Recherche textuelle

DIU Enseignement de l'informatique au lycée
Bloc 5 : Algorithmique avancée

Bruno Grenet – Université de Montpellier

Juin 2021

Problème

Recherche de motif

Entrées : Un texte t de longueur n, un motif x de longueur $m < n$

Sortie : Toutes les positions de x dans t

Problème

Recherche de motif

Entrées : Un texte t de longueur n, un motif x de longueur $m < n$

Sortie : Toutes les positions de x dans t

$$abaa \quad \text{dans} \quad acaabbabaaa \ ?$$

Un premier algorithme

RechercheNaïve

Entrées : t et x

1. $P \leftarrow$ liste vide
2. Pour $i = 0$ à $n - m$:
3. $\quad j \leftarrow m - 1$
4. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
5. $\quad\quad j \leftarrow j - 1$
6. $\quad$ Si $j = -1$: Ajouter i à P
7. Renvoyer P

$x = abaa$

$a\ c\ a\ a\ b\ b\ a\ b\ a\ a\ a$

Un premier algorithme

RechercheNaïve

Entrées : t et x

1. $P \leftarrow$ liste vide
2. Pour $i = 0$ à $n - m$:
3. $\quad j \leftarrow m - 1$
4. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
5. $\quad\quad j \leftarrow j - 1$
6. $\quad$ Si $j = -1$: Ajouter i à P
7. Renvoyer P

$x = abaa$

$a\ c\ a\ a\ b\ b\ a\ b\ a\ a\ a$

Théorème

L'algorithme RechercheNaïve renvoie la liste des positions de x dans t en temps $O(mn)$.

Notations

- Longueur : $|w|$
- Lettres : $w_0, \ldots, w_{|w|-1}$
- Fenêtre de taille m en position i : $w_{[i,i+m[} = w_i w_{i+1} \cdots w_{i+m-1}$
- x apparaît en position i dans t : $x = t_{[i,i+|x|[}$

La règle du mauvais caractère : exemple

$x = abaa$

$a\ c\ a\ a\ b\ b\ a\ b\ a\ a\ a$

Dernière occurence non finale

$d_x(\ell)$: position de la dernière occurrence de ℓ dans x, non finale

Dernière occurence non finale

$d_x(\ell)$: position de la dernière occurrence de ℓ dans x, non finale

Exemple : $x = abaabcab$

ℓ	a	b	c	d	$\ldots$
$d_x(\ell)$	6	4	5	-1	-1

Règle du mauvais caractère

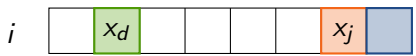
Théorème

Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.

Règle du mauvais caractère

Théorème

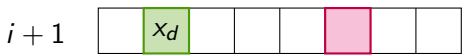
Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.



Règle du mauvais caractère

Théorème

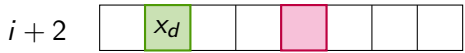
Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.



Règle du mauvais caractère

Théorème

Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.



Règle du mauvais caractère

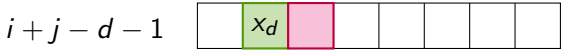
Théorème

Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.

Règle du mauvais caractère

Théorème

Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.



Règle du mauvais caractère

Théorème

Soit $t_{[i,i+m[}$ la fenêtre courante, avec $x_j \neq t_{i+j}$ et $x_k = t_{i+k}$ pour $k > j$ et $d = d_x(t_{i+j})$. Alors si $j > d$, $x \neq t_{[i_0,i_0+m[}$ pour $i < i_0 < i + j - d$.

Algorithme de Boyer et Moore simplifié – version 1

BoyerMoore-MC

Entrées : t, x et d_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad i \leftarrow i +$ $\max(1, j - d_x(t_{i+j}))$
10. Renvoyer P

Algorithme de Boyer et Moore simplifié – version 1

BoyerMoore-MC

Entrées : t, x et d_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad i \leftarrow i +$ $\max(1, j - d_x(t_{i+j}))$
10. Renvoyer P

Théorème

L'algorithme BoyerMoore-MC renvoie la liste des positions de x dans t en temps $O(mn)$.

Algorithme de Boyer et Moore simplifié – version 1

BoyerMoore-MC

Entrées : t, x et d_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad i \leftarrow i +$ $\max(1, j - d_x(t_{i+j}))$
10. Renvoyer P

Théorème

L'algorithme BoyerMoore-MC renvoie la liste des positions de x dans t en temps $O(mn)$.

- même pire cas que la recherche naïve :
  - motif $ba \cdots a$
  - texte $a \cdots\cdots\cdots a$

Algorithme de Boyer et Moore simplifié – version 1

BoyerMoore-MC

Entrées : t, x et d_x

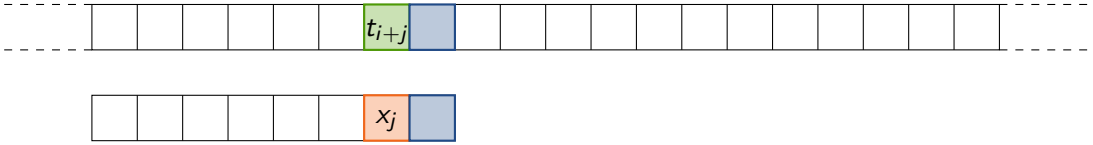
1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad i \leftarrow i +$ $\max(1, j - d_x(t_{i+j}))$
10. Renvoyer P

Théorème

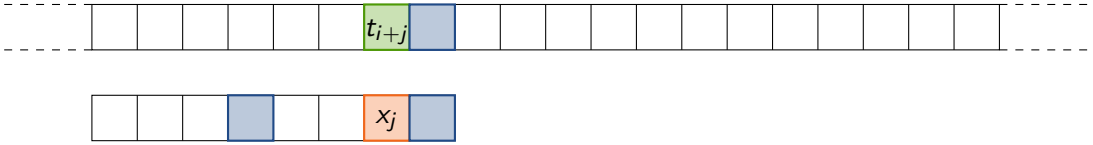
L'algorithme BoyerMoore-MC renvoie la liste des positions de x dans t en temps $O(mn)$.

- même pire cas que la recherche naïve :
  - motif $ba \cdots a$
  - texte $a \cdots\cdots\cdots a$
- *entrée supplémentaire* : tableau d des dernières occurences non finales
  - exercice !

Variantes

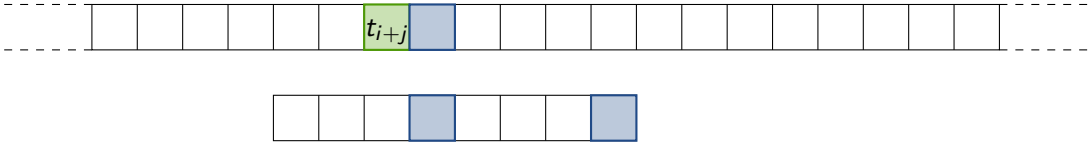


Variantes



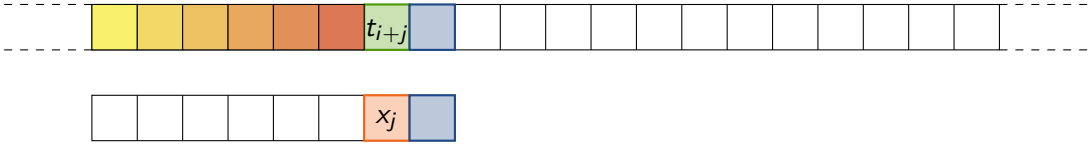
Variantes

Variantes



- Algorithme de Horspool : alignement de la dernière lettre de la fenêtre $O(mn)$

Variantes



- Algorithme de Horspool : alignement de la dernière lettre de la fenêtre $O(mn)$
- Alignement le plus à droite possible, en tenant compte de toutes les lettres $O(mn)$

Règle du bon suffixe : exemple

$x = abaaaa$

a	b	b	c	a	a	c	a	a	a	a	b	a	a	a	a

Règle du bon suffixe : p_x et s_x

- $p_x(j)$: longueur du plus long préfixe de x ($\neq x$) qui soit aussi suffixe de $x_{[j,m[}$
- $s_x(j)$: position la plus à droite d'une copie de $x_{[j,m[}$, non précédée de x_{j-1}

Règle du bon suffixe : p_x et s_x

- $p_x(j)$: longueur du plus long préfixe de x ($\neq x$) qui soit aussi suffixe de $x_{[j,m[}$
- $s_x(j)$: position la plus à droite d'une copie de $x_{[j,m[}$, non précédée de x_{j-1}

Exemple

x	b	a	b	a	b	a
j	0	1	2	3	4	5
$p_x(j)$	4	4	4	2	2	0
$s_x(j)$	-1	-1	0	-1	0	-1

Règle du bon suffixe : p_x et s_x

- $p_x(j)$: longueur du plus long préfixe de x ($\neq x$) qui soit aussi suffixe de $x_{[j,m[}$
- $s_x(j)$: position la plus à droite d'une copie de $x_{[j,m[}$, non précédée de x_{j-1}

Exemple

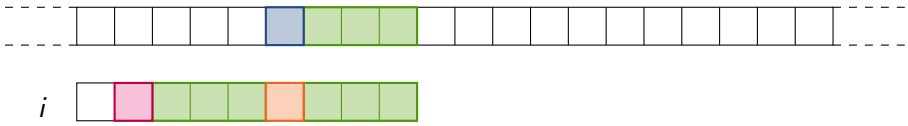
x	b	a	b	a	b	a
j	0	1	2	3	4	5
$p_x(j)$	4	4	4	2	2	0
$s_x(j)$	-1	-1	0	-1	0	-1

x	a	b	b	a	a	b	a	b	a
j	0	1	2	3	4	5	6	7	8
$p_x(j)$	1	1	1	1	1	1	1	1	1
$s_x(j)$	-1	-1	-1	-1	-1	-1	4	2	4

Règle(s) du bon suffixe (1/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) \geq 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + j + 1 - s_x(j)$.



Règle(s) du bon suffixe (1/2)

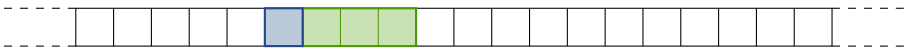
Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) \geq 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + j + 1 - s_x(j)$.

Règle(s) du bon suffixe (1/2)

Théorème

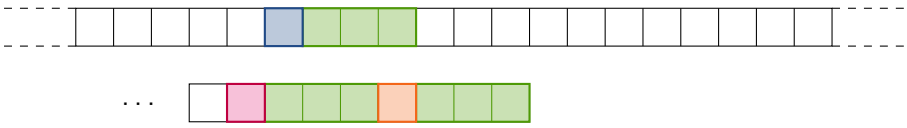
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) \geq 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + j + 1 - s_x(j)$.



Règle(s) du bon suffixe (1/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) \geq 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + j + 1 - s_x(j)$.



Règle(s) du bon suffixe (1/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) \geq 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + j + 1 - s_x(j)$.

$i + j + 1 - s$

Règle(s) du bon suffixe (2/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.

Règle(s) du bon suffixe (2/2)

Théorème

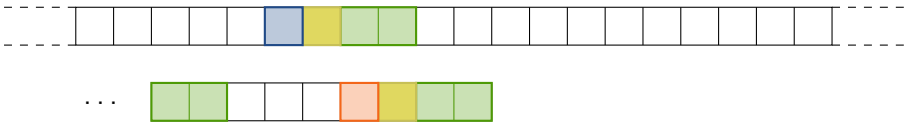
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.

$i+1$ 

Règle(s) du bon suffixe (2/2)

Théorème

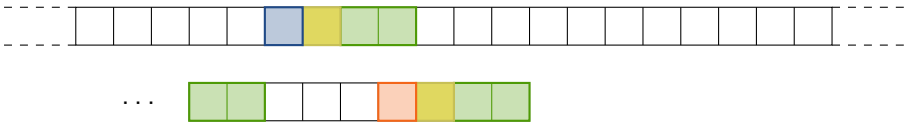
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.



Règle(s) du bon suffixe (2/2)

Théorème

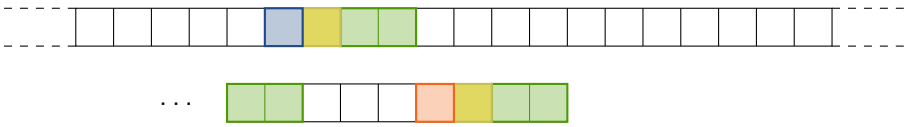
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.



Règle(s) du bon suffixe (2/2)

Théorème

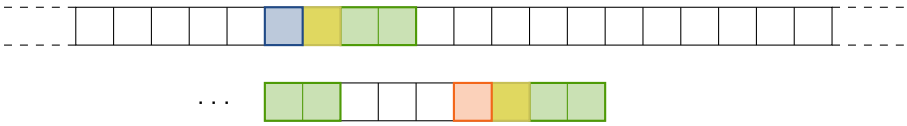
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.



Règle(s) du bon suffixe (2/2)

Théorème

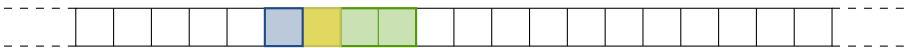
Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.



Règle(s) du bon suffixe (2/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.



$i + m - p - 1$

Règle(s) du bon suffixe (2/2)

Théorème

Soit $t_{[i,i+m[}$ *la fenêtre courante, avec* $x_j \neq t_{i+j}$ *et* $x_{[j+1,m[} = t_{[i+j+1,i+m[}$.
Si $s_x(j) < 0$, $x \neq t_{[i_0,i_0+m[}$ *pour* $i < i_0 < i + m - p_x(j)$.

$i + m - p$

Algorithme de Boyer et Moore simplifié – version 2

BoyerMoore-BS

Entrées : t, x, p_x et s_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad\quad i \leftarrow i + m - p_x(1)$
10. $\quad$ Sinon si $s_x(j+1) \geq 0$:
11. $\quad\quad i \leftarrow i + j + 1 - s_x(j+1)$
12. $\quad$ Sinon : $i \leftarrow i + m - p_x(j)$
13. Renvoyer P

Algorithme de Boyer et Moore simplifié – version 2

BoyerMoore-BS

Entrées : t, x, p_x et s_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad\quad i \leftarrow i + m - p_x(1)$
10. $\quad$ Sinon si $s_x(j+1) \geq 0$:
11. $\quad\quad i \leftarrow i + j + 1 - s_x(j+1)$
12. $\quad$ Sinon : $i \leftarrow i + m - p_x(j)$
13. Renvoyer P

Théorème

L'algorithme BoyerMoore-BS renvoie la liste des positions de x dans t en temps $O(mn)$.

Algorithme de Boyer et Moore simplifié – version 2

BoyerMoore-BS

Entrées : t, x, p_x et s_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad\quad i \leftarrow i + m - p_x(1)$
10. $\quad$ Sinon si $s_x(j+1) \geq 0$:
11. $\quad\quad i \leftarrow i + j + 1 - s_x(j+1)$
12. $\quad$ Sinon : $i \leftarrow i + m - p_x(j)$
13. Renvoyer P

Théorème

L'algorithme BoyerMoore-BS renvoie la liste des positions de x dans t en temps $O(mn)$.

- même pire cas que la recherche naïve

Algorithme de Boyer et Moore simplifié – version 2

BoyerMoore-BS

Entrées : t, x, p_x et s_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad\quad i \leftarrow i + m - p_x(1)$
10. $\quad$ Sinon si $s_x(j+1) \geq 0$:
11. $\quad\quad i \leftarrow i + j + 1 - s_x(j+1)$
12. $\quad$ Sinon : $i \leftarrow i + m - p_x(j)$
13. Renvoyer P

Théorème

L'algorithme BoyerMoore-BS renvoie la liste des positions de x dans t en temps $O(mn)$.

- même pire cas que la recherche naïve
- tableaux s et p
  - calcul en temps $O(m^2)$ (exercice !)
  - difficile : temps linéaire $O(m)$

Algorithme de Boyer et Moore – version complète !

BoyerMoore

Entrées : t, x, d_x, p_x et s_x

1. $P \leftarrow$ liste vide
2. $i \leftarrow 0$
3. Tant que $i \leq n - m$:
4. $\quad j \leftarrow m - 1$
5. $\quad$ Tant que $j \geq 0$ et $x_j = t_{i+j}$:
6. $\quad\quad j \leftarrow j - 1$
7. $\quad$ Si $j = -1$:
8. $\quad\quad$ Ajouter i à P
9. $\quad\quad i \leftarrow i + m - p_x(1)$
10. $\quad$ Sinon :
11. $\quad\quad$ Si $s_x(j+1) \geq 0$: $i \leftarrow i + \max(j + 1 - s_x(j+1), j - d_x(t_{i+j}))$
12. $\quad\quad$ Sinon : $i \leftarrow i + \max(m - p_x(j), j - d_x(t_{i+j}))$
13. Renvoyer P

Propriétés de l'algorithme de Boyer et Moore

Théorème

L'algorithme BoyerMoore renvoie la liste de toutes les positions de x dans t en temps $O(mn)$.

Propriétés de l'algorithme de Boyer et Moore

Théorème

L'algorithme BoyerMoore renvoie la liste de toutes les positions de x dans t en temps $O(mn)$.

- **Tout ça pour ça !**

Propriétés de l'algorithme de Boyer et Moore

Théorème

L'algorithme BoyerMoore renvoie la liste de toutes les positions de x dans t en temps $O(mn)$.

- **Tout ça pour ça !**

- Très rapide en pratique
- Complexité est $O(m+n)$ dans bcp de situations
- Existe une variante en complexité $O(m+n)$ dans tous les cas

Conclusion

- Autres algorithmes de complexité $O(m+n)$:
  - Knuth-Morris-Pratt (automates finis)
  - Karp-Rabin (technique probabiliste)
- Extensions :
  - Expressions régulières (`grep`, etc.) $a^*bc^*.^*a$
  - Mots bidimensionnels (plus connus sous le nom d'*images*)
  - Motifs non contigus $a\_\_\_\_\_b\_\_\_\_\_a$
- Utilisation :
  - Ctrl + F dans bcp de logiciels
  - Détection de gênes dans une séquence ADN
  - ...