Course Staff

Instructor:
• Mohammed El-Kebir (melkebir)
• Office hours: Wednesdays, 3:15-4:15pm in SC 3216

TA:
• Ashwin Ramesh (aramesh7)
• Office hours: Fridays, 11-11:59am in SC 3405

Developing combinatorial algorithms to study all stages of cancer progression.
Course Organization

Course website: 
www.el-kebir.net/teaching/cs466

Syllabus:
• Prerequisites: CS 225 and its prerequisites
• Textbook

Grading:
• 5 written/programming assignments
• Midterm
• Final
• Research project

Piazza: (please sign up)
• https://piazza.com/class - fall2019/cs466
Course Objectives

Learn:
• Learn underlying ideas of common algorithms in bioinformatics.
• Learn to translate a biological problem into a computational problem.
• Learn to read scientific papers, propose and conduct independent research.

Not learn:
• Will not learn to run popular bioinformatics packages.
• Will not learn how to program.
Homework Assignments

• 5 homework assignments
• Each homework assignment is a combination of written/programming exercises
• LaTeX highly recommended for homework assignments
• Python for programming exercises

Late policy:
• Students may request one 3-day extension in the semester for full credit
• Late submission within 3 days 80%
Molecular Biology is the field of biology that studies the composition, structure and interactions of cellular molecules – such as nucleic acids and proteins – that carry out the biological processes essential for the cell's functions and maintenance.

Cellular molecules:
1. DNA
2. RNA
3. Protein

https://www.nature.com/subjects/molecular-biology
DNA

Each strand composed of sequence of covalently bonded **nucleotides** (bases).

Structure of DNA:

A <-> T,  C <-> G  Watson-Crick base-pairing

Four nucleotides:

- A (adenine)
- C (cytosine)
- T (thymine)
- G (guanine)
DNA

Each strand composed of sequence of covalently bonded **nucleotides (bases)**.

5' ...ACGTGACTGAGGACCGTG... 3'
... |||||||||||||||||| |||| |
3' ...TGCACTGACTCCTGGCAC... 5'

Pair of strings from 4 character alphabet

Single string from 4 character alphabet
RNA

- Single-stranded
  - A (adenine)
  - C (cytosine)
  - U (uracil)
  - G (guanine)

- Can fold into structures due to base complementarity.
  \[ A \leftrightarrow U, \quad C \leftrightarrow G \]

- Comes in many flavors:
  - mRNA, tRNA, tmRNA, snRNA, snoRNA, scaRNA, aRNA, asRNA, piwiRNA, etc.
Protein

- String of amino acids: 20 letter alphabet

...DTIGDWNPSFFGIQLVSSVHT TLWYRENAFPVLLGGFSWLWSFNW HNMGGYYYVHYHIGYPMIRCGLHL VPMQFAPFSIARSFALVWHNAPM VLKINPHERQDPVFWPCLYYSVD IRSMHIGYPMIENCYQA...

<table>
<thead>
<tr>
<th>Amino Acid</th>
<th>3-Letters</th>
<th>1-Letter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine</td>
<td>Ala</td>
<td>A</td>
</tr>
<tr>
<td>Arginine</td>
<td>Arg</td>
<td>R</td>
</tr>
<tr>
<td>Asparagine</td>
<td>Asn</td>
<td>N</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>Asp</td>
<td>D</td>
</tr>
<tr>
<td>Cysteine</td>
<td>Cys</td>
<td>C</td>
</tr>
<tr>
<td>Glutamic acid</td>
<td>Glu</td>
<td>E</td>
</tr>
<tr>
<td>Glutamine</td>
<td>Gln</td>
<td>Q</td>
</tr>
<tr>
<td>Glycine</td>
<td>Gly</td>
<td>G</td>
</tr>
<tr>
<td>Histidine</td>
<td>His</td>
<td>H</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>Ile</td>
<td>I</td>
</tr>
<tr>
<td>Leucine</td>
<td>Leu</td>
<td>L</td>
</tr>
<tr>
<td>Lysine</td>
<td>Lys</td>
<td>K</td>
</tr>
<tr>
<td>Methionine</td>
<td>Met</td>
<td>M</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>Phe</td>
<td>F</td>
</tr>
<tr>
<td>Proline</td>
<td>Pro</td>
<td>P</td>
</tr>
<tr>
<td>Serine</td>
<td>Ser</td>
<td>S</td>
</tr>
<tr>
<td>Threonine</td>
<td>Thr</td>
<td>T</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>Trp</td>
<td>W</td>
</tr>
<tr>
<td>Tyrosine</td>
<td>Tyr</td>
<td>Y</td>
</tr>
<tr>
<td>Valine</td>
<td>Val</td>
<td>V</td>
</tr>
</tbody>
</table>
Protein

• String of amino acids: 20 letter alphabet
• Folds into 3D structures to perform various functions in cells
Three fundamental molecules:

1. **DNA**
   - Information storage.

2. **RNA**
   - Old view: Mostly a “messenger”.
   - New view: Performs many important functions.

3. **Protein**
   - Perform most cellular functions (biochemistry, signaling, control, etc.)
Central Dogma of Molecular Biology

DNA → RNA → Protein: The process by which cells “read” the genome

First proposed by Francis Crick in 1956.
Transcription and Translation


Transcription and Translation


http://bioinfo.bisr.res.in/project/crat/pictures/codon.jpg
Computational biology and bioinformatics is an interdisciplinary field that develops and applies computational methods to analyze large collections of biological data, such as genetic sequences, cell populations or protein samples, to make new predictions or discover new biology.

https://www.nature.com/subjects/computational-biology-and-bioinformatics
Technology and Bioinformatics are Transforming Biology

Until late 20th Century

Hypothesis Generation and Validation

21st Century and Beyond

Algorithms

Hypothesis Generation and Validation

High throughput technologies
A Deluge of Data

Cost per Genome

What happened here?

Log Scale

NIH National Human Genome Research Institute

genome.gov/sequencingcosts
A Deluge of Data
A Deluge of Data

Biologists propose to sequence the DNA of all life on Earth

By Elizabeth Pennisi | Feb. 24, 2017, 1:15 PM

Outer ring color scheme:
Red: Completed genome
Light Blue: Low resolution genome
Question: What does it mean that we can sequence a genome?

No technology exists that can sequence a complete (human) genome from end to end!

Genome
Millions -billions nucleotides

Next-generation DNA sequencing

10-100’s million noisy reads
Reads: 30-1000 nucleotides

Making sense of this data absolutely requires the use and development of algorithms!
Why Study Computational Biology?

Interdisciplinary
Biology
Computer Science
Mathematics
Statistics
= FUN!

Best Jobs
1. Actuary
2. Audiologist
3. Mathematician
4. Statistician
5. Biomedical Engineer
6. Data Scientist
7. Dental Hygienist
8. Software Engineer
9. Occupational Therapist
10. Computer Systems Analyst

Worst Jobs
200. Newspaper reporter
199. Lumberjack
198. Enlisted Military Personnel
197. Cook
196. Broadcaster
195. Photojournalist
194. Corrections Officer
193. Taxi Driver
192. Firefighter
191. Mail Carrier

Why choose just 1?

“I can’t be as confident about computer science as I can about biology. *Biology easily has 500 years of exciting problems to work on. It’s at that level.*”

Donald Knuth
Professor emeritus of Computer Science at Stanford University
Turing Award winner
“father of the analysis of algorithms.”
**Question:** How do we compare two genes/genomes?

**Human Genome:**
...ACTCGACTGAGAGGATTTCGAGCATGA...
≈3.2 x 10^9 bp

**Mouse Genome:**
...ACTCAACTGAGATTCGAGCTTCAATGA...
≈2.8 x 10^9 bp
Course Topic #2: Genome Assembly

Question: How do we put all the pieces back together?
Course Topic #3: Phylogenetics

Phylogenetic Tree of Life

Question: Can we reconstruct the evolutionary history of different species?

Question: Can we recover how a tumor has evolved over time?

https://en.wikipedia.org/wiki/Phylogenetic_tree

https://scientificbsides.wordpress.com/2014/06/09/infering-tumour-evolution-2-comparison-to-classical-phylogenetics/
Question: How do we start to make sense of all these sequences?

Suffix Trees

Motif Finding

Burrows Wheeler Transform

Course Topic #4: Pattern Matching

Question: How do we start to make sense of all these sequences?

Suffix Trees

Burrows Wheeler Transform

Motif Finding

Question: How can we analyze available data to determine what drives tumor growth and how to treat or prevent it?
Course Topics

1. Sequence alignment
   *How do we compare two genes/genomes?*

2. Genome assembly
   *How do we put all the pieces back together?*

3. Phylogenetics
   *What is the evolutionary history of different sequences?*

4. Pattern matching
   *How do we start to make sense out of all these sequences?*

5. Cancer genomics
   *How do we identify what drives tumor growth and how to treat/prevent it?*
Course Topics

1. Sequence alignment
   Dynamic programming: edit distance

2. Genome assembly
   Graphs: de Bruijn graph, Eulerian and Hamiltonian paths

3. Phylogenetics
   Trees and distances: distance matrices, neighbor joining, hierarchical clustering.
   Phylogenies: Sankoff/Fitch algorithms, perfect phylogeny and compatibility

4. Pattern matching
   Suffix trees/arrays. Burrows-Wheeler transform, Hidden Markov Models (HMMs)

5. Cancer genomics
   Cancer phylogenies: Integer linear optimization and graph algorithms
Problem $\Pi$ with instance $X$ and solution set $\Pi(X)$:

- **Decision problem:**
  - Is $\Pi(X) = \emptyset$?

- **Optimization problem:**
  - Find $y^* \in \Pi(X)$ s.t. $f(y^*)$ is optimum.

- **Counting problem:**
  - Compute $|\Pi(X)|$.

- **Sampling problem:**
  - Sample uniformly from $\Pi(X)$.

- **Enumeration problem:**
  - Enumerate all solutions in $\Pi(X)$

**Algorithms:**

Set of instructions for solving problem.

- Exact
- Heuristic
The Change Problem

• Suppose we have three coins:

  5 cent  3 cent  1 cent

• What is the minimum number of coins needed to make change for \( M \) cents?
The Change Problem

• Suppose we have three coins:

\[ c = (5 \text{ cent}, 3 \text{ cent}, 1 \text{ cent}) \]

• What is the minimum number of coins needed to make change for \( M \) cents?

Change Problem: Given amount \( M \in \mathbb{N} \setminus \{0\} \) and coins \( c = (c_1, \ldots, c_n) \in \mathbb{N}^n \) s.t. \( c_n = 1 \) and \( c_i \geq c_{i+1} \) for all \( i \in [n-1] = \{1, \ldots, n-1\} \), find \( d = (d_1, \ldots, d_n) \in \mathbb{N}^n \) s.t. (i) \( M = \sum_{i=1}^{n} c_i d_i \) and (ii) \( \sum_{i=1}^{n} d_i \) is minimum
Idea #1: Choose largest coin possible

GreedyChange($M, c_1, ..., c_n$)

1. for $i \leftarrow 1$ to $n$
2. $d_i \leftarrow \lfloor \frac{M}{c_i} \rfloor$
3. $M \leftarrow M - d_i c_i$
Idea #1: Choose largest coin possible

GreedyChange($M, c_1, ..., c_n$)

1. for $i \leftarrow 1$ to $n$
2. $d_i \leftarrow \lfloor M/c_i \rfloor$
3. $M \leftarrow M - d_ic_i$

Is this a good algorithm? Two properties of a good algorithm:
Idea #1: Choose largest coin possible

GreedyChange\((M, c_1, \ldots, c_n)\)

1. for \(i \leftarrow 1\) to \(n\)
2. \(d_i \leftarrow \lfloor \frac{M}{c_i} \rfloor\)
3. \(M \leftarrow M - d_i c_i\)

Is this a good algorithm? Two properties of a good algorithm:

**Correctness:** gives the correct output for any input.
- Works for \(c = (5, 3, 1)\) and \(M = 8\).
- But what about \(c = (5, 4, 1)\) and \(M = 8\)?

**Efficient:** running time of the algorithm does not increase too rapidly with input size.
Idea #2: Don’t be smart, apply brute force

**Change Problem:** Given amount \( M \in \mathbb{N} \setminus \{0\} \) and coins \( \mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{N}^n \) s.t. \( c_n = 1 \) and \( c_i \geq c_{i+1} \) for all \( i \in [n - 1] = \{1, \ldots, n - 1\} \), find \( \mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n \) s.t. (i) \( M = \sum_{i=1}^{n} c_i d_i \) and (ii) \( \sum_{i=1}^{n} d_i \) is minimum.

\[ \mathbf{c} = (5\text{ cent}, 4\text{ cent}, 1\text{ cent}) \]

- Check all possible solutions:
  - \( 11 = 5 + 5 + 1 \)
  - \( 11 = 5 + 4 + 1 + 1 \)
  - \( 11 = 5 + 1 + 1 + 1 + 1 + 1 + 1 \)
  - \( 11 = 4 + 4 + 1 + 1 + 1 \)
  - ...

Correct? yes
Efficient? no

**ExhaustiveChange**\((M, c_1, \ldots, c_n)\)
1. for \((d_1, \ldots, d_n) \in [[M/c_1]] \times \ldots \times [[M/c_n]]\)
2. if \( \sum_{i=1}^{n} c_i d_i = M \)
3. return \((d_1, \ldots, d_n)\)
Idea #3: Recursion

\[ c = (\text{5 cent} , \text{3 cent} , \text{1 cent} ) \]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
</table>

Optimal substructure:
Optimal solution is obtained from optimal solutions of subproblems
Idea #3: Recursion

\[ c = (5, 3, 1) \]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
</table>

• This example can be expressed using a recurrence relation
• Let \( \text{minNumCoins}(M) \) be the minimum number of coins to make change for \( M \) cents

\[
\text{minNumCoins}(M) = \min \left\{ \begin{array}{l}
\text{minNumCoins}(M - 1) + 1, \\
\text{minNumCoins}(M - 3) + 1, \\
\text{minNumCoins}(M - 5) + 1.
\end{array} \right.
\]
Idea #3: Recursion

**Change Problem:** Given amount $M \in \mathbb{N} \setminus \{0\}$ and coins $\mathbf{c} = (c_1, \ldots, c_n) \in \mathbb{N}^n$ s.t. $c_n = 1$ and $c_i \geq c_{i+1}$ for all $i \in [n-1] = \{1, \ldots, n-1\}$, find $\mathbf{d} = (d_1, \ldots, d_n) \in \mathbb{N}^n$ s.t. (i) $M = \sum_{i=1}^{n} c_i d_i$ and (ii) $\sum_{i=1}^{n} d_i$ is minimum

\[
\text{minNumCoins}(M) = \min \left\{ \begin{array}{l} \text{minNumCoins}(M - c_1) + 1, \\ \text{minNumCoins}(M - c_2) + 1, \\ \ldots \\ \text{minNumCoins}(M - c_n) + 1. \end{array} \right. 
\]
Idea #3: Recursion

Given coins $c = (1, 3, 7)$ and amount $M = 77$, find $d = (d_1, ..., d_n) \in \mathbb{N}^n$ such that: (i) $M = \sum_{i=1}^n c_i d_i$ and (ii) $\sum_{i=1}^n d_i$ is minimum.

\[
\begin{align*}
\text{minNumCoins}(77) &= \min \left\{ \text{minNumCoins}(77 - 1) + 1, \right. \\
& \quad \left. \text{minNumCoins}(77 - 3) + 1, \right. \\
& \quad \left. \text{minNumCoins}(77 - 7) + 1, \right. \\
\text{minNumCoins}(76) &= \min \left\{ \text{minNumCoins}(76 - 1) + 1, \right. \\
& \quad \left. \text{minNumCoins}(76 - 3) + 1, \right. \\
& \quad \left. \text{minNumCoins}(76 - 7) + 1, \right. \\
\vdots \\
\text{minNumCoins}(7) &= 1 \\
\text{minNumCoins}(3) &= 1 \\
\text{minNumCoins}(1) &= 1
\end{align*}
\]
Idea #3: Recursion

RecursiveChange\( (M, c_1, ..., c_n) \)

1. if \( M = 0 \)
2. return 0
3. bestNumCoins \( \leftarrow \infty \)
4. for \( i \leftarrow 1 \) to \( n \)
5. if \( M \geq c_i \)
6. numCoins \( \leftarrow \) RecursiveChange\( (M - c_i, c_1, ..., c_n) \)
7. if numCoins + 1 < bestNumCoins
8. bestNumCoins \( \leftarrow \) numCoins + 1
9. return bestNumCoins

Correct but inefficient: Same subproblem is solved many times!
Idea #3: Recursion

RecursiveChange($M, c_1, \ldots, c_n$)

1. if $M = 0$
2. return 0
3. bestNumCoins $\leftarrow \infty$
4. for $i \leftarrow 1$ to $n$
5. if $M \geq c_i$
6. numCoins $\leftarrow$
   RecursiveChange($M - c_i, c_1, \ldots, c_n$)
7. if numCoins $+ 1 < $ bestNumCoins
8. bestNumCoins $\leftarrow$ numCoins $+ 1$
9. return bestNumCoins

Correct but inefficient:
Same subproblem is solved many times!

Solutions:
• Remember previously computed values: memoization
• Bottom up computation: dynamic programming
Idea #4: Solve recurrence with dynamic programming

Fill in table “bottom up”: from smallest to largest.

\[ c = (5, 3, 1) \]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min # coins</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{minNumCoins}(M) = \min \left\{ \text{minNumCoins}(M - 1) + 1, \text{minNumCoins}(M - 3) + 1, \text{minNumCoins}(M - 5) + 1 \right\}.
\]

Only one coin is needed to make change for the values 1, 3 and 5
Idea #4: Solve recurrence with dynamic programming

Fill in table “bottom up”: from smallest to largest.

\[ c = (5, 3, 1) \]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min # coins</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{minNumCoins}(M) = \min \left\{ \text{minNumCoins}(M - 1) + 1, \right.
\text{minNumCoins}(M - 3) + 1, \left. \text{minNumCoins}(M - 5) + 1. \right\}
\]

Two coins are needed to make change for the values 2, 4 and 6.
Idea #4: Solve recurrence with dynamic programming

Fill in table “bottom up”: from smallest to largest.

\[c = (5, 3, 1)\]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min # coins</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{minNumCoins}(M) = \min \left\{ \text{minNumCoins}(M - 1) + 1, \text{minNumCoins}(M - 3) + 1, \text{minNumCoins}(M - 5) + 1 \right\}
\]

Three coins are needed to make change for the value 7
Idea #4: Solve recurrence with dynamic programming

Fill in table “bottom up”: from smallest to largest.

minNumCoins(M) = \min \left\{ \begin{align*}
\minNumCoins(M - 1) + 1, \\
\minNumCoins(M - 3) + 1, \\
\minNumCoins(M - 5) + 1.
\end{align*} \right. 

Optimal substructure: Optimal solution obtained from optimal subsolutions
Idea #4: Solve recurrence with dynamic programming

Fill in table “bottom up”: from smallest to largest.

\[ c = (5, 3, 1) \]

<table>
<thead>
<tr>
<th>Value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min # coins</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\text{minNumCoins}(M) = \min \left\{ \text{minNumCoins}(M - 1) + 1, \right. \\
\left. \text{minNumCoins}(M - 3) + 1, \text{minNumCoins}(M - 5) + 1. \right\}
\]

Optimal substructure: Optimal solution obtained from optimal subsolutions
Idea #4: Solve recurrence with dynamic programming

**Change Problem:** Given amount $M \in \mathbb{N} \setminus \{0\}$ and coins $c = (c_1, ..., c_n) \in \mathbb{N}^n$ s.t. $c_n = 1$ and $c_i \geq c_{i+1}$ for all $i \in [n-1] = \{1, ..., n-1\}$, find $d = (d_1, ..., d_n) \in \mathbb{N}^n$ s.t. (i) $M = \sum_{i=1}^{n} c_id_i$ and (ii) $\sum_{i=1}^{n} d_i$ is minimum

DPChange($M, c_1, ..., c_n$)
1. for $m \leftarrow 1$ to $M$
2. minNumCoins[$m$] $\leftarrow \infty$
3. for $i \leftarrow 1$ to $n$
4. minNumCoins[$c_i$] $\leftarrow 1$
5. for $m \leftarrow 1$ to $M$
6. for $i \leftarrow 1$ to $n$
7. if $m > c_i$
8. minNumCoins[$m$] $\leftarrow \min(1 + \text{minNumCoins}[m - c_i], \text{minNumCoins}[m])$
9. return minNumCoins[$M$]

Correct? yes  
Efficient? yes
### Change Problem:
Given amount $M \in \mathbb{N} \setminus \{0\}$ and coins $c = (c_1, \ldots, c_n) \in \mathbb{N}^n$ s.t. $c_n = 1$ and $c_i \geq c_{i+1}$ for all $i \in [n - 1] = \{1, \ldots, n - 1\}$, find $d = (d_1, \ldots, d_n) \in \mathbb{N}^n$ s.t. (i) $M = \sum_{i=1}^{n} c_i d_i$ and (ii) $\sum_{i=1}^{n} d_i$ is minimum.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Correct?</th>
<th>Efficient?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy algorithm [GreedyChange]</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Exhaustive enumeration [ExhaustiveChange]</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Recursive algorithm [RecursiveChange]</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Dynamic programming [DPChange]</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Summary

• DNA, RNA and proteins are sequences
  • Central dogma of molecular biology: DNA -> RNA -> protein

• Problem != algorithm

• Different algorithm techniques
  • Greedy
  • Exhaustive search/brute force
  • Recursive algorithm
  • Dynamic programming algorithm

• Reading:
  • “Biology for Computer Scientists” by Lawrence Hunter
    (http://www.el-kebir.net/teaching/CS466/Hunter_BIO_CS.pdf)
  • Jones and Pevzner: Chapters 2.1, 2.3, 2.4, 6.2
Sources

- CS 362 by Layla Oesper (Carleton College)
- CS 1810 by Ben Raphael (Brown/Princeton University)
- An Introduction to Bioinformatics Algorithms book (Jones and Pevzner)