# Computational Biology (MAT4930/MAT5932)

## Lecture 3: Branch-and-bound Algorithms

Abdulmelik Mohammed

University of South Florida

01/21/2021

# Gene regulation in eukaryotes

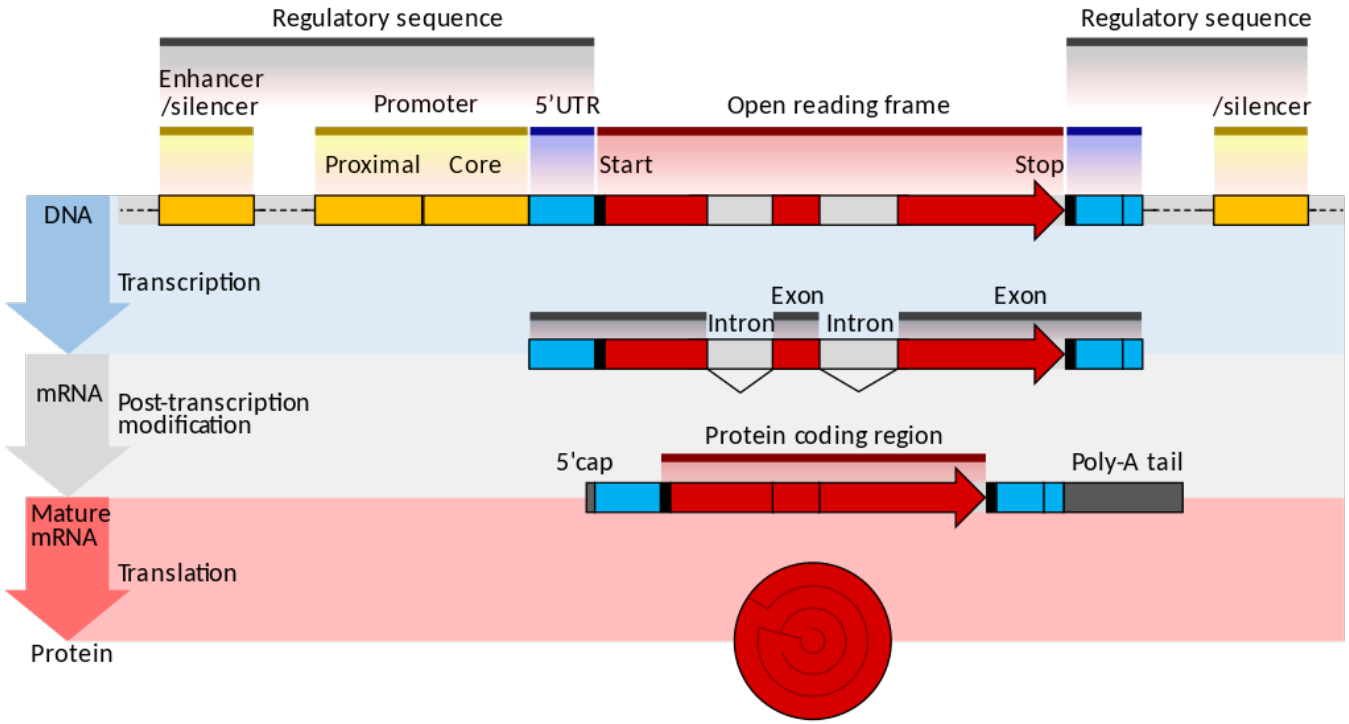


Figure: Image by Thomas Shafee 2015.

- Transcription of a gene is regulated by *transcription factors* that bind *regulatory regions*.

# The transcription factor NF-$\kappa$B homodimer

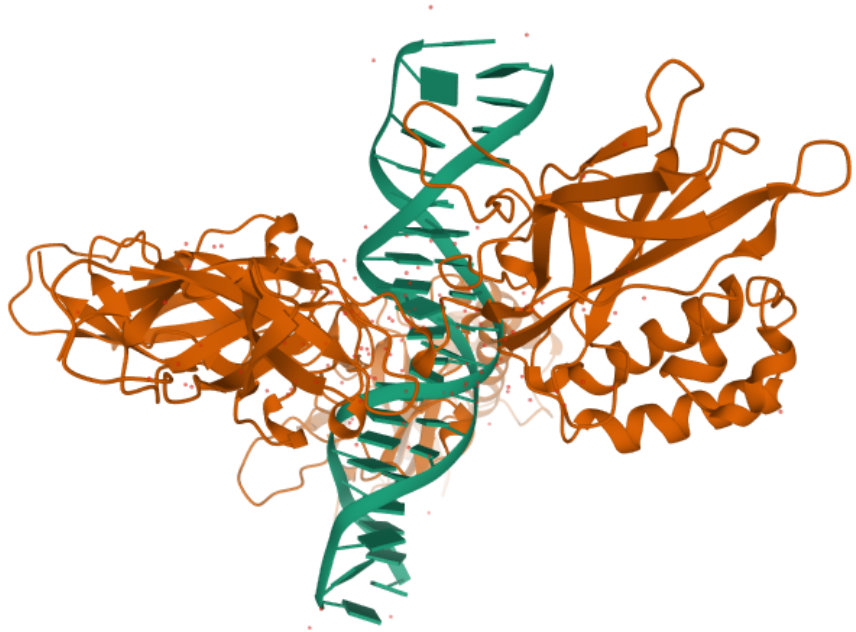

Figure: PBD structure 1SVC (`https://www.rcsb.org/3d-view/1SVC`).

- For e.g. NF-$\kappa$B is active in the switching on of immunity genes in fruit flies when the flies get infected by a pathogen.

# Mechanism of NF-$\kappa$B

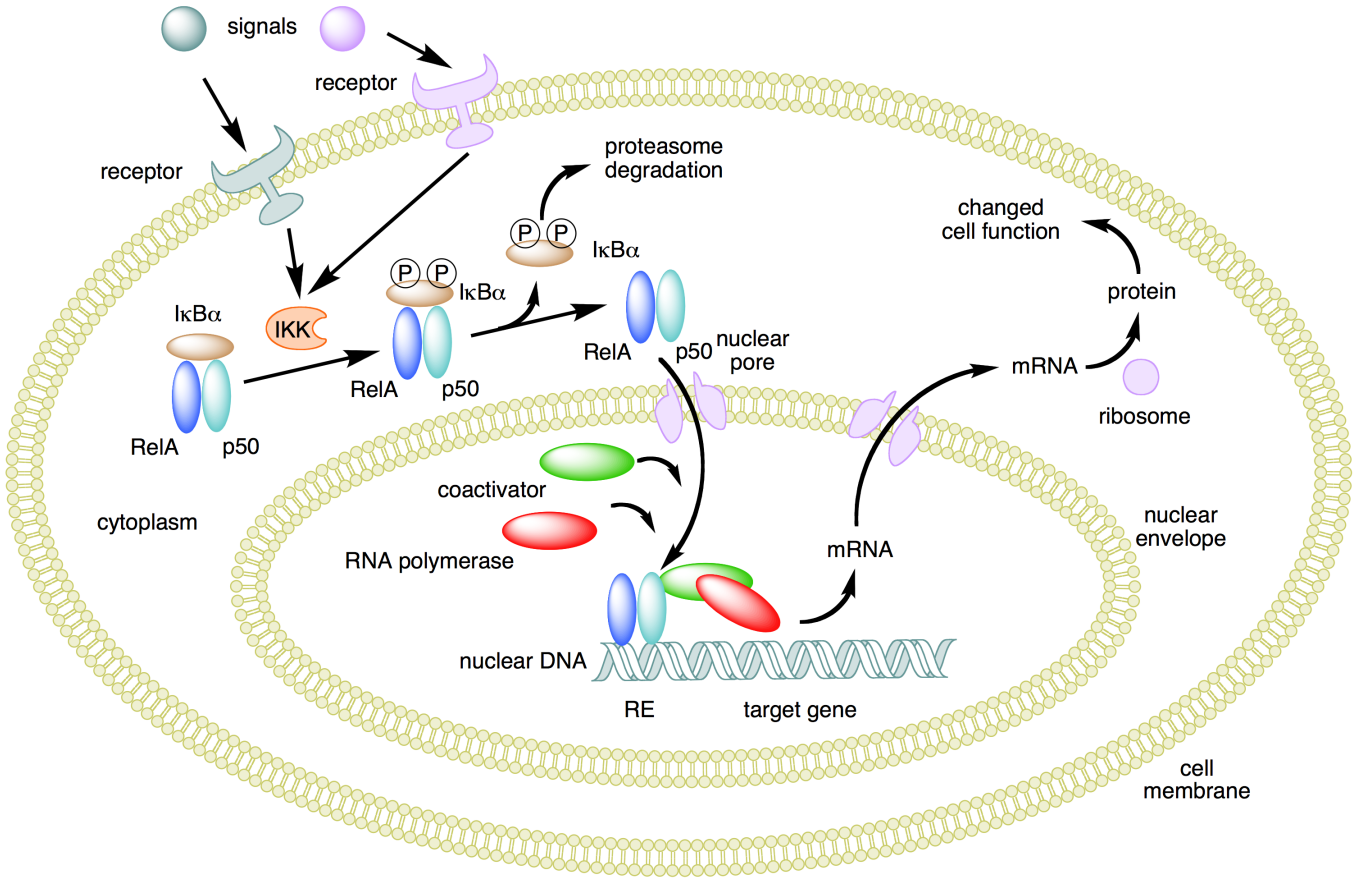


Figure: Image reprinted from Wikipedia 2007.

# Gene regulatory motifs

- The switching is regulated by NF-$\kappa$B binding sites, which have a sequence pattern close to TCGGGGATTTCC upstream of the gene.
- By infecting fruit flies and obtaining the genes (and upstream sequences) that get switched on, we can get a set of sequences in which the motif appears frequently, even though we do not yet know the location and sequence of the motif.
- *Motif finding* is the problem of finding the location and sequence of regulatory motifs from a given genome, or a set of sequences from a genome.

# A collection of putative NF-$\kappa$B binding sites

T C G G G G A T T T C A
A C G G G G A T T T T T
T C G G T A C T T T A C
T T G G G G A C T T T T
C C G G T G A T T C C C
G C G G G G A A T T T C
T C G G G G A T T C C T
T C G G G G A T T C C T
T A G G G G A A C T A C
T C G G G T A T A A A C
T C G G G G G T T T T T
C C G G T G A C T T A C
C C A G G G A C T C C C
A A G G G G A C T T C C
T T G G G G A C T T T T
T T T G G G A G T C C C
T C G G T G A T T T C C
T A G G G G A A G A C C

---

T C G G G G A T T T C C

# Consider a set of seven random sequences

CGGGGCTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAACCAAAGCGGACAAA
GGGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCTC
CTGCTGTACAACTGAGATCATGCTGCTTCAAC
TACATGATCTTTTGTGGATGAGGGAATGATGC

# Suppose we implant the sequence ATGCAACT in the set

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC
TACATGATCTTTTTGATGCAACTTGGATGAGGGAATGATGC

# It would be difficult to identify the implanted sequence

CGGGGCTATGCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAAATGCAACTCCAAAGCGGACAAA
GGATGCAACTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGATGCAACTCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCCATGCAACTTC
CTGCTGTACAACTGAGATCATGCTGCATGCAACTTTCAAC
TACATGATCTTTTGATGCAACTTGGATGAGGGAATGATGC

# In real regulatory motifs there could be additional mutations due to evolution

CGGGGCTATcCAACTGGGTCGTCACATTCCCCTTTCGATA
TTTGAGGGTGCCCAATAAggGCAACTCCAAAGCGGACAAA
GGATGgAtCTGATGCCGTTTGACGACCTAAATCAACGGCC
AAGGAaGCAACcCCAGGAGCGCCTTTGCTGGTTCTACCTG
AATTTTCTAAAAAGATTATAATGTCGGTCtTGgAACTTC
CTGCTGTACAACTGAGATCATGCTGCATGCcAtTTTCAAC
TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC

# Profile matrices and consensus string

- An *l-mer* is a subsequence of length $l$.
- Consider a set of $t$ DNA sequences, each of length $n$.
- The $l$-mers starting at position $\mathbf{s} = (s_1, s_2, \ldots, s_t)$ can be aligned to create a $t \times l$ *alignment matrix* $A(\mathbf{s})$ whose $(i, j)$ entry is the nucleotide in the $(s_i + j - 1)$th element in the $i$th sequence.
- The *profile matrix* $P(\mathbf{s})$ is the $4 \times l$ matrix whose $(i, j)$th element is the number of times nucleotide $i$ (e.g. A=1, T=2, G=3, C=4) appears in column $j$ of $A(\mathbf{s})$.
- The *consensus* string of an alignment $\mathbf{s}$ is the $l$-mer whose $j$th nucleotide is the most frequent nucleotide in the $j$th column of $A(\mathbf{s})$.

# Aligning the random sequences with the mutated implanted sequence

```
                                CGGGGCTATcCAgCTGGGTCGTCACATTCCCCTT...
                    TTTGAGGGTGCCCAATAAggGCAACTCCAAAGCGGACAAA
                                    GGATGgAtCTGATGCCGTTTGACGACCTA...
                                  AAGGAaGCAACcCCAGGAGCGCCTTTGCTGG...
AATTTTCTAAAAAGATTATAATGTCGGTCCtTGgAACTTC
    CTGCTGTACAACTGAGATCATGCTGCATGCcAtTTTCAAC
                    TACATGATCTTTTGATGgcACTTGGATGAGGGAATGATGC
```

Figure: Reprinted from Jones and Pevzner 2004.

# Profile matrix and consensus sequence of mutated implanted sequence

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Alignment** | | A | T | C | C | A | G | C | T |
| | | G | G | G | C | A | A | C | T |
| | | A | T | G | G | A | T | C | T |
| | | A | A | G | C | A | A | C | C |
| | | T | T | G | G | A | A | C | T |
| | | A | T | G | C | C | A | T | T |
| | | A | T | G | G | C | A | C | T |
| **Profile** | **A** | 5 | 1 | 0 | 0 | 5 | 5 | 0 | 0 |
| | **T** | 1 | 5 | 0 | 0 | 0 | 1 | 1 | 6 |
| | **G** | 1 | 1 | 6 | 3 | 0 | 1 | 0 | 0 |
| | **C** | 0 | 0 | 1 | 4 | 2 | 0 | 6 | 1 |
| **Consensus** | | A | T | G | C | A | A | C | T |

Figure: Reprinted from Jones and Pevzner 2004.

# The Motif Finding Problem

- Let $M_{P(\mathbf{s})}(j)$ denote the largest count in the $j$th column of the profile matrix $P(\mathbf{s})$.
- The *consensus score* is defined as:

$$Score(\mathbf{s}, DNA) = \sum_{j=1}^{l} M_{P(\mathbf{s})}(j).$$

- For the starting positions $\mathbf{s} = (8, 19, 3, 5, 31, 27, 15)$ that we have seen, the consensus score is:

$$Score(\mathbf{s}, DNA) = 5 + 5 + 6 + 4 + 5 + 5 + 6 + 6 = 42.$$

- The Motif Finding Problem is then defined as:

**Input:** A $t \times n$ matrix of DNA, and $l$, the length of the motif to search.

**Output:** An array of $t$ starting positions $\mathbf{s} = (s_1, s_2, \ldots, s_t)$ maximizing $Score(\mathbf{s}, DNA)$.

**Problem 1:** Motif Finding Problem.

# Median String Problem

- A formulation with equivalent results is the *Median String Problem*.
- The *Hamming distance* $d_H(v, w)$ between two $l$-mers $v$ and $w$ is the number of nucleotides that differ in the two sequences.
- Let $d_H(v, \mathbf{s})$ denote the sum of the Hamming distances between $v$ and all the $l$-mers starting at positions in $\mathbf{s}$.
- Let $TotalDistance(v, DNA) = \min_{\mathbf{s}}(d_H(v, \mathbf{s}))$ denote the minimum possible total Hamming distance between a given string v and any set of starting positions in the DNA.
- The *median string* is the $l$-mer $v$ that minimizes $TotalDistance(v, DNA)$.
- If $w$ is the consensus string of $\mathbf{s}$, then $d_H(w, \mathbf{s}) = lt - Score(\mathbf{s}, DNA)$.

# Median String Problem

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **A** | **T** | C | **C** | **A** | G | **C** | **T** |
| G | G | **G** | **C** | **A** | **A** | **C** | **T** |
| **A** | **T** | **G** | G | **A** | T | **C** | **T** |
| **A** | A | **G** | **C** | **A** | **A** | **C** | C |
| T | **T** | **G** | G | **A** | **A** | **C** | **T** |
| **A** | **T** | **G** | **C** | C | **A** | T | **T** |
| **A** | **T** | **G** | G | C | **A** | **C** | **T** |

Figure: Bold nucleotides agree with the consensus sequence, while others do not. Image reprinted from Jones and Pevzner, 2004.

- The consensus string minimizes $d_H(v, \mathbf{s})$ over all choices of $v$.
- We have the relation:

$$\min_v \min_{\mathbf{s}} d_H(v, \mathbf{s}) = \min_{\mathbf{s}} \min_v d_H(v, \mathbf{s}) = lt - \max_{\mathbf{s}} Score(\mathbf{s}, DNA).$$

- Thus the median string of the $t$ DNA sequences correspond to the consensus sequence of the solution to the Motif Finding Problem.

# Motif finding vs median string

- In motif finding, the search space contains $(n - l + 1)^t$ possible starting positions.
- In median string, there are $4^l$ possibilities for the median string.
- Typically, the input sequences are long but the motif to find is short. Thus $n \gg l$.
- By reformulating the problem, the search space can be significantly reduced.

# Search trees

- In computational problems, the *search space* is the set of possible solutions among which we search for the output.
- In branch-and-bound and backtracking algorithms, we explore the search space through a *search tree* whose internal nodes represent partial solutions and whose leaves represent the complete candidate solutions.

# Search trees for motif finding and median string

- The search space of both motif finding and median string can be viewed as *L-mers* from a finite alphabet of size $k$.
- In motif finding the $L$-mers are starting positions of the form $(1, 1, \ldots, 1), (1, 1, \ldots, 2), \ldots, (n-l+1, n-l+1, \ldots, n-l+1)$ and $k = n - l + 1$.
- In median string, the $L$-mers are strings of the form $AA\ldots A, AA\ldots T, \ldots, CC\ldots C$, or equivalently $(1, 1, \ldots, 1), (1, 1, \ldots, 2), \ldots, (4, 4\ldots 4)$. Here $k = 4$.
- We next see an example of a search tree for $k = 2$.

# Search tree example

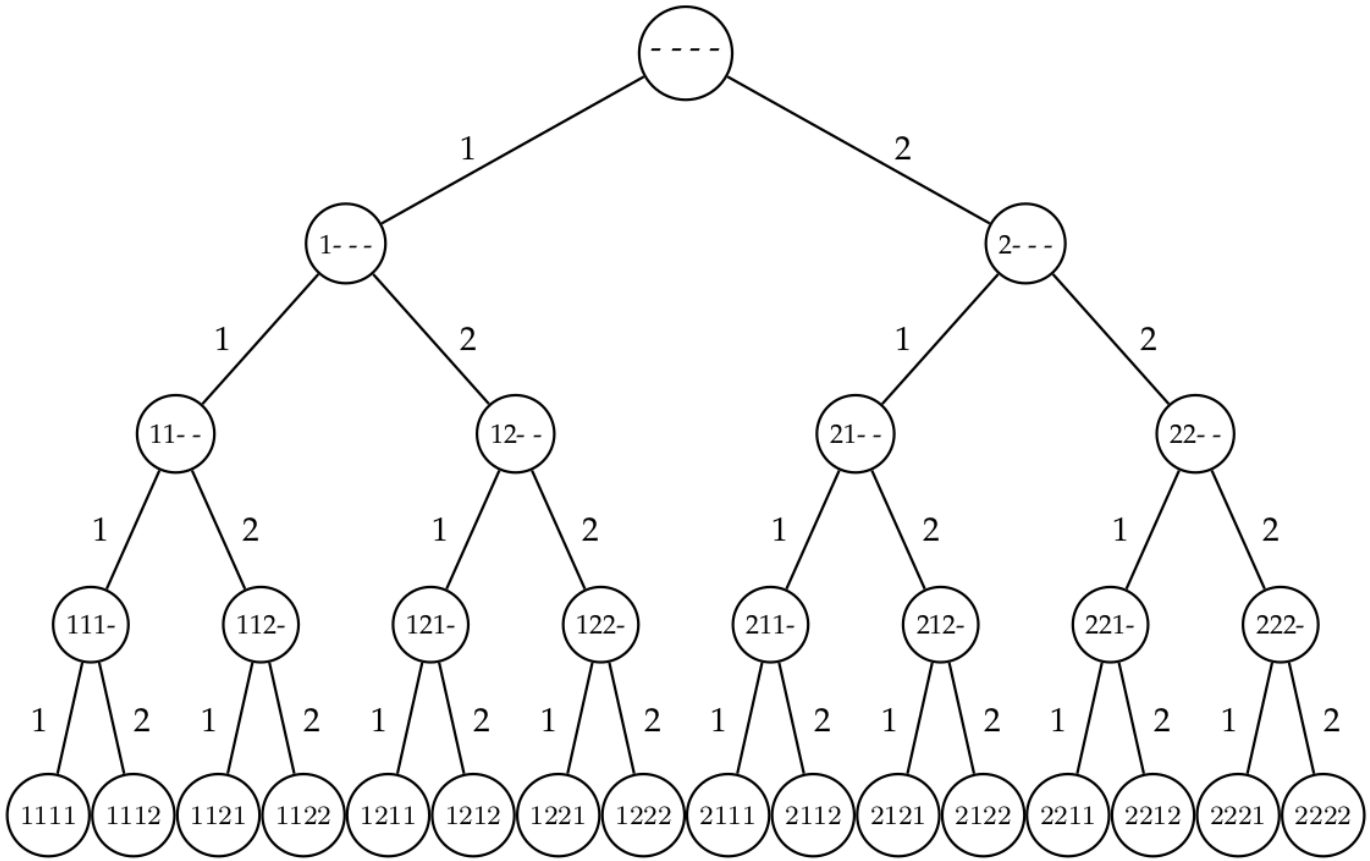


Figure: Search tree for exploring 4-mers from a two letter alphabet. Image reprinted from Jones and Pevzner, 2004.

# Preorder search-tree traversal

- In branch-and-bound, we explore the search tree systematically and thus need an order of traversing the nodes in the search tree.
- A commonly used order is the *preorder*, where we first explore the subtree rooted in the left-most child and when done we explore subtree in the next left child and so on. A preorder traversal of a binary tree is presented below.

```
1 PREORDER(v)
2 |   Output v
3 |   if v has children then
4 |   |   PREORDER(left child of v)
5 |   |   PREORDER(right child of v)
6 |   end
```

**Algorithm 2:** Preorder exploration of binary trees.

# Preorder traversal of search tree for 4-mers from a two letter alphabet

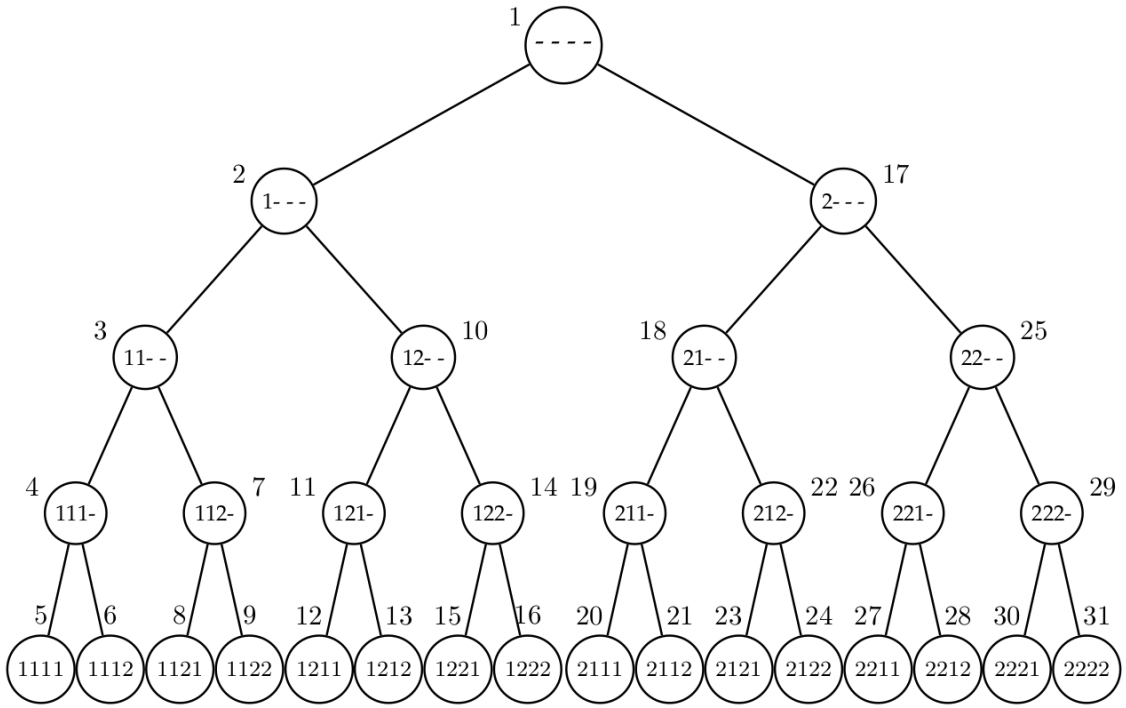


Figure: Image reprinted from Jones and Pevzner, 2004

# Iterative implementation of preorder traversal for motif finding and median string

- As we saw, the $L$-mers can be represented by a vector $\mathbf{a} = (a_1, \ldots, a_L)$ and these are the leaves in the search tree.
- The internal vertices at level $i$ can then be represented by vectors of the form $(a_1, \ldots, a_i)$. An internal vertex is then a prefix of all the nodes in the subtree rooted at that vertex. Thus, the internal vertex can equivalently be represented by the pair $(\mathbf{a}, i)$.
- To get the next vertex in the preorder, we define the subroutine NextVertex which takes the 4 arguments $\mathbf{a}$,$i$,$L$,$k$ and returns a pair $(\mathbf{a}, i)$.

```
1  NextVertex(a, i, L, k)
2      if i < L then
3          a_{i+1} ← 1
4          return (a, i + 1)
5      else
6          for j ← L to 1 do
7              if a_j < k then
8                  a_j ← a_j + 1
9                  return (a, j)
10             end
11         end
12     end
```

**Algorithm 3:** Iterative algorithm for finding next vertex in pre-order.

# Bypassing subtrees in branch-and-bound

- In the branch-and-bound formulation, we aim to skip entire subtrees rooted at certain internal vertices when the most optimistic score we can get from the current $i$-mer cannot exceed the current best score we have maintained thus far.
- For this purpose, we use a subroutine BYPASS to skip the entire subtree rooted at the current vertex under consideration.

```
1 BYPASS(a, i, L, k)
2     for j ← i to 1 do
3         if a_j < k then
4             a_j ← a_j + 1
5             return (a, j)
6         end
7     end
8     return (a, 0)
```

# Partial solutions we aim to bypass in branch-and-bound

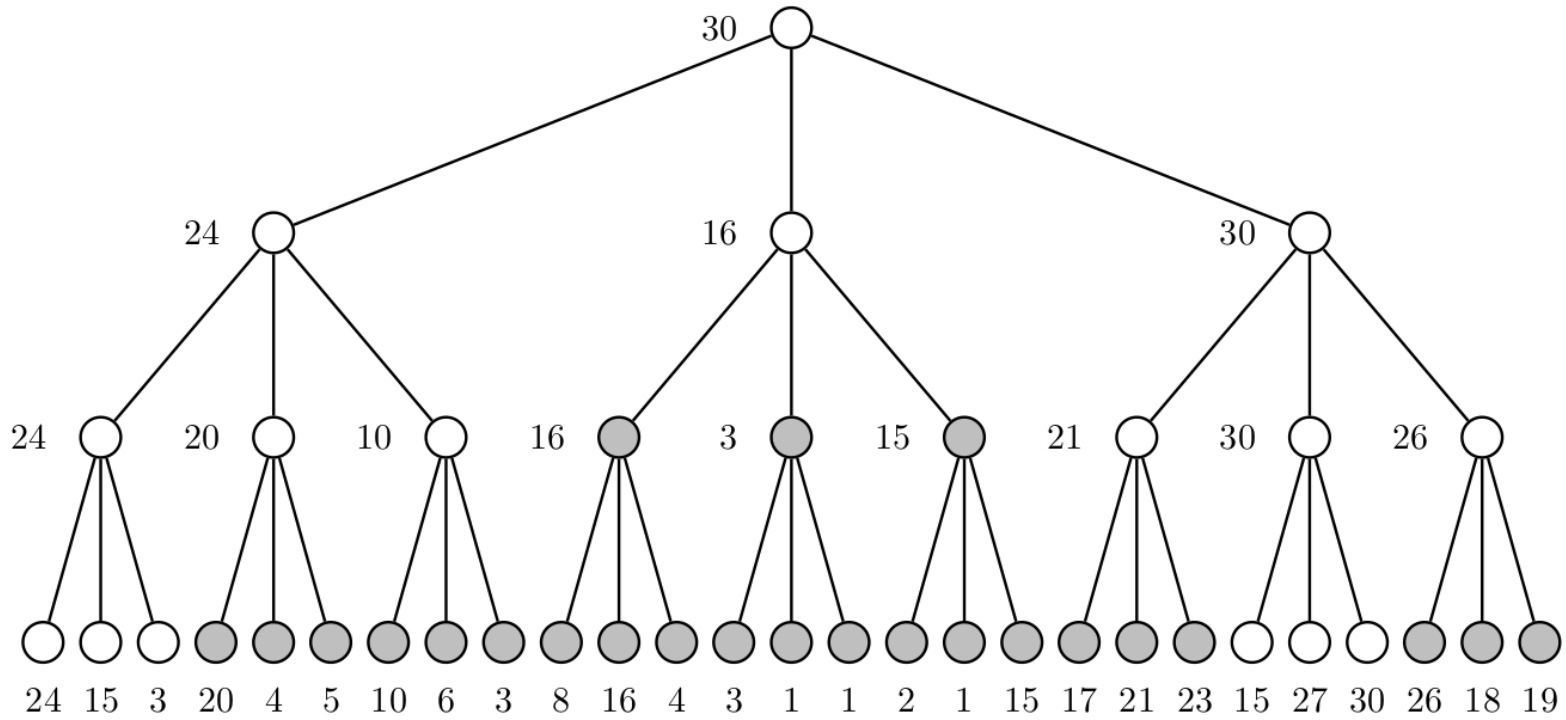


Figure: Image reprinted from Jones and Pevzner, 2004

# Partial scores for motif finding

- Given $\mathbf{s} = (s_1, \ldots, s_t)$, define the *partial consensus score* $Score(\mathbf{s}, i, DNA)$ to be the consensus score of the $i \times l$ alignment matrix involving only the first $i$ DNA sequences with the corresponding starting positions $(s_1, \ldots, s_i, -, \ldots, -)$.
- The remaining sequences can add at most $(t - i) \cdot l$ to the partial score and thus any extension of a partial solution with partial score $Score(\mathbf{s}, i, DNA)$ is at most $Score(\mathbf{s}, i, DNA) + (t - i) \cdot l$.

# Bounding for motif finding

- If the most optimistic extension $Score(\mathbf{s}, i, DNA) + (t - i) \cdot l$ of the partial score is less than the best complete score recorded so far, all the possible extensions of the current partial solution can be discarded.
- In other terms, the search tree can be *pruned* at the internal vertex corresponding to the current partial solution. This is an example of *bounding* in a branch-and-bound algorithm.

# Branch-and-bound motif finding

```
1  BranchAndBoundMotifSearch(DNA, t, n, l)
2      s ← (1, ..., 1)
3      bestScore ← 0
4      i ← 1
5      while i > 0 do
6          if i < t then
7              optimisticScore ← Score(s, i, DNA) + (t − i) · l
8              if optimisticScore < bestScore then
9                  (s, i) ← ByPass(s, i, t, n − l + 1)
10             else
11                 (s, i) ← NextVertex(s, i, t, n − l + 1)
12             end
13         else
14             if Score(s, DNA) > bestScore then
15                 bestScore ← Score(s)
16                 bestMotif ← (s_1, ..., s_t)
17             end
18             (s, i) ← NextVertex(s, i, t, n − l + 1)
19         end
20     end
21     return bestMotif
```

# Branch-and-bound median string search

```
1  BranchAndBoundMedianSearch(DNA, t, n, l)
2      s ← (1, ..., 1)
3      bestDistance ← ∞
4      i ← 1
5      while i > 0 do
6          if i < l then
7              prefix ← sequence corresponding to (s_1, ..., s_i)
8              optimisticDistance ←
                TotalDistance(prefix, DNA)
9              if optimisticDistance > bestDistance then
10                 (s, i) ← ByPass(s, i, t, 4)
11             else
12                 (s, i) ← NextVertex(s, i, t, 4)
13             end
14         else
15             word ← sequence corresponding to (s_1, ..., s_i)
16             if TotalDistance(word, DNA) < bestDistance
                then
17                 bestDistance ← TotalDistance(word, DNA)
18                 bestWord ← word
19             end
20             (s, i) ← NextVertex(s, i, t, 4)
21         end
22     end
23     return bestWord
```

# Analysis

- In the worst-case, we might have to explore the whole search tree in branch-and-bound, which provide no better performance than brute force.
- If we bound at level $i$ in motif finding, we batch reject $(n - l + 1)^{t-i}$ candidate starting positions, which we know are not going to provide an optimal solution.
- In median string, if we bound at level $i$, we reject $4^{l-i}$ candidate $l$-mers.
- Thus, in practice, branch-and-bound algorithms can provide significant run-time improvements compared to brute force algorithms.