Part 2: Sequence search and comparison
Some books


Sequence alignment
Sequence comparison

- Sequence comparison: most ubiquitous task in bioinformatics (*comparative genomics*)
  - genome analysis: gene prediction, phylogeny reconstruction, repeats, …
  - RNA analysis
  - protein analysis
- *Main assumptions*:
  - similar sequences correspond to similar biological functions
  - similar sequences witness phylogenetic proximity
  - similar sequences fold to similar structures
Example: insulin

- **elephant**

- **hamster**

- **whale**

- **alligator**
Another example

Sequence alignment

- Given two sequences RDISLVKNAGI and RNILVSDAKNVGI

3 types of columns corresponding to 3 elementary evolutionary events
- matches
- substitution (mismatch)
- insertion, deletion (indel)

Assign a score (positive or negative) to each event. Alignment score = sum of scores over all columns. Optimal alignment = one that maximizes the score
Sequence alignment: scoring

- sequences: RDISLVLKAGI, RNILVSDAKNVGI
- scoring function:

<table>
<thead>
<tr>
<th>Mismatch</th>
<th>Match</th>
<th>Indel</th>
</tr>
</thead>
<tbody>
<tr>
<td>AV, LD : 0</td>
<td>R, K : 5</td>
<td>-5</td>
</tr>
</tbody>
</table>

Score=19
Score=-11
Score=25
Sequence alignment: scoring

- BLOSUM62 matrix for protein sequences

|     | C   | S   | T   | P   | A   | G   | N   | D   | E   | Q   | H   | R   | K   | M   | I   | L   | V   | F   | Y   | W   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| C   | 9   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| S   | -1  | 4   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| T   | -1  | 1   | 5   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| P   | -3  | -1  | -1  | 7   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| A   | 0   | 1   | 0   | -1  | 4   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| G   | -3  | 0   | -2  | -2  | 0   | 6   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| N   | -3  | 1   | 0   | -2  | -2  | 0   | 6   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| D   | -3  | 0   | -1  | -1  | -2  | -1  | 6   | 1   | 6   |     |     |     |     |     |     |     |     |     |     |     |     |
| E   | -4  | 0   | -1  | -1  | -1  | -2  | 2   | 5   |     |     |     |     |     |     |     |     |     |     |     |     |     |
| Q   | -3  | 0   | -1  | -1  | -1  | -2  | 0   | 0   | 2   | 5   |     |     |     |     |     |     |     |     |     |     |     |
| H   | -3  | -1  | -2  | -2  | -2  | -2  | 1   | -1  | 0   | 0   | 8   |     |     |     |     |     |     |     |     |     |     |
| R   | -3  | -1  | -1  | -2  | -1  | -2  | 0   | -2  | 0   | 1   | 0   | 5   |     |     |     |     |     |     |     |     |     |
| K   | -3  | 0   | -1  | -1  | -1  | -2  | 0   | -1  | 1   | 1   | -1  | 2   | 5   |     |     |     |     |     |     |     |     |
| M   | -1  | -1  | -1  | -2  | -1  | -3  | -2  | -3  | -2  | 0   | -2  | -1   | -1  | 5   |     |     |     |     |     |     |     |
| I   | -1  | -2  | -1  | -3  | -1  | -4  | -3  | -3  | -3  | -3  | -3  | -3  | -3  |     | 1   | 4   |     |     |     |     |     |
| L   | -1  | -2  | -1  | -3  | -1  | -4  | -3  | -4  | -3  | -2  | -3  | -2  | -2  |     | 2   | 2   | 4   |     |     |     |     |
| V   | -1  | -2  | 0   | -2  | 0   | -3  | -3  | -2  | -2  | -2  | -3  | -3  | -2  |     | 1   | 3   | 1   | 4   |     |     |     |
| F   | -2  | -2  | -2  | -4  | -2  | -3  | -3  | -3  | -3  | -1  | -3  | -3  | -3  | -1  | 0   | 0   | 0   | -1  | 6   |     |
| Y   | -2  | -2  | -2  | -3  | -2  | -3  | -2  | -3  | -1  | 2   | -2  | -2  | -2  | -1  | -1  | -1  | -1  | -1  | 3   | 7   |
| W   | -2  | -3  | -2  | -4  | -3  | -2  | -4  | -4  | -3  | -2  | -2  | -3  | -3  | -1  | -3  | -2  | -3  | -3  | 1   | 2   | 11  |
LCS: Longest Common Subsequence

-AGGCTCACCTGACT-CCAGGC-CGA---TGCC---

TAG-CTCAC--GAC-GC--GG-TCGATTGCCGAC
LCS: Longest Common Subsequence

- consider score match: 1, indel: 0, mismatch: -1

- \text{AGGCTCACCTGACT-CCAGGC-CGA--TGCC---}

- \text{TAG-CTCAC--GAC-GC--GG-TCGATTGTGCCGAC}

- optimal alignment \sim \text{longest common subsequence (LCS)}
- LCS(AGCGA,CAGATAGAG)=4
- Score(S,T)=LCS(S,T)
- d(S,T)=|S|+|T|-2\cdot LCS(S,T)

minimum number of indels required to transform S into T
Levenshtein (edit) distance

- consider score match: 0, indel: -1, mismatch: -1

\[-\text{AGGCTCACCTGACTCCAGGCCGA} \rightarrow \text{TGCC} \rightarrow \rightarrow \]
\[\text{| | | | | | | | | | | | | | | | }\]
\[\text{TAG} \rightarrow \text{CTCAC} \rightarrow \text{GACGC} \rightarrow \text{GGTCGA} \rightarrow \text{TTTGCCGAC}\]

- optimal alignment ~ Levenshtein (edit) distance
- minimum number of indels and substitutions required to transform S into T
- edit(S,T) = -Score(S,T)
- edit(ACAGT,CCGA) = 3

\[\text{ACAGT} \rightarrow \text{CC} \rightarrow \text{GA}\]
Bioinformatics: "CIGAR strings"

part of SAM format

RefPos: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Reference: C C A T A C T G A A C T G A C T A A C
Read: A C T A G A A T G G C T
POS: 5
CIGAR: 3M1I3M1D5M
Computing Score(S,T)

- Assume \( d \) is indel penalty (usually \( d < 0 \)), \( s(x,y) \) score of aligning \( x \) and \( y \) (match or mismatch), \( S[1..n] \) and \( T[1..m] \) are input strings.

- **Idea**: compute \( \text{Score}[i,j] \): optimal score between \( S[1..i] \) and \( T[1..j] \)

\[
\text{Score}[i,j] = \max \begin{cases} 
\text{Score}[i-1,j-1] + s(S[i],T[j]) \\
\text{Score}[i-1,j] + d \\
\text{Score}[i,j-1] + d 
\end{cases}
\]
Computing Score(S,T)

- Assume $d$ is indel penalty (usually $d < 0$), $s(x,y)$ score of aligning $x$ and $y$ (match or mismatch), $S[1..n]$ and $T[1..m]$ are input strings.

- **Idea**: compute $\text{Score}[i,j]$: optimal score between $S[1..i]$ and $T[1..j]$

\[
\text{Score}[i,j] = \max \begin{cases} 
\text{Score}[i-1,j-1] + s(S[i],T[j]) \\
\text{Score}[i-1,j] + d \\
\text{Score}[i,j-1] + d 
\end{cases}
\]

- Initialization:
Computing Score(S,T)

- assume d is indel penalty (usually $d<0$), $s(x,y)$ score of aligning $x$ and $y$ (match or mismatch), $S[1..n]$ and $T[1..m]$ are input strings
- **Idea**: compute $Score[i,j]$: optimal score between $S[1..i]$ and $T[1..j]$

\[
Score[i,j] = \max \begin{cases} 
Score[i-1,j-1] + s(S[i],T[j]) \\
Score[i-1,j] + d \\
Score[i,j-1] + d 
\end{cases}
\]

- **initialization**: $Score[0,0]=0$, $Score[0,j]=jd$, $Score[i,0]=id$
- resulting score: $Score[n,m]$
- **Implementation**: Dynamic Programming!
Example

\[ s(x, x) = 2, \ s(x, y) = -1 \text{ for } x \neq y, \ d = -2 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \quad d = -2 \]

\[
\begin{array}{cccccccccc}
A & C & G & G & C & T & A & T \\
\hline
0 & 0 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -16 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
A & 1 & -2 &  &  &  &  &  &  &  \\
C & 2 & -4 &  &  &  &  &  &  &  \\
T & 3 & -6 &  &  &  &  &  &  &  \\
G & 4 & -8 &  &  &  &  &  &  &  \\
T & 5 & -10 &  &  &  &  &  &  &  \\
A & 6 & -12 &  &  &  &  &  &  &  \\
T & 7 & -14 &  &  &  &  &  &  &  \\
\end{array}
\]
**Example**

\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \ d = -2 \]

<table>
<thead>
<tr>
<th></th>
<th>0</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>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td>-14</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>-4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>3</td>
<td>-6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>4</td>
<td>-8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>5</td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>6</td>
<td>-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>7</td>
<td>-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \ d = -2 \]

\[
\begin{array}{cccccccccc}
& A & C & G & G & C & T & A & T \\
\hline
A & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
0 & 0 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -16 \\
1 & -2 & 2 & 0 & -2 & -4 & -6 & -8 & -10 & -12 \\
2 & -4 & 0 & 4 & 2 & 0 & -2 & -4 & -6 & -8 \\
3 & -6 & & & & & & & & \\
4 & -8 & & & & & & & & \\
5 & -10 & & & & & & & & \\
6 & -12 & & & & & & & & \\
7 & -14 & & & & & & & & \\
\end{array}
\]
Example

\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \quad d = -2 \]

\[
\begin{array}{cccccccccc}
 & A & C & G & G & C & T & A & T \\
\hline
A & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
0 & 0 & -2 & -4 & -6 & -8 & -10 & -12 & -14 & -16 \\
1 & -2 & 2 & 0 & -2 & -4 & -6 & -8 & -10 & -12 \\
2 & -4 & 0 & 4 & 2 & 0 & -2 & -4 & -6 & -8 \\
3 & -6 & -2 & 2 & 3 & 1 & -1 & 0 & -2 & -4 \\
4 & -8 & -4 & 0 & 4 & 5 & 3 & 1 & -1 & -3 \\
5 & -10 & -6 & -2 & 2 & 3 & 4 & 5 & 3 & 1 \\
6 & -12 & -8 & -4 & 0 & 1 & 2 & 3 & 7 & 5 \\
7 & -14 & -10 & -6 & -2 & -1 & 0 & 4 & 5 & 9 \\
\end{array}
\]
Example

\( s(x, x) = 2, s(x, y) = -1 \) for \( x \neq y \), \( d = -2 \)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td>-14</td>
<td>-16</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>-12</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>-14</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Score(S, T)
How to recover the alignment?
\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \quad d = -2 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
</tr>
</tbody>
</table>

Score(S,T) = 9
How to recover the alignment?

\[ s(x, x) = 2, s(x, y) = -1 \text{ for } x \neq y, \ d = -2 \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>G</th>
<th>G</th>
<th>C</th>
<th>T</th>
<th>A</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
<td>-12</td>
<td>-14</td>
</tr>
<tr>
<td>1</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
<td>-8</td>
<td>-10</td>
</tr>
<tr>
<td>2</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>-4</td>
<td>-6</td>
</tr>
<tr>
<td>3</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>-3</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>-12</td>
<td>-8</td>
<td>-4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>-14</td>
<td>-10</td>
<td>-6</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Score(S,T)
Alignment: graph formulation

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

2 match

-1 mismatch

-2 indels
Alignment: graph formulation

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

-2
indels

2
match

-1
mismatch

max-cost path cost=9
Exercise

- Compute all optimal alignments between ACCGTTG and CGAATGAA if the match score is 2, the mismatch penalty is -1 and the gap penalty (indel score) is -2
Comments

- algorithm known as Needleman-Wunsch algorithm (1970)
- note that optimal alignment is generally not unique
- the problem considered is called global alignment
- both time and space complexity is $O(n^2)$
- space complexity is $O(n)$ if only the optimal score has to be computed (e.g. line-by-line, keep two lines at a time)
- time can be reduced to $O(n^2 \log^2 n)$ (assuming RAM model) [Masek, Paterson 80] using “four-russians technique” (another solution in [Crochemore, Landau, Ziv-Ukelson 03])
- proved to be unlikely solvable in time $O(n^{2-\varepsilon})$ [Abboud, Williams, Weimann 14] (by reduction from 3SUM to some versions of alignment problem)
- same result for the edit distance [Backurs, Indyk 15]
Output-sensitive variant [Ukkonen 85]

- Consider $S[1..n]$ and $T[1..n]$ (for simplicity, $m = n$)
- Assume also that matches cost 0, mismatches and indels cost negative (penalties)
- **Lemma 1**: Assume $D^* = Score[n, n] \leq 0$. Then the optimal DP path lies within a “strip” of width $|D^*|/\delta$, where $\delta$ is the smallest indel penalty
- **Lemma 2**: There exists an algorithm that, given $D$, tests if $Score[n, n] \geq D$ in time $O(|D|n)$. If yes, it also outputs the optimal path (i.e. alignment).
Output-sensitive variant [Ukkonen 85]

- Consider $S[1..n]$ and $T[1..n]$ (for simplicity, $m = n$)
- Assume also that matches cost 0, mismatches and indels cost negative (penalties)

**Lemma 1**: Assume $D^* = \text{Score}[n, n] \leq 0$. Then the optimal DP path lies within a “strip” of width $|D^*|/\delta$, where $\delta$ is the smallest indel penalty

**Lemma 2**: There exists an algorithm that, given $D$, tests if $\text{Score}[n, n] \geq D$ in time $O(|D|n)$. If yes, it also outputs the optimal path (i.e. alignment).

**Theorem**: There exists an algorithm that computes $D^* = \text{Score}[n, n]$ and the optimal path in time $O(|D^*|n)$.

**Proof**: Apply Lemma 2 for $|D| = 1, 2, 4, ...$ until $|D| = 2^k \geq |D^*|$
Exercises (1)

- **End-space free alignment of \( S \) and \( T \)**
  - compute the best alignment of \( S \) and \( T \) such that spaces at string borders contribute 0
Exercises (2)

- Approximate occurrences of $P$ in $T$
  - compute all alignments such that $\text{Score}(S, T[i..j]) > \delta$
Exercise

- search P=gcgta tcg in T=tattggcctata cggtt if the match score is 1, the mismatch penalty is -1 and the gap penalty (indel score) is -1
Exercises (2)

- Approximate occurrences of \( P \) in \( T \)
  - compute all alignments such that \( \text{Score}(S, T[i..j]) > \delta \)

![Diagram of alignments]

- Particular cases
  - edit distance (\(< k\)): \( O(kn) \) [Landau\&Vishkin 85, Galil\&Park 89, …]
  - Hamming distance: \( O(n \cdot \log(m)) \) [Fischer\&Paterson 73], \( O(nk) \) [Galil\&Giancarlo 86], \( O(n \sqrt{k \log k}) \) [Amir\&Lewenstein\&Porat 04], …
Computing alignment in linear space

- Hirschberg (1975) proposed a nice trick in order to compute the optimal alignment in linear space (at the price of doubling the time)

- Key observation:

```plaintext
\[ T \]
\[ S \]
\[ k^* \]
\[ n/2 \]
Computing alignment in linear space

- Hirschberg (1975) proposed a nice trick in order to compute the optimal alignment in linear space (at the price of doubling the time)

- Key observation:

\[ k^* = \text{argmax}_k (\text{Score} \left( \frac{n}{2}, k \right) + \text{Score}^R \left( \frac{n}{2}, m - k \right)) \]
Computing alignment in linear space

- Hirschberg (1975) proposed a nice trick in order to compute the optimal alignment *in linear space* (at the price of doubling the time)

- Key observation:

\[ k^* = \text{argmax}_k (\text{Score} \left[ \frac{n}{2}, k \right] + \text{Score}^R \left[ \frac{n}{2}, m - k \right]) \]
\[ k^* = \text{argmax}_k (\text{Score}_{\frac{n}{2}, k} + \text{Score}^R_{\frac{n}{2}, m - k}) \]
Resulting complexity

- if the Score computation on a $p \times q$ matrix takes time $c \cdot pq$, then computing the first “cut” takes $2 \cdot c \cdot (n/2) \cdot m = c \cdot nm$
- the first halving results in time
  \[ c \cdot (n/2) \cdot k^* + c \cdot (n/2) \cdot (m - k^*) = 1/2 \cdot c \cdot nm \]
- all recursive calls take time
  \[ c \cdot nm + 1/2 \cdot c \cdot nm + 1/4 \cdot c \cdot nm + \cdots \leq 2c \cdot nm \]
Biologists are mostly interested in *local alignments* that may ignore arbitrary prefixes and suffixes of input sequences.

![Local alignment diagram](image-url)
Local alignment

- Biologists are mostly interested in *local alignments* that may ignore arbitrary prefixes and suffixes of input sequences.

*Problem:* Compute all *significant* local alignments, i.e. all alignments of score above a threshold.
Smith-Waterman algorithm (1981)

- Assume matches are scored positively and mismatches/indels are scored negatively
- Score[i,j]: maximum score over all substrings of S that end at position i and all substrings of T that end at position j
- Initialization: Score[0,j]=Score[i,0]=0

\[
\text{Score}[i,j] = \max \begin{cases}
0 \\
\text{Score}[i-1,j-1] + s(S[i], T[j]) \\
\text{Score}[i-1,j] - d \\
\text{Score}[i,j-1] - d
\end{cases}
\]
Smith-Waterman: example

EAWACQGKL vs ERDAWCQPGKWy

s(x,x)=1, s(x,y)=-3 for x≠y, d=-1

resulting local alignment: 

\[
\begin{pmatrix}
A & W & A & C & Q & - & G & K \\
A & W & - & C & Q & P & G & K
\end{pmatrix}
\]
Score matrix is important

The average value of score matrix should be negative

There exists a statistical model [Karlin, Altschul 90] that allows to relate the score of a local alignment and the probability for this alignment to appear in random sequences ($p$-value)
More complex gap penalty systems

- **Affine gap penalty**: $h + q \cdot i$
  - $h$: gap opening penalty
  - $q$: gap extension penalty
  - $O(mn)$ algorithm [Gotoh 82]

- **Convex gap penalty**
  - $O(mn \cdot \log n)$

- **Arbitrary gap penalty**
  - $O(mn^2 + nm^2)$