TP1 - Sous-séquences maximales

Sébastien Delecraz Éloi Perdereau

16 septembre 2013

1 Analyse théorique

1.1 Algorithme naïf

Entrées: T, n
Sorties: Sous-séquence maximale
début
  $S_{max} \leftarrow -\infty$
  pour tous les $1 \leq k \leq n$ **faire**
    pour tous les $k \leq l \leq n$ **faire**
      $S \leftarrow 0$
      pour tous les $k \leq j \leq l$ **faire**
        $S \leftarrow S + T[j]$
      fin
      si $S > S_{max}$ **alors**
        $S_{max} \leftarrow S$
      fin
    fin
  fin
  retourner S_{max}
fin

Algorithm 1: Naïf

Complexité

$$O(n^3)$$

1.2 Algorithme moins naïf

Complexité

$$O(n^2)$$

Entrées: T, n
Sorties: Sous-séquence maximale
début
    $S_{max} \leftarrow -\infty$
    pour tous les $1 \leq k \leq n$ **faire**
        $S \leftarrow 0$
        pour tous les $k \leq l \leq n$ **faire**
            $S \leftarrow S + T[k]$
            si $S > S_{max}$ **alors**
                $S_{max} \leftarrow S$
            fin
        fin
    fin
    retourner S_{max}
fin

Algorithm 2: Moins naïf

1.3 Algorithme diviser pour régner

Complexité

$$O(n \log n)$$

1.4 Algorithme incrémental

Complexité

$$O(n)$$

Entrées: T, k, l
Sorties: Sous-séquence maximale
début
    si $l - k = 1$ **alors retourner** $T[k]$;
    si $l - k = 2$ **alors retourner** $max\{T[k],\ T[k+1],\ T[k] + T[k+1]\}$;
    $j \leftarrow \frac{l-k}{2}$
    $S_1 \leftarrow Diviser\_pour\_regner(T, k, k + j - 1)$
    $S_2 \leftarrow Diviser\_pour\_regner(T, k + j + 1, l)$
    $S_3 \leftarrow S_{tmp} \leftarrow T[j]$
    pour *i variant de j-1 à k descendant* **faire**
        $S_{tmp} \leftarrow S_{tmp} + T[i]$
        si $S_{tmp} > S_3$ **alors**
            $S_3 \leftarrow S_{tmp}$
        fin
    fin
    $S_4 \leftarrow S_{tmp} \leftarrow T[j]$
    pour *i variant de j+1 à l montant* **faire**
        $S_{tmp} \leftarrow S_{tmp} + T[i]$
        si $S_{tmp} > S_3$ **alors**
            $S_4 \leftarrow S_{tmp}$
        fin
    fin
    $S_0 \leftarrow S_3 + S_4 - T[j]$
    retourner $\max S_0, S_1, S_2$
fin

Algorithm 3: Diviser pour régner

```
Entrées: T, n
Sorties: Sous-séquence maximale
début
    S ← T[1]
    queue ← 1
    pour tous les 2 ≤ i ≤ n faire
        si T[i] > 0 alors
            S_0 ← 0
            pour j variant de i-1 à queue+1 descendant faire
                S_0 ← S_0 + T[j]
            fin
            si S_0 > 0 alors
                queue ← i  S ← S + S_0
            fin
        sinon
            si S < T[i] alors
                S ← T[i]
            fin
        fin
    fin
fin
```

Algorithm 4: Incrémental