Une solution de complexité n^3 au problème de la grille de mots croisés

Une solution de complexité n^3 au problème de la grille de mots croisés (voir TP correspondant) a été proposée par Claire Kenyon. En voici une description, ainsi que le program Caml correspondant.

1 Une première solution de complexité n^4

Une première solution consiste à calculer les valeurs booléennes $T_{i,j,k,l}$ telles que $T_{i,j,k,l}$ vaut Vrai si et seulement si le rectangle d'extrémité (i,j), de largeur k et de hauteur l contient au moins une case noire. On remarque alors que l'on peut calculer $T_{i,j,k,l}$ de proche en proche, et en temps constant pour chaque quadruplet (i,j,k,l). En effet, on a :

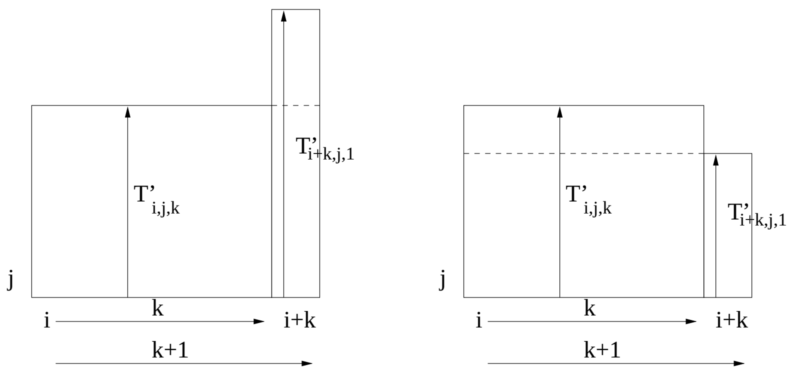
$$\begin{array}{rcl} T_{i,j,k+1,l} & = & T_{i,j,k,l} \vee T_{i+k,j,1,l} \\ T_{i,j,1,l+1} & = & T_{i,j,1,l} \vee T_{i,j+l,1,1} \\ T_{i,j,1,1} & = & \textsf{Vrai} \text{ si et seulement si la case } (i,j) \text{ est noire} \end{array}$$

Il suffit alors d'allouer un tel tableau T et de calculer ses valeurs grâce aux équations ci-dessus, en conservant au fur et à mesure le plus grand rectangle de cases blanches rencontré (On peut aussi le rechercher *a posteriori*, c'est-à-dire après le calcul de T). La complexité d'un tel algorithme est clairement n^4 (c'est la taille de T).

2 Une solution de complexité n^3

Mais on peut faire mieux, en calculant dans un tableau $T'_{i,j,k}$ la plus grande valeur l pour laquelle $T_{i,j,k,l}$ soit Faux. Il s'avère que, là encore, on peut calculer les valeurs $T'_{i,j,k}$ de proche en proche, en temps constant pour $k > 1$ et en temps n pour $k = 1$, comme le montre les équations:

$$\begin{array}{rcl} T'_{i,j,k+1} & = & \min(T'_{i,j,k}, T'_{i+k,j,1}), \quad \text{voir figure ci-dessous} \\ T'_{i,j,1} & = & \text{le plus grand } l \text{ tel que les cases } (i,j),(i,j+1),\ldots,(i,j+l-1) \text{ soient blanches} \end{array}$$



La complexité d'un tel algorithme est n^3. En effet, le calcul des $T'_{i,j,1}$ est de coût n pour chaque (i,j), donc de coût total n^3, et le calcul des $T'_{i,j,k}$ pour $k > 1$ se fait en temps constant de proche en proche et est donc de coût total n^3 également.

Le programme Caml implantant cet algorithme est le suivant, en conservant les définitions préliminaires données dans le corrigé:

```
let T' = make_matrix N N [||];;
for i = 0 to N-1 do
  for j = 0 to N-1 do
    T'.(i).(j) <- make_vect (succ N) 0
  done
done;;

let init_T' () =
  let rec colonne i j l =
    if j+l<N & not (grille.(i).(j+l)) then colonne i j (succ l) else l in

  for i = 0 to N-1 do
    for j = 0 to N-1 do
      T'.(i).(j).(1) <- colonne i j 0 ;
      for k = 2 to N do  T'.(i).(j).(k) <- 0  done
    done
  done
;;

let trouve' () =
  init_T'() ;
  let sol = ref (0,0,0) and aire = ref 0 in

  let garde_max ((i,j,k) as s) =
    let a = k * T'.(i).(j).(k) in
    if a > !aire then begin sol := s ; aire := a end in

  for i = 0 to N-1 do  for j = 0 to N-1 do
    garde_max (i,j,1) ;
    for k = 2 to N-i do
      T'.(i).(j).(k) <- min  T'.(i).(j).(pred k)  T'.(i+k-1).(j).(1) ;
      garde_max (i,j,k)
    done
  done done ;

  let (i,j,k) = !sol in Rect (Interv (i,k), Interv (j,T'.(i).(j).(k)))
;;
```