Hirschberg’s Algorithm for Edit Distance Computation and Alignment

By
Arun Kumar (114708780)
Archit Saxena (112682655)

Edit Distance: Hirschberg’s Algorithm Overview

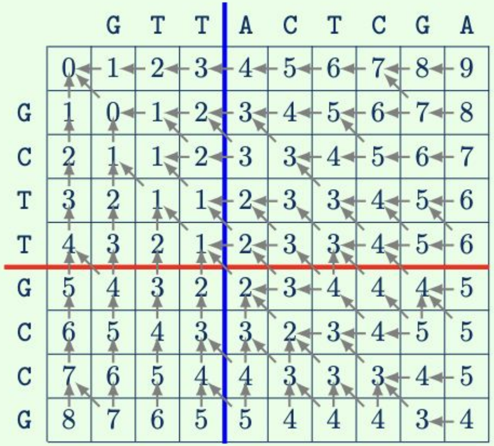
Observation

- Let $n, m > 1$. Denote $x = \lfloor \frac{m}{2} \rfloor$,
- There exists $y \in [1\,..\,n)$ such that: $\text{ED}(X, Y) = \text{ED}(X[1\,..\,x], Y[1\,..\,y]) + \text{ED}(X(x\,..\,m], Y(y\,..\,n])$.

Hirschberg’s Algorithm: Overview

- In: X, m, Y, n,
- Out: Alignment of X to Y of cost $\text{ED}(X, Y)$,
- First compute y (as above) in $\mathcal{O}(mn)$ time,
- Then, recursively find the alignment of $X[1..x]$ to $Y[1..y]$ and $X(x..m]$ to $Y(y..n]$,
- Finally, concatenate alignments (encoded, e.g., using **Alg #2**) returned by the recursion,
- Key difficulty is thus finding y.

Example



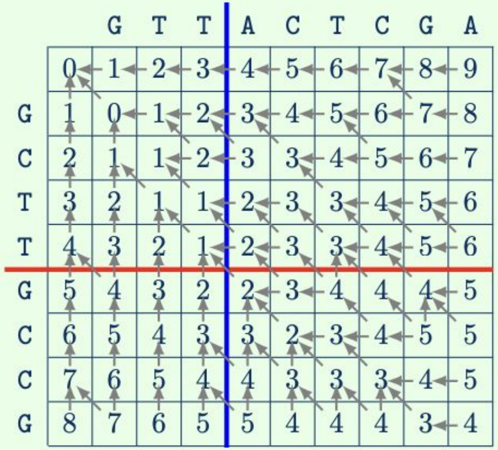
		G	T	T	A	C	T	C	G	A
	0	1	2	3	4	5	6	7	8	9
G	1	0	1	2	3	4	5	6	7	8
C	2	1	1	2	3	3	4	5	6	7
T	3	2	1	1	2	3	3	4	5	6
T	4	3	2	1	2	3	3	4	5	6
G	5	4	3	2	2	3	4	4	4	5
C	6	5	4	3	3	2	3	4	5	5
C	7	6	5	4	4	3	3	3	4	5
G	8	7	6	5	5	4	4	4	3	4

Edit Distance: Hirschberg's Algorithm Overview

Time Complexity

- Estimate total work at $i$the level of recursion,
  - 0th: $\mathcal{O}(mn)$ time,
  - 1st: $\mathcal{O}(\frac{m}{2}y) + \mathcal{O}(\frac{m}{2}(n-y)) = \mathcal{O}(\frac{mn}{2})$,
  - 2nd: $\mathcal{O}(\frac{m}{4}y') + \mathcal{O}(\frac{m}{4}(y-y'))$ $+\mathcal{O}(\frac{m}{4}y'')+\mathcal{O}(\frac{m}{4}(n-y-y'')) = \mathcal{O}(\frac{mn}{4})$,
  - ith: $\mathcal{O}(\frac{mn}{2^i})$.
- Overall: $mn\sum_{i=0}^{\log m}\frac{1}{2^i} < 2mn$ operations, i.e., Hirschberg's algorithm runs run $\mathcal{O}(mn)$ time.

Example



		G	T	T	A	C	T	C	G	A
	0	1	2	3	4	5	6	7	8	9
G	1	0	1	2	3	4	5	6	7	8
C	2	1	1	2	3	3	4	5	6	7
T	3	2	1	1	2	3	3	4	5	6
T	4	3	2	1	2	3	3	4	5	6
G	5	4	3	2	2	3	4	4	4	5
C	6	5	4	3	3	2	3	4	5	5
C	7	6	5	4	4	3	3	3	4	5
G	8	7	6	5	5	4	4	4	3	4

Literature on Edit Distance and recovering alignment:

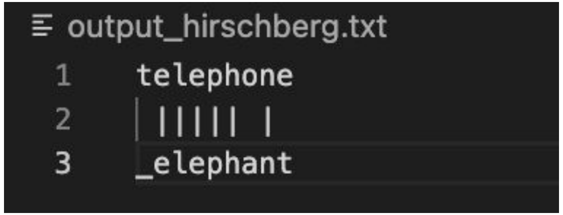
- Our Standard Dynamic Programming **O(nm) space** and **O(nm) time**
- Hirschberg's Algorithm (1975) helps in recovering the alignment in **O(nm) time** and **O(n) space**
- Extending the Four Russian Algorithm to Compute the Edit Script in Linear **Space O(n)** time and **O(n^2 /log(n))** (2008) Vamsi Kundeti & Sanguthevar Rajasekaran
- Fast & Space-Efficient Approximations of Language Edit Distance and RNA folding: An Amnesic Dynamic Programming Approach (Barna Saha, 2017) recovers alignment using Map reduce, parallelization and other tricks
- Similarly, we have online implementation like **Edlib**: a C/C++ library for fast, exact sequence alignment using edit distance. This uses Myer’s bitvector parallelizable algorithm. Complexity: **O(nm/w)**
- Using a “bit vector” type dp matrix, can further reduce the space requirements by a factor of log(n), and using wavelet trees to store the strings can further reduce the space requirement

Experimental Setup and Design:

- All the experiments were done on Mac M1 AIR(2020), with 8GB RAM and 256GB SSD.
- Our language of choice was C++11, with g++ compiler
- We have the following implementations which we used for profiling:
  - Naive Implementation: standard DP with O(nm) time and O(mn) space
  - Our implementation of Hirschberg's Algorithm with O(mn) time and O(m) space which is based on the algorithm presented in class
  - Fast EDLIB library implementation available online (C/C++)(Hirschberg + bit-vector parallelization). Based on SeqAn library but optimized further for edit distance.
- For correctness, we compared the edit distance of our Naive, Hirschberg's algorithm and EDLIB implementation online on random/artificial strings, repetitive and non repetitive corpus
- For measuring RAM usage we used **"/usr/bin/time -l"** command and ensure that no other significant programs were running in the machine during testing
- We found that compiler optimization (O3, O2 etc. flag) do affect the results, in analysis below we have not applied any optimization

Experimental Design and Details

- We created three different binaries for Naive, Hirschberg and Edlib using “make”
- These can be created by running “make” in terminal from the top level folder.
- To Calculate the Edit Distance & Alignment, we can run:
    - ./bin/Hirschberg inputfile1.txt inputfile2.txt
    - It creates a new file output.txt with the corresponding globally aligned strings
    - Eg. for two strings **“elephant”** and **“telephone”**, our binaries will output



output_hirschberg.txt

```
telephone
 ||||| |
_elephant
```

- In all our implementation is easy to use
- We further compare the edit distance obtained from naive, hirschberg and edlib with our strings after obtaining alignment

Testing: correctness test

- We used the “out.txt” (provided in HW4) to check the correctness of our implementation both Naive and hirschberg to validate our implementation
- We wrote scripts to compare our Naive and Hirschberg on variety of test cases. These includes
  a. Single length strings “A” & “B” (test1.sh)
  b. Repetitive unary strings “AAAAAAAAAAAA” and “AAAAA” (test2.sh)
  c. Strings like “telephone” and “elephant” (test3.sh)
  d. Unary strings (test4.sh)
  e. Randomly generated two strings of length 50000 (test5.sh) 10 times
- From the above tests we concluded that, Our Hirschberg algorithm was correctly implemented
- All scripts details are available on the readme submitted on github

Results

Experimental Results on Artificial(random) strings

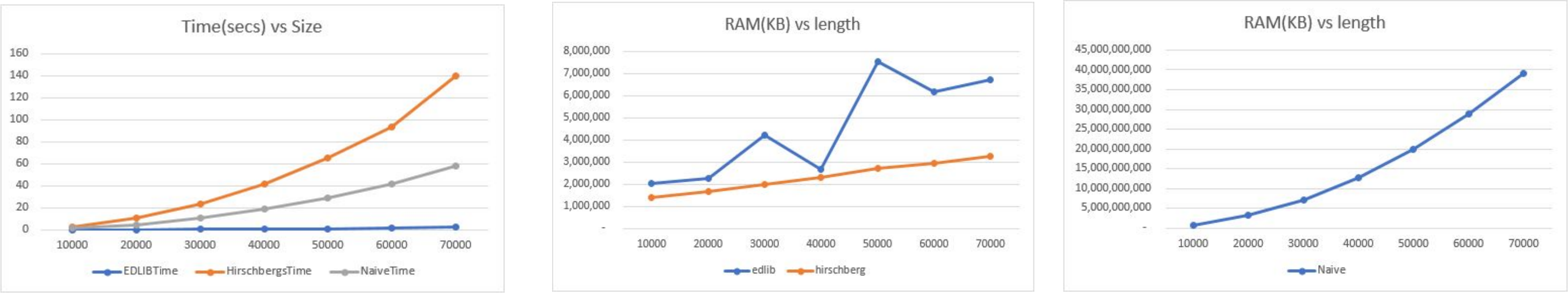


Fig1. Artificially generated alphanumeric strings strings with **σ = 20**

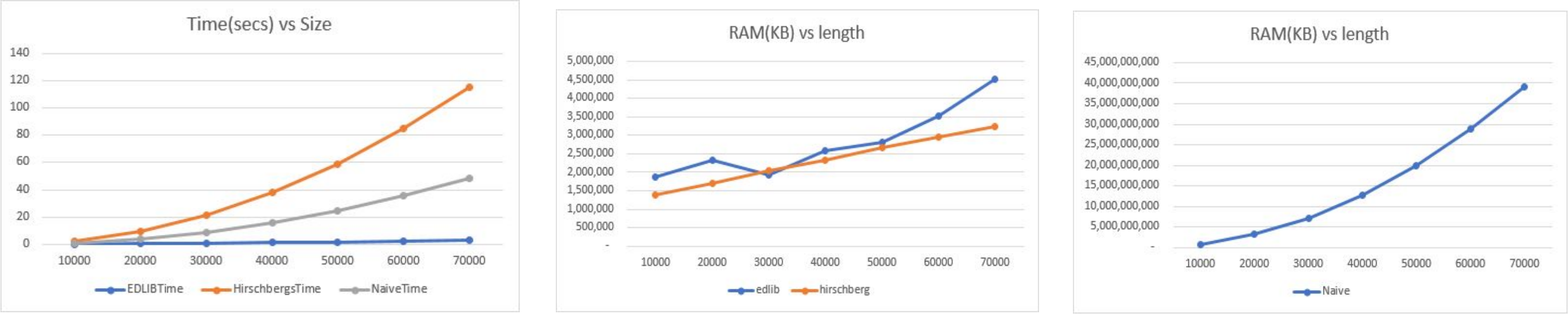
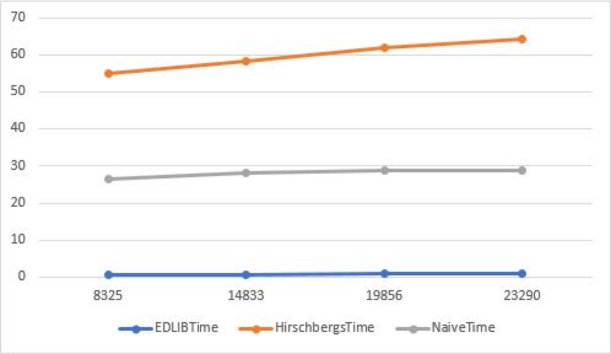


Fig2. Artificially generated alphanumeric strings strings with **σ = 4**

Experiments on Artificial strings with varying edit Distance



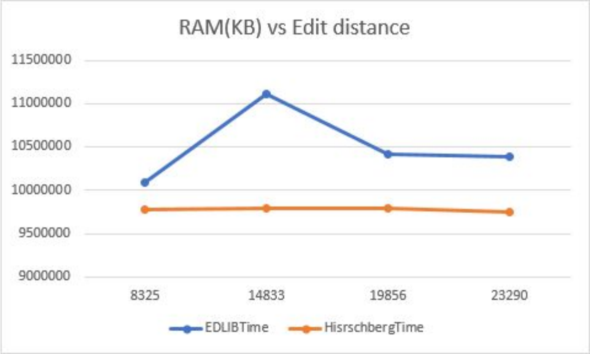
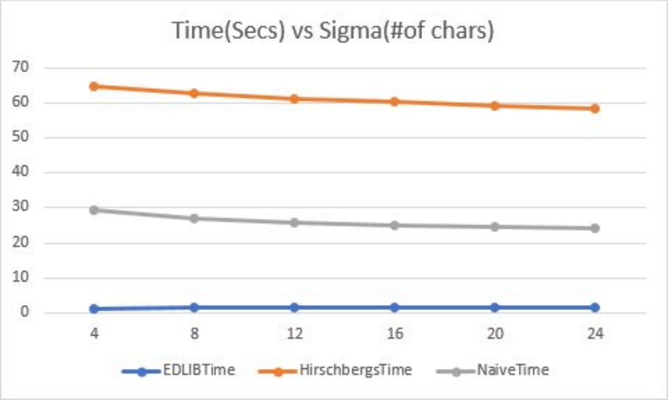


Fig4. Artificially generated alphanumeric strings strings with **n = 50000, σ = 20**

- We randomly changes the letters at the location of the same string to vary the edit Distance

Experiments on Artificial strings with different letters



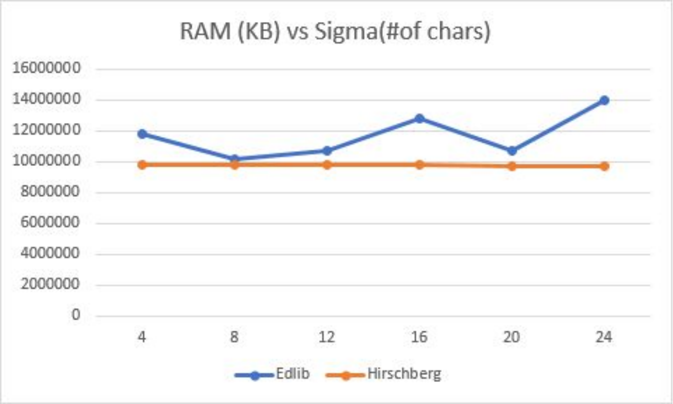
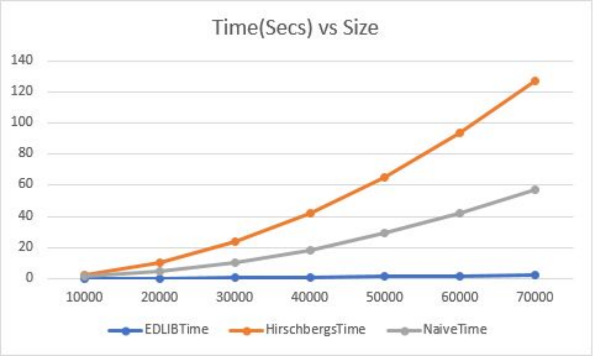
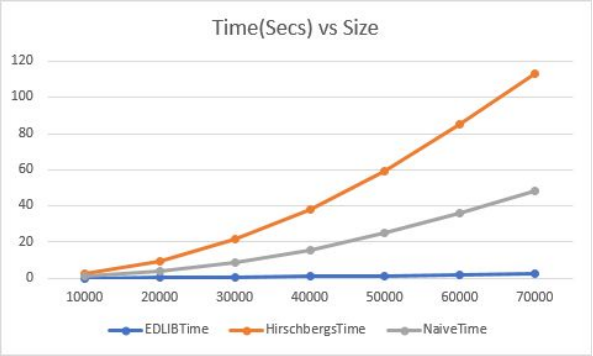


Fig4. Artificially generated alphanumeric strings strings with **n** = **50000** and increasing **σ**

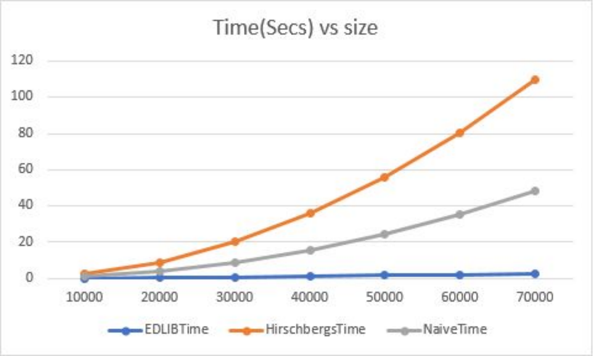
Experiments on DNA, proteins, English (Non Repetitive Corpus)



DNA, **σ = 4**

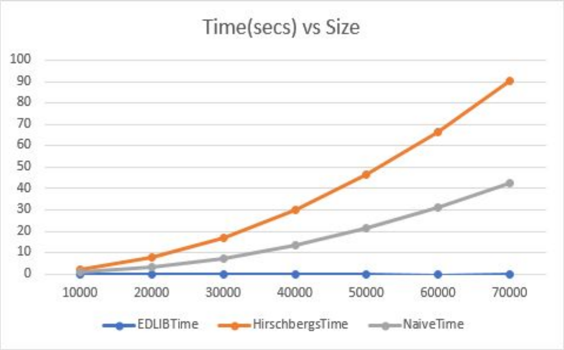


Proteins, **σ = 20**

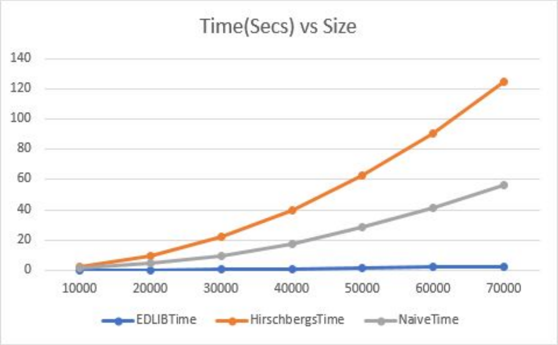


English

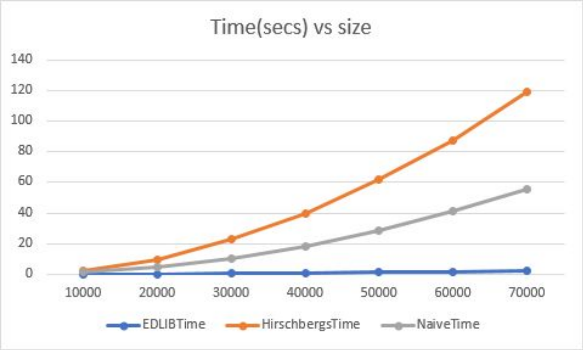
Experiments on DNA, proteins (Repetitive Corpus)



English: einstein.en.txt.gz, **σ** = 139



Pera: **σ** = 5



Influenza: **σ** = 15

References

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5408825/ [Github]
- https://github.com/2108arunk/Compbio_project
- https://en.wikipedia.org/wiki/Hirschberg%27s_algorithm
- https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.562.2911&rep=rep1&type=pdf

QnA