Somme minimale d'un segment de tableau

Soit t un tableau de n entiers. Un segment de t est une suite non vide constitué d'éléments de t consécutifs : $t(i..j) = (t.(i), \ldots, t.(j))$ avec $0 \leqslant i \leqslant j < n$. L'objet du TP est de programmer divers algorithmes permettant de déterminer le minimum des sommes de tous les segments de t.

Pour les essais, on pourra utiliser la fonction suivante qui retourne un tableau de longueur n contenant des entiers aléatoires entre a et b :

```
let random_vect n a b =
  let t = make_vect n 0 in
  for i=0 to n-1 do t.(i) <- a + random__int(b-a+1) done;
  t
;;
```

Par ailleurs, on rappelle que en Caml, le minimum de deux entiers x et y s'obtient par l'expression `min` x y.

1) Algorithme en n^3

a) Écrire une fonction `coupe_3` prenant en arguments un tableau t et deux entiers i, j et retournant la somme des éléments du segment $t(i..j)$. Il sera supposé que les indices i, j passés en arguments sont valides pour le tableau t.

b) A l'aide de `coupe_3`, écrire une fonction `minisomme_3` prenant en argument un tableau de longueur n et retournant le minimum des sommes des segments de t.

2) Algorithme en n^2

On améliore l'algorithme précédent de la manière suivante : si l'on connaît la somme s d'un segment $t(i..j)$ alors on peut en déduire la somme s' du segment $t(i..j+1)$ par la relation $s' = s + t.(j+1)$, ce qui est plus rapide que de calculer $t.(i) + t.(i+1) + \ldots + t.(j+1)$.

a) Écrire une fonction `coupe_2` prenant en arguments un tableau t de longueur n et un indice i et retournant le minimum des sommes $t(i..j)$ lorsque j varie de i à $n-1$.

b) En déduire une fonction `coupe_2` calculant le minimum des sommes des segments d'un tableau t donné en argument. Constater que cette fonction s'exécute plus rapidement que `coupe_3` lorsque n est grand (de l'ordre de quelques centaines).

3) Algorithme en n

Soit la fonction suivante :

```
let minisomme_1 t =
  let x = ref(t.(0)) and s = ref(t.(0)) in
  for i=1 to vect_length(t)-1 do
    x := min (!x + t.(i)) t.(i);
    s := min !s !x
  done;
  !s
;;
```

Saisir cette fonction et constater qu'elle calcule effectivement le minimum des sommes des segments de t et qu'elle s'exécute bien plus rapidement que `somme_2`. Justifier par le raisonnement la validité de cette fonction.

4) Somme la plus proche d'un entier donné

On se donne un tableau t et un entier k. Écrire une fonction qui calcule la somme d'un segment de t la plus proche de k (ou l'une des plus proches s'il y en a deux).