Algorithms in Comp. Bio

Motifs & Profiles
Median Strings

Recap

- Still in Ch 4 (brute force)
- HW - a question for you:
  coding, or paper problems?
  (Still coming...)
- Next week: greedy algorithms (Ch. 5 of book)
  & approximations (also in 5)

Formalize: • t DNA sequences, ℓ-mers, n nucleotides each
• Select a position in each: $(s_1, s_2, \ldots, s_t)$
$1 \le s_i \le n - \ell + 1$

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

(a) Superposition of the seven highlighted 8-mers from figure 4.2 (d).

Alignment (Matrix)

	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 matrix

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 string

A	T	G	C	A	A	C	T

(b) The alignment matrix, profile matrix and consensus string formed from the 8-mers starting at positions $\mathbf{s} = (8, 19, 3, 5, 31, 27, 15)$ in figure 4.2 (d).

Notation :

$P(\vec{s})$:= profile matrix wrt starting position vector $\vec{s}$

$M_{P(\vec{s})}(j)$:= largest count in column j of $P(\vec{s})$

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
$P(\vec{s})$ →	**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

$M_{P(\vec{s})}(1) = 5$

$M_{P(\vec{s})}(2) = 5$

$M_{P(\vec{s})}(8) = 6$

Consensus score:

$$Score(\vec{s}, DNA) = \sum_{j=1}^{t} M_{P(s)}(j)$$

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

Here,

$$Score(\vec{s}, DNA) = 5+5+6+4+5+5+6+6$$

Why? Strength of a profile:

$l \cdot t$ means best possible alignment - same letter in each spot

$\frac{lt}{4}$: worst alignment - equal mix of nucleotides per spot

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 $t \times n$ matrix of DNA, and l, the length of the pattern to find.

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

Note: In reality, often use entropy:

Let $p_{i,j}$ be $(i,j)^{th}$ entry in profile.

$$\text{Entropy} = \sum_{j=1}^{l} \sum_{i=1}^{4} \left[\frac{p_{i,j}}{t} \log \frac{p_{i,j}}{t} \right]$$

where t = # sequences

This is more statistically robust measure

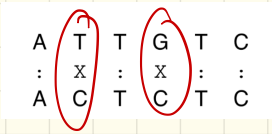
(but algorithm is essentially unchanged)

Another view: Median Strings

2 ℓ-mers v + w

Hamming distance $d_H(v,w) :=$ # of positions that differ

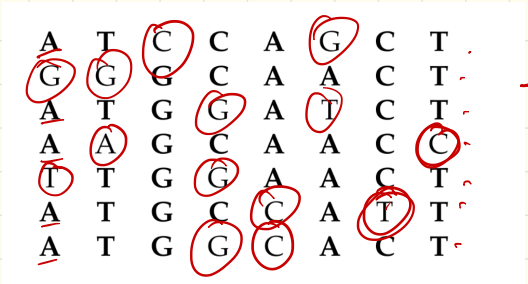
Ex: $d_H(ATTGTC, ACTCTC) = 2$



If we have t ℓ-mers (indicated by $\vec{s}$ again) & one more v,

$$d_H(v, \vec{s}) = \sum_{i=1}^{t} d_H(v, s_i)$$

Ex:



7 8-mers

$v = ATGCAACT$

Finally:

Median String Problem:
Given a set of DNA sequences, find a median string.

Input: A $t \times n$ 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.

But wait...

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

Ham dist

candidate median

Claim: Motif finding + Median String are equivalent.

"Pf":

For any $\vec{s} = (s_1, \ldots, s_t)$, let w be consensus string.

$$d_H(w, \vec{s}) = \ell \cdot t - \text{Score}(\vec{s}, DNA)$$

So: best (max) score that minimizes Hamming distance

Now - back to branch + bound:

Motif finding: $(n-l+1)^t$ vectors $\vec{s}$

$$
\begin{array}{cccccc}
(& 1, & 1, & \ldots, & 1, & 1 \ ) \\
(& 1, & 1, & \ldots, & 1, & 2 \ ) \\
(& 1, & 1, & \ldots, & 1, & 3 \ ) \\
 & & & \vdots & & \\
(& 1, & 1, & \ldots, & 1, & n-l+1 \ ) \\
(& 1, & 1, & \ldots, & 2, & 1 \ ) \\
(& 1, & 1, & \ldots, & 2, & 2 \ ) \\
(& 1, & 1, & \ldots, & 2, & 3 \ ) \\
 & & & \vdots & & \\
(& 1, & 1, & \ldots, & 2, & n-l+1 \ ) \\
 & & & \vdots & & \\
(& n-l+1, & n-l+1, & \ldots, & n-l+1, & 1 \ ) \\
(& n-l+1, & n-l+1, & \ldots, & n-l+1, & 2 \ ) \\
(& n-l+1, & n-l+1, & \ldots, & n-l+1, & 3 \ ) \\
 & & & \vdots & & \\
(& n-l+1, & n-l+1, & \ldots, & n-l+1, & n-l+1 \ )
\end{array}
$$

Median string:

4^l l-mers:

AA$\cdots$ AA	$(1,1,\ldots,1,1)$
AA$\cdots$ AT	$(1,1,\ldots,1,2)$
AA$\cdots$ AG	$(1,1,\ldots,1,3)$
AA$\cdots$ AC	$(1,1,\ldots,1,4)$
AA$\cdots$ TA	$(1,1,\ldots,2,1)$
AA$\cdots$ TT	$(1,1,\ldots,2,2)$
AA$\cdots$ TG	$(1,1,\ldots,2,3)$
AA$\cdots$ TC	$(1,1,\ldots,2,4)$
$\vdots$	$\vdots$
CC$\cdots$ GG	$(4,4,\ldots,3,3)$
CC$\cdots$ GC	$(4,4,\ldots,3,4)$
CC$\cdots$ CA	$(4,4,\ldots,4,1)$
CC$\cdots$ CT	$(4,4,\ldots,4,2)$
CC$\cdots$ CG	$(4,4,\ldots,4,3)$
CC$\cdots$ CC	$(4,4,\ldots,4,4)$

Goal: BFB will need to enumerate these in some order

(along with partial scores, eventually)

k^ℓ ℓ-mers

← here, 4^ℓ

$(1,1,\ldots,1,1)$
$(1,1,\ldots,1,2)$
$(1,1,\ldots,1,3)$
$(1,1,\ldots,1,4)$
$(1,1,\ldots,2,1)$
$(1,1,\ldots,2,2)$
$(1,1,\ldots,2,3)$
$(1,1,\ldots,2,4)$
$\vdots$
$(4,4,\ldots,3,3)$
$(4,4,\ldots,3,4)$
$(4,4,\ldots,4,1)$
$(4,4,\ldots,4,2)$
$(4,4,\ldots,4,3)$
$(4,4,\ldots,4,4)$

Easier to visualize over a 2 "letter" alphabet:

1111 1112 1121 1122 1211 1212 1221 1222 2111 2112 2121 2122 2211 2212 2221 2222

Figure 4.5 All 4-mers in the alphabet of $\{1,2\}$.

↑ 2^4 of these

Note: I'll stick to 2 for now (principle is same)

Pseudocode written for arbitrary k

So: k^ℓ ℓ-mers

$(1, 4, 1, 1)$

If at ℓ-mer $(a_1, a_2, \ldots, a_L)$

how to get to "next"?

$(1, 1, *, *)$

2 1

```
NextLeaf(a, L, k)
1  for i ← L to 1
2      if a_i < k
3          a_i ← a_i + 1
4          return a
5      a_i ← 1
6  return a
```

$(*, *, *, *)$

$(2, 1, 1, 1)$

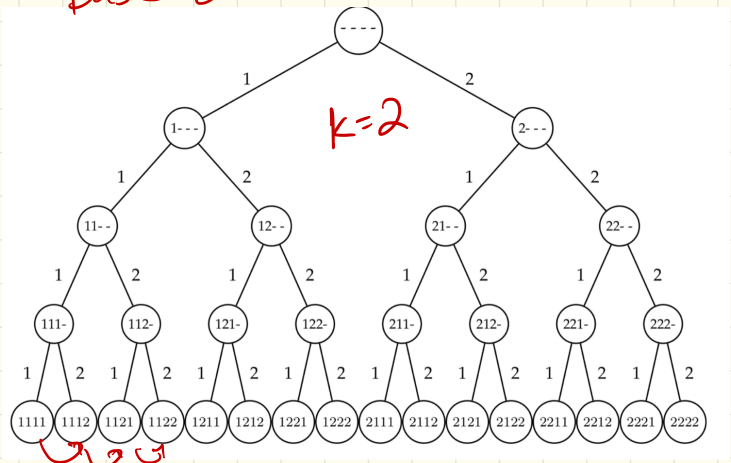
How to get all?

∞-loop

```
AllLeaves(L, k)
1  a ← (1, ..., 1)
2  while forever
3      output a
4      a ← NextLeaf(a, L, k)
5      if a = (1, 1, ..., 1)
6          return
```

Called "next leaf" b/c of B+B tree:

base 2 version:



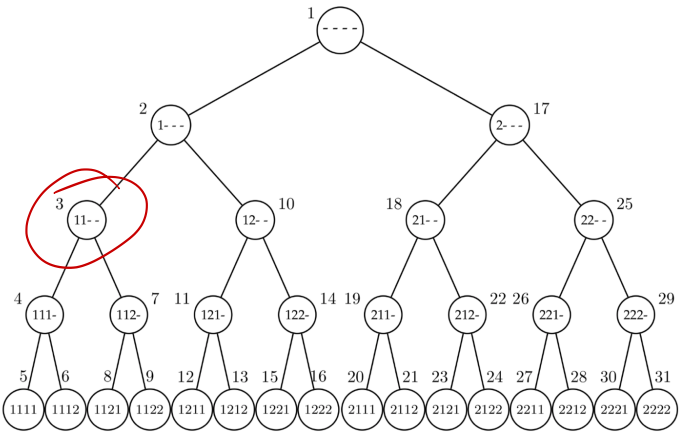

```
NEXTLEAF(a, L, k)
1  for i ← L to 1
2      if a_i < k
3          a_i ← a_i + 1
4          return a
5      a_i ← 1
6  return a
```

← use to list leaves of tree

how to list all vertices?

Pre (or post) order

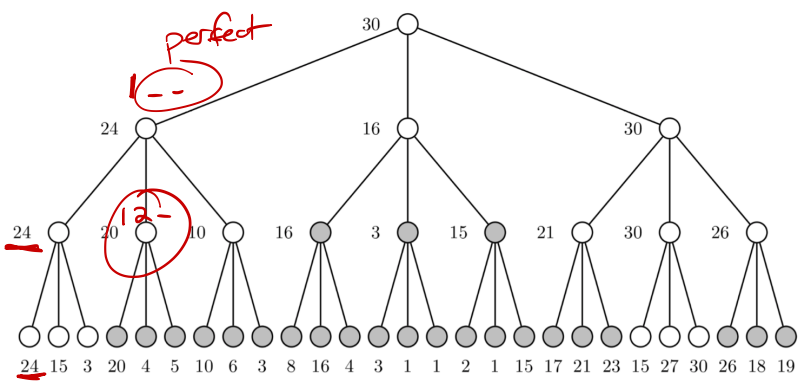
Preorder traversal:




```
PREORDER(v)
1  output v
2  if v has children
3      PREORDER( left child of v )
4      PREORDER( right child of v )
```

```
(-,-,-,-)
(1,-,-,-)
(1,1,-,-)
(1,1,1,-)
(1,1,1,1)
(1,1,1,2)
(1,1,2,-)
(1,1,2,1)
(1,1,2,2)
(1,2,-,-)
(1,2,1,-)
(1,2,1,1)
(1,2,1,2)
(1,2,2,-)
(1,2,2,1)
(1,2,2,2)
(2,-,-,-)
(2,1,-,-)
(2,1,1,-)
(2,1,1,1)
(2,1,1,2)
(2,1,2,-)
(2,1,2,1)
(2,1,2,2)
(2,2,-,-)
(2,2,1,-)
(2,2,1,1)
(2,2,1,2)
(2,2,2,-)
(2,2,2,1)
(2,2,2,2)
```

Need a way to skip:



NEXTLEAF can't do this!

New routine:

```
BYPASS(a, i, k)
1  for j ← i to 1
2      if a_j < k
3          a_j ← a_j + 1
4          return (a, j)
5  return (a, 0)
```

skip subtree at vertex (a,i)

Now: back to motifs

Brute force: first try

```
BRUTEFORCEMOTIFSEARCH(DNA, t, n, l)
1  bestScore ← 0
2  for each (s_1, ..., s_t) from (1, ..., 1) to (n − l + 1, ..., n − l + 1)
3      if Score(s, DNA) > bestScore
4          bestScore ← Score(s, DNA)
5          bestMotif ← (s_1, s_2, ..., s_t)
6  return bestMotif
```

Runtime: $(n-l+1)^t$ possible $\vec{s}$'s

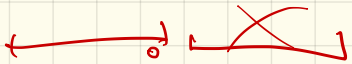
For each, $O(l)$ to get score

Total: $O(ln^t)$

How to do line 2?

```
BRUTEFORCEMOTIFSEARCHAGAIN(DNA, t, n, l)
1  s ← (1, 1, ..., 1)
2  bestScore ← Score(s, DNA)
3  while forever
4      s ← NEXTLEAF(s, t, n − l + 1)
5      if Score(s, DNA) > bestScore
6          bestScore ← Score(s, DNA)
7          bestMotif ← (s_1, s_2, ..., s_t)
8      if s = (1, 1, ..., 1)
9          return bestMotif
```

B+B: Key:



- If first i starting positions $s_1, \ldots, s_i$ are weak, then $s_{i+1}, \ldots, s_t$ might not matter!

Partial score: SCORE(s, i, DNA) is first i rows of alignment matrix:

Here, 5+5+6 (for $i=3$)

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

t

$i=3$

ℓ

remaining rows: at most ~~$(t-i)\cdot\ell$~~ $t(\ell-i)$

Here: 5·7

So: if $SCORE(\vec{s}, i, DNA) + (t-i)l$ is less than current score - skip!

(ie call BYPASS)

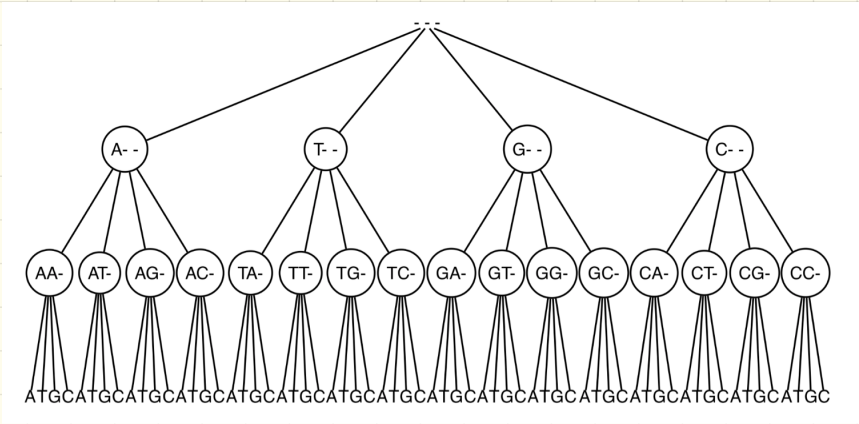
saves $(n-l+1)^{t-i}$ leaves!

```
BRANCHANDBOUNDMOTIFSEARCH(DNA, t, n, l)
 1  s ← (1, ..., 1)
 2  bestScore ← 0
 3  i ← 1
 4  while i > 0
 5      if i < t
 6          optimisticScore ← Score(s, i, DNA) + (t − i) · l
 7          if optimisticScore < bestScore
 8              (s, i) ← BYPASS(s, i, t, n − l + 1)
 9          else
10              (s, i) ← NEXTVERTEX(s, i, t, n − l + 1)
11      else
12          if Score(s, DNA) > bestScore
13              bestScore ← Score(s)
14              bestMotif ← (s_1, s_2, ..., s_t)
15          (s, i) ← NEXTVERTEX(s, i, t, n − l + 1)
16  return bestMotif
```

Runtime? same $O(l n^t)$ (?)

7 to 16 500-1000

Flipping to median strings: How many?



4^l total (+ similar tree)

Brute force:

```
BRUTEFORCEMEDIANSEARCH(DNA, t, n, l)
1  bestWord ← AAA···AA
2  bestDistance ← ∞
3  for each l-mer word from AAA...A to TTT...T
4      if TOTALDISTANCE(word, DNA) < bestDistance
5          bestDistance ← TOTALDISTANCE(word, DNA)
6          bestWord ← word
7  return bestWord
```

4^l

$O(nt)$

Total: $O(nt4^l)$ (not ln^t)

(Note: $l \in (8,15)$, but $n \in (500, 1000)$)

B+B version:

Tree version:

```
SIMPLEMEDIANSEARCH(DNA, t, n, l)
 1  s ← (1, 1, ..., 1)
 2  bestDistance ← ∞
 3  i ← 1
 4  while i > 0
 5      if i < l
 6          (s, i) ← NEXTVERTEX(s, i, l, 4)
 7      else
 8          word ← nucleotide string corresponding to (s_1, s_2, ... s_l)
 9          if TOTALDISTANCE(word, DNA) < bestDistance
10              bestDistance ← TOTALDISTANCE(word, DNA)
11              bestWord ← word
12          (s, i) ← NEXTVERTEX(s, i, l, 4)
13  return bestWord
```

And b+b:

```
BRANCHANDBOUNDMEDIANSEARCH(DNA, t, n, l)
 1  s ← (1, 1, ..., 1)
 2  bestDistance ← ∞
 3  i ← 1
 4  while i > 0
 5      if i < l
 6          prefix ← nucleotide string corresponding to (s_1, s_2, ..., s_i)
 7          optimisticDistance ← TOTALDISTANCE(prefix, DNA)
 8          if optimisticDistance > bestDistance
 9              (s, i) ← BYPASS(s, i, l, 4)
10          else
11              (s, i) ← NEXTVERTEX(s, i, l, 4)
12      else
13          word ← nucleotide string corresponding to (s_1, s_2, ... s_l)
14          if TOTALDISTANCE(word, DNA) < bestDistance
15              bestDistance ← TOTALDISTANCE(word, DNA)
16              bestWord ← word
17          (s, i) ← NEXTVERTEX(s, i, l, 4)
18  return bestWord
```

Same running time