Schémas algorithmiques

Dans ce chapitre, on présente différents paradigmes d'algorithmes qui permettent de développer des algorithmes résolvant des problèmes (d'optimisation) mettant en jeu des textes, des noms, des listes, des graphes, des expressions algorithmiques, des points du plan, des ordonnancements ...

1 Introduction

On rappelle qu'un problème d'optimisation consiste à maximiser ou minimiser (on se placera, pour la suite du cours, dans le premier cas) une fonction à valeurs réelles donnée, sur un ensemble de solutions donné. Ainsi, un tel problème $\mathcal{P}$ est défini par la fonction dite **fonction objectif** f, et l'ensemble des solutions $\mathscr{S}$.

On appelle alors **valeur du problème** $\mathcal{P}$, notée $\mathrm{val}(\mathcal{P})$, la donnée $\max\{f(X) \mid X \in \mathscr{S}\}$, et **ensemble des solutions optimales** l'ensemble $\mathrm{argmax}_{x\in\mathscr{S}} f(x) = \{X \in \mathscr{S} \mid f(X) = \mathrm{val}(\mathcal{P})\}$

Remarque: Il y a unicité de la valeur du problème, mais pas forcément de la solution optimale.

Si $\mathscr{S} \subset \widetilde{\mathscr{S}} \subset D_f$ (où D_f désigne l'ensemble de définition de f), on dit que $\widetilde{\mathcal{P}} = (\widetilde{\mathscr{S}}, f)$, de solution optimale $\widetilde{P} = \mathrm{argmax}_{x\in\widetilde{\mathscr{S}}} f(x) = \max_{x\in\widetilde{\mathscr{S}}}(f(x))$, est un **relâché de** $\mathcal{P}$, et on a alors $\mathrm{val}(\widetilde{\mathcal{P}}) \geq \mathrm{val}(\mathcal{P})$.

2 Algorithmes gloutons

2.1 Définitions

On dit d'un algorithme qu'il est **glouton** lorsqu'il construit une solution à un problème (d'optimisation généralement), en prenant des décisions *localement* pertinentes (optimales) à chaque étape, et qu'il ne revient pas sur ses décisions. Dans le cadre d'un problème d'optimisation, on dit qu'un algorithme glouton est **optimal** ssi il renvoie toujours une solution optimale.

Exemple: L'algorithme de Huffman ou "rempli-sac" dans le cas fractionnaire sont des algorithmes gloutons optimaux.

Remarque: Attention, selon les problèmes, un même algorithme peut être optimal ou non (sac à dos fractionnaire vs sac à dos entier). En général, les algorithmes gloutons sont efficaces (de faible complexité, à fortiori polynomiale). Ils ne sont donc pas optimaux pour des problèmes difficiles (sac à dos entier). Cependant, ils peuvent être utiles pour obtenir rapidement une borne supérieure ou inférieure (c'est à dire un majorant ou un minorant de la valeur optimale).

Remarque: Pour savoir si un problème d'optimisation peut être résolu par un algorithme glouton, on se pose deux questions :

1. La fonction objectif est-elle décomposable comme somme de fonctions sous des sous parties de l'ensemble des solutions ?
2. L'ensemble des solutions est-il décomposable comme produit cartésien ou au contraire défini par des contraintes liantes ?

2.2 Exemples d'algorithmes gloutons

2.2.1 Sac à dos

Présentation

Sac à dos
- Entrée: $P \in \mathbb{R}_+^*, v = (v_i)_{i\in[\![1,n]\!]} \in (\mathbb{R}_+^*)^n$ et $p = (p_i)_{i\in[\![1,n]\!]} \in (\mathbb{R}_+^*)^n$
- Sortie: $\max\limits_{\substack{\delta\in\{0,1\}^n \\ \delta\cdot p\leq P}} \underbrace{\delta.v}_{=f(v)}$

Sac à dos fractionnaire
- Entrée: $P \in \mathbb{R}_+^*, v = (v_i)_{i\in[\![1,n]\!]} \in (\mathbb{R}_+^*)^n$ et $p = (p_i)_{i\in[\![1,n]\!]} \in (\mathbb{R}_+^*)^n$
- Sortie: $\max\limits_{\substack{\delta\in[0,1]^n \\ \delta\cdot p\leq P}} \underbrace{\delta.v}_{=f(v)}$

Remarque: L'opérateur $a \cdot b$ correspond au produit scalaire : pour deux n-uplets $a = (a_1, ..., a_n)$ et $b = (b_1, ..., b_n)$, $a \cdot b = (a_1b_1, a_2b_2, ..., a_nb_n)$.

Remarque: "$\delta \in \{0,1\}^n$" est une **contrainte d'intégrité** (son rôle est de vérifier la validité des données), "$\delta.p \leq P$" est une **contraine linéaire** (c'est une relation linéaire qui relie les différentes variables).

Pseudo-code pour le sac à dos fractionnaire

Rempli-sac :
- Entrée: $(p_i, v_i)_{i\in[\![1,n]\!]}$ une suite finie de réels strictement positifs, P un réel.
- Sortie: $\max\limits_{\substack{\delta\in[0,1]^n \\ \delta\cdot p\leq P}} \delta.(v_1, ..., v_n)$

Trier les objets par rapport $\frac{v_i}{p_i}$ décroissant
En déduire $\sigma \in \mathfrak{S}([\![1,n]\!])$ telle que $\left(\frac{v_{\sigma(k)}}{p_{\sigma(k)}}\right)_{k\in[\![1,n]\!]}$ décroît.
$s \leftarrow 0$
$\delta \leftarrow$ tableau de réels indicé par $[\![1,n]\!]$ initialisé à 0.
Invariant: $s = \sum_{i=1}^n \delta_i \times p_i$
$k = 1$
Tant que $s + p_{\sigma(k)} \leq P$
    $s \leftarrow s + p_{\sigma(k)}$
    $\delta_{\sigma(k)} \leftarrow 1$
    $k \leftarrow k + 1$
si $k \leq n$
    $\delta_{\sigma(k)} \leftarrow \frac{P-s}{s_{\sigma(k)}}$
    $s \leftarrow s + \delta_{\sigma(k)} \times p_\sigma(k)$

Preuve de optimalité On montre l'optimalité de cette stratégie par argument d'échange. On pose :

- Un entier $n \in \mathbb{N}^*$,
- Deux suites $p = (p_i)_{i\in[\![1,n]\!]}$ et $v = (v_i)_{i\in[\![1,n]\!]}$ telles que $\left(\frac{p_i}{v_i}\right)$ décroît strictement,
- Un réel $P \in \mathbb{R}_+^*$,
- L'ensemble $\mathscr{S} = \{\delta \in [0,1]^n \mid \delta.p \leq P\}$.

On suppose que $p \cdot \mathbb{1} > P$ (i.e $\sum_{i=1}^n p_i \times 1 > P$, donc $\mathbb{1} \notin \mathscr{S}$).

Remarque: On note $\mathbb{1}_k = (0, 0, ..., 0, \underbrace{1}_{\text{en position } k}, 0, ..., 0, 0)$, et $\mathbb{1}_{[\![1,k]\!]} = (\underbrace{1, 1, ..., 1, 1}_{\text{jusqu'à la position } k}, 0, 0, ..., 0, 0)$.

Soit $\delta^* \in \text{argmax}_{\delta\in\mathscr{S}}\, v \cdot \delta$. On sait que $\mathbb{1} \notin \mathscr{S}$ donc $\delta^* \neq \mathbb{1}$, donc $\{i \in [\![1,n]\!] \mid \delta_i^* < 1\} \neq \varnothing$. On peut alors définir $m = \min\{i \in [\![1,n]\!] \mid \delta_i^* < 1\}$. Par définition de m, $\forall i \in [\![1, m[\![, \delta_i^* = 1$.

Lemme: $\forall i \in [\![m+1, n]\!], \delta_i^* = 0$.

- ▷ Par l'absurde, on suppose qu'il existe $i_0 \in [\![m+1, n]\!]$ tel que $\delta^*_{i_0} \neq 0$. Posons $\varepsilon = \min(p_{i_0}\delta^*_{i_0}, p_m(1-\delta^*_m))$, et $\hat{\delta} = \delta^* - \frac{\varepsilon}{p_{i_0}}\mathbb{1}_{i_0} + \frac{\varepsilon}{p_m}\mathbb{1}_m$. Soit $i \in [\![1, n]\!]$.
  - Si $i \notin \{i_0, m\}, \hat{\delta}_i = \delta^*_i \in [0, 1]$.
  - $\hat{\delta_{i_0}} = \delta^*_{i_0} - \frac{\varepsilon}{p_{i_0}} \leq \delta^*_{i_0} \leq 1$. De plus, $\frac{\varepsilon}{p_{i_0}} \leq \frac{1}{p_{i_0}} \times p_{i_0}\delta^*_{i_0} = \delta^*_{i_0}$. Donc $\delta^*_{i_0} - \frac{\varepsilon}{p_{i_0}} \geq 0$ soit $\hat{\delta_{i_0}} \geq 0$.
  - $\hat{\delta_m} = \delta^*_m + \frac{\varepsilon}{p_m} \geq \delta^*_m \geq 0$. De plus $\frac{\varepsilon}{p_m} \leq 1 - \delta^*_n$ soit $\delta^*_m + \frac{\varepsilon}{p_m} \leq 1$ soit $\hat{\delta_m} \leq 1$.

  Ainsi $\hat{\delta} \in [0, 1]^n$. De plus,

$$\hat{\delta} \cdot p = (\delta^* + \frac{\varepsilon}{p_m}\mathbb{1}_m - \frac{\varepsilon}{p_{i_0}}\mathbb{1}_{i_0}) \cdot p = (\delta^* \cdot p) + \frac{\varepsilon}{p_m}(\mathbb{1}_m \cdot p) - \frac{\varepsilon}{p_{i_0}}(\mathbb{1}_{i_0} \cdot p) = (\delta^* \cdot p + \varepsilon - \varepsilon) = \delta^* \cdot p \overset{\delta^* \in \mathscr{S}}{\leq} P.$$

  Ainsi $\hat{\delta} \in \mathscr{S}$. Par ailleurs,

$$\hat{\delta} \cdot v = \delta^* \cdot v + \frac{\varepsilon}{p_m}(\mathbb{1}_m \cdot v) - \frac{\varepsilon}{p_{i_0}}(\mathbb{1}_{i_0} \cdot v) = \delta^* \cdot v + \varepsilon(\frac{v_m}{p_m} - \frac{v_{i_0}}{p_{i_0}}) > \delta^* \cdot v$$

  Ce qui est absurde car $\delta^* \in \mathrm{argmax}_{\delta \in \mathscr{S}}\, v.\delta$.

Lemme: $\delta^*_m = \frac{P - \sum_{k=1}^{m-1} p_k}{p_m}$.

- ▷ Supposons par l'absurde que $\delta^*_m \neq \frac{P - \sum_{i=1}^{m-1} p_k}{p_m}$. Notons $w = \frac{P - \sum_{i=1}^{m-1} p_k}{p_m} = \frac{P - (p \cdot \delta^* - p_m\delta^*_m)}{p_m}$ d'après ce que l'on a montré précédemment.
- ▷ Si $\delta^*_m < w$, alors $\delta^*_m p_m < P - (p \cdot \delta^* - p_m\delta^*_m)$, soit $p \cdot \delta^* < P$. Posons $\hat{\delta} = \delta^* + \frac{\varepsilon}{p_m}\mathbb{1}_m$ avec $\varepsilon = P - p \cdot \delta^* > 0$.
  - On alors $\hat{\delta} \in \mathscr{S}$. En effet, on a :

$$\hat{\delta} \cdot p = \delta^* \cdot p + \frac{\varepsilon}{p_m}(\mathbb{1}_m \cdot p) = \delta^* \cdot p + \varepsilon = P$$

    Par ailleurs, pour $k \in [\![1, n]\!] \setminus \{m\}, \hat{\delta}_k = \delta^*_k \in [0, 1]$. Ensuite, $\hat{\delta}_m = w = \delta^*_m + \frac{\varepsilon}{p_m} > \delta^*_m \geq 0$. $\hat{\delta}_m = \delta^*_m + \frac{P - p \cdot \delta^*}{p_m} = \frac{\delta^*_m p_m + P - p \cdot \delta^*}{p_m} = w$. Enfin, $w < 1$, i.e $P < \sum_{k=1}^m p_k$, car sinon $\mathbb{1}_{[\![1,m]\!]} \in \mathscr{S}$ serait une solution plus optimale que δ^* ($\mathbb{1}_{[\![1,m]\!]} \cdot v - \delta^* \cdot v = (1 - p_m)v_m > 0$ donc $\mathbb{1}_{[\![1,m]\!]} \cdot v > \delta^* \cdot v$), contredisant le fait que δ^* soit optimale.
  - $\hat{\delta}$ est alors une solution vérifiant $\hat{\delta} \cdot v = (\delta^* + \frac{\varepsilon}{p_m}\mathbb{1}_m) \cdot v = \delta^* \cdot v + \frac{v_m}{p_m}\varepsilon = \delta^* \cdot v + v_m \overbrace{(w - \delta^*_m)}^{>0 \text{ car } \delta^*_m < w} > \delta^* \cdot v$, ce qui contredit l'optimalité de δ^*.
- ▷ Si $\delta^*_m > v$, on a alons $p \cdot \delta^* > P$, ce qui est absurde car $\delta^* \in \mathscr{S}$.

Propriété: *Sous les hypothèses et les notations de la propriété précédente,* $\mathrm{argmax}_{\delta \in \mathscr{S}}\, v \cdot \delta$ *est réduite à la solution donnée par* $M = \min\{i \in [\![1, m]\!] \mid \sum_{k=1}^i p_k > P\}$

- ▷ Soit $\delta^* \in \mathrm{argmax}_{\delta \in \mathscr{S}}\, \delta \cdot v$. Soit $m = \min\{i \in [\![1, n]\!] \mid \delta^*_i < 1\}$. On veut montrer que $m = M$. Puisque $\delta^* \in \mathscr{S}, \delta^* \cdot p \leq P$ soit $\sum_{k=1}^m \delta^*_k p_k \leq P$. soit $\delta^*_m p_m + \sum_{k=1} m - 1\delta_k p_k \leq P$ donc $\sum_{k=1}^{i=1}$

3 L'algorithme des k plus proches voisins

TRI	Entrée:	$(x_i)_{i \in [\![1,n]\!]} \in X^n$ où $(X, \leq)$ est un ensemble totalement ordonné.
	Sortie:	$\sigma \in \mathscr{S}$ tel que $(x_{\sigma(i)})_{i \in [\![1,n]\!]})$ est croissante pour $\leq$.

Remarque: $\min\limits_{\sigma \in \mathfrak{S}_n} \left(\sum_{i=1}^n \max\{x_{\sigma(i)} - x_{\sigma(i+1)}, 0\}\right) = 0.$

3.1 Tri par insertion

Tri-sélection

Entrée: (x_i) une famille d'éléments comparables avec $\leq$ en $\theta(1)$, rangés dans un tableau.

Sortie: $\sigma \in \mathscr{S}$ tel que $(x_{\sigma(i)})_{i\in[\![1,n]\!]})$ est croissante pour $\leq$.

```
Soit $T$ une copie du tableau (i.e $\forall i \in [\![1,n]\!], T[i] = x_i$)
Soit $I$ le tableau identité de $[1..n]$
Soit $\sigma$ un tableau d'entiers indicé par $[\![1,n]\!]$ initialisé à 0
Pour $i$ allant de 1 à $n$:
                    Invariant: $\forall i \in [\![1,n]\!], T[i] = x_{I(i)}$ et $\begin{cases} T[1..k-1] \text{est trié} \\ \forall i \in [\![k,n]\!] T[i] \geq \max\{T[j] \mid j \in [\![1,k-1]\!]\} \\ \forall i \in [\![1,k-1]\!], T[i] = x_{\sigma[i]} \end{cases}$
    Trouver $i_0 \in \mathrm{argmin}_{i\in[\![k,n]\!]} T[i]$
    $\sigma[k] \to i_0$
    Échanger $T[k]$ et $T[i_0]$
Renvoyer $\sigma$
```

Implémentation en Ocaml :

```ocaml
let tri_selec (t_init : 'a array) : int array =
  (* hypothèse : Array.length t_init >0 *)
  (* calcule une permutation qui trie les valeurs de t_init*)

  let n =  Array.length t_init in
  let t = Array.make n t_init.(0) in
  let sigma = Array.make n 0 in
  for i = 0 to (n-1) do
    t.(i) <- t_init.(i) ;
    sigma.(i) <- i
  done;
  (* à ce stade t est une copie de t_init et sigma est l'identité*)

  (* invariant : sigma est une permutation,
     et pour i de k à n t.(sigma.(i))=t_init.(i)
     et t_init.(sigma.(i)) pour i de 1 à k-1 est croissante
     et si k>1, pour tout i de k à n, t.(k-1) <= t.(i) *)
  for k=0 to n-1 do
    (* trouvons i0 l'argmin de t[k..n]*)
    let i0 = ref k in
    for j = k+1 to n-1 do
      if t.(j) < t.(!i0) then i0:=j else()
    done;
    (* on veut retenir ds sigma.(k) l'indice ds t_init de la valeur t.(i0),
       retenir  ds sigma.(k) = sigma.(i0)
       de plus on veut séparer cette valeur des valeurs restant à trier dans t
       donc on échange  sigma.(i0) et sigma.(k) et  t.(i0) et t.(k)*)
    let temp = t.(k) in t.(k) <- t.(!i0) ; t.(!i0) <- temp;
    let temp = sigma.(k) in sigma.(k) <- sigma.(!i0); sigma.(!i0) <- temp;
  done;
  sigma

```

```
let test_tri_selec :unit =
  assert(tri_selec [|1;2;3|] = [|0;1;2|]);
  assert(tri_selec [|4;2;3|] = [|1;2;0|]);
  assert(tri_selec [|'a';'d';'c';'e';'f'|] = [|0;2;1;3;4|])
```

Propriété: *Soit I un ensemble fini non vide de cardinal n (typiquement $[\![1,n]\!]$ ou $[\![0,n-1]\!]$). Soit $(x_i)_{i\in I} \in X^I$ où $(X,\leq)$ est un ensemble totalement ordonné. Soit $i_1 \in \text{argmin}_{i\in I}\, x_i$. On note $\tilde{I} = I \setminus \{i_1\}$. Si $\tilde{\sigma} \in Bij([\![1,n-1]\!],\tilde{I})$ est telle que $(x_{\tilde{\sigma}(i)})$ est croissante, alors le prolongement σ de $\tilde{\sigma}$ défini ci-dessous est une bijection telle que $(x_{\sigma(i)})_{i\in[\![1,n]\!]}$ est croissante.*

$$\sigma = \begin{pmatrix} [\![1,n]\!] & \to & I \\ 1 & \mapsto & i_1 \\ i\geq 2 & \mapsto & \tilde{\sigma}(i-1) \end{pmatrix}$$

▷ Comme $\tilde{\sigma}$ est bijective, on a card $\tilde{I}$ = card $[\![1,n-1]\!] = n-1$. Par ailleurs, comme $i_1 \notin \tilde{I}$, card I = card $\tilde{I}\cup\{i_1\}$ = card $\tilde{I}+1 = n$.

▷ Par définition de σ, $\sigma(1) = i_1$ et $\sigma([\![2,n]\!]) = \tilde{\sigma}([\![1,n-1]\!])$, or comme $\tilde{\sigma}$ est en particulier surjective, $\tilde{\sigma}([\![1,n-1]\!]) = \tilde{I}$. On a donc $\sigma(I) = \tilde{I}\cup\{i_1\} = I$, autrement dit σ est surjective. σ est alors une application surjective entre deux ensembles de même cardinal fini, σ est donc bijective.

▷ Soit $(i,j)\in[\![1,n]\!]^2$ tel que $i<j$. Si $i=1$, nécessairement $j\in[\![2,n]\!]$, donc $\sigma(i)=i_1$ et $\sigma(j) = \tilde{\sigma}(j-1)\in\tilde{I}$ donc $x_{\sigma(i)} = x_{i_1} \leq x_{\sigma(j)}$ par définition de i_1 comme argmin. Sinon, $(i,j)\in[\![2..n]\!]^2$ donc $\sigma(i) = \tilde{\sigma}(i-1)$ et $\sigma(j) = \tilde{\sigma}(j-1)$, or par croissance de $(x_{\tilde{\sigma}(i)})_{i\in[\![1,n-1]\!]}$, on a $x_{\tilde{\sigma}(i-1)}$ et $x\tilde{\sigma}(j-1)$ donc $x\sigma(i)\leq x\sigma(j)$. Dans les deux cas on a $x_{\sigma(i)}\leq x_{\sigma(j)}$. On en déduit que $(x_{\sigma(i)})_{i\in[\![1,n]\!]}$ est croissante.

3.2 Autres exemples d'algorithmes gloutons

- L'algorithme de Huffman, qui permet de générer un codage de score minimal (voir DM n°3)

4 Programmation dynamique

4.1 Le problème du sac à dos entier

Voir TP.

4.2 Plus long sous-mot commun

Soit Σ un alphabet (i.e un ensemble fini de symboles). Soit $(u,v)\in\Sigma^*\times\Sigma^*$. Soit $n=|u|$ et $m=|v|$. On dit que u est un **sous-mot** de v, et on note $u\leq v$ ssi il existe une extractrice $\varphi\in\mathcal{F}([\![1,n]\!],[\![1,m]\!])$ strictement croissante telle que $u = v_{\varphi(1)}v_{\varphi(2)}...v_{\varphi(n)}$, i.e $\forall i\in[\![1,n]\!], u_i = v_{\varphi(i)}$.

Lemme: *Soit $(u,v)\in\Sigma^*\times\Sigma^*$ de longueurs respectives n et m. Si u* **Propriété:** *Soit $(u,v\in\Sigma^*\times\Sigma^*)$, de longueur n et m. Pour $(i,j)\in[\![1,n]\!]\times[\![1,m]\!]$, on note $\mathscr{L}(i,j) = \max\{|w|\mid w\in\Sigma^*, w\leq u_1...u_i, w\leq v_1...v_j\}$. On a alors :*

1. $\forall i\in[\![0,n]\!], \mathscr{L}(i,0) = |\varepsilon| = 0$
2. $\forall j\in[\![0,n]\!], \mathscr{L}(0,j) = 0$
3. $\forall (i,j)\in[\![1,n]\!]\times[\![1,m]\!], \mathscr{L}(i,j) = \begin{cases}\mathscr{L}(i-1,j-1)+1 \text{ si } u_i = v_j\\ \max(\mathscr{L}(i-1,j),\mathscr{L}(j-1,i))\end{cases}$

▷ Soit $i\in[\![0,n]\!]$.

$$\mathscr{L}(i,0) = \max\{|w|\mid w\in\Sigma^*, w\leq u_1...u_i, w\leq\varepsilon\} \leq \max\{|w|\mid w\in\Sigma^*, w\leq\varepsilon\} = |\varepsilon| = 0$$

Or $\varepsilon\leq u_1...u_i$ et $\varepsilon\leq\varepsilon$ donc $|\varepsilon|\leq\mathscr{L}(1,0)$, soit $0\leq\mathscr{L}(i,0)$ d'où $\mathscr{L}(i,0)=0$ par double inégalité. On prouve (2) de la même façon.

▷ Soit $(i, j) \in [\![1, n]\!] \times [\![1, m]\!]$. Supposons que $u_i = v_j$.

$$\begin{aligned}\mathscr{L}(i,j) &= \max\{|w| \mid w \in \Sigma^*, w \leq u_1u_2...u_{i-1}u_i, w \leq v_1v_2...v_{j-1}v_j\} \\ &\geq \max\{|w| \mid w \in \Sigma^*, w_{|w|} = u_i, w \leq u_1u_2...u_{i-1}u_i, w \leq v_1v_2...v_{j-1}v_j\} \\ &= \max\{|x| + 1 \mid x \in \Sigma^*, x \leq u_1...u_{i-1}, x \leq v_1...v_{j-1}\} = \mathscr{L}(i-1, j-1) + 1\end{aligned}$$

Inversement, montrons que $\mathscr{L}(i,j) - 1 \leq \mathscr{L}(i-1, j-1)$. Il existe $w \in \mathscr{S}_{i,j}$ tq $|w| = \mathscr{L}(i,j)$. Si $w = \varepsilon$, alors $\mathscr{L}(i,j) = |\varepsilon| = 0$, or $u_i = v_j \in \mathscr{S}_{i,j}$ donc $\mathscr{L}(i,j) \geq |u_1| = 1$, ce qui est absurde. Plaçons-nous maintenant dans le cas contraire. On pose $p = |w|$ et $x = w_1w_2...w_{p-1}$. Alors $|x| = p - 1 = |w| - 1$. Montrons que $x \in \mathscr{S}_{i-1,j-1}$.

- Si $w_p = u_i = v_j$, alors $x \leq u_1, ...u_{i-1}$ et $x \leq v_1...v_{j-1}$ par identification.
- Si $w_p \neq u_i$, alors d'après le lemme $w \leq u_1...u_{i-1}$ et $w \leq v_1...v_{j-1}$. De plus, comme $x \leq w$, par transitivité on a $x \leq u_1...u_{i-1}$ et $x \leq v_1...v_{j-1}$.
- Dans les deux cas, $x \in \mathscr{S}_{i-1,j-1}$ donc $|x| \leq \mathscr{L}(i-1, j-1)$ soit $\mathscr{L}(i,j) - 1 = |w| - 1 = |x| \leq \mathscr{L}(i-1, j-1)$