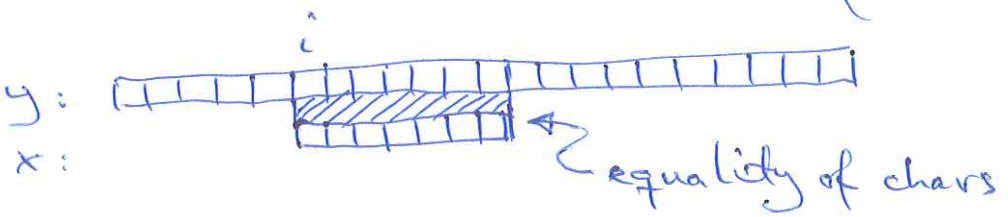
Exact Pattern Matching

Find string x (the pattern) in a string y.

Occurrence at position i : ($n = |y|$, $m = |x|$, $m \leq n$)



"Sliding window" idea : test occurrences for increasing i.

Naive : increment i by one each time

Better : try to increment by more (many algorithms / methods).

Naive:

```
for i = 0 to n - m
    match = true
    for j = 0 to m - 1
        if y[i+j] ≠ x[j]
            match = false
            break
    if match
        report occurrence at pos. i
```

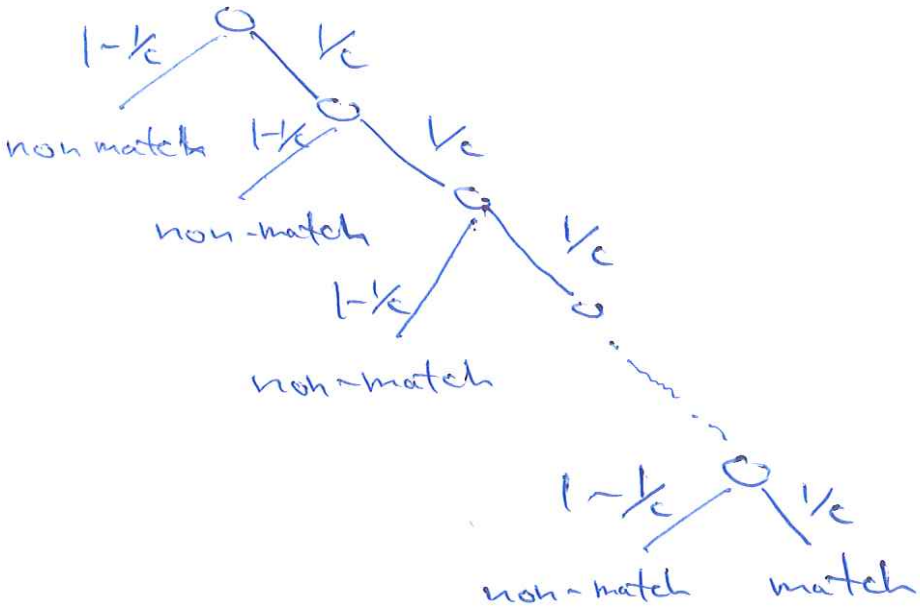
Runtime of naive: $O(n \cdot m)$ worstcase

Example:

$y = aaa.....a$

$x = aaa..b$

In practice often better. For instance, if strings are random, i.e. each char of y is drawn randomly from Σ, then for any pattern the various possible executions of the inner loops, with probabilities for each case, are [$c = |\Sigma|$]:



The expected number of iterations for each inner loop is therefore ($c = |\Sigma|$)

$$\sum_{i=1}^{|x|} i \cdot \text{propability}(i \text{ iterations})$$

$$\leq \left(1 - \frac{1}{c}\right) \cdot \left(1 + \frac{1}{c} \cdot 2 + \left(\frac{1}{c}\right)^2 \cdot 3 + \left(\frac{1}{c}\right)^3 \cdot 4 + \dots\right)$$

$$= \left(1 - \frac{1}{c}\right) \cdot \frac{1}{\left(1 - \frac{1}{c}\right)^2} = \frac{1}{1 - \frac{1}{c}}$$

$$= \frac{1}{\frac{c-1}{c}} = \frac{c}{c-1} \leq 2$$

for $c \geq 2$.

So total time expected is $O(2 \cdot n)$ $= O(n)$ for this type of random y (and any x).

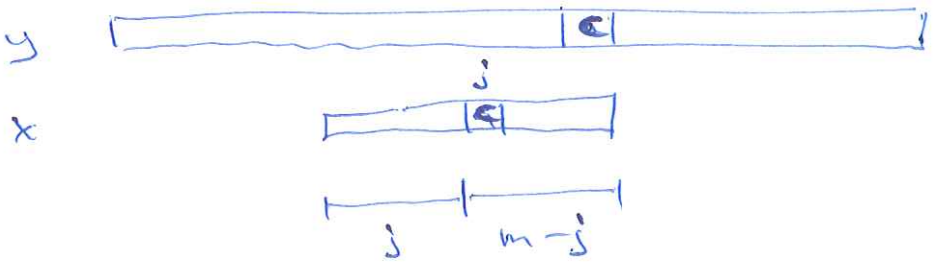
Heuristics for speeding up the naive algorithm on many inputs:

i) First scan y for occurrences of one of x's characters [eg. the/a least frequent character of x].

Then only check of $\#(\text{occ in } y) \times \#(\text{occ in } x)$ window positions are necessary

ii) Hoorspool heuristic:

Preprocessing by finding for each $c \in \Sigma$ its rightmost occurrence pos. in $x[0..|x|-2]$ (ie. not counting last char of x).



After test of current window position (whether match or non-match), we can safely shift window $(m-j)-1$ positions to the right without skipping a valid match.

Note: if a char $c \in \Sigma$ does not appear in x, we can shift window by m, which corresponds to a value $j = -1$.

Space requirement: a table of size $|\Sigma|$.

Preprocessing (find table):

```
for i = 0 to |Σ|
    table[i] = -1

for j = 0 to |x| - 1
    table[ x[j] ] = j
```

Time: $O(m + |\Sigma|)$

We assume Σ can be understood as $\{0, 1, \ldots, |\Sigma|-1\}$