**CS 594: Representations in Algorithm Design** **Spring 2022**

Lecture on 03/15/2022

*Lecturer: Xiaorui Sun* *Scribe: Muhammed Adeem Shaik*

# 1 Last Lecture's Review

• Approximate of Edit Distance the way it is solved in Dynamic Program.
• Faster algorithm for Edit Distance and the approximation factor and the way it is evolved during the years.

# 2 Today's Lecture

• Continuation of Approximation of Edit Distance.
• Problems on Graph

# 3 Approximation of Edit Distance

## 3.1 Problem Definition

To find the the Edit distance between for given two strings x, y over z and these two sequences of characters come from the alphabet z. Now we have to see how many operations are needed to transform x to y.
Available Operations are -
• Add a character.
• Delete a character.
• Replace a character in a string by another character.
• Here x = y = n: which are of same length.

This problem can be solved in O($n^2$) running time and the classic approximate edit distance in time is O ($n^{1.618}$).

Expression for row edit distance :

$$\boxed{ED(x,y) \leq ED\ (x,y) \leq C.ED(x,y)}$$

## 3.2 Idea

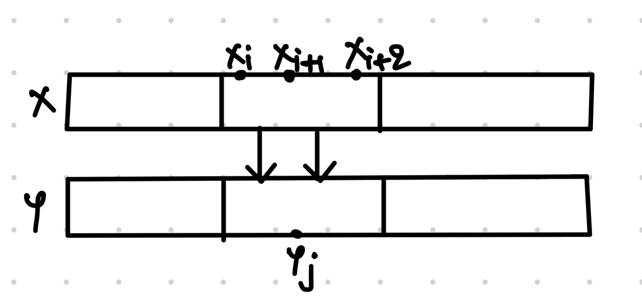
The Idea is Reducing to compute Edit distance between the sub-strings i.e
- Break string x, y into $x_i$, $y_j$ such that $|x_i| = |y_j| =$ w where w is $n^{0.19}$.
- This implies that $x_i$ and $y_j$ are of same length as w.
- Sub-string of $x_i$ starts from the $i^{th}$ character of length w.

$$x_i = x_i + x_i + 1 \text{ ..... } x_i + w - 1$$

- Fact is that Edit distance is the smallest mapping of indices of $1^{st}$ string to the $2^{nd}$ string i.e. –

$$\boxed{\text{ED} \approx \frac{\min_{\Pi i \sim j} \sum_{i \in N} ed[x_i, y_\Pi]}{w}}$$

Consider that we have an entire string of x y and assume that we have an optimal solution.

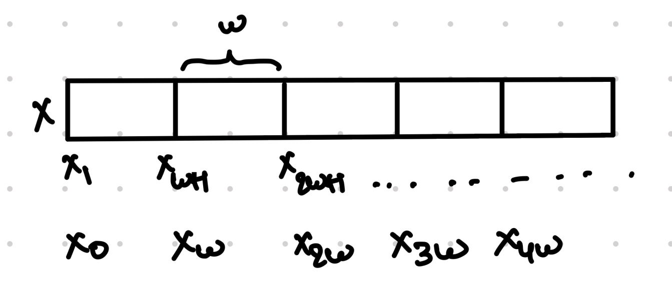



- The above string maps from $x_i$ to $y_j$.
- For each sub-string of length w, it matches to the same sub-string in other string. (Characters making between 2 (x to y)).
- From the above expression match such that the above quantity is minimized.

• If we have the Edit distance between (xi, yj) for all the possible i, j then we can compute Edit distance.

$$\mathrm{ED}(x_i,\, y_j)\ \forall i, j.$$

• Now we figure out the minimum possible matching, Roughly we have n different sub-strings $x_i$ and n different sub-strings $y_j$.
• We need to consider $n^2$ different pairs in order to know the distance and If we compute edit distance between all pairs it becomes worse => $n^2.w^2$ which is a very bad running time.(where $w^2$ is a fixed value)
• First step is Now we reduce possible number of 'i' i.e., the possible number of pairs
• Consider i is a multiple of w.




Dividing into block of length W.

• Edit distance is the best matching such that all i's are multiple of w for the edit distance between $x_i$ and $y_{(i)}$.

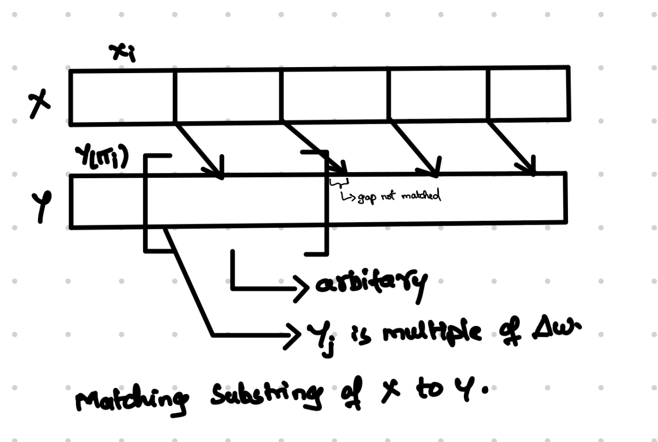
$$\boxed{\mathrm{ED} \approx \min_{\Pi} \sum ed[x_i, y_{\Pi(i)}] i \in u, w\ldots}$$

• By this we don't need w that is we are normalizing and the condition for overlapping is

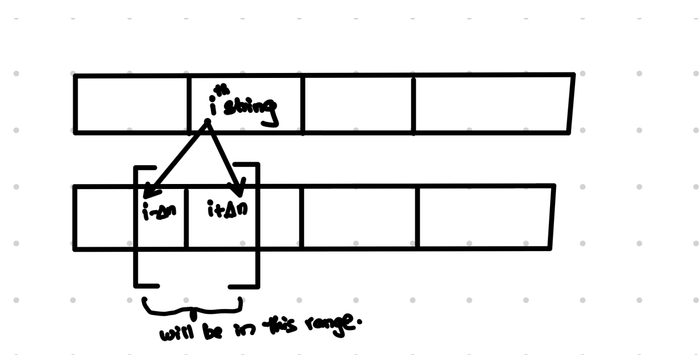
$$\boxed{\pi(i+1) \geq \pi(i) + w}$$

• Number of subsets from x becomes n/w.n≈ $n^2/w$
• Running time $n^2/w.w^2$ => $n^2w$ which is worse than previous where w = number of sub-strings.

• Second step is to reduce number of strings in y.
• It is okay to have an $\Delta n$ additive error, then we only need to consider $n/w.\Delta$ sub-strings in y.
• Idea behind this is :




• Run the y sub-string to multiple of $\Delta w$ and if the sub-string is multiple of $\Delta w$ then shifting Gives error of $\Delta$ times w together the error is $\Delta n$.
• If Edit distance is $\Delta n$ then there is an algorithm to compute edit distance in time $\Delta.n^2$, If Edit distance is small then $x_i$ is roughly matched to $i-\Delta n$ to at most $i+\Delta n$.




• If *Editdistance* $\leq 0.9$ then you have algorithm with running time $n^{0.9}$ and in this $\Delta$ is assumed to be $\Delta \geq 1/n^{0(1)}$.
• Considering n/w different $x_i$ and $n/w.\Delta$ different $y_j$.

# 4 Problem Definition

- Edit Distance [$x_i$,$y_j$]
- $x_i$ is multiple of w
- $y_j$ is multiple of $\Delta$ times w [$\Delta$.w]
- n/w.n/ $\Delta$.w pairs => $n^2$/ $\Delta w^2.w^2 \approx n^2$/ $\Delta$ running time.

## 4.1 Algorithm

1.Partition of $x_i$ and $y_j$ – Running time for this case is O(n).

2.Compute Edit distance $(x_i, y_j)$ - Running time for this case is $n/w.n/w\Delta.w2$.

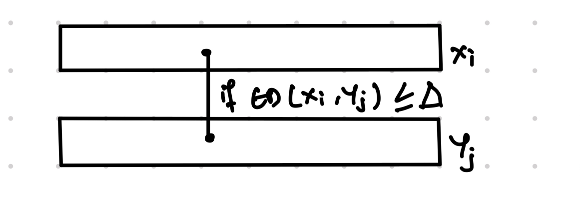
3.Approximate Edit distance based on ED $(x_i, y_j)$. (Roughly dynamic programming in time $n/w.n/w\Delta$).

- To get a good algorithm we need to improve the '2' step mentioned above and key idea for this is not to compute $ED(x_i,y_j)\forall i, j$. we should compute small pairs $O(n^2/w^2)$ pairs.
- Assuming we have string pairs i.e. $x_i, y_j, y_k$.
- Assuming the edit distance between these $ED(x_i, y_j)$, $ED(y_j, y_k)$.
- Edit distance between $ED(x_i, y_k)$ is the one which we have to compute.
- Edit distance between $x_i, y_k$ is upper bounded by $x_i, y_j$ and $y_j, y_k$.

$\boxed{\text{ED } (\mathrm{x}_i, y_k) \leq ED(x_i, y_j) \leq ED(y_j, y_k) \text{ – Triangular inequality.}}$

# 5 Graph Problem

Graph: Vertices are xi $\cup yj$.

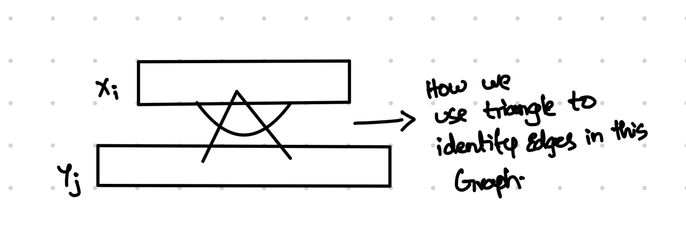


- It is a Bipartite Graph

• Assuming that we have an Graph of G1, G2, G4, G8 ...... Gw then it is sufficient to calculate the edit distance. (W is upper bound of the sub-strings)
• $ED(x_i, y_j)$ belongs to $G\Delta$ but not in $G\Delta/2$ than $ED(x_i, y_j) \approx \Delta$.
• At the end we want to roughly estimate the graph and now the question becomes approximate $G\Delta$ for given $\Delta$?
• We use triangular inequality but we get error.
• Approximation $\sim G\Delta$ $Edge(x_i,y_j)$ in $G\Delta$ if ED(1) $\leq\Delta$ and if $(x_i,y_j)$ in $\sim G\Delta$ then ED $(x_i,y_j)$ $\leq$3.$\Delta$ .

## 5.1 Easy Case

• In $G\Delta$ which is a bipartite graph every vertex $x_i$ has the degree $\geq$ $constant\times n^E$.




• Here every vertex will have some neighbors and every vertex in xi has relatively large degrees.
• $G\Delta$ implies that if sample $y_j$ with probability $1/n^E$ then for each $x_i$ at least one neighbor in $y_j$ is sampled.
• This sample substring will help to find distance between the other strings (x,x).

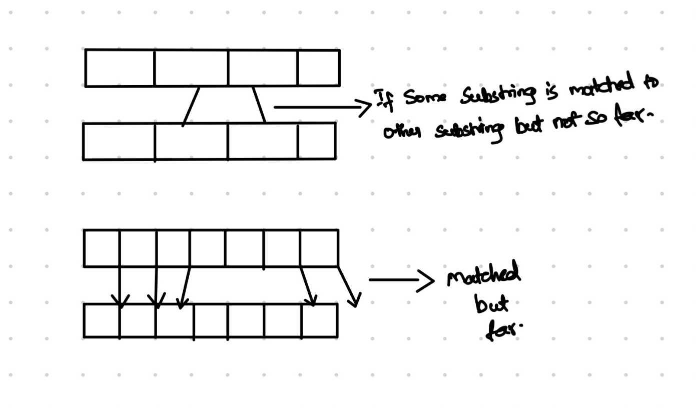
## 5.2 Algorithm

• S = sample strings in $y_j$ $w.plogn/n^E$.
• Compute the Edit distance for $ED(y_j, y_j), ED(y_j, x_i)$.
• Use Triangular Inequality to estimate the $\sim G\Delta$.
• Which means that if Edit distance between $(x_i,y_j)$ and the Edit distance between $(y_j.y_j)$ is less than or equal to delta then add an edge $(x_i,y_j)$.

That is if $ED(x_i,y_j) \leq \Delta, ED(y_j.y_j) \leq \Delta$ then Add Edge $(x_i,y_j)$.

# 6 Finding Neighbors of Edges in Graph

• In a graph there are lot of edges and goal is to represent all edges in a simpler way and also to identify neighbours.
• The first idea to estimate the distance between the pair of substrings is to identify the neighbors as there are lot of edges in graph and the goal is to represent all edges in a simpler way.
• Second idea is to assume that every vertex in $G\Delta$ has degree $\leq$n$^E$ and here no triangular inequality is used because the degree is small.
• We use Edit distance structure i.e optimal matching substring.




## 6.1 Algorithm

• Approach for this case is to sample $x_i$ and compute $ED(x_i,y_j)$.
• If $ED(x_i,y_j)$ is small then $(x_i + \Delta, y_j + \Delta)$ are potential edges in $G\Delta$.
• Now check if these potential edges are real edges

This approach does not identify all edges but its safe to ignore all the edges.

————————————END————————————