Problème : un algorithme « diviser pour régner »

Problème : Recherche de pic

1. Recherche des pics

Question 1. Avec une fonction `test_pic` que l'on réutilisera dans la suite.

```
let test_pic t i = (i=0 || t.(i) >= t.(i-1)) && (i=Array.length t - 1 || t.(i) >= t.(i+1)) ;;

let indices_pics t =
  let q=ref [] in
  for i=0 to Array.length t - 1 do
    if test_pic t i then q:= i :: !q
  done ;
  !q
;;
```

2. Un seul pic en dimension 1

Question 2. Considérons le maximum du tableau entre l'indice 0 et l'indice $i-1$, et notons j son indice :

— si $j < i-1$, alors j est bien l'indice d'un pic de `t`.

— sinon, on a d'une part `t.(j)` $<$ `t.(j-1)` (si $j > 0$), mais d'autre part `t.(j)` $<$ `t.(j+1)` par hypothèse, doinc j est également l'indice d'un pic de `t`.

Question 3. Voici :

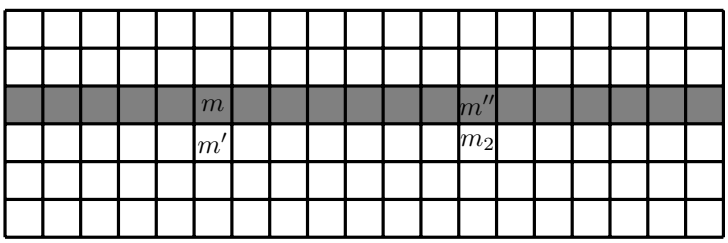
```
let chercher_pic t =
  let rec aux g d =
    let m=(g+d)/2 in
    if test_pic t m then m
    else if t.(m)<t.(m+1) then aux (m+1) d
    else aux g (m-1)
  in aux 0 (Array.length t -1)
;;
```

On maintient l'invariant de boucle « `t` possède un pic entre les indices g et d inclus ». Si m n'est pas un pic, on a nécessairement $g < d$, donc $g \leq m < d$ (avec égalité éventuelle si $d = g+1$). Le test `t.(m)<t.(m+1)` ne fait donc pas de dépassement d'indice.

3. Un seul pic en dimension supérieure

Question 4.

- Terminaison : montrons que si t est un tableau à $N \geq 1$ lignes et $M \geq 1$ colonnes, alors l'algorithme termine sur t. C'est le cas si $N = 1$, car le maximum est un pic. Sinon, on se rappelle sur un tableau non vide, possédant strictement moins de lignes que t : l'algorithme termine.
- Correction : on montre par récurrence sur $N \geq 1$ que l'algorithme renvoie bien un pic du tableau fourni en paramètre (avec $M \geq 1$ quelconque), ce qu'on fait par récurrence forte. C'est le cas si $N = 1$. Si le maximum de la ligne médiane n'est pas un pic, supposons (c'est symétrique), que l'algorithme se rappelle sur le tableau t_2 formé par les lignes en desosus de la ligne médiane. Par hypothèse de récurrence, l'algorithme fournit bien un pic de ce tableau, il s'agit de justifier que c'est aussi un pic de t. Le seul cas non trivial est celui où le pic renvoyé est situé sur la première ligne de t_2, comme dans l'image ci-dessous :



Notons :

- m l'élément maximal de la ligne médiane ;
- m' l'élément situé en dessous : on a donc $m' \geq m$ car l'algorithme s'est rappelé sur t_2 constitué des lignes sous la ligne médiane ;
- m_2 le maximum de la ligne où est situé m' : c'est le pic renvoyé par l'algorithme, on a donc $m_2 \geq m'$. Puisque m_2 est un pic de t_2, il est supérieur à l'élément éventuellement situé en dessous ;
- avec m'' l'élément situé au dessus de m_2, il reste à vérifier que $m_2 \geq m''$ pour justifier que m_2 est bien un pic. Mais comme m est le maximum de sa ligne, on a $m \geq m''$. Ainsi $m'' \leq m \leq m' \leq m_2$, et m_2 est bien un pic de t.

Question 5. Notons $C(N, M)$ la complexité sur un tableau à N lignes et M colonnes. On a pour $N \geq 2$:

$$C(N,M) = \underbrace{C(\lfloor N/2 \rfloor)}_{\text{appel récursif}} + \underbrace{O(N)}_{\text{extraction des lignes}} + \underbrace{O(M)}_{\text{calcul du maximum de la ligne médiane}}$$

Résolvons cette récurrence pour N une puissance de 2 : on pose $N = 2^p$, on a alors $C(2^p, M) = C(2^{p-1}, M) + O(2^p) + O(M)$ pour tout $p \geq 1$. On peut télescoper : $C(2^p, M) = \sum_{i=1}^{p} \underbrace{(C(2^i, M) - C(2^{i-1}, M))}_{O(M)+O(2^i)} + \underbrace{C(1, M)}_{O(M)} = O(Mp) + O(2^p)$ car $\sum_{i=1}^{p} 2^i = 2^{p+1} - 1 = O(2^p)$. Or $p = \log_2(N)$, d'où $C(N, M) = O(N) + O(M \log N)$.
Remarque : on pouvait avoir l'intuition de ce résultat directement : la recherche va se produire sur $O(\log N)$ lignes du tableaux, chaque calcul de maximum se faisant avec une complexité $O(M)$. Chaque extraction de portion se fait en temps linéaire en le nombre de lignes, et au total il faut extraire $N/2 + N/4 + N/8 + \cdots = O(N)$ lignes. Une extraction de lignes se fait en temps constant car seule l'adresse mémoire de la ligne est recopiée, pas l'ensemble des éléments.

Question 6. On écrit d'abord une fonction de recherche d'indice de maximum sur un tableau à une dimension.

```
let imax t=
  let n=Array.length t and m=ref 0 in
  for i=1 to n-1 do
    if t.( !m) < t.(i) then m:= i
  done ;
  !m
;;
```

Puis on écrit la fonction elle-même.

```
let rec cherche_pic_2 t=
  let n=Array.length t and m=Array.length t.(0) in
  let d=n/2 in let im=imax t.(d) in
  if d<n-1 && t.(d+1).(im) > t.(d).(im) then let a,b=cherche_pic_2 (Array.sub t (d+1) (n-d-1)) in (a+d,b)
  else if d>0 && t.(d-1).(im) > t.(d).(im) then cherche_pic_2 (Array.sub t 0 d)
  else (d,im)
;;
```

Remarque : attention, le fait d'extraire des lignes change l'indexation, d'où le rajout de d à la première composante lorsqu'on travaille sous la ligne médiane.

4. Un algorithme optimal en dimension 2

Question 7. Si le maximum sur la croix n'est pas un pic, il est plus petit qu'un élément situé dans une des quatre zones blanches : il suffit de chercher un pic dans cette zone. La correction se montre en suivant le même principe que pour l'algorithme précédent. Sans recopie d'éléments, on a la complexité suivante (la croix possède $O(N + M)$ éléments).

$$C(N,M) = C(\lfloor N/2 \rfloor, \lfloor M/2 \rfloor) + O(N+M)$$

En admettant la croissance, on a donc en notant $p = N + M$ la relation $C(p) = C(\lfloor p/2 \rfloor) + O(p)$, dont la solution est $C(p) = O(p)$ (voir le cours, ou simplement sommer : $C(p) = O(p + p/2 + p/4 + \cdots) = O(2p) = O(p)$). L'algorithme a donc une complexité $O(N + M)$.