SEQUENCE MOTIF FINDING

Phạm Quang Dũng

Bộ môn Khoa học máy tính - Khoa Công nghệ thông tin

Trường Đại học Nông nghiệp Hà Nội

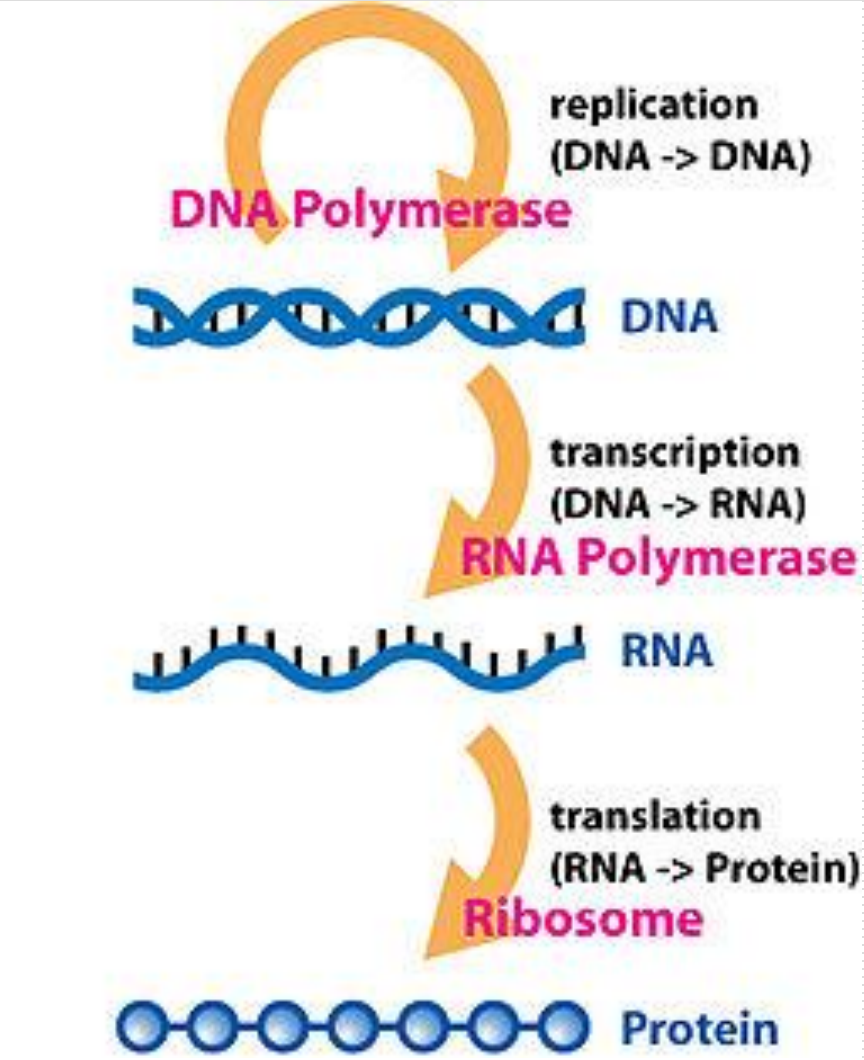
email: pqdung@hua.edu.vn

website: www.hua.edu.vn/khoa/fita/pqdung

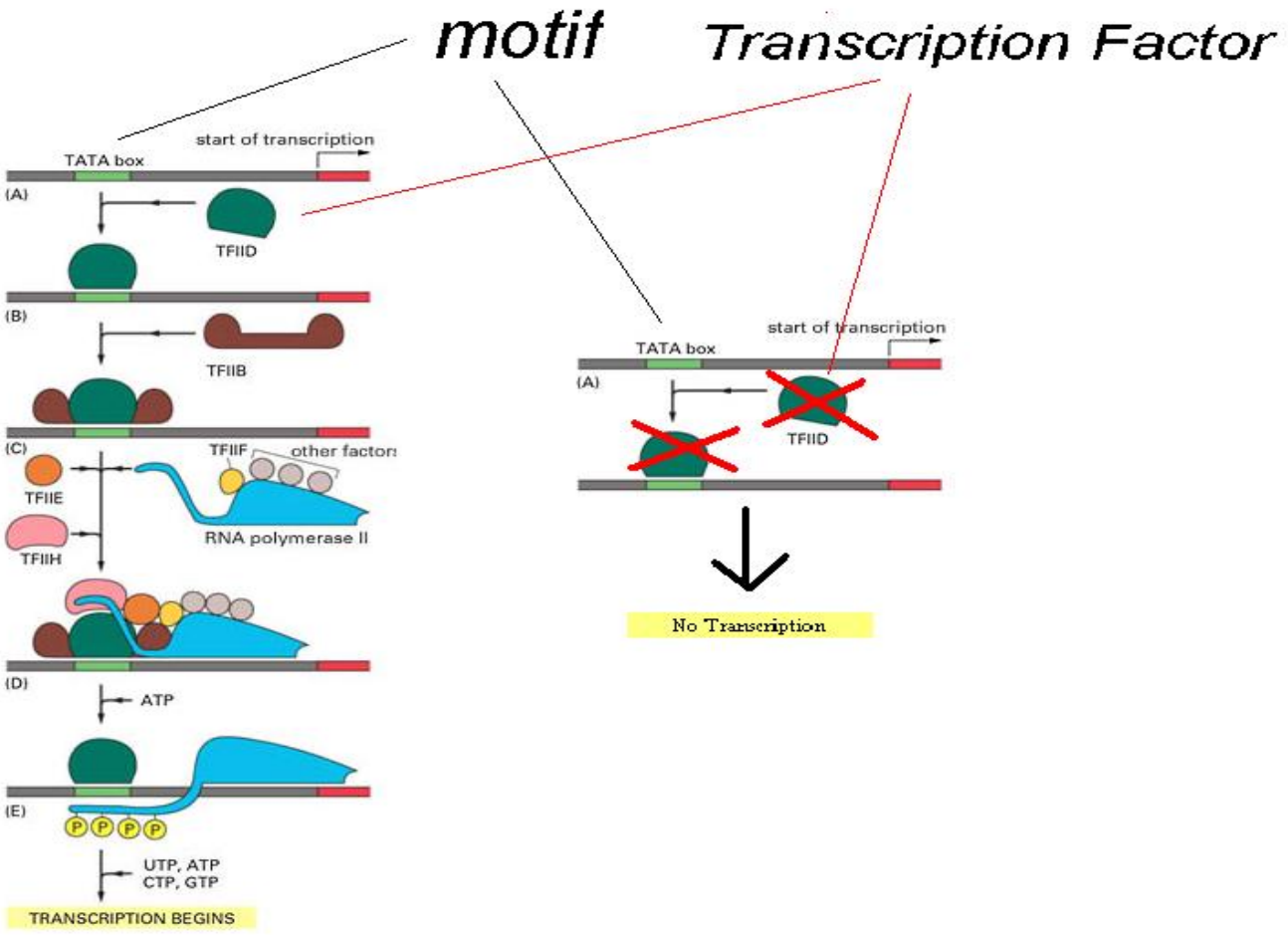
Content

1. Biology & maths. background
2. Problem statement
3. Sequence motif finding algorithms
4. Considering some exact algorithms

Central dogma



Transcription Factors and Motifs



Transcription Factor Binding Sites

- Every gene contains a regulatory region (RR) upstream of the transcriptional start site
- Located within the RR are the *Transcription Factor Binding Sites* (TFBS), also known as *motifs*, specific for a given transcription factor
- A TFBS can be located anywhere within the Regulatory Region (RR).
- A single TF can regulate multiple genes if those genes' RRs contain corresponding TFBS
  - Can find regulated genes via knock out experiments

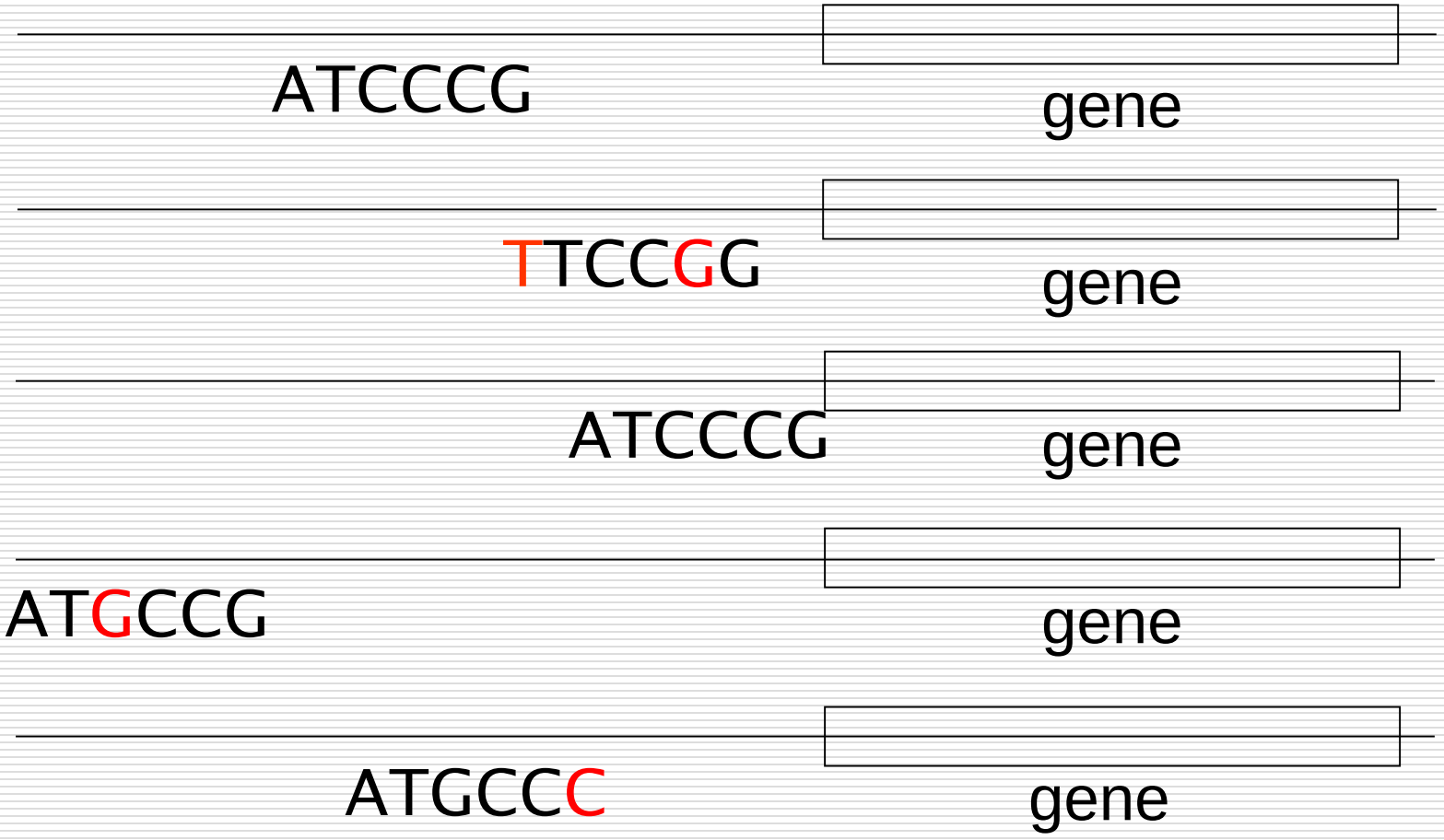
Problem statement

- Sequence motifs are short, recurring patterns in DNA/RNA/protein that are presumed to have a biological function.
- The characterization and localization of motifs is a fundamental approach to a better understanding of the structure, function and evolutionary relationships of the corresponding genes or proteins.
  - Eg.: they indicate sequence-specific binding sites for proteins such as nucleases and transcription factors (TF).
  - Others are involved in important processes at the RNA level, including ribosome binding, mRNA processing (splicing, editing, polyadenylation) and transcription termination.

Identifying Motifs: Complications

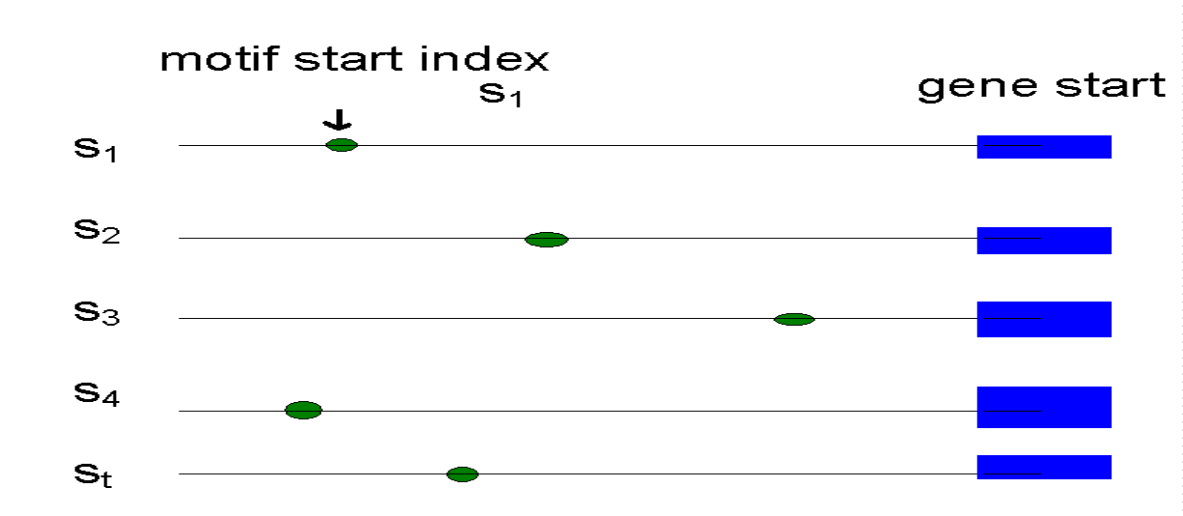
- We do not know the motif sequence
  - May know its length
- We do not know where it is located relative to the genes start
- Motifs can differ slightly from one gene to the next
  - Non-essential bases could mutate…
- How to discern functional motifs from random ones?

Motifs and Transcriptional Start Sites

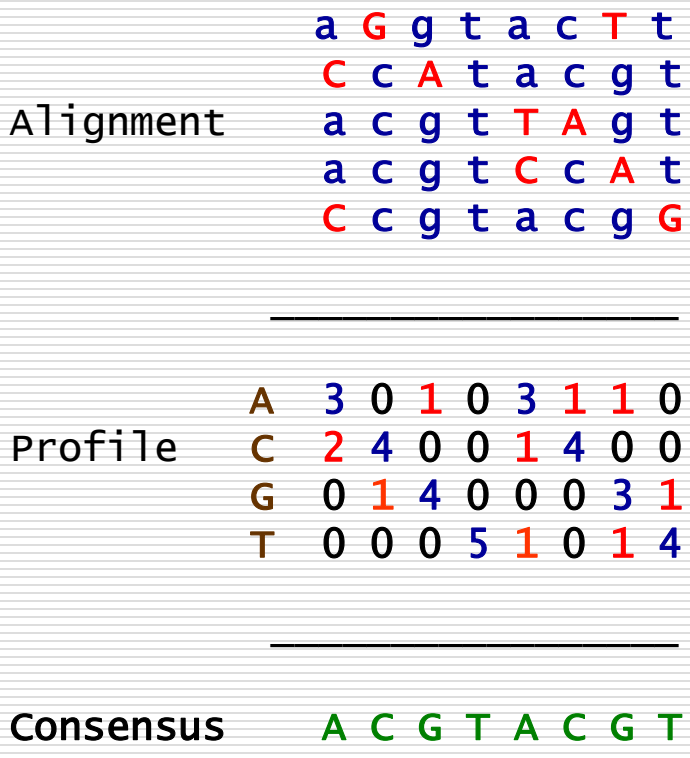


Defining Motifs

- To define a motif, lets say we know where the motif starts in the sequence
- The motif start positions in their sequences can be represented as $\boldsymbol{s} = (s_1, s_2, s_3, \ldots, s_t)$



Motifs: Profiles and Consensus



- Line up the patterns by their start indexes

$$\mathbf{s} = (s_1, s_2, \ldots, s_t)$$

- Construct matrix profile with frequencies of each nucleotide in columns
- Consensus nucleotide in each position has the highest score in column
  - **Think of consensus as an "ancestor" motif, from which mutated motifs emerged**

Evaluating Motifs

- We found the consensus sequence, but how "good" is this consensus?

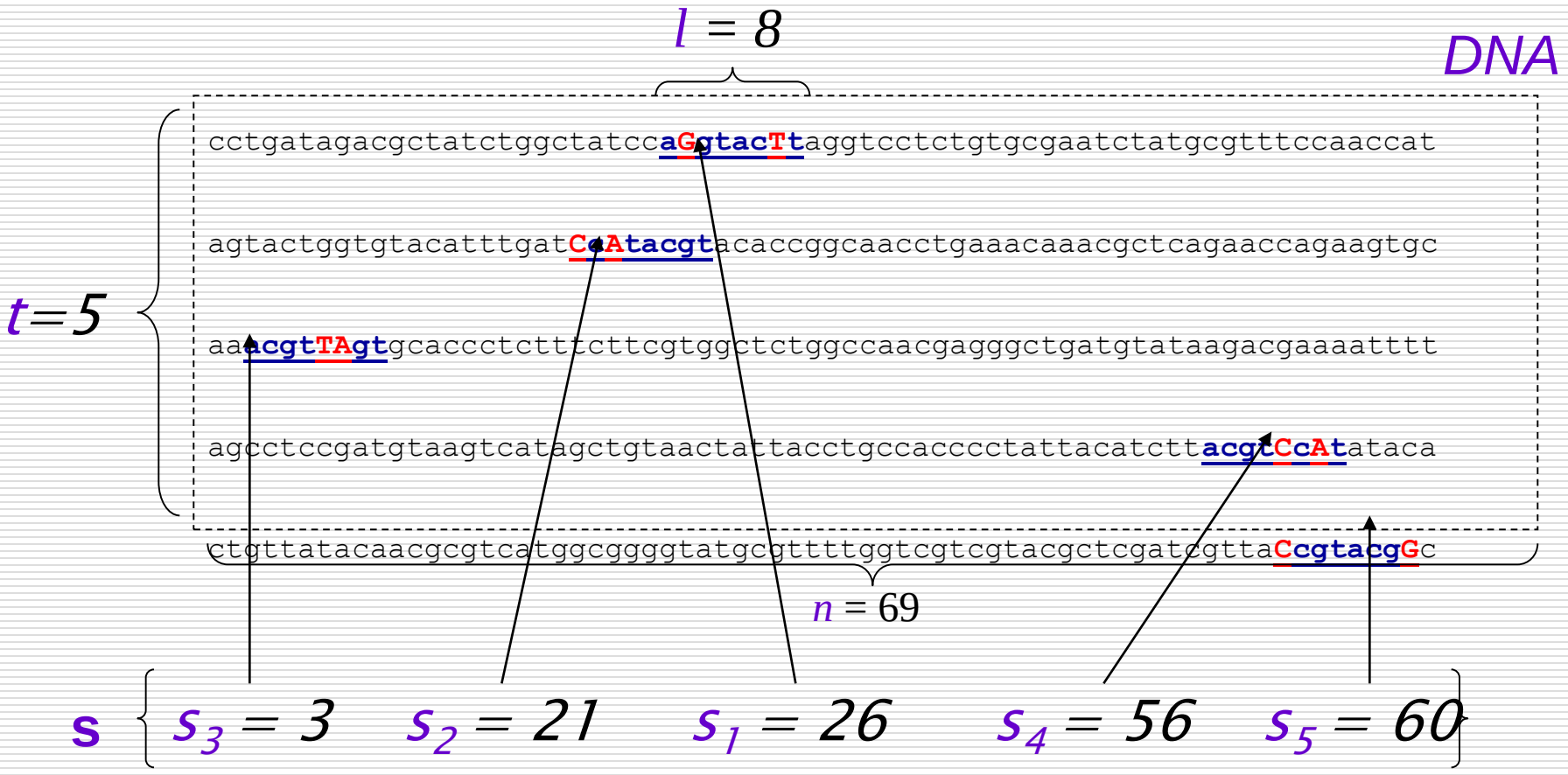
- Need to introduce a scoring function

Some Notations

- ***t*** - number of sample DNA sequences
- ***n*** - length of each DNA sequence
- ***DNA*** - sample of DNA sequences (t x n array)

- ℓ - length of the motif (ℓ-mer)
- s_i - starting position of an ℓ-mer in sequence i
- $\mathbf{s}=(s_1, s_2, \ldots s_t)$ - array of motif's starting positions

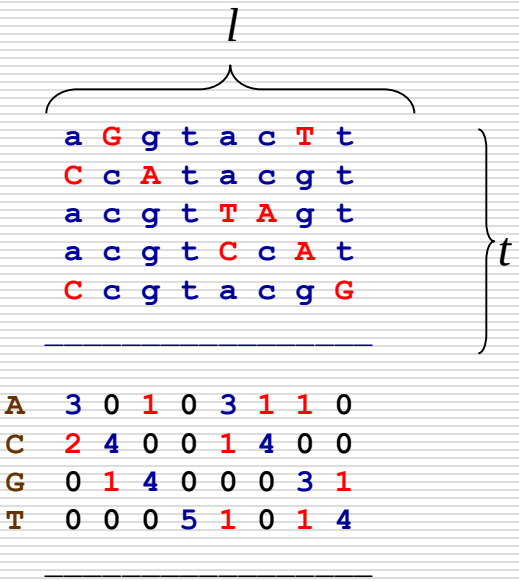
Example


l = 8
DNA
cctgatagacgctatctggctatccaGgtacTtaggtcctctgtgcgaatctatgcgtttccaaccat
agtactggtgtacatttgatCcATacgtacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtTAgtgcaccctctttcttcgtggctctggccaacgagggctgatgtataagacgaaaatttt
agcctccgatgtaagtcatagctgtaactattacctgccacccctattacatcttacgtCcAtataca
ctgttatacaacgcgtcatggcggggtatgcgttttggtcgtcgtacgctcgatcgttaCcgtacgGc
t=5
n = 69
s
s3 = 3
s2 = 21
s1 = 26
s4 = 56
s5 = 60

Scoring Function

- Given **s** = $(s_1, \ldots s_t)$ and ***DNA***:

$$Score(\mathbf{s}, \mathbf{DNA}) = \sum_{i=1}^{l} \max_{k \in \{A,T,C,G\}} count(k,i)$$



The Motif Finding Problem

- If starting positions **s**=(s_1, s_2,… s_t) are given, the problem is easy even with mutations in the sequences because we can simply construct the profile to find the motif (consensus)

- But… the starting positions **s** are usually not given. How can we align the patterns and compute the "best" profile matrix?

The Motif Finding Problem: Formulation

The Motif Finding Problem: Given a set of DNA sequences, find a set of ℓ-mers, one from each sequence, that maximizes the consensus score

- Input: A ***t*** x ***n*** matrix of *DNA*, and ℓ, the length of the pattern to find

- Output: An array of ***t*** starting positions **s** = (s_1, s_2, … s_t) maximizing *Score*(s,*DNA*)

The Motif Finding Problem: Brute Force Solution

- Compute the scores for each possible combination of starting positions ***s***
- The best score will determine the best profile and the consensus pattern in ***DNA***
- The goal is to maximize *Score*(***s***,***DNA***) by varying the starting positions s_i, where:

$$1 \leq s_i \leq n-l+1]$$
$$i = 1, \ldots, t$$

Pseudocode for Brute Force Motif Search

BruteForceMotifSearch(***DNA, t, n, l***)

bestScore ← 0

for each **s**=(s_1,s_2 , . . ., s_t) from (1,1 . . . 1)
to (***n***–***l***+1, . . ., ***n***–***l***+1)

if (*Score*(**s**,***DNA***) > ***bestScore***)

bestScore ← *score*(**s**, ***DNA***)

bestMotif ← (s_1,s_2 , . . . , s_t)

return **bestMotif**

Brute Force Approach: Running Time

- Varying $(n - \ell + 1)$ positions in each of $\boldsymbol{t}$ sequences, we're looking at $\boldsymbol{(n - \ell + 1)^t}$ sets of starting positions
- For each set of starting positions, the scoring function makes ℓ operations, so complexity is $\ell (n - \ell + 1)^t = \boldsymbol{O(\ell n^t)}$

Running Time of BruteForceMotifSearch

- That means that for $t = 8$, $n = 1000$, $\ell = 10$
  - Must perform 7.322E+25 computations
  - Assuming each computation takes a cycle on a 3 GHz CPU, it would take 7.33 billion years to search all the possibilities
- This algorithm is not practical
- Lets explore some ways to speed it up

The Median String Problem

- Given a set of ***t*** DNA sequences find a pattern that appear in all ***t*** sequences with the minimum number of mutations

- This pattern will be the motif

Hamming Distance

- Hamming distance:
  - $d_H(\boldsymbol{v},\boldsymbol{w})$ is the number of nucleotide pairs that do not match when ***v*** and ***w*** are aligned. For example:

$$d_H(\text{AAAAAA}, \text{ACAAAC}) = 2$$

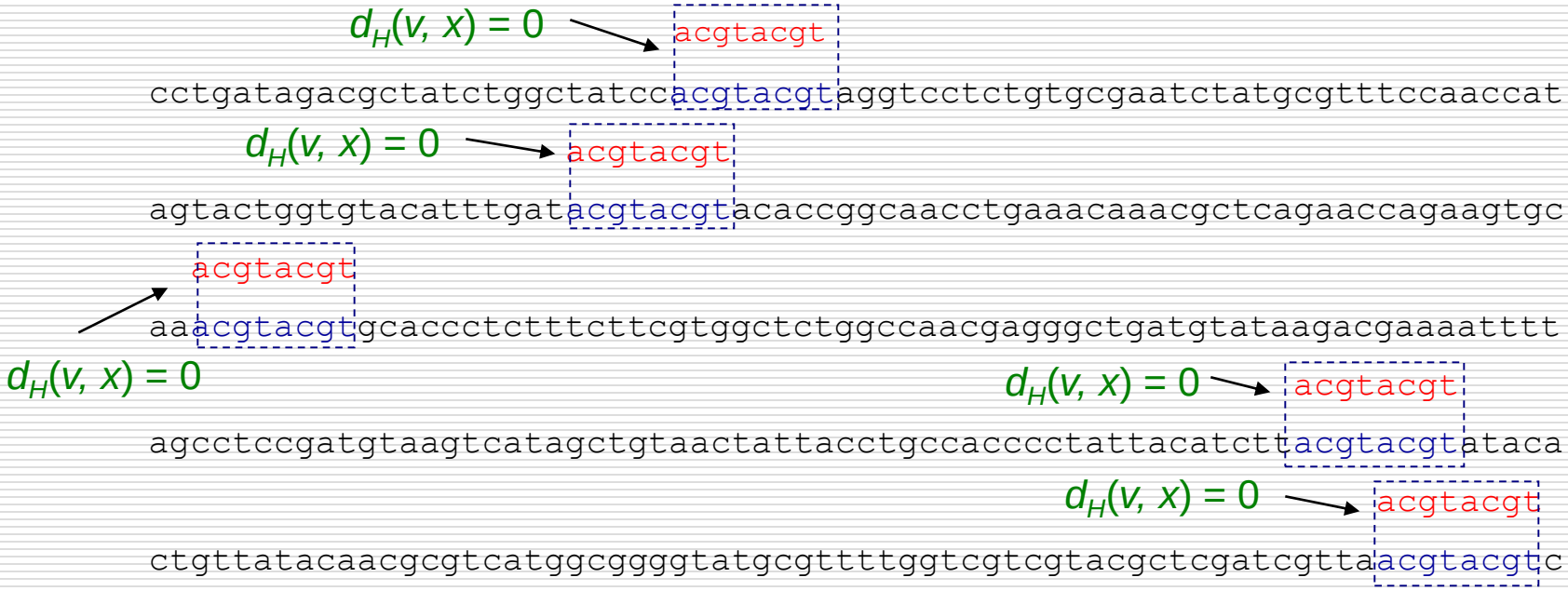
Total Distance

- For each DNA sequence ***i***, compute all $d_H(\boldsymbol{v}, \boldsymbol{x})$, where ***x*** is an ℓ-mer with starting position s_i ($1 \leq s_i \leq \boldsymbol{n} - \ell + 1$)

- *TotalDistance*(***v,DNA***) is the sum of the **minimum Hamming distances** for each DNA sequence ***i***

Total Distance: An Example

- Example 1, given ***v*** = "acgtacgt" and **s**

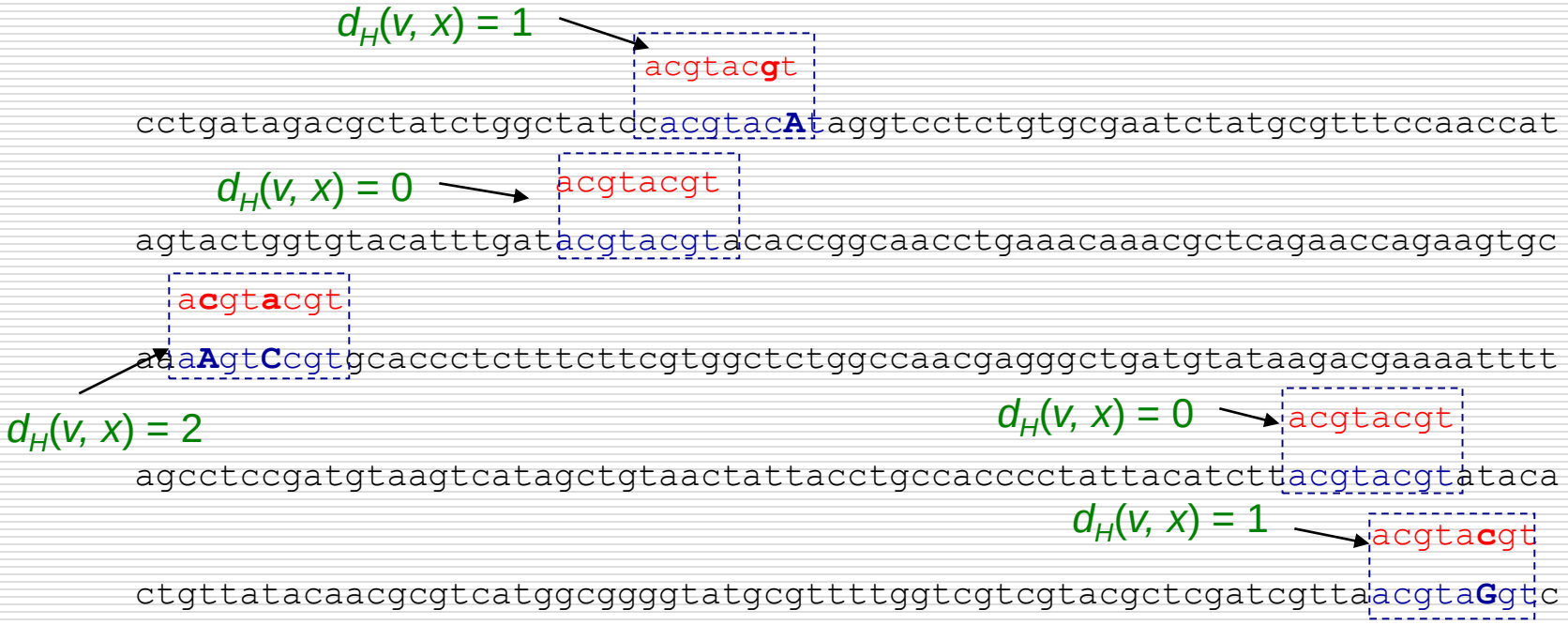


v is the sequence in red, x is the sequence in blue

- $TotalDistance(\mathbf{v}, \mathbf{DNA}) = 0$

Total Distance: Another Example

- Example 2, given ***v*** = "acgtacgt" and **s**



v is the sequence in red, x is the sequence in blue

- $TotalDistance(\mathbf{v}, DNA) = 1 + 2 + 1 = 4$

The Median String Problem: Formulation

The Median String Problem:

- Given a set of DNA sequences, find a median string
- Input: A ***t*** x ***n*** matrix DNA, and ℓ, the length of the pattern to find
- Output: A string ***v*** of ℓ nucleotides that minimizes *TotalDistance(**v**,**DNA**)* over all strings of that length

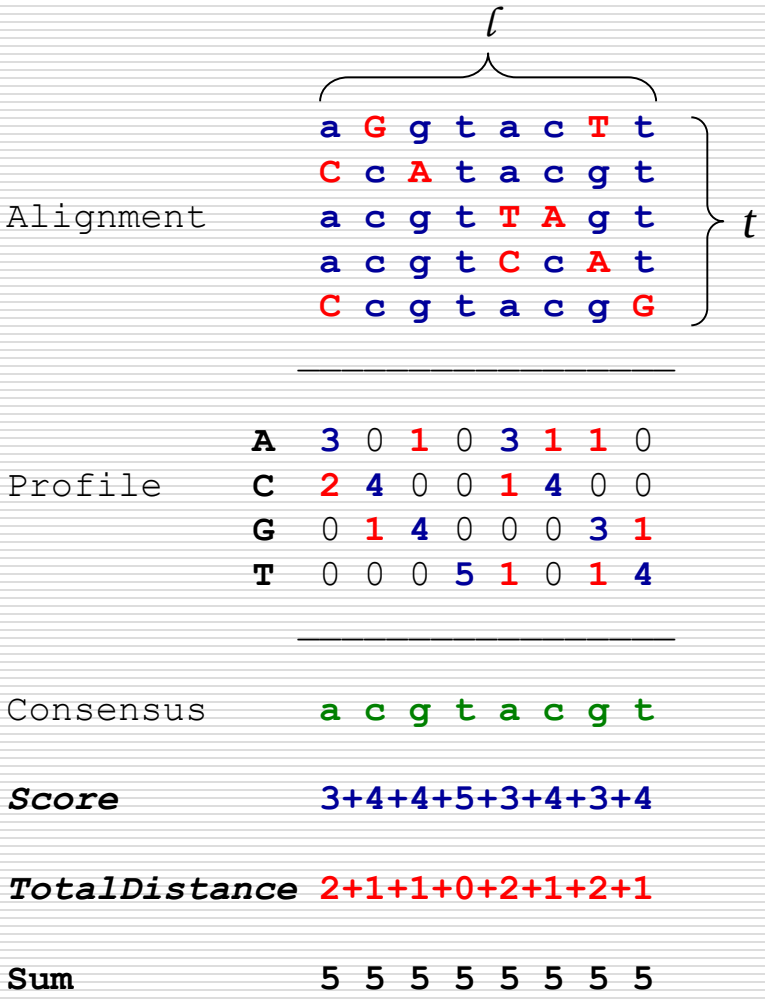
Motif Finding Problem == Median String Problem

- The *Motif Finding* and *Median String* problems are computationally equivalent

- Proof:

  Need to show that minimizing *TotalDistance* is equivalent to maximizing *Score*

We are looking for the same thing



- At any column ***i***

  $Score_i + TotalDistance_i = \boldsymbol{t}$

- Because there are l columns

  $Score + TotalDistance = \ell * \boldsymbol{t}$

- Rearranging:

  $Score = \ell * \boldsymbol{t} - TotalDistance$

- $\ell * \boldsymbol{t}$ is constant the minimization of the right side is equivalent to the maximization of the left side

The Motif Finding Problem vs. The Median String Problem

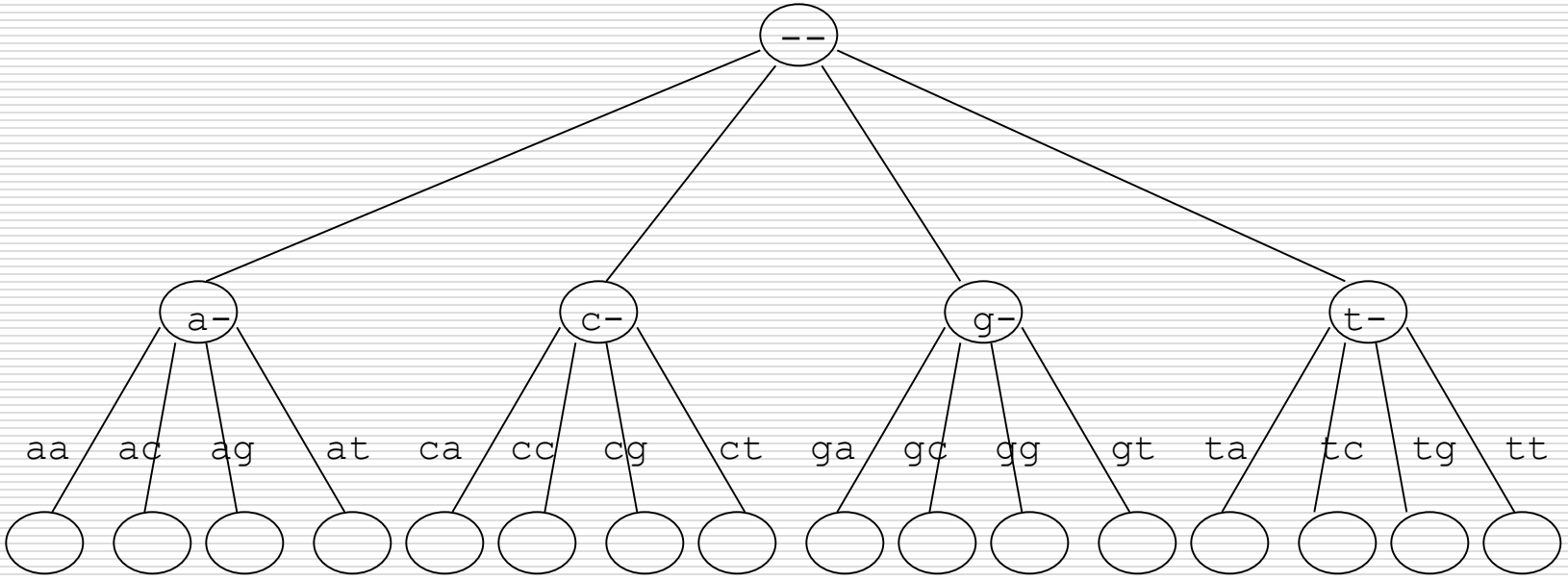
- Why bother reformulating the motif finding problem into the median string problem?
  - The Motif Finding Problem needs to examine all the combinations for **s**. That is $(\boldsymbol{n} - \ell + 1)^t$ combinations!!!
  - The Median String Problem needs to examine all 4^ℓ combinations for $\boldsymbol{v}$. This number is relatively smaller

Brute Force Median String Algorithm

1. MedianStringSearch (***DNA, t, n, l***)
2. ***bestWord*** ← AAA…A
3. ***bestDistance*** ← ∞
4. for each *l*-mer **s from** AAA…A to TTT…T
   if *TotalDist*(**s,***DNA***)** < ***bestDistance***
   bestDistance←*TotalDist*(**s,*DNA*)**
   bestWord ← **s**
5. **return** *bestWord*

Search Trees

- Group candidate sequences by their prefixes

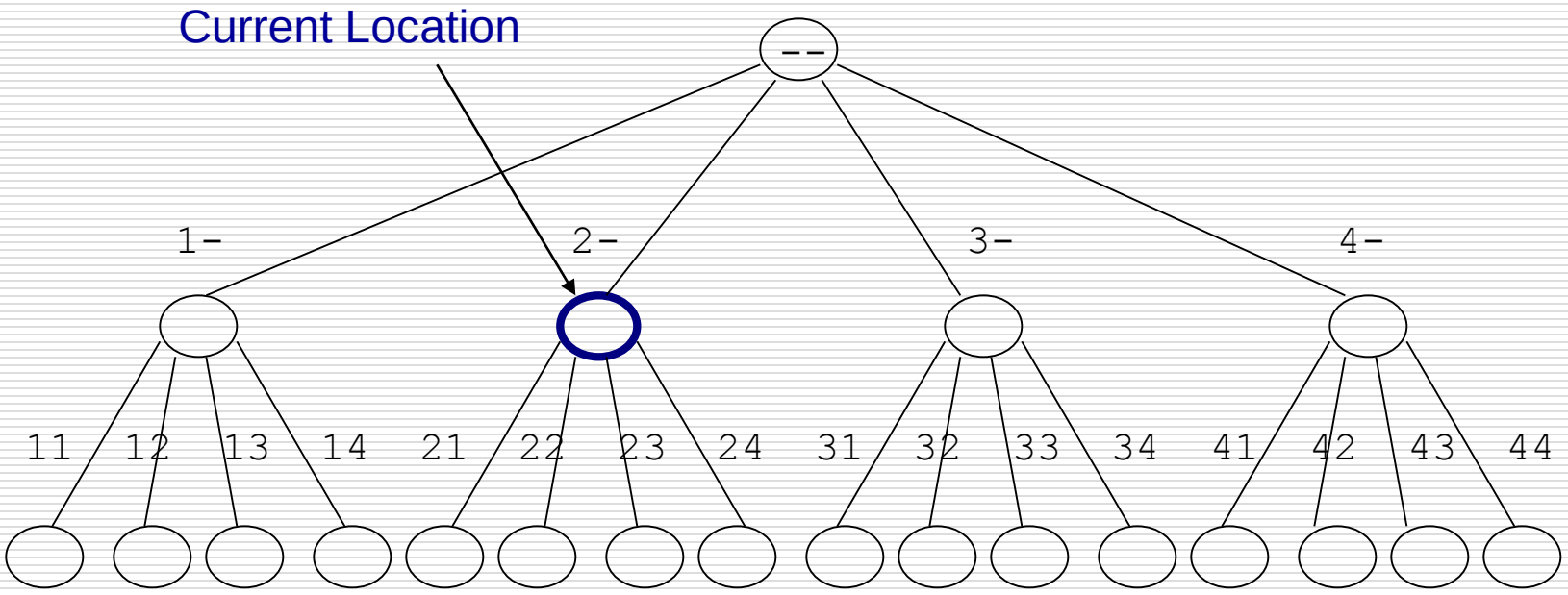


Moving through the Search Trees

- Once the tree is built, we need to design algorithms to move through the tree
- Four common moves in a search tree that we are about to explore:
  - Move to the next leaf
  - Visit all the leaves
  - Visit the next node
  - Bypass the children of a node

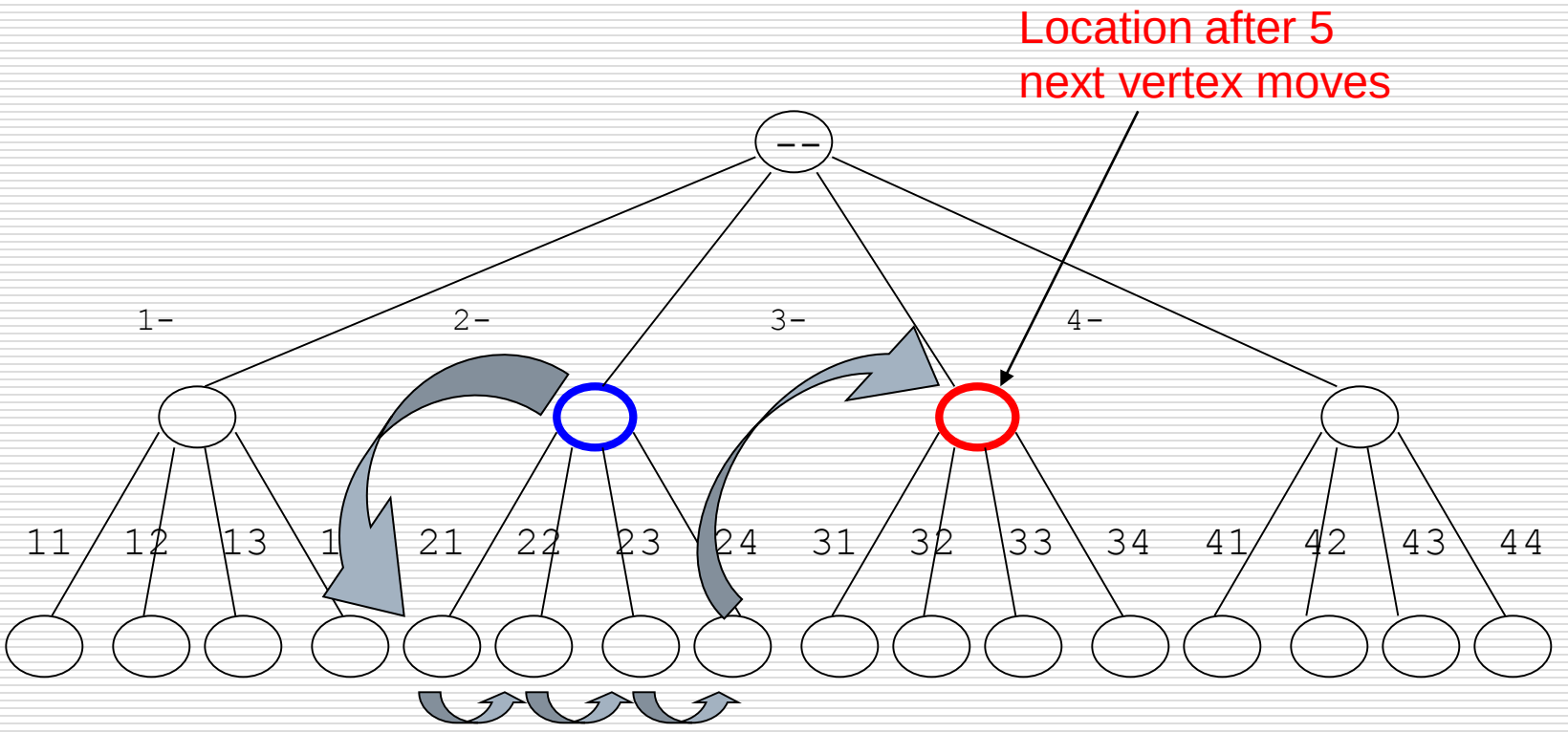
Example

- Moving to the next vertex:



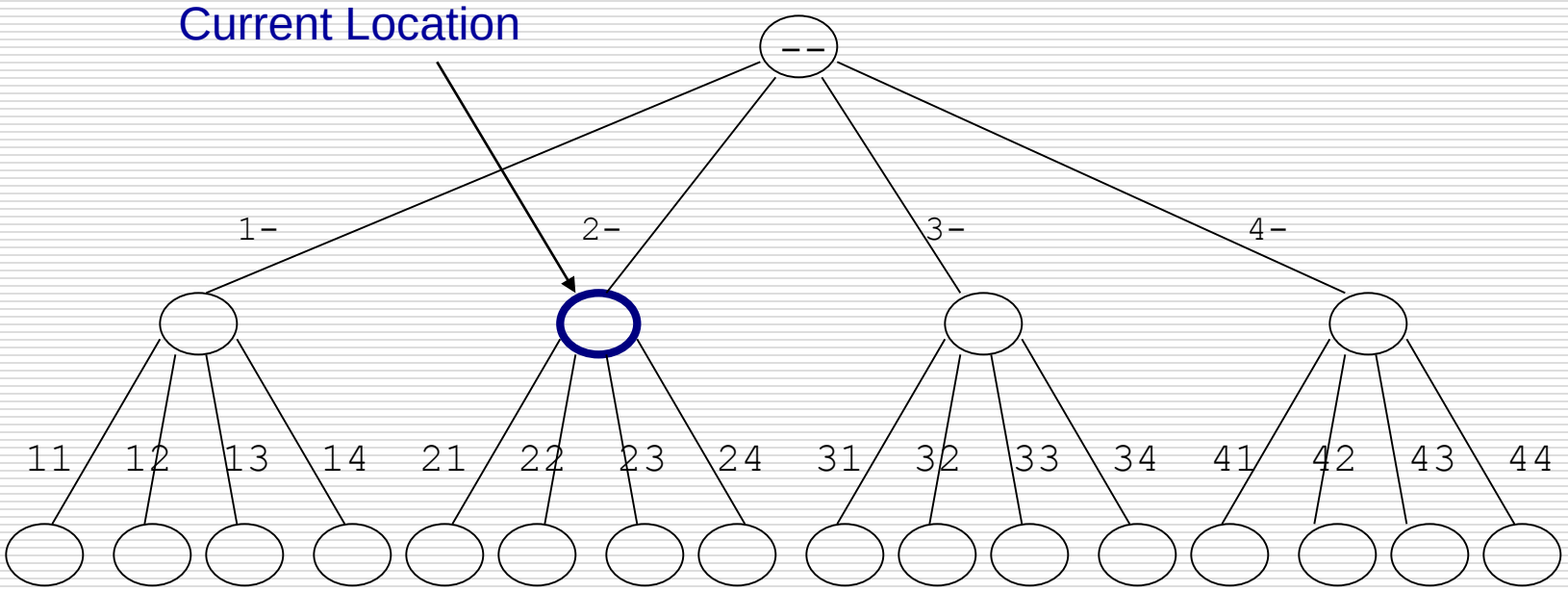
Example

- Moving to the next vertices:



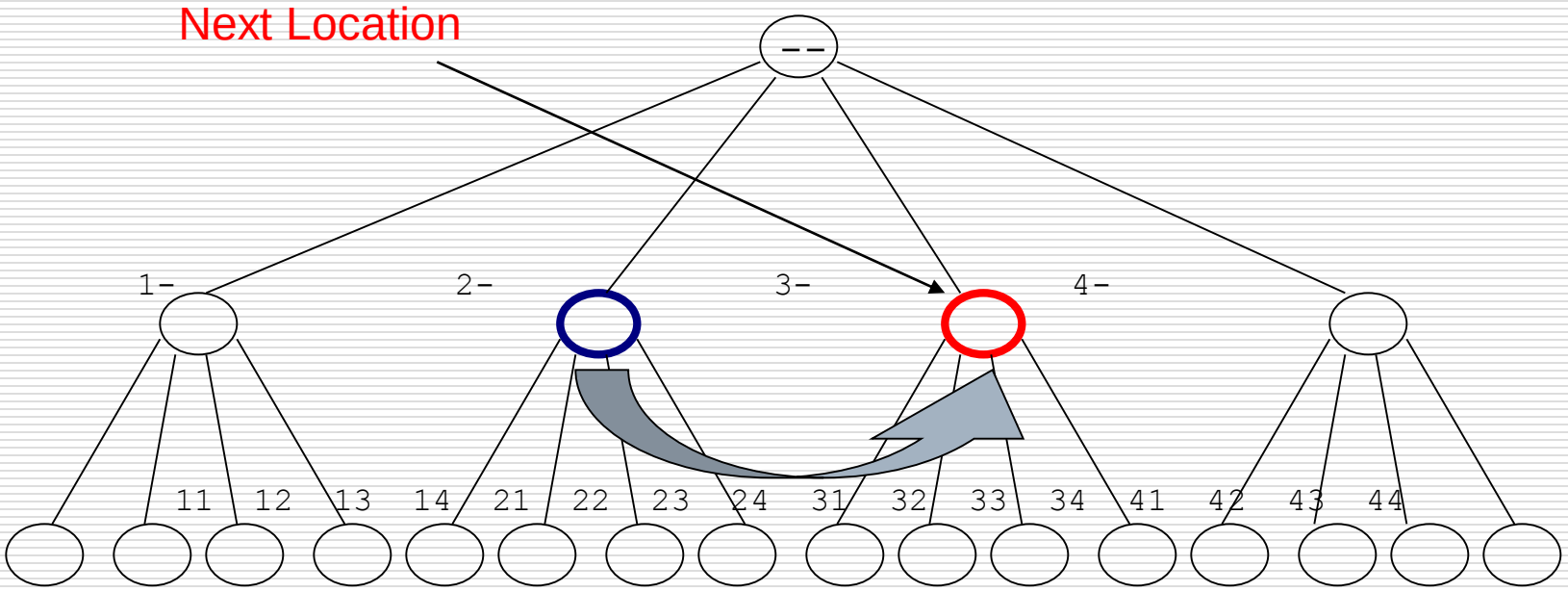
Bypass Move: Example

- Bypassing the descendants of "2-":



Example

- Bypassing the descendants of "2-":



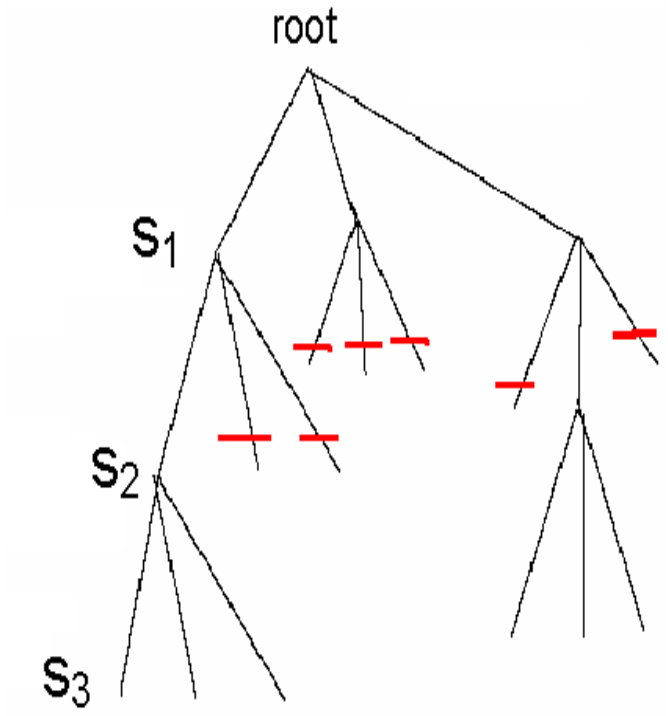
Branch and Bound Applied to Median String Search

- Note that if the total distance for a prefix is greater than that for the best word so far:

  TotalDistance (*prefix*, *DNA*) + ZERO > *BestDistance*

  there is no use exploring the remaining part of the word

- We can eliminate that branch and BYPASS exploring that branch further



Bounded Median String Search

```
1.  BranchAndBoundMedianStringSearch(DNA,t,n,ℓ)
2.  s ← (1,…,1)
3.  bestDistance ← ∞
4.   i ← 1
5.  while i > 0
6.    if i < ℓ
7.       prefix ←  nucleotide string of s
8.       optimisticDistance ← TotalDistance(prefix,DNA)
9.                 if optimisticDistance > bestDistance
10.                  (s, i ) ← Bypass(s,i, ℓ, 4)
11.                else
12.                  (s, i ) ← NextVertex(s, i, ℓ, 4)
13.      else
14.                word ← nucleotide string for s
15.                if  TotalDistance(s,DNA) < bestDistance
16.                   bestDistance ← TotalDistance(word, DNA)
17.                   bestWord ← word
18.              (s,i ) ← NextVertex(s,i,ℓ, 4)
19.  return bestWord
```

Two classes of sequence motif finding prob.

- **Quorum Planted Motif Search** (qPMS): Given n input strings $s_1,\ldots ,s_n$ of length m each, three integer parameters l, d and q, find all the (l,d,q)-motifs of the input strings.
  - A string M of length l is called an (l, d, q)-motif of the strings if there are at least q out of the n strings such that the Hamming distance between each one of them and M is no more than d.
- **Planted Motif Search** (PMS): a special case of the qPMS problem when q=n.

PMS algoritms

- An exact PMS algorithm always finds all the (l, d)-motifs present in the input sequences.
  - NP-hard
  - Algorithms: PMS6, Pampa, **PMSPrune**, RISSOTO
- Typically, approximate PMS algorithms employ heuristics such as local search, Gibbs sampling, expectation optimization, etc.
  - usually tend to be faster
  - Algorithms: MEME, Projection, GibbsDNA, **PairMotif+**, etc.

qPMS algorithms

- The larger the values of l and d that a qPMS algorithm can handle, the more accurate will be the motifs it finds.
- Is harder than the PMS problem.
- exact algorithms:
  - qPMSPrune (2007): l=17, d=5, q=n/2;
  - qPMS7 (6/2012): can solve larger instances, 10 times faster, also best for PMS problem.
  - PairMotif (10/2012): pattern-driven algorithm for (l,d) DNA motif search.

qPMSPrune – pseudo-code

- For any l-mer x, represents it's d-neighborhood as a tree $T_d(x)$

Algorithm qPMSPrune

For each $x \in_\ell s_i, 1 \le i \le n-q+1$ do:

Traverse the tree $\mathcal{T}_d(x)$ in a depth-first manner. At each node (t,p), do the following steps.

i. Incrementally compute $d_H(t,s_j)$ from its parent for $1 \le j \le n, j \ne i$.

ii. Let q' be the number of input strings s_j such that $d_H(t,s_j) \le d$. If $q' \ge q-1$, output t.

iii. Let q'' be the number of input strings s_j such that $d_H(t,s_j) \le 2d - d_H(t,x)$. If $q'' < q-1$, then prune the subtree rooted at node (t,p). Otherwise, explore its children.

qPMSPrune - time complexity

- $O((n-q+1)nm^2N(l,d))$
- Trong đó:

$$\mathcal{N}(\ell,d) = \sum_{i=0}^{d} \binom{\ell}{i} (|\Sigma|-1)^i$$

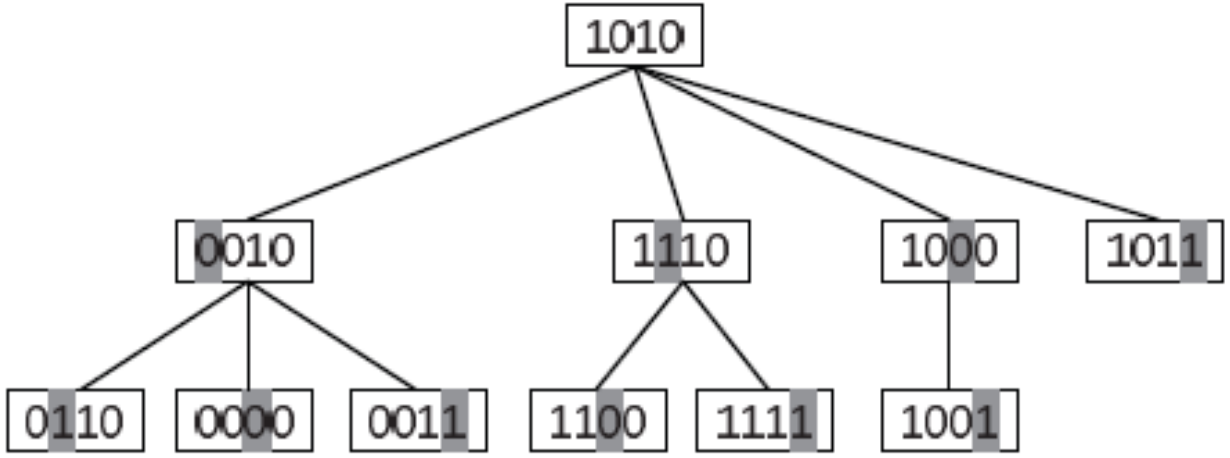


Figure 1. Traverse the tree in qPMSPrune. $\mathcal{T}_2(1010)$ with alphabet $\Sigma=\{0,1\}$. The value of p at each node is the location of its shaded letter. For example, $p=1$ at node 0010, $p=4$ at node 0011.

qPMS7 improves the runtime of qPMSPrune

- reduce the time taken for computing Hamming distances $dH(t,s_j)$ in step (1) of qPMSPrune.
  - the operation takes at least $\Omega(nm)$ time in Algorithm qPMSPrune because it considers every l-mer in each input string s_j.
  - some l-mers can be ignored without changing the result since they just need to count q' and q''.
  - a l-mer z in s_j can be ignored if $d_H(t,z)>2d-d_H(t,x)$.
  - The runtime of the operation now depends on the sizes of the lists of surviving l-mers.

Table 3. Time comparison of different algorithms on the challenging instances of protein sequences for the special case - PMS Problem.

Algorithm	(11,5)	(13,6)	(15,7)	(17,8)	(19,9)
qPMS7	1 m	1.4 m	1.9 m	6.8 m	7.5 m
qPMSPrune1	4.5 m	21 m	2.4 m	17 h	–
qPMSPrune	12 m	104 m	16 h	–	–

The alphabet size $|\Sigma| = 20$, $n = 20$, $m = 600$, and $q = n = 20$.

Table 6. Results on real datasets for transcription factor-binding sites discovery.

Data	Predicted Motifs	Matched Binding Sites
mus05r	AGAGGAAAAAAAAAGGAG	s_1: GGAAAAACAAAGGTAATG
mus07r	CTGCCCACCCTCTGCAACCC	s_4: CCCAACACCTGCTGCCTGAGCC
mus11r	AGGGCGGGGGGCGGAGCG	s_2: GCCGCCGGGGTGGGGCTGAG
		s_3: GGGGGGGGGGGCGGGGC
		s_4: GTGGGGGCGGGGCCTT
		s_9: GAACAGGAAGTGAGGCGG
hm03r	AAAAGAAAAAAAAATAAACAA	s_1: TCAAGCAAAAAAAATAAATAAATACCTATGCAA
		s_2: ACAAGCAAACAAAATAAATATCTGTGCAATAT
		s_3: TATGAGCAAACAAAATAAATACCTGTGCAA
hm08r	CGTGCAGTCCCCTTCAT	s_{10}: TATGGTCATGACGTCTGACAGAGC
hm19r	CCCTTCCACCACCCACAG	s_2: CACTTTTAGCTCCTCCCCCCA
hm26r	CCCCCCGCCTCCCGCTCCC	s_3: CCCCGCCTCAGGCTCCCGGGG
		s_7: CTCAGCCTGCCCCTCCCAGGGATTAAG
		s_8: GCGCCGAGGCGTCCCCGAGGCGC

The datasets are from mouse (resp. human) if their names start with "mus" (resp. "hm").

References

1. L.A.A. Meira, V.R. Maximo, A.L. Fazenda, A.F. da Conceicao. *acc-Motif Detection Tool*. arXiv:1203.3415 [cs.DS], Apr 2013.
2. Xin Li, DouglasS. Stones, Haidong Wang, Hualiang Deng, Xiaoguang Liu, Gang Wang. *NetMODE: Network Motif Detection without Nauty*. PLOS ONE, Dec 2012, Vol 7, Iss 12.
3. Qiang Yu, Hongwei Huo, Yipu Zhang, Hongzhi Guo. *PairMotif: A new pattern-driven algorithm for planted (l,d) DNA Motif Search*. PLOS ONE, Oct 2012, Vol 7, Iss 10.

References (cont')

4. Hieu Dinh, S. Rajasekaran, J. Davila. qPMS7: A fast algorithm for Finding (l,d)-Motifs in DNA and protein sequences. PLOS ONE, Jul 2012, Vol 7, Iss 7.
5. Wong E, Baur B, Quader S, Huang CH. *Biological network motif detection: principles and practice*. Oxford University Press, Briefings in Bioinformatics. doi:10.1093/bib/bbr033.
6. Davila J., Balla S., Rajasekaran S. *Fast and practical algorithms for planted (l, d) motif search*. IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol.4, no.4, pp.544-552, Oct.-Dec. 2007.

Thanks for your attention!