Institut for Matematik og Datalogi
Syddansk Universitet, Odense

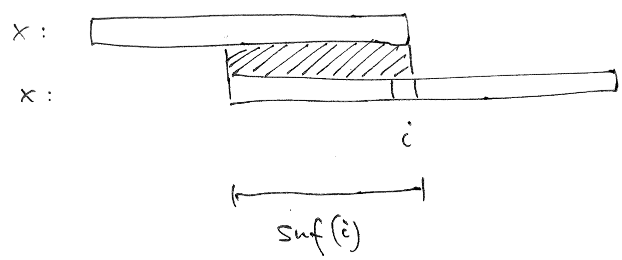
28. september 2016
RF

Finding the BM Shift Table

For a string x of length m we for $-1 \leq i \leq m-1$ define the value $\mathrm{suf}(i)$ by

$$\mathrm{suf}(i) = |\,\mathrm{lcs}(x, x[0..i])|,$$

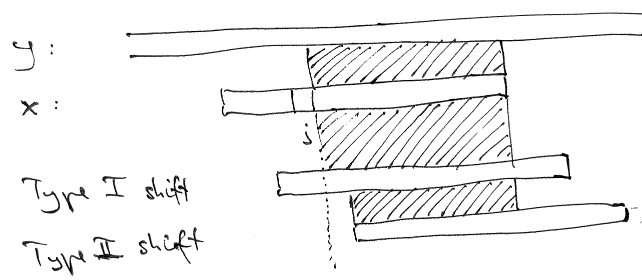
where $\mathrm{lcs}(x, y)$ denotes the longest common suffix of the two strings x and y. The following picture illustrates the definition.



Since the value of $\mathrm{suf}(i)$ is the same as $\mathrm{pref}(m-1-i)$ for the reversed string (compare figure above to figure for pref()), the $O(m)$ time algorithm from last lecture for finding the table of pref values implies a $O(m)$ time algorithm for finding the suf values.

We want to find $\mathrm{BMShift}(j)$, which for an unsuccesful attempt with the negative character test happening at position j in the pattern x is the minimum legal shift.

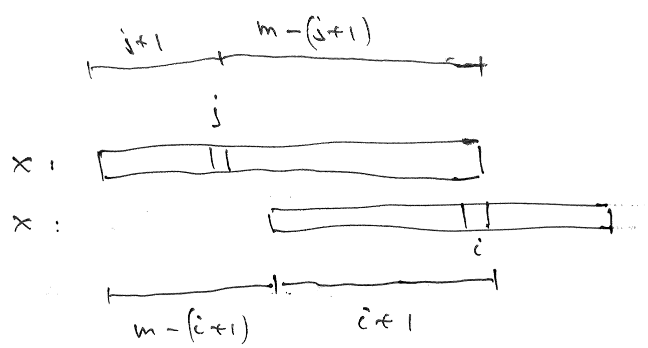
The possible legal shifts after an unsuccesful attempt in the BM algorithm can be divided into two types, I and II, depending on whether the shift is strictly less than $j-1$ (Type I) or at least $j-1$ (Type II).



Note that for a given j, any Type I shift is smaller than any Type II shift. Our algorithm will first for each j find the minimum over all Type II shifts, and then for each j update with the minimum over all Type I shifts (if any). There is always at least one shift of Type II possible, namely a shift of distance m.

Type II Shifts

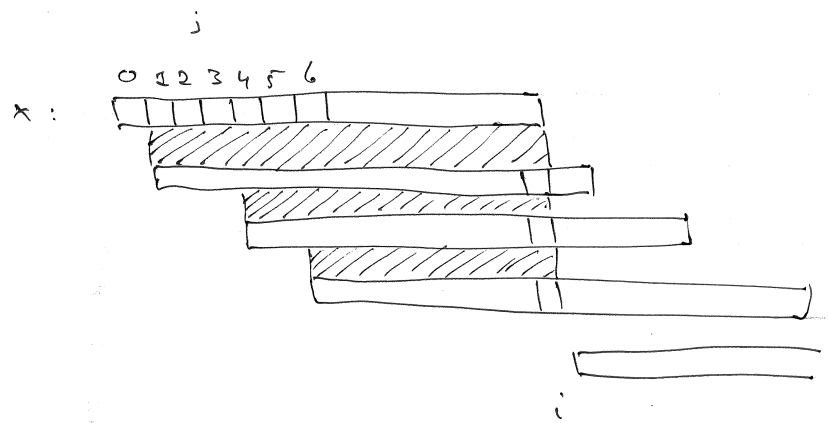
Let (j, i) for $0 \leq j \leq m-1$ (attempt is unsuccesful) and $-1 \leq i \leq m-1$ (the shift should be at least one) be the possible configurations of the following type.



This is a legal Type II shift (of distance $m-(i+1)$) for j iff the following two conditions are satisfied by (j, i).

1. $\text{suf}(i) = i + 1$
2. $i + 1 \leq m - (j + 1)$

As an example, assume that condition 1 is satisfied for shifts $m-(i+1)$ of sizes 1, 4, 6, and m (that is, for $i = m-2, m-5, m-6, -1$)



It can be seen from the figure that all these shifts (i.e., values of i) fulfill condition 2 for $j = 0$, that the last three shifts fulfill it for $j = 0, 1, 2, 3$, that the last two shifts fulfill it for $j = 0, 1, 2, 3, 4, 5$, and that the last shift fulfill it for all j.

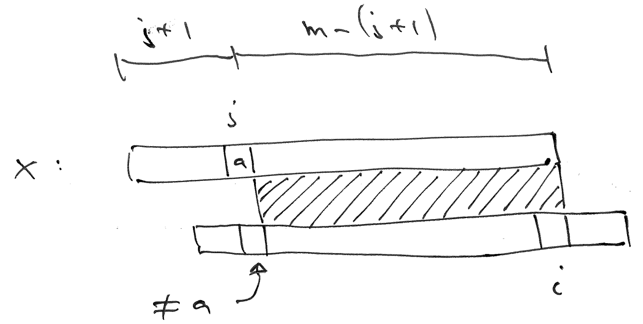
For a given j we want the *smallest* shift (which means largest i). This means the first shift $j = 0$, that second shift for $j = 1, 2, 3$, third shift for $j = 4, 5$, and the last shift for the rest of the j's.

Thus, the following code makes the table `BMShift[j]` contain the smallest possible Type II shift for each `j`.

```
j=0
FOR i=m-2 DOWN TO -1
  IF suf[i] == i+1
    WHILE j < m-1-i
      BMShift[j] = m-(i+1)
      j++
```

Type I Shifts

Let (j, i) for $0 \le j \le m-1$ and $-1 \le i \le m-1$ be the possible configurations of the following type.



This is a legal Type I shift (of distance $m-(i+1)$) for j iff the following two conditions are satisfied by (j,i).

1. $\text{suf}(i) = m-(j+1)$
2. $i+1 \geq m-(j+1)+1$

For each i there is exactly one value of j fulfilling condition 1, namely $j = m - \text{suf}(i) - 1$. However, several values of i can fulfill condition 1 for the same j. We would like to check these for decreasing shift lengths, i.e., increasing values of i, such that the last one checked will be the smallest shift.

Since $i+1 \geq \text{suf}(i)$ always, condition 1 implies $i+1 \geq m-(j+1)$, so the only way for (j,i) to fulfill condition 1 but not condition 2 is to have $i+1 = m-(j+1)$. As can be seen from previous figures, this a valid Type II shift for j, and all other valid Type II shifts are larger. Thus, this is actually the current value for BMShift(j), based on the the Type II shifts. Hence it is fine to just check for condition 1, and set values for BMShift(j) based on this. Any real Type I (condition 1 and 2) for a value of j will be met later (for larger i, i.e., shorter shifts) and thus overwrite the value. Conversely, if no Type I exists for that value of j, no harm was done to BMShift(j).

In short, the code above just needs to be extended with the following code in order to find the final values for `BMShift[j]` based on both Type I and Type II shifts.

```
FOR i=-1 TO m-2
  BMShift[m-suf[i]-1] = m-(i+1)
```