(n) \leq \frac{2}{n}\sum_{p=n/2}^{n-1} M(p) + \mathcal{O}(n).$$

En déduire que $M(n) = \mathcal{O}(n)$.

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Corrigé Q1 2016

UFR IM²AG
UNIVERSITÉ Grenoble Alpes

Barème

1. Exercice Pic
   1.1 : 1pt
   1.2 : 3pts
2. Exercice Delta
   2.1 : 1pt
   2.2 : 2pts
   2.3 : 3pts
3. Problème rang k
   3.1 : 1 pt
   3.2 : 2 pts
   3.3 : 3 pts
   3.4 : 2 pts
   3.5 : 2pts
4. Rédaction, présentation : 2 pts

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