

INTRODUCTION TO ALGORITHMS

THE CHANGE PROBLEM

CONVERT SOME AMOUNT OF MONEY INTO THE FEWEST
NUMBER OF COINS

Input: amount of money, M , in cents

Output: Smallest number of coins whose values add up to M

```
ZARCHANGE( $M$ )
while  $M > 0$ 
   $c$  = Largest coin that is  $\leq$  to  $M$ 
  Give coin  $c$  to customer
   $M = M - c$ 
```

THE CHANGE PROBLEM

ASSUME: - CHANGE IS LESS THAN R1
- COINS: 5c, 10c, 20c, 50c

1. GIVE THE INTEGER PART OF $M/50$
2. M = REMAINDER ($M/50$)
3. GIVE THE INTEGER PART OF $M/20$
4. M = REMAINDER ($M/20$)
5. GIVE THE INTEGER PART OF $M/10$
6. M = REMAINDER ($M/10$)
7. GIVE THE INTEGER PART OF $M/5$

THE CHANGE PROBLEM

PSEUDOCODE

```
ZARCHANGE( $M$ )
 $r = M$ 
 $p_1 = \text{floor}(r/50)$ 
 $r = r - 50 * p_1$ 
 $p_2 = \text{floor}(r/20)$ 
 $r = r - 20 * p_2$ 
 $p_3 = \text{floor}(r/10)$ 
 $r = r - 10 * p_3$ 
 $p_4 = \text{floor}(r/5)$ 
return ( $p_1, p_2, p_3, p_4$ )
```

- lacks elegance
- lacks generality

THE CHANGE PROBLEM

General solution

- d denominations represented by array $\mathbf{c} = (c_1, \dots, c_d)$
- eg: $\mathbf{c} = (50, 20, 10, 5)$

Input: An amount of money, M , and an array of d denominations $\mathbf{c} = (c_1, c_2, \dots, c_d)$ in decreasing order of value ($c_1 > c_2 > \dots > c_d$).

Output: A list of d integers i_1, i_2, \dots, i_d such that $c_1 i_1 + c_2 i_2 + \dots + c_d i_d = M$, and $i_1 + i_2 + \dots + i_d$ is as small as possible.

THE CHANGE PROBLEM

```
BETTERCHANGE( $M, \mathbf{c}, d$ )
 $r = M$ 
for  $k = 1$  to  $d$ 
     $i_k = r/c_k$ 
     $r = r - c_k i_k$ 
return  $(i_1, i_2, \dots, i_d)$ 
```

Problem: not correct in all cases

consider 40 cents with $\mathbf{c} = (25, 20, 10, 5, 1)$

Solution: 1 x 25, 1 x 10, 1 x 5 is not optimal

THE CHANGE PROBLEM

Brute Force

- enumerate all possibilities
- every possible combination of coins with \mathbf{c} denominations
- select the combination with fewest coins

THE CHANGE PROBLEM

Brute Force

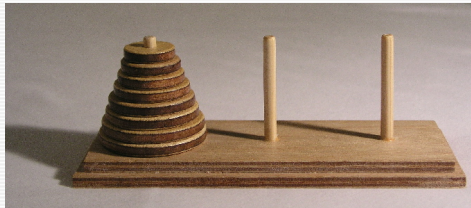
```
BRUTEFORCE( $M, \mathbf{c}, d$ )
 $smallest\_n = 1e+27$ 
for each  $(i_1, \dots, i_d)$  from  $(0, \dots, 0)$  to  $(M/c_1, \dots, M/c_d)$  {
     $value = \sum_{k=1}^d i_k c_k$ 
    if  $value = M$ 
         $n = \sum_{k=1}^d i_k$ 
        if  $n < smallest\_n$ 
             $smallest\_n = n$ 
    }
return  $(i_1, i_2, \dots, i_d)$ 
```

RECURSIVE ALGORITHMS

TOWERS OF HANOI PROBLEM

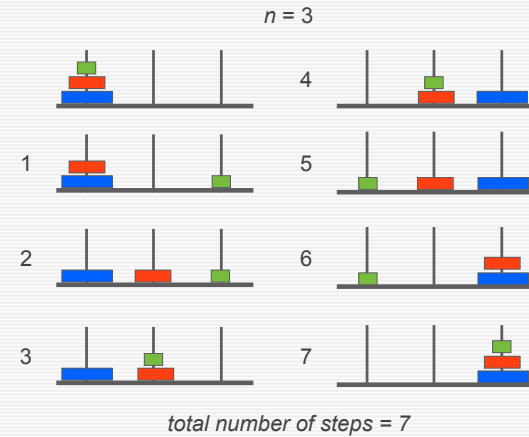
INPUT: AN INTEGER, N , FOR THE NUMBER OF DISCS

OUTPUT: SEQUENCE OF MOVES THAT WILL SOLVE THE N -DISK TOWERS OF HANOI PROBLEM



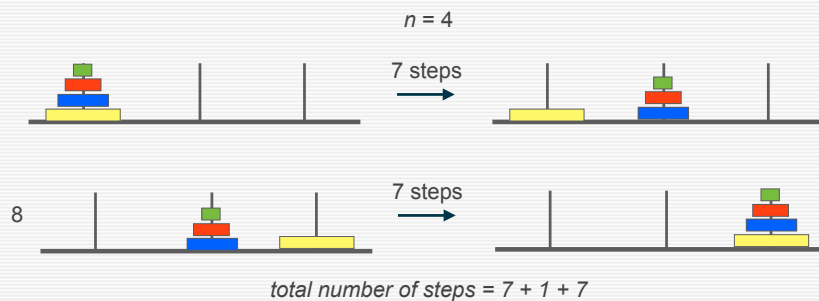
RECURSIVE ALGORITHMS

TOWERS OF HANOI PROBLEM



RECURSIVE ALGORITHMS

TOWERS OF HANOI PROBLEM



To move a stack of size n

1. Move a stack of size $n-1$ to the middle peg
2. Move the n th disk to right peg
3. Move a stack of size $n-1$ to the right peg

RECURSIVE ALGORITHMS

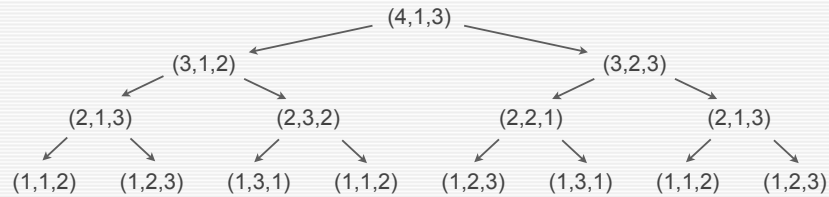
TOWERS OF HANOI PROBLEM

```

HANOITOWERS( $n$ , fromPeg, toPeg)
if  $n = 1$ 
  output "Move disk from peg fromPeg to peg toPeg"
  return
unusedPeg = 6 - fromPeg - toPeg
HANOITOWERS( $n-1$ , fromPeg, unusedPeg)
output "Move disk from peg fromPeg to peg toPeg"
HANOITOWERS( $n-1$ , unusedPeg, toPeg)
return
    
```

RECURSIVE ALGORITHMS

RECURSION TREE



```

HANOITOWERS(n, fromPeg, toPeg)
  if n = 1
    output "Move disk from peg fromPeg to peg toPeg"
    return
  unusedPeg = 6 - fromPeg - toPeg
  HANOITOWERS(n-1, fromPeg, unusedPeg)
  output "Move disk from peg fromPeg to peg toPeg"
  HANOITOWERS(n-1, unusedPeg, toPeg)
  return
    
```

ITERATIVE ALGORITHMS

SORT A LIST OF INTEGERS

Input: a list of n distinct integers $\mathbf{a} = (a_1, a_2, \dots, a_n)$
Output: Sorted list of integers $\mathbf{b} = (b_1, b_2, \dots, b_n)$ from \mathbf{a} such that $b_1 < b_2 < \dots < b_n$.

```

SELECTIONSORT(a, n)
  for i = 1 to n-1
    aj = smallest element among ai, ai+1, ... an
    swap ai and aj
  return a
    
```

(7,92,87,1,4,3,2,6)

(1,92,87,7,4,3,2,6)
 (1,2,87,7,4,3,92,6)
 (1,2,3,7,4,87,92,6)
 (1,2,3,4,7,87,92,6)
 (1,2,3,4,6,87,92,7)
 (1,2,3,4,6,7,92,87)
 (1,2,3,4,6,7,87,92)

ITERATIVE ALGORITHMS

```

SELECTIONSORT(a, n)
  for i = 1 to n-1
    j = INDEXOFMIN(a, i, n)
    swap ai and aj
  return a
    
```

```

INDEXOFMIN(array, first, last)
  idx = first;
  for k = first+1 to last
    if arrayk < arrayidx
      idx = k
  return idx
    
```

```

RECURSIVESELECTIONSORT(a, first, last)
  if first < last
    idx = INDEXOFMIN(a, first, last)
    swap afirst with aidx
    a = RECURSIVESELECTIONSORT(a, first+1, last)
  return a
    
```

```

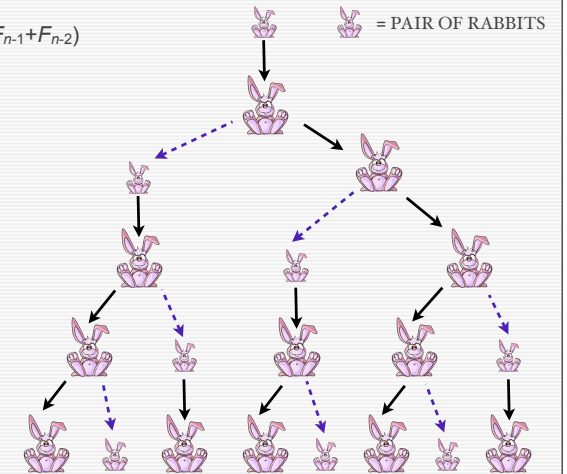
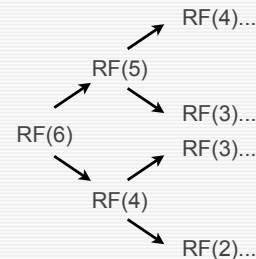
INDEXOFMIN(array, first, last)
  idx = first;
  for k = first+1 to last
    if arrayk < arrayidx
      idx = k
  return idx
    
```

RECURSIVE ALGORITHMS

FIBONACCI SERIES = (1,1,2,3,5,8,...,F_{*n*-1}+F_{*n*-2})

```

RECURSIVEFIBONACCI(n)
  if n = 1 or n = 2
    return 1
  else
    a = RECURSIVEFIBONACCI(n-1)
    b = RECURSIVEFIBONACCI(n-2)
    return a+b
    
```



RECURSIVE ALGORITHMS

FIBONACCI SERIES = (1,1,2,3,5,8,...,F_{n-1}+F_{n-2})

```

FIBONACCI(n)
F1 = 1
F2 = 1
for i = 3 to n
    Fi = Fi-1 + Fi-2
return Fn
    
```

EXECUTION TIME

- count the approximate number of operations

```

BRUTEFORCE(M,c,d)
smallest_n = 1e+27
for each (i1,..., id) from (0,...,0) to (M/c1,...,M/cd) {
    value =
    if value = M
        n =
        if n < smallest_n
            smallest_n = n
}
return(i1,i2,...,id)
    
```

$$\frac{M}{c_1} \frac{M}{c_2} \dots \frac{M}{c_d}$$

$$d \frac{M^d}{c_1 c_2 \dots c_d}$$

- d is a parameter therefore exponential algorithm
- M^k, where k is a constant is polynomial
- M¹: linear, M²: quadratic, M³: cubic

ALGORITHM DESIGN

- What approach is the best to solve a problem
- Trade off of accuracy and speed
 - Exhaustive (Brute-Force)
 - Branch-and-Bound
 - Greedy Algorithms
 - Dynamic Programming
 - Divide-and-Conquer
 - Machine Learning
 - Randomized

FINDING REGULATORY MOTIFS

```

atgaccgggatactgataaaaaaaggggggggctacacattagataaacgtatgaag
tacgttagactcggcgccgcccccctatTTTTTgagcagatttagtgacctggaaaaa
aaatttgagtacaaaacttttccgaataaaaaaaaggggggatgagatccctgggat
gacttaaaaaaaaggggggggtgctctcccgatTTTTgaatgtaggatcattgccagg
gtccgagctgagaattggatgaaaaaaaggggggtccacgcaatcgcaaccaacgcg
gacccaaaggcaagaccgataaaggagatcccttttgcggtaatgtgccgggaggtggt
tacgtaggaagccctaacggacttaataaaaaaaaggggggcttataggccaatcatg
ttcttTgaatgatttaaaaaaaagggggggaccgcttggcgcacccaaattcagtgT
ggcgagcgcaacggTTTTggcccttgTtagaggccccgtaaaaaaaagggggggcaat
tatgagagagctaatactatcgcgTgctgttcataacttgagTtaaaaaaaaggggggc
tggggcacatacaaggagTcttcccttatcagTtaatgctgtatgacactatgtattgg
ccattggctaaaaagcccaacttgacaaaTggaagatagaatccttgcataaaaaaaagg
ggggaccgaaagggaagctggtgagcaacgacagattcttacctgatttagctcgcttc
cggggatctaatagcacgaagcttaaaaaaaaggggggga
    
```

FINDING REGULATORY MOTIFS

atgaccgggatactgat**AAAAAAAGGGGGG**ggcgtacacattagataaacgtatgaag
 tacgttagactcggcgccgaccctatTTTTTgagcagatttagtgacctggaaaaa
 aaatttgagtacaaaactTTTCCgaata**AAAAAAAGGGGGG**atgagatccctgggat
 gactt**AAAAAAAGGGGGG**tgctctccgattTTTTgaatatgtaggacattcgccag
 gtccgagctgagaattggat**AAAAAAAGGGGGG**tccacgcaatcggaaccaacgcg
 gacccaaaggcaagaccgataaaggagatccTTTTgcggaatgtgccgggagctggt
 tacgtaggaagccctaacggacttaat**AAAAAAAGGGGGG**ccttaggtcaatcatg
 ttcttTggaatgatt**AAAAAAAGGGGGG**gaccgcttgccgacccaaattcagtg
 gggcgagcgcaacggtTTTggccctTgttagaggccccgt**AAAAAAAGGGGGG**caat
 tatgagagagctaactctatcgctgctgttcataactTgatt**AAAAAAAGGGGGG**c
 tggggcacatacaagaggagtcttcttatcagttaatgctgtatgacactatgtattgg
 cccattggctaaaagcccaacttgacaaatggaagatagaatcctTgcata**AAAAAAGG**
GGGGaccgaaaggaagctggtgagcaacgacagattcttactgTcattagctcgctc
 cgggatctaatagcacgaagctt**AAAAAAAGGGGGG**a

FINDING REGULATORY MOTIFS

atgaccgggatactgat**AtAgAAcAGGGcGGG**ggcgtacacattagataaacgtatgaag
 tacgttagactcggcgccgaccctatTTTTTgagcagatttagtgacctggaaaaa
 aaatttgagtacaaaactTTTCCgaata**AAAcAAcAGGtGcGG**atgagatccctgggat
 gactt**AAaAAcAAtGGGtG**tgctctccgattTTTTgaatatgtaggacattcgccag
 gtccgagctgagaattggat**AAAaAAAtGGGcGaG**tccacgcaatcggaaccaacgcg
 gacccaaaggcaagaccgataaaggagatccTTTTgcggaatgtgccgggagctggt
 tacgtaggaagccctaacggacttaat**AtAAgAAAGaGGG**aGcttaggtcaatcatg
 ttcttTggaatgatt**AAcAAAgAGGGtGG**caccgcttgccgacccaaattcagtg
 gggcgagcgcaacggtTTTggccctTgttagaggccccgt**AcAAcAAAGtGGG**aGcaat
 tatgagagagctaactctatcgctgctgttcataactTgatt**AAA**t**AAgcGGGcGG**c
 tggggcacatacaagaggagtcttcttatcagttaatgctgtatgacactatgtattgg
 cccattggctaaaagcccaacttgacaaatggaagatagaatcctTgcata**ActAAAAcGG**
GGaGGaccgaaaggaagctggtgagcaacgacagattcttactgTcattagctcgctc
 cgggatctaatagcacgaagctt**AcAAAcAAaGGGG**aGa

FINDING REGULATORY MOTIFS

atgaccgggatactgatatagaacaggcggggcgtacacattagataaacgtatgaag
 tacgttagactcggcgccgaccctatTTTTTgagcagatttagtgacctggaaaaa
 aaatttgagtacaaaactTTTCCgaataaaacaacaggtgcgatgagatccctgggat
 gacttaaaacaatgggtgtgctctccgattTTTTgaatatgtaggacattcgccag
 gtccgagctgagaattggatgaaaaaatggcgagtccacgcaatcggaaccaacgcg
 gacccaaaggcaagaccgataaaggagatccTTTTgcggaatgtgccgggagctggt
 tacgtaggaagccctaacggacttaataaagaaaggagcttaggtcaatcatg
 ttcttTggaatgatttaacaaagggtggcgaccgcttgccgacccaaattcagtg
 gggcgagcgcaacggtTTTggccctTgttagaggccccgtacaacaagtgggagcaat
 tatgagagagctaactctatcgctgctgttcataactTgattaaataagcggggcgc
 tggggcacatacaagaggagtcttcttatcagttaatgctgtatgacactatgtattgg
 cccattggctaaaagcccaacttgacaaatggaagatagaatcctTgcatactaaacgg
 ggaggaccgaaaggaagctggtgagcaacgacagattcttactgTcattagctcgctc
 cgggatctaatagcacgaagcttacaacaaaggggaga

FINDING REGULATORY MOTIFS

- For example: Transcription Factor Binding Sites (TFBS)
- We do not know the motif
- We don't know where the motif is located
- Motifs can mutate at non-important bases
- Are there sequence motifs that occur more often than chance?

NF-κB (nuclear factor kappa-light-chain-enhancer of activated B cells) is a protein complex that controls the transcription of DNA. NF-κB is found in almost all animal cell types and is involved in cellular responses to stimuli such as stress, cytokines, free radicals, ultraviolet irradiation, oxidized LDL, and bacterial or viral antigens. NF-κB plays a key role in regulating the immune response to infection. Conversely, incorrect regulation of NF-κB has been linked to cancer, inflammatory and autoimmune diseases, septic shock, viral infection, and improper immune development. NF-κB has also been implicated in processes of synaptic plasticity and memory.



MOTIF FINDING ANALOGY

- Edgar Allan Poe (1809-1849) in "The Gold Bug"
- William Legrand

53+++305)6*;4826)4+.)4+);806*;
 48+8¶60)85;1+(;:†*8+83(88)5*†;46
 (;88*96*?;8)†+(;485);5*†2:†+(;
 4956*2(5*-4)8¶8*;4069285);)6†8)
 4††;1(†9;48081;8:8†1;48†85;4)
 485†528806*81(†9;48;(88;4(†?
 34;48)4†;161;:188;†?;



MOTIF FINDING ANALOGY

- English
- Each symbol corresponds to one letter in the English alphabet
- No punctuation marks are encoded

53+++305)6*;4826)4+.)4+);806*;
 48+8¶60)85;1+(;:†*8+83(88)5*†;46
 (;88*96*?;8)†+(;485);5*†2:†+(;
 4956*2(5*-4)8¶8*;4069285);)6†8)
 4††;1(†9;48081;8:8†1;48†85;4)
 485†528806*81(†9;48;(88;4(†?
 34;48)4†;161;:188;†?;

- frequency of each symbol
- frequency of each letter in the English language
- Map

8	:	4)	+	*	5	6	(!		0	2	9	3	:	?	`	-]	.
34	25	19	16	15	14	12	11	9	8	7	6	5	5	4	4	3	2	1	1	1

e t a o i n s r h l d c u m f p g w y b v k x j q z

most frequent

least frequent

MOTIF FINDING ANALOGY

sfiiifcsoorntaeuroaikoaiotecrntaeleyrcooestvenpinelefheeosnitarhteenmrn
 wteonihtaesotsnlpnihtamsrnuhsnbaoyentacrmuesotorleoaiidhtimtaeced
 tepeidtaelestaoaeslsueecrnedhimtaetheetahiwfa taeoaiidrdtpdeetiwt

53+++305)6*;4826)4+.)4+);806*;
 48+8¶60)85;1+(;:†*8+83(88)5*†;46
 (;88*96*?;8)†+(;485);5*†2:†+(;
 4956*2(5*-4)8¶8*;4069285);)6†8)
 4††;1(†9;48081;8:8†1;48†85;4)
 485†528806*81(†9;48;(88;4(†?
 34;48)4†;161;:188;†?;

- frequency of *n*-tuples
- "The" > "48"
- substitute all occurrences of ; 4 8

53+++305)6*the26)h+.)h+)te06*thee`60))e5]e*:*e!e3(ee)5*!t h6(tee*96*?
 te)*+(the5)5*!2:*+(th956*2(5*h)e`e*th0692e5)t)6!e)h+++t1(+9the0e1te:e
 +1thele5th)he5!52ee06*e1(+9thet(eeth(+?3hthe)h+t161t:1eet+?t

MOTIF FINDING ANALOGY

53+++305)6*the26)h+.)h+)te06*thee`60))e5]e*:*e!e3(ee)5*!t h6(tee*96*?
 te)*+(the5)5*!2:*+(th956*2(5*h)e`e*th0692e5)t)6!e)h+++t1(+9the0e1te:e
 +1thele5th)he5!52ee06*e1(+9thet(eeth(+?3hthe)h+t161t:1eet+?t

- "the(ee" : "thetree",
- Therefore "e" maps to "r"
- ... and "th(+?3h" > "thr+?3h"
- what are "+", "?", "3"?

AGOODGLASSINTHEBISHOPSHOSTELINTHEDEVILSSEATWENYONEDEGREESANDTHIRTE
 ENMINUTESNORTHEASTANDBYNORTHMAINBRANCHSEVENTHLIMBEASTSIDESHOOTFRO
 MTHELEFTEYEOFTHEDEATHSHEADABEELINEFROMTHETREETHROUGHTHESHOTFIFTYFE
 ETOUT

A good glass in the bishop's hostel in the devil's seat forty-one degrees and thirteen minutes
 northeast and by north main branch seventh limb east side shoot from the left eye of the death's-
 head a bee line from the tree through the shot fifty feet out.

MOTIF FINDING ANALOGY

The Gold Bug	Finding Motifs
Symbols	Nucleotides
Words	Motifs
Analyze frequencies of n -tuples	
Knowing some n -tuples reduces complexity	
complete dictionary	incomplete dictionary
grammar	no standard grammar
all n -tuples of symbols make up words	not all n -tuples of nucleotides are motifs

PROFILES

```

1      10      20      30      40      50      60
cctgataagaatgcaacttggctatccacgtacgtaggtcctctgtgccaatctatgcggtttccaacca
agtactgggtgtacatttgaatgcaacttacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccatgcaactctgatgtataagacgaaaatctt
agcctccgatgtaaatgcaacttgaactattacctgccaccctattacatcttacgtacgtatataca
ctgttatacaacgcgctcatggcggggtatgcggttttggctcgtcgtacgctcgatcgtaatgcaactc
    
```

- find a sequence of length (l) = 8 that is repeated in each of the sequences

```

1      10      20      30      40      50      60
cctgatagaatgcaacttggctatccacgtacgtaggtcctctgtgccaatctatgcggtttccaacca
agtactgggtgtacatttgaatgcaacttacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccatgcaactctgatgtataagacgaaaatctt
agcctccgatgtaaatgcaacttgaactattacctgccaccctattacatcttacgtacgtatataca
ctgttatacaacgcgctcatggcggggtatgcggttttggctcgtcgtacgctcgatcgtaatgcaactc
    
```

total nucleotides = $68 \times 5 = 340$

PROFILES

- find a sequence of length (l) = 8 that is repeated in each of the sequences and allow some sites to be mutated

```

1      10      20      30      40      50      60
cctgatagaagcgagcttggctatccacgtacgtaggtcctctgtgccaatctatgcggtttccaacca
agtactgggtgtacatttgaatgcaacctacaccggcaacctgaaacaaacgctcagaaccagaagtgc
aaacgtacgtgcaccctctttcttcgtggctctggccatggagctctgatgtataagacgaaaatctt
agcctccgatgtaaatgcatcgtgtaactattacctgccaccctattacatcttacgtacgtatataca
ctgttatacaacgcgctcatggcggggtatgcggttttggctcgtcgtacgctcgatcgtaatgtagctc
    
```

- the string `atgcaact` cannot be used as a search string
- use a profile matrix

PROFILES

```

ACGCAGCT
TTGCAACC
ATGGACCT
ATGCATCC
ATGTAGCT
    
```

- we knew the starting position of the motif
- we can construct profile matrices for all starting positions of an 8-mer string

A	4	0	0	0	5	1	0	0
C	0	1	0	3	0	1	5	2
G	0	0	5	1	0	2	0	0
T	1	4	0	1	0	1	0	3
	A	T	G	C	A	G	C	T

Motif Finding Problem: Find the starting position, s , corresponding to the most conserved profile

where $\mathbf{s} = (s_1, s_2, \dots, s_l)$ with $1 \leq s_i \leq n-l+1$

MOTIF FINDING

ACGCAGCT
TTGCAACC
ATGGACCT
ATGCATCC
ATGTAGCT

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

- best score = lt , where t is the number of sequences
- worst score = $lt/4$

$P(s) =$

A	4	0	0	0	5	1	0	0
C	0	1	0	3	0	1	5	2
G	0	0	5	1	0	2	0	0
T	1	4	0	1	0	1	0	3
consensus	A	T	G	C	A	G	C	T
$M_{P(s)}(j)$	4	4	5	3	5	2	5	3

$$= 4 + 4 + 5 + 3 + 5 + 2 + 5 + 3$$

$$= 31$$

$$\frac{lt}{4} = 10 \quad lt = 40$$

MOTIF FINDING

Input: A $t \times n$ matrix of DNA and length, l , of a motif to find

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

- compute scores for each possible combination of starting positions

MOTIF FINDING

t : number of sample DNA sequences

n : length of each DNA sequence

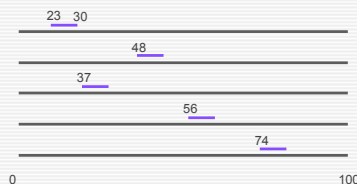
DNA : sample of sequences ($t \times n$ array)

l : length of motif (l -mer)

s_i : starting position of an l -mer in sequence i

$\mathbf{s} = (s_1, s_2, \dots, s_t)$: array of motif starting positions in t sequences

$t = 5$
 $n = 100$
 $l = 8$
 $\mathbf{s} = (23, 48, 37, 56, 74)$



MOTIF FINDING

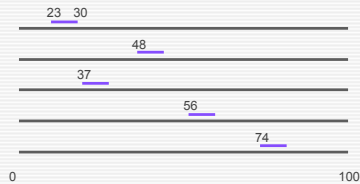
```

BRUTEFORCEMOTIFSEARCH(DNA, t, n, l)
  bestScore = 0
  for each (s1, s2, ..., st) from (1, 1, ..., 1) to (n-l+1, ..., n-l+1) {
    if Score(s, DNA) > bestScore
      bestScore = Score(s, DNA)
      bestMotif = (s1, s2, ..., st)
  }
  return(bestMotif)
    
```

- $(n-l+1)$ positions for each of t sequences
- $(n-l+1)^t$ sets of starting positions \mathbf{s}
- for each set, \mathbf{s} , the Score function makes l operations (i.e. l -mer)
- $l(n-l+1)^t = O(\ln^t)$
- $t=5, n=1000, l=8$ has approx. $7.7e+15$ operations!

MOTIF FINDING

- Reformulate *Motif Search* as a *Median String* problem
- given two *l*-mers *v* and *w*
- calculate the *Hamming Distance*, the number of positions at which two strings differ
- eg: $d_H(\text{AATCCT}, \text{AACCAT}) = 2$
- given *t* DNA sequences find a pattern of length, *l*, that **minimizes** the number of mutations



given $v = \text{"AATCCT"}$

Total Hamming Distance

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

MOTIF FINDING

- For each DNA sequence *i*, compute all $d_H(v, x)$, where *x* is an *l*-mer with starting position, *s_i*
- Find the minimum of $d_H(v, x)$ among all *l*-mers in sequence *i*
- *TotalDistance(v, DNA)* is the sum of the minimum *Hamming* distances for each DNA sequence *i*

Goal: Given a set of DNA sequences, find a median string

Input: A $t \times n$ matrix, and *l*, the length of the *l*-mer to find

Output: A string *v* of *l* nucleotides that minimizes *TotalDistance(v, DNA)* over all strings of that length

Double Minimization

$\min(\text{all choices of } l\text{-mers } v) \text{ AND } \min(\text{all choices of starting positions } \mathbf{s})$

MOTIF FINDING

```

MEDIANSTRINGSEARCH(DNA, t, n, l)
  bestWord = AA...A
  bestDistance = 1e+27
  for each l-mer v from AAA...A to TTT...T {
    if TotalDistance(v, DNA) < bestDistance
      bestDistance = TotalDistance(v, DNA)
      bestWord = v
  }
  return bestWord
    
```

- *Motif Finding* needs to examine $(n-l+1)^t$ combinations for \mathbf{s}
- *Median String* examines 4^l combinations for *v*

MOTIF FINDING

- given *consensus* = ATGCAGC
- *MotifSearch* and *MedianString* are equivalent

ACGCAGCT
 TTGCAACC
 ATGGACCT
 ATGCATCC
 ATGTAGCT

A	4	0	0	0	5	1	0	0
C	0	1	0	3	0	1	5	2
G	0	0	5	1	0	2	0	0
T	1	4	0	1	0	1	0	3
consensus	A	T	G	C	A	G	C	T
Score _(j)	4	4	5	3	5	2	5	3
TotalDistance _(j)	1	1	0	2	0	3	0	2
Sum	5	5	5	5	5	5	5	5

MOTIF FINDING

- Median String problem
- need to consider all 4^l possible l -mers
- i.e. for $l=4$: *aaaa, aaac, aaag, aaat tttt*
- What is the best way to organize this search?

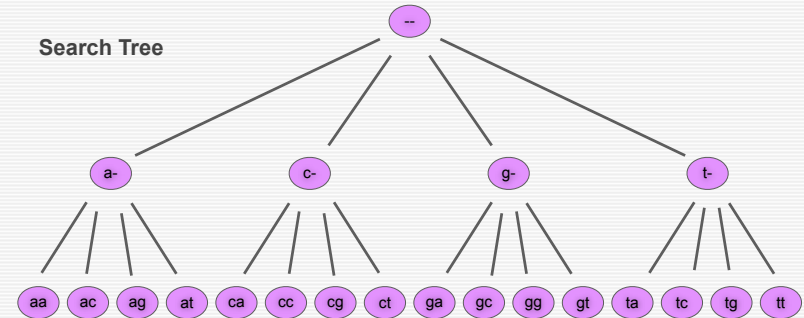
Linked List?



- need to visit all predecessors to get to a sequence

MOTIF FINDING

Search Tree



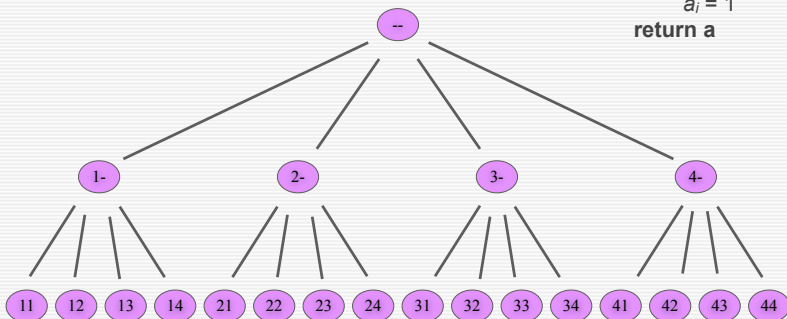
- sequences are the leaves
- parent of a node is the prefix of its children
- move to next leaf
- visit all leaves
- visit the next node
- bypass the children of a node

MOTIF FINDING

- recode ACGT as integers (i.e. A=1, C=2, G=3, T=4)
- $\mathbf{a} = (a_1, a_2, \dots, a_L)$ is an array of non-zero integers
- L = length of array \mathbf{a}
- k = maximum digit value (or alphabet size)

```

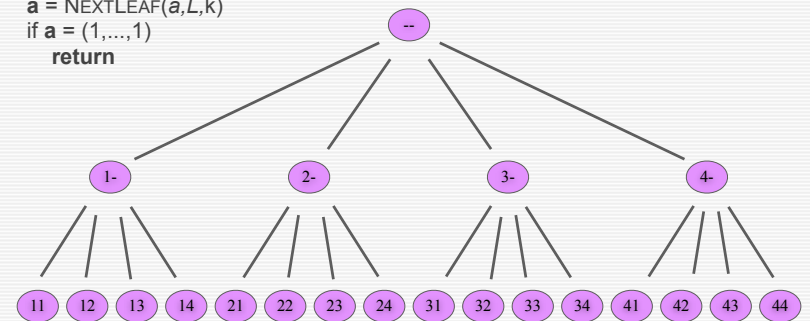
NEXTLEAF(a,L,k)
  for i = L to 1
    if a_i < k
      a_i = a_i + 1
    return a
  a_i = 1
  return a
    
```



MOTIF FINDING

```

ALLEAVES(L,k)
  a = (1,...,1)
  while forever
    output a
    a = NEXTLEAF(a,L,k)
  if a = (1,...,1)
    return
    
```

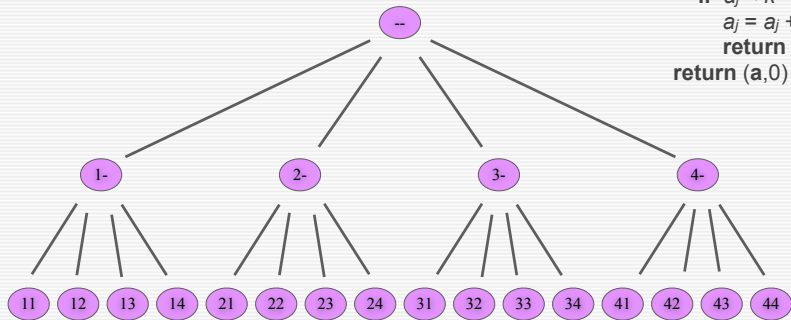


MOTIF FINDING

- $\mathbf{a} = (a_1, a_2, \dots, a_L)$ is an array of non-zero integers
- L = length of array \mathbf{a} (also levels in the tree)
- k = maximum digit value (or alphabet size)

```

NEXTVERTEX( $\mathbf{a}, i, L, k$ )
  if  $i < L$ 
     $a_{i+1} = 1$ 
    return ( $\mathbf{a}, i+1$ )
  else
    for  $j = L$  to 1
      if  $a_j < k$ 
         $a_j = a_j + 1$ 
        return ( $\mathbf{a}, j$ )
    return ( $\mathbf{a}, 0$ )
  
```



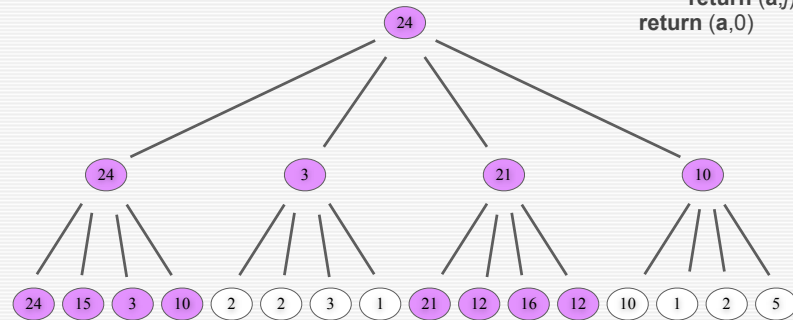
MOTIF FINDING

Branch-and-Bound

- $\mathbf{a} = (a_1, a_2, \dots, a_L)$ is an array of non-zero integers
- L = length of array \mathbf{a}
- k = maximum digit value (or alphabet size)

```

BYPASS( $\mathbf{a}, i, L, k$ )
  for  $j = i$  to 1
    if  $a_j < k$ 
       $a_j = a_j + 1$ 
      return ( $\mathbf{a}, j$ )
  return ( $\mathbf{a}, 0$ )
  
```



MOTIF FINDING

```

BRUTEFORCEMOTIFSEARCH( $DNA, t, n, l$ )
  bestScore = 0
  
```

```

  for each ( $s_1, s_2, \dots, s_t$ ) from (1,1,...,1) to ( $n-l+1, \dots, n-l+1$ ) {
  
```

```

    if Score( $\mathbf{s}, DNA$ ) > bestScore
      bestScore = Score( $\mathbf{s}, DNA$ )
      bestMotif = ( $s_1, s_2, \dots, s_t$ )
  }
  
```

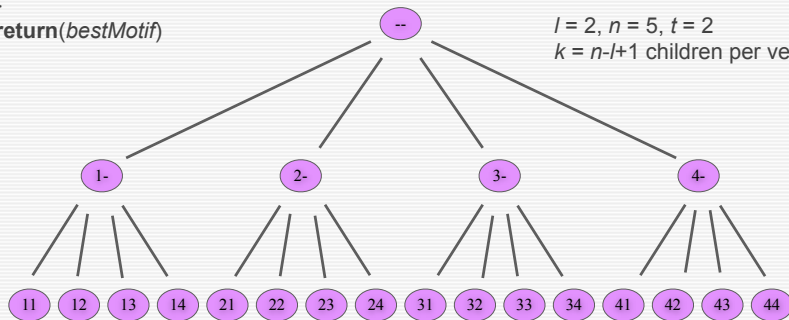
```

  return(bestMotif)
  
```

- $(n-l+1)^t$ sets of starting positions \mathbf{s}

- t levels
- $k = n-l+1$ children per vertex

$l = 2, n = 5, t = 2$
 $k = n-l+1$ children per vertex



MOTIF FINDING

```

BRUTEFORCEMOTIFSEARCHMODIFY( $DNA, t, n, l$ )
  
```

```

   $\mathbf{s} = (1, 1, \dots, 1)$ 
  bestScore = 0
  while forever
     $\mathbf{s} = \text{NEXTLEAF}(\mathbf{s}, t, n-l+1)$ 
    if Score( $\mathbf{s}, DNA$ ) > bestScore
      bestScore = Score( $\mathbf{s}, DNA$ )
      bestMotif = ( $s_1, s_2, \dots, s_t$ )
  }
  if  $\mathbf{s} = (1, 1, \dots, 1)$ 
    return(bestMotif)
  
```

- allows us to move between leaves
- but we want to reduce the amount of computation
- rather move between vertices and use branch and bound

MOTIF FINDING

BRANCHANDBOUNDMOTIFSEARCH(DNA, t, n, l)

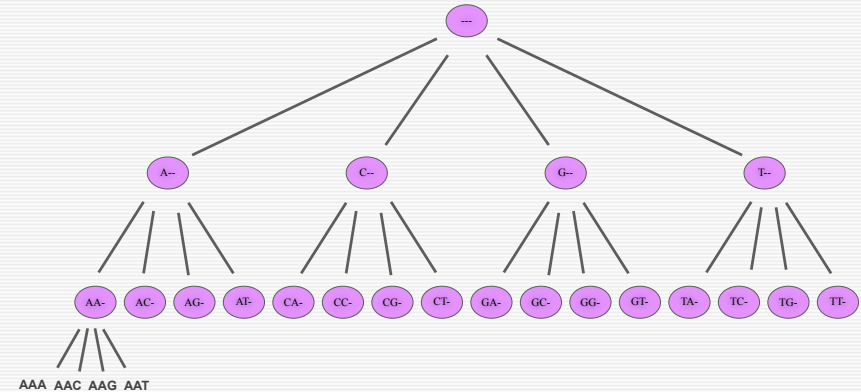
```

s = (1, 1, ..., 1)
bestScore = 0
i = 1
while i > 0
  if i < t
    optimisticScore = Score(s, i, DNA) + (t-i)*l
    if optimisticScore < bestScore
      (s, i) = BYPASS(s, i, t, n-l+1)
    else
      (s, i) = NEXTVERTEX(s, i, t, n-l+1)
  else
    if Score(s, DNA) > bestScore
      bestScore = Score(s, DNA)
      bestMotif = (s1, s2, ..., sl)
    (s, i) = NEXTVERTEX(s, i, t, n-l+1)
return bestMotif
    
```

RECALL: max Score = $l * l$

MOTIF FINDING

- But *Median String Problem* is a better alternative
- consider a string v with length, l , of 3
- Each vertex can only have four children (A,C,G,T) as opposed to $n-l+1$ for *Motif Search*



MOTIF FINDING

BRUTEFORCEMEDIANSEARCH(DNA, t, n, l)

```

bestWord = AAA...A
bestDistance = 1e+27
for each l-mer word from AAA...A to TTT...T
  if TOTALDISTANCE(word, DNA) < bestDistance
    bestDistance = TOTALDISTANCE(word, DNA)
    bestWord = word
return bestWord
    
```

MOTIF FINDING

SIMPLEMEDIANSEARCH(DNA, t, n, l)

```

v = (1, 1, ..., 1) // i.e AA...A
bestDistance = 1e+27
i = 1
while i > 0
  if i < l
    (v, i) = NEXTVERTEX(v, i, l, 4)
  else
    word = nucleotide string for (v1, v2, ..., vl)
    if TOTALDISTANCE(word, DNA) < bestDistance
      bestDistance = TOTALDISTANCE(word, DNA)
      bestWord = word
    (v, i) = NEXTVERTEX(v, i, l, 4)
return bestWord
    
```

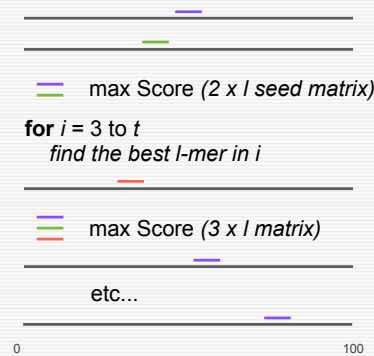
MOTIF FINDING

```

BRANCHANDBOUNDMEDIANSEARCH(DNA,t,n,l)
  v = (1,1,...,1) // i.e AA...A
  bestDistance = 1e+27
  i = 1
  while i > 0
    if i < l
      prefix = nucleotide string for (v1,v2,...vl)
      optimisticDistance = TOTALDISTANCE(prefix,DNA)
      if optimisticDistance > bestDistance
        (v,i) = BYPASS(v,i,l,4)
      else
        (v,i) = NEXTVERTEX(v,i,l,4)
    else
      word = nucleotide string for (v1,v2,...vl)
      if TOTALDISTANCE(word,DNA) < bestDistance
        bestDistance = TOTALDISTANCE(word,DNA)
        bestWord = word
      (v,i) = NEXTVERTEX(v,i,l,4)
  return bestWord
    
```

GREEDY ALGORITHMS

- *MotifSearch* and *MedianString* are exact algorithms
- consider an approximate algorithm
- best solution at each iteration
- scans each DNA sequence only once



```

GREEDYMOTIFSEARCH(DNA,t,n,l)
  bestMotif = (1,1,...,1)
  s = (1,1,...,1)
  for s1 = 1 to n-l+1
    for s2 = 1 to n-l+1
      if Score(s,2,DNA) > Score(bestMotif,2,DNA)
        bestMotif1 = s1
        bestMotif2 = s2
  s1 = bestMotif1
  s2 = bestMotif2
  for i = 3 to t
    for si = 1 to n-l+1
      if Score(s,i,DNA) > Score(bestMotif,i,DNA)
        bestMotifi = si
  si = bestMotifi
  return bestMotif
    
```

GREEDY ALGORITHMS

- may miss optimal motifs
- CONSENSUS is a software tool that uses a greedy algorithm to find motifs
- major difference between this algorithm and CONSENSUS
- starting sequences are assigned randomly
- multiple (n=1000) seed matrices are used
- more on greedy phylogenetic algorithms later in course
- we frequently use greedy Genetic Algorithms
- Recombination detection, Model Selection