Pattern Matching under Hamming Distance

Chapters 9.1 and 9.4 of Dan Gusfield: *Algorithms on strings, trees, and sequences*

Giulia Bernardini
giulia.bernardini@units.it

Algorithmic Design, Advanced Algorithms for Scientific Computing, Algorithmic Data Mining
a.y. 2023/2024

The k-mismatch problem

IN: a text T of length n, a pattern P of length m<n, an integer k<m

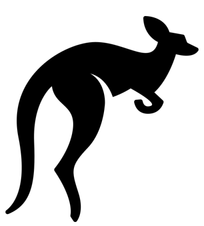
OUT: all positions i in T such that $d_H(T[i..i+|P|-1],P)\leq k$

```
        7 10 13 16
T=AMBARABACCICCICCOCCO ; P=COCCO ; k=3
        xx  x
        COCCO
            x  x
           COCCO
               x
              COCCO
                 COCCO
```

Output: {7,10,13,16}

The kangaroo algorithm for k-mismatch

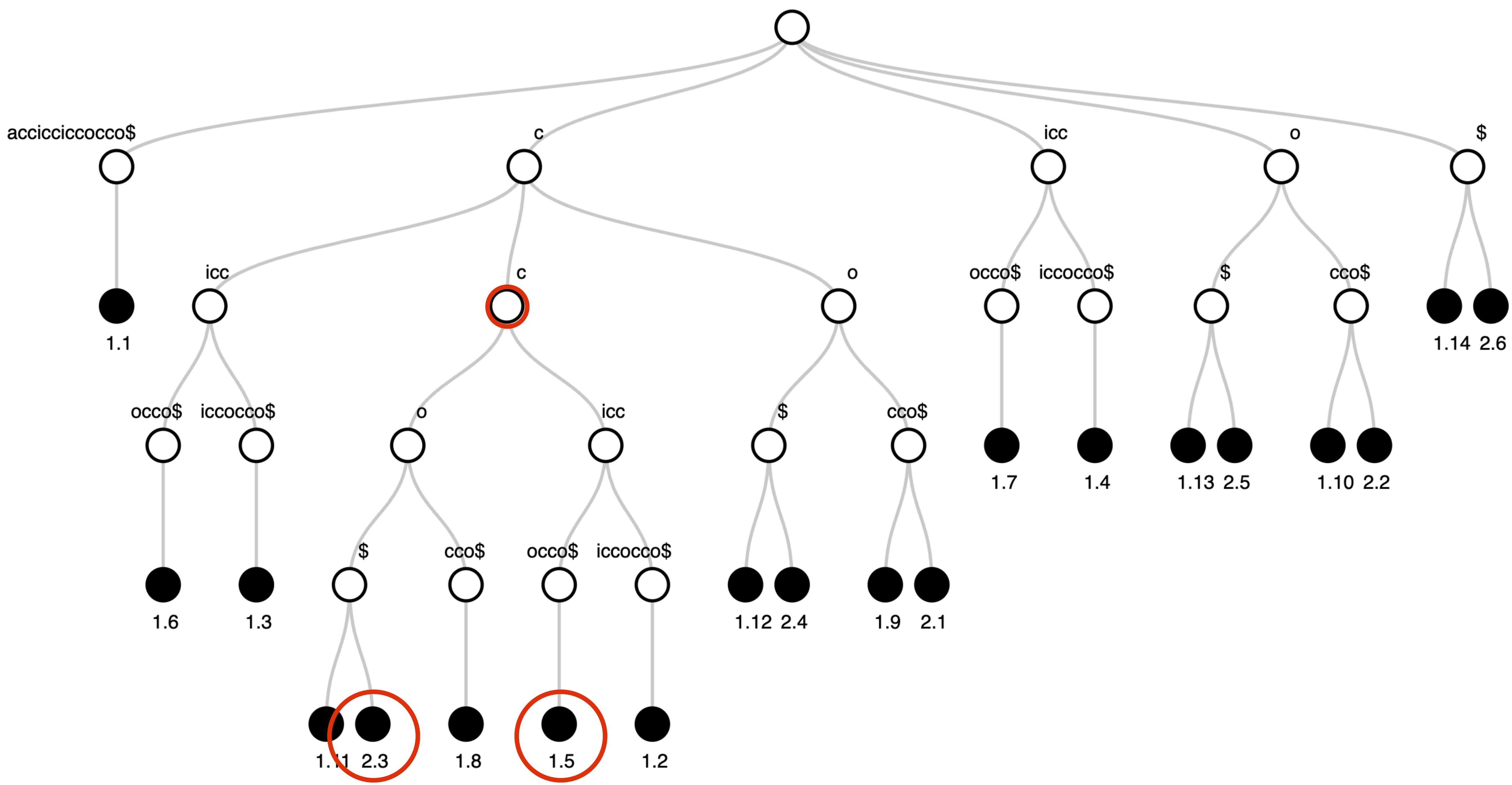
```
kMISMATCH(T,P,k)
    sol ← ∅;
    for all i=1,…,|T|
        count←0; match←0;
        while count≤k and match+count<|P|
            ext←LCE_{T,P}( match+count+i , match+count+1 );
            match←match + ext;
            if match+count=|P|
                sol.append(i);
            else
                count←count +1;
    return sol;
```



The k-mismatch problem

T=ACCICCICCOCCO ; P=COCCO ; k=3

$LCE_{T,P}(5,3)=2$



Edit Distance

Chapters 11.2 and 11.3 of Dan Gusfield: *Algorithms on strings, trees, and sequences*

Giulia Bernardini
giulia.bernardini@units.it

Algorithmic Design, Advanced Algorithms for Scientific Computing, Algorithmic Data Mining
a.y. 2023/2024

Computing edit distance: recursion

Given strings S of length n and T of length m, we denote by D(i,j) the edit distance between S[1..i] and T[1..j]. D(n,m) denotes the edit distance between the whole S and T.

Base conditions: $D(i,0)=i$ (i deletions) and $D(0,j)=j$ (j insertions).

Let $d:[1,n]\times[1,m]\rightarrow\{0,1\}$ a function such that $d(i,j)=1$ if $S[i]\neq T[j]$, $d(i,j)=0$ otherwise. Then it holds the following

Recursion: $D(i,j)=\min\{ D(i-1,j)+1 , D(i,j-1)+1 , D(i-1,j-1)+d(i,j) \}$ for any $i\in[1,n]$, $j\in[1,m]$.

Computing edit distance: dynamic programming

The dynamic programming algorithm for computing edit distance consists in computing all values D(i,j) bottom-up, starting from the smallest possible i and j and storing the computed values in a dynamic programming table that has the letters of S at the columns and the letters of T at the rows (plus an extra row and column to account for i=0 and j=0).

Computing edit distance: dynamic programming

The optimal transcript highlighted in grey is MIIMRMMM, corresponding to the alignment

```
S  unday
Saturday
MIIMRMMM
```

		S	u	n	d	a	y
	0	1	2	3	4	5	6
S	1	0	1	2	3	4	5
a	2	1	1	2	3	3	4
t	3	2	2	2	3	4	4
u	4	3	2	3	3	4	5
r	5	4	3	3	4	4	5
d	6	5	4	4	3	4	5
a	7	6	5	5	4	3	4
y	8	7	6	6	5	4	**3**