CS 466
Introduction to Bioinformatics

Instructor: Jian Peng
Teaching Assistant: Wesley Qian
BLAST:
Basic Local Alignment Search Tool
Hashing

A hash function maps a key to a value
Hash table

• Hash table is a data structure: a way to store key-value pairs, and a way to retrieve them

• Based on the idea of a hash function. This maps a key or an object (e.g., a string, or a more complex record) to an integer, the “address”

• The value of the key is then stored at that address in memory
Hashing: an example

- Key: (AAACGTAT, 1234321)
  - i.e., a 8 bp-string and its location in genome
- We want to store many such strings and their locations
- and later retrieve all locations of a particular string really quickly
- Hash function $h(\text{AAACGTAT}) = 435$
Hashing: an example

- Let’s assume that there are $4^8 = 64$K memory locations available.

- The first time we see (AAACGTAT, *), we store it at address $h$([AAACGTAT]) = 435.

- The next time we see (AAACGTAT, *), we compute $h$([AAACGTAT]), go to 435, find it already occupied. A collision!
How to handle collisions

• Buckets: Address 435 can store multiple keys/objects (e.g., as a linked list)

• Linear probing: If an address is occupied, store the key/object in next available location

• Multiple hashing: have an army of hash functions. If the first one ("h") led to a collision, try another hash function ("h2")
Bucketing and Chaining

- Rather than searching for a free entry, make each entry in the table an ARRAY (bucket) or LINKED LIST (chain) of items/entries
- Buckets
  - How big should you make each array?
  - Too much wasted space
- Chaining
  - Each entry is a linked list
Open addressing and linear probing

- Open addressing means an item with key, k, may not be located at h(k)
- Assume, location 2 is occupied with another item
- If a new item hashes to location 2, we need to find another location to store it
- Linear Probing
  - Just move on to location h(k)+1, h(k)+2, h(k)+3,...
Preprocessing and hash

Preprocessing:
store exact matches of all short patterns on the text by a hash table

\[ ATC \rightarrow h \rightarrow \text{address1} \rightarrow \{1, 6, 100, 2000, 5454, \ldots, \} \]
\[ AAA \rightarrow h \rightarrow \text{address2} \rightarrow \{15, 21, 30, 785, 3434, \ldots, \} \]
\[ TTC \rightarrow h \rightarrow \text{address3} \rightarrow \{5, 164, 220, 502, 943, \ldots, \} \]
BLAST: finding maximal segment pairs

• Given two sequences of same length, the similarity score of their alignment (without gaps) is the sum of similarity values for each pair of aligned residues

• Maximal segment pair (MSP): Highest scoring pair of identical length segments from the two sequences being compared (“query” and “subject”)

• The similarity score of an MSP is called the MSP score

• BLAST heuristically aims to find them
Goal: report database sequences that have MSP score above some threshold $S$.

Thus, sequences with at least one locally maximal segment pair that scores above $S$. 

Query: HBA_HUMAN Hemoglobin alpha subunit
Sbjct: SPAC869.02c [Schizosaccharomyces pombe]

Score = 33.1 bits (74), Expect = 0.24
Identities = 27/95 (28%), Positives = 50/95 (52%), Gaps = 10/95 (10%)
High scoring pairs (or local maximal segment pairs)

- A molecular biologist may be interested in all conserved regions shared by two proteins, not just their highest scoring pair.
- A segment pair (segments of identical lengths) is locally maximal if its score cannot be improved by extending or shortening in either direction.
- BLAST attempts to find all locally maximal segment pairs above some score cutoff.
A quick way to find MSPs

- Homologous sequences tend to have very similar or even identical substrings, also called seeds.
- From a seed, it is possible to construct a local HSP/MSP by extending to flanking regions.

---

```
FAKDFLAGGVAASISKTVAPIERVKLLLQVQHAASKQITADKQYKGIIDCVVRIPKEQGV
FD+GGAAA+SKTAVAPIERVKLLLQVQASKIDK+YKGI+D++R+PKEQGV
FLIDLASSGTAAVSKTAVAPIERVKLLLQVQDASKIAVDKRYKGIIMDVLRVPKEQGV
```
Efficient algorithm?
1. Break query sequence into words

Break query into words:

MEA
EAA
AVK
VKE
KKE
EEI
EIS
ISV
...

MEAAVKEEISVEDEAVDKNI
2. Find database hits

- Find exact matches to query words
- Can be done in efficiently
  - Hashing
  - Alternatively AC finite state machine

ATC $\rightarrow$ h $\rightarrow$ address1 $\rightarrow$ retrieve $\rightarrow$ \{1, 6, 100, 2000, 5454, …, \}

AAA $\rightarrow$ h $\rightarrow$ address2 $\rightarrow$ retrieve $\rightarrow$ \{15, 21, 30, 785, 3434, …, \}

TTC $\rightarrow$ h $\rightarrow$ address3 $\rightarrow$ retrieve $\rightarrow$ \{5, 164, 220, 502, 943, …, \}
2. Find database hits
3. Extend hits

1. Find "seeds" (initial matches) of a fixed length (e.g. 11)
2. Try extending an alignment from each seed
How to handle possible mismatches in words?

MVRE\textsubscript{R}KCI\textsubscript{L}C\textsubscript{H}I\textsubscript{V}YGSKK\textsubscript{E}MDEHMR\textsubscript{S}MLH\textsubscript{H}RELE\textsubscript{N}LKGR\textsubscript{D}IS

Query word, $W=3$ for proteins

(\(W=11\) for nucleotides)

<table>
<thead>
<tr>
<th>Word</th>
<th>Score (BL-62)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSK</td>
<td>15</td>
</tr>
<tr>
<td>GAK</td>
<td>12</td>
</tr>
<tr>
<td>GNK</td>
<td>12</td>
</tr>
<tr>
<td>GTK</td>
<td>12</td>
</tr>
<tr>
<td>GSR</td>
<td>12</td>
</tr>
<tr>
<td>GDK</td>
<td>11</td>
</tr>
<tr>
<td>GQK</td>
<td>11</td>
</tr>
<tr>
<td>GEK</td>
<td>11</td>
</tr>
<tr>
<td>GGK</td>
<td>11</td>
</tr>
<tr>
<td>GKK</td>
<td>11</td>
</tr>
<tr>
<td>GSQ</td>
<td>11</td>
</tr>
<tr>
<td>GSE</td>
<td>11</td>
</tr>
</tbody>
</table>

Neighbor words

First step:

For each position $p$ of the query, find the list or words of length $w$ scoring more than $T$ when paired with the word starting at $p$:

List of words of length $w$, scoring more than $T$ with the $p$-word.
How to handle possible mismatches in words?

Second step:
For each words list, identify all exact matches with DB sequences:

```
p-word words list                                      DB sequences
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```
How to handle possible mismatches in words?

Third step:
For each word match («hit»), extend ungapped alignment in both directions. Stop when S decreases by more than X from the highest value reached by S.

HSP = High Scoring Segment Pair
Parameters

- Word length: 3 for protein, 11 for DNA/RNA
- Thresholds $T$ and $S$:
  - BLAST minimizes time spent on database sequences whose similarity with the query has little chance of exceeding this cutoff $S$.
  - Main strategy: seek only segment pairs (one from database, one query) that contain a word pair with score $\geq T$
  - Intuition: If the sequence pair has to score above $S$, its most well matched word (of some predetermined small length) must score above $T$
- Lower $T$ => Fewer false negatives
- Lower $T$ => More pairs to analyze