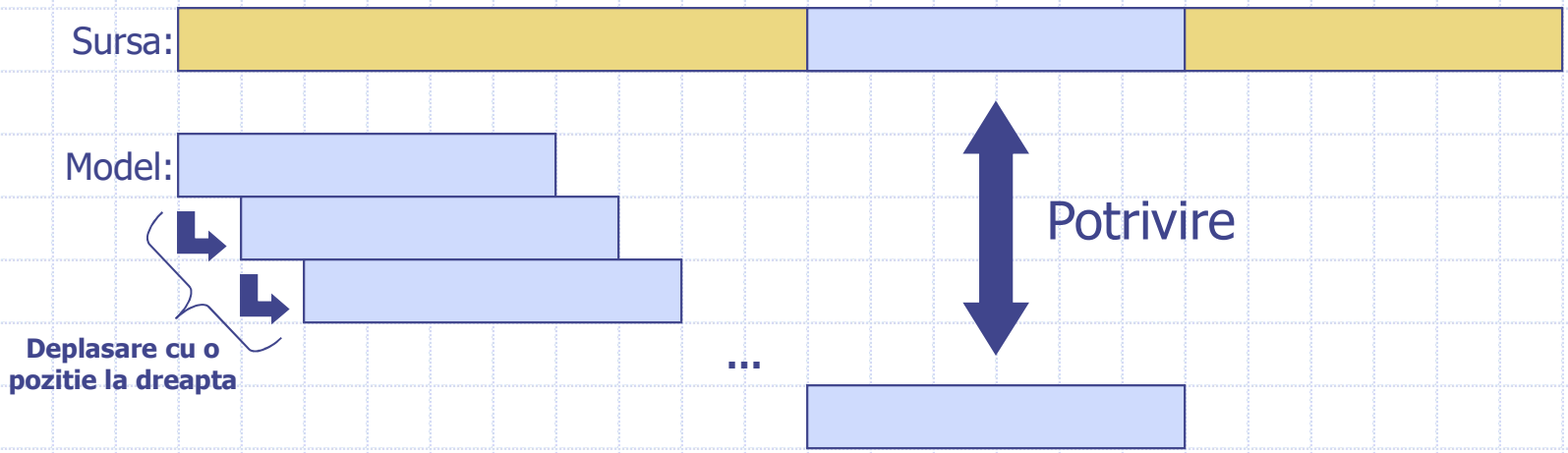
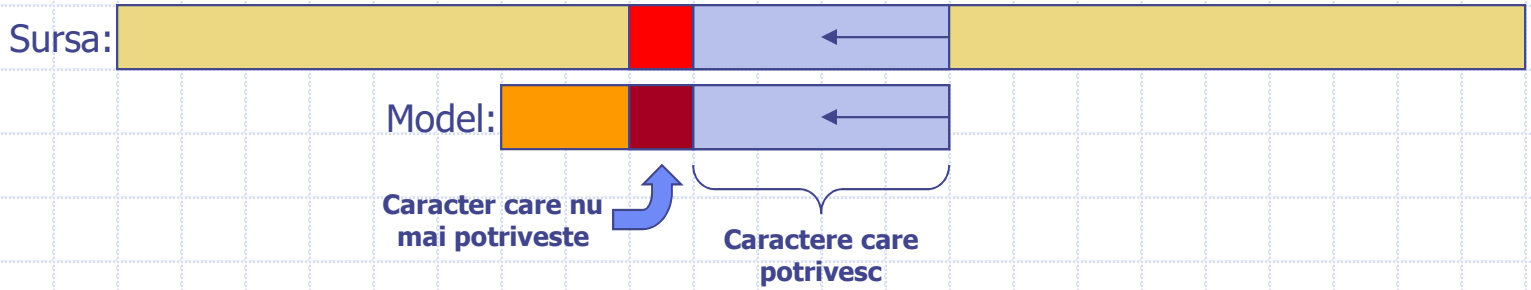
Algoritmul Boyer-Moore

- Algoritmul imaginat de Boyer si Moore reprezinta o tehnica avansata de cautare a unui sir de caractere model (pattern) intr-un sir de caractere sursa



Algoritmul Boyer-Moore

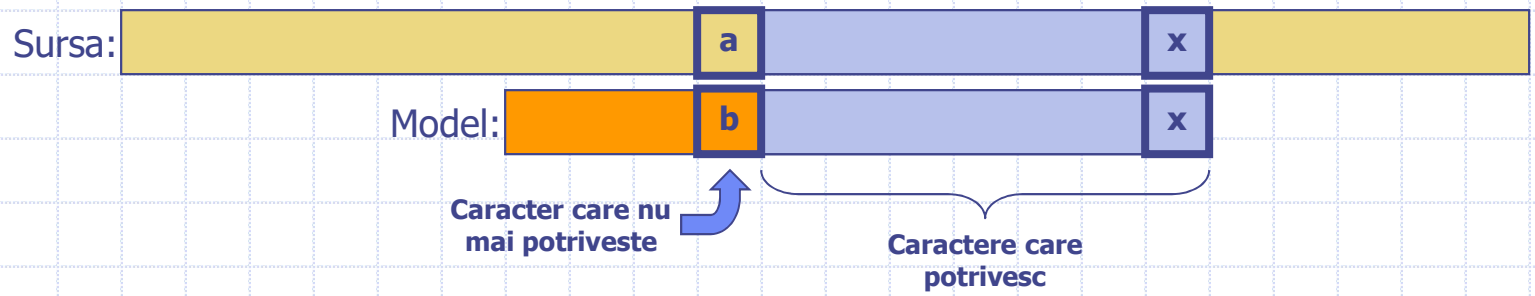
- Spre deosebire de varianta clasica, in care modelul este deplasat de-a lungul sursei cu cate o pozitie pana la gasirea unei potriviri, algoritmul Boyer-Moore incearca deplasarea modelului cu un numar mai mare de pozitii in momentul depistarii unei nepotriviri



- In figura de mai sus, in momentul depistarii nepotrivirii, algoritmul detine informatii despre caracterele din portiunea de culoare albastru-deschis, atat din sursa cat si din model
- Aceste informatii ar putea fi folosite pentru mutarea "inteligenta" a modelului spre dreapta cu mai mult de o pozitie
- Dupa cum se observa, depistarea unei nepotriviri se face examinand modelul de la dreapta la stanga, nu de la stanga la dreapta

Algoritmul Boyer-Moore

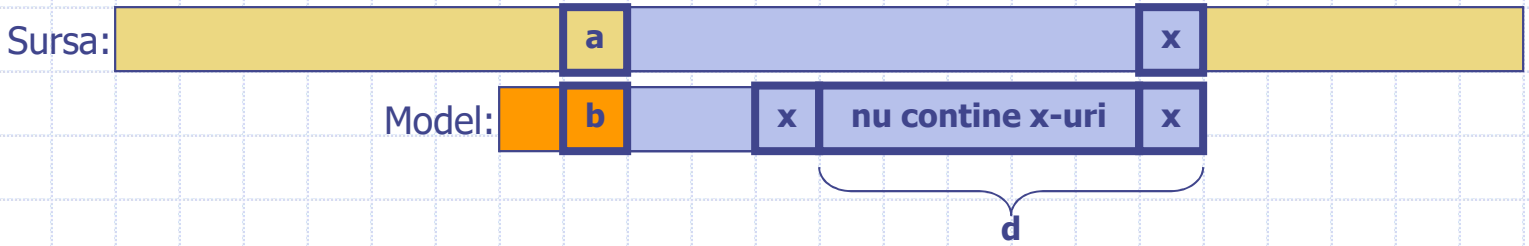
- De fiecare data, in cazul unei nepotriviri intre sursa si model, situatia se va prezenta ca in figura de mai jos:



- O conditie necesara este **a** != **b**
- Algoritmul are mai multe variante, in varianta cea mai simpla deplasarea modelului spre dreapta facandu-se cu atatea pozitii incat sub caracterul **x** din sursa (vezi figura) sa se pozitioneze urmatorul caracter **x** din model
- Daca un astfel de caracter **x** nu exista, modelul se va deplasa spre dreapta cu intreaga sa lungime

Algoritmul Boyer-Moore

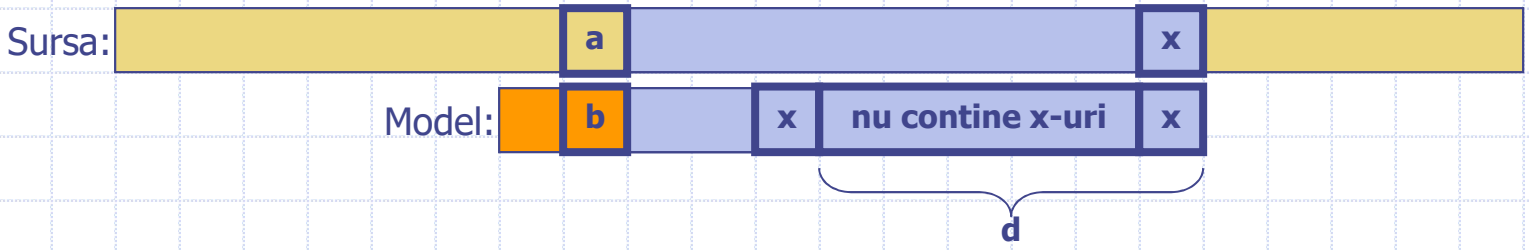
- De exemplu, in cazul de mai jos deplasarea modelului se va face pe distanta **d**



- Indiferent unde a aparut nepotrivirea intre sursa si model, se ia in considerare caracterul din sursa corespunzator ultimului caracter al modelului – acesta este caracterul **x** din figura
- Se parcurge modelul ***de la dreapta la stanga*** si se cauta prima aparitie a lui **x**
- Exceptia o constituie situatia in care **x** este chiar la sfarsitul modelului – in acest caz se cauta a doua aparitie a lui **x** in model
- Se muta modelul spre dreapta astfel incat **x**-ul gasit in model sa ajunga sub cel din sursa
- Daca nu este gasit un **x** corespunzator in model, modelul va fi deplasat cu intreaga sa lungime spre dreapta

Algoritmul Boyer-Moore

- Justificarea celor enuntate anterior este simpla



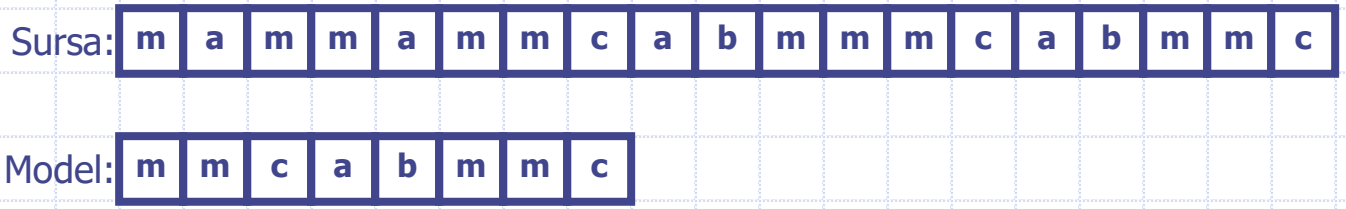
- Daca mutam modelul cu mai putin de **d** pozitii spre dreapta, sub caracterul **x** din sursa (vezi figura) va ajunge un caracter diferit de **x** din model, deci nu se poate pune problema unei potriviri
- Daca modelul nu mai contine nici un caracter **x**, atunci orice deplasare mai scurta decat lungimea modelului va pozitiona un caracter al modelului (diferit de **x**) sub **x**-ul din sursa, deci, din nou nu se poate pune problema unei potriviri

Algoritmul Boyer-Moore

- Evident, exista o varianta mult mai buna decat cautarea in model a caracterului **x** la fiecare nepotrivire intre sursa si model
- Deoarece exista un numar mic de caractere posibile (256, daca vorbim de setul ASCII), modelul ar putea fi precompilat inainte de inceperea algoritmului, astfel incat sa obtinem un tabel **tabDepl:array[char] of integer** (in notatie Pascal), construit dupa urmatoarele reguli:
  - **tabDepl[c]** = distanta dintre ultima aparitie a lui **c** in model si sfarsitul modelului, daca **c** apare in model, dar nu pe ultima pozitie
  - **tabDepl[c]** = distanta dintre penultima aparitie a lui **c** in model si sfarsitul modelului, daca **c** apare in model de mai multe ori, inclusiv pe ultima pozitie
  - **tabDepl[c]** = lungimea modelului, daca **c** apare in model **numai** pe ultima pozitie
  - **tabDepl[c]** = lungimea modelului, daca **c** nu apare in model
- La depistarea unei nepotriviri, modelul se va deplasa spre dreapta cu **tabDepl[x]**, unde **x** este caracterul din sursa corespunzator ultimului caracter al modelului
- Daca modelul a fost precompilat, lungimea deplasarii poate fi aflata in O(1), fiind **tabDepl[x]**

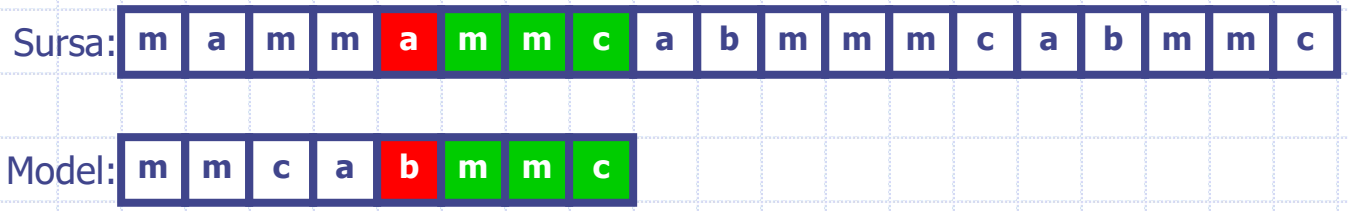
Algoritmul Boyer-Moore

- Vom studia mersul algoritmului pentru urmatoarea situatie:



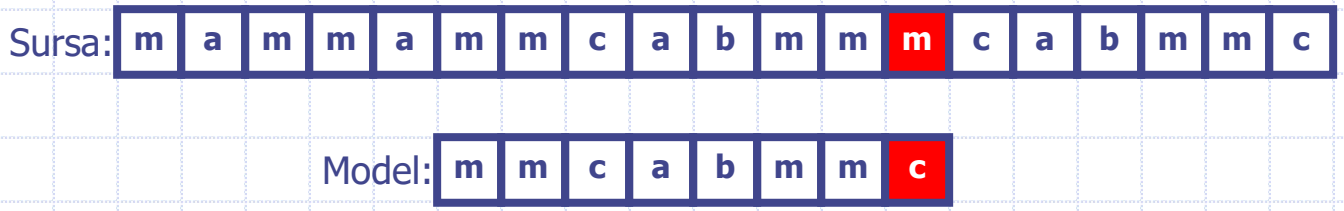
- Precompiland modelul dupa regulile enuntate, obtinem:
  - tabDepl[‘a’] = 4
  - tabDepl[‘b’] = 3
  - tabDepl[‘c’] = 5
  - tabDepl[‘m’] = 1
  - tabDepl[<orice altceva>] = 8

Algoritmul Boyer-Moore



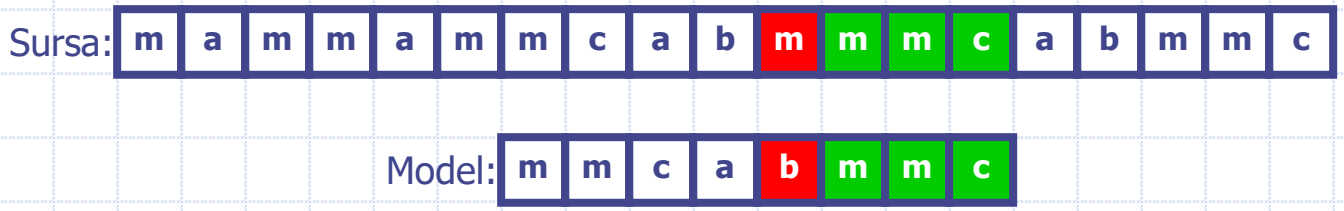
- Prima nepotrivire apare intre caracterele ‘a’ si ‘b’
- **x** = ‘c’
- **tabDepl[x]** = 5
- Vom muta modelul spre dreapta cu 5 pozitii

Algoritmul Boyer-Moore



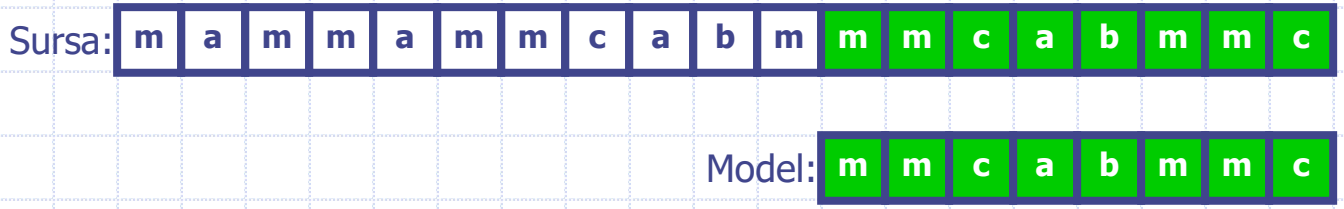
- Prima nepotrivire apare intre caracterele ‘m’ si ‘c’
- **x** = ‘m’
- **tabDepl[x]** = 1
- Vom muta modelul spre dreapta cu 1 pozitie

Algoritmul Boyer-Moore



- Prima nepotrivire apare intre caracterele ‘m’ si ‘b’
- **x** = ‘c’
- **tabDepl[x]** = 5
- Vom muta modelul spre dreapta cu 5 pozitii

Algoritmul Boyer-Moore



- S-a gasit o potrivire intre sursa si model
- In acest moment, algoritmul se poate incheia, sau poate continua pentru gasirea altor potriviri
- Daca se decide continuarea, se va gasi **x** = 'c' ceea ce va impune o deplasare a modelului spre dreapta cu 5 pozitii si reluarea procesului
- In cazul prezentat, continuarea nu va duce la gasirea altor potriviri, deoarece am ajuns la sfarsitul sirului sursa

Algoritmul Boyer-Moore

- Varianta prezentata este una dintre cele mai simple ale algoritmului, care deplaseaza modelul in cazul unei nepotriviri folosind o tehnica numita “bad character shift”
- Exista variante mai complexe care folosesc si o alta tehnica de deplasare in caz de nepotrivire numita “good suffix shift”, pe langa “bad character shift”
- Tehnica “good suffix shift” este oarecum similara cu tehnica folosita de algoritmul Knuth-Morris-Pratt pentru a deplasa modelul
- In combinatie, cele 2 tehnici fac ca algoritmul Boyer-Moore sa fie cel mai competitiv algoritm de cautare de siruri, cazul cel mai favorabil ajungand la O(N/M), unde M este lungimea sirului model iar N este lungimea sirului sursa