21 Reductions

Tuesday, April 1, 2025 14:03

Many fast algorithms are already implemented
and available via libraries
Easier to use these compared to developing new algorithms.

Examples from HW:
- pattern matching / vector distances
  $O(n \log n)$ algo by using polynomial multiplication

- Inequalities with 2 variables $x_i - x_j \le c_{ij}$
  using LP algorithm would have been difficult
  Instead we just ran Bellman Ford (finding neg. cycle in graph)

These are <u>reductions</u>
Solve Problem A via algorithm for Problem B

Pattern Matching reduces to polynomial multiplication
2 var per inequality LP reduces to negative weight shortest path/cycle

Benefits:
1) Makes algorithm design easier. Use existing algos.
2) Allows us to argue how easy/hard problems are relative to each other.

If we believe that $O(n \log n)$ time is the best possible for pattern matching
then we must also believe that $O(n \log n)$ for polynomial multiplication is the best.

Reason: If there was faster (eg $O(n)$ time) algo for poly multiplication
then we could solve pattern matching in $O(n)$
but that would contradict our belief..

Referred to as "conditional lower bound" because the lower bound

Referred to as "conditional lower bound" because the lower bound
of $\Omega(n \log n)$ for poly. mult. depends on the condition that
pattern matching needs $\Omega(n \log n)$

You came up with algo
but it is slow.
Should we spend more effort/time finding something faster?
Can we explain why it's hard?
Can we find some necessary condition/relaxation (eg approximate output)
that is needed for a faster algorithm?

Example: Nearest Neighbor Data Structure

store each v_i $O(nd)$

- Init $(v_1 v_2 \dots v_n \in \mathbb{R}^d)$

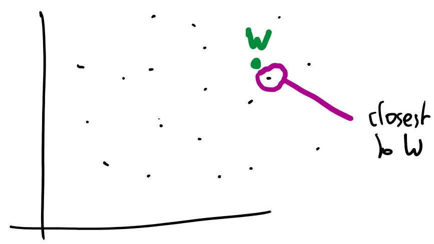
- Query $(W \in \mathbb{R}^d)$ Return v_i that is closest to W

for-loop try each v_i

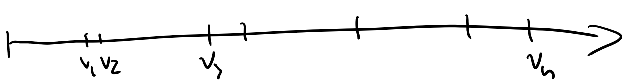
$O(nd)$

$\min_i \|W - v_i\|_2$

$$\|u - v\|_2 = \sqrt{\sum_{i=1}^{d} (u_i - v_i)^2}$$



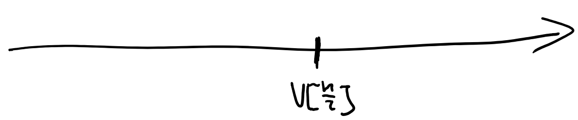
Compare to $d=1$ case
$v_1 \dots v_n$ are $\in \mathbb{R}$ so can sort them



then binary search during query

Init $O(n \log n)$
Query $O(\log n)$

Search $(V[1 \dots n], W)$
  if $V[\frac{n}{2}] > W$
    recurse on $V[1 \dots \frac{n}{2}]$
  else
    recurse on $V[\frac{n}{2}, \dots n]$



Next: Argue that we cannot do better than $O(nd)$ search/query.

Def: OV-Problem (orthogonal vector)

Input $v_1 v_2 \dots v_n \in \{0,1\}^d$
$w_1 w_2 \dots w_n \in \{0,1\}^d$

Output Yes/No if there exists $w_i^T v_j = 0$
ie pair of orthogonal vectors

Naive algo, try all n^2 pairs
$O(n^2 d)$ time.

Question is this optimal? Could we do $O(n^{1.999} \text{poly}(d))$?

Def: OV-conjecture
there exists no algorithm for the OV-Problem
with $O(n^{2-\varepsilon} \text{poly}(d))$ time.

Remark $O(\frac{n^2}{\log n} \text{poly}(d))$ exists

$$\frac{n^2}{\log n} = n^{2 - \frac{\log \log n}{\log n}}$$ goes to 0 for large inputs

Conjecture basically says that no better than log improvements are possible.

Reduction from OV to NN data structure

Use NN data str. to solve OV problem.

$OV(v_1 v_2 \dots v_n \in \{0,1\}^d, w_1 \dots w_n \in \{0,1\}^d)$

for $i = 1 \dots n$

$\tilde{v}_i = \begin{pmatrix} v_i \\ \mathbb{1} - v_i \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \{0,1\}^{3d}$ // $\mathbb{1} - v_i$ is vector v_i but we flip the 0s and 1s

$\tilde{w}_i = \begin{pmatrix} -w_i \\ 0 \\ 0 \end{pmatrix} \in \{-1,0,1\}^{3d}$

$= d + d - 2 \tilde{v}_j^T \tilde{w}_i$

$$\bar{w}_i = \begin{pmatrix} -w_i \\ 0 \\ \vdots \\ 0 \\ \mathbb{1} - w_i \end{pmatrix} \in \{-1,0,1\}^{3d}$$

$$= d + d - 2\bar{v}_j^T \bar{w}_i$$

$$// \|\bar{v}_j - \bar{w}_i\|_2^2 = \|\bar{v}_j\|_2^2 + \|\bar{w}_i\|_2^2 - 2\bar{v}_j^T \bar{w}_i$$

$$\sum_{k=1}^{d} (\bar{v}_j)_k^2 = \#\text{ of 1s in } \bar{v}_j = d$$

NN.Init($\bar{v}_1 \, \bar{v}_2 \dots \bar{v}_3$)

for $i = 1 \dots n$

    $\bar{v}_j$ = NN.Query($\bar{w}_i$)

    if $\|\bar{v}_j - \bar{w}_i\|_2 == \sqrt{2d}$

        return True // $v_j^T w_i = 0$

return false // no orthogonal pair exists.

- $\|\bar{v}_j - \bar{w}_i\|_2^2 = 2d \iff \bar{v}_j^T \bar{w}_i = 0 \iff v_j^T w_i = 0$
- $\bar{v}_j^T \bar{w}_i \le 0 \quad \forall i,j, \quad v_j^T w_i \ge 0$
- $\|\bar{v}_j - \bar{w}_i\|_2^2 = 2d - 2\bar{v}_j^T \bar{w}_i = 2d + 2v_j^T w_i \ge 2d$

If there exists $v_j^T w_i = 0$

then $\|\bar{v}_j - \bar{w}_i\|_2^2 = 2d \le \|\bar{v}_{j'} - \bar{w}_{i'}\| \quad \forall i', j'$

so algo returns true

If there is no $v_j^T w_i = 0 \Rightarrow v_j^T w_i \ge 1$

$\Rightarrow \|\bar{v}_j - \bar{w}_i\|_2^2 = 2d + 2v_j^T w_i \ge 2d + 2 \quad \forall i,j$

$\Rightarrow$ so algo must return false

Time complexity

$P(n,d)$ time for Init of NN

$Q(n,d)$ time for query of NN

then OV can be solved in time $O(nd + P(n,d) + nQ(n,d))$

If $P(n,d) = n^{2-\varepsilon}\,\text{poly}(d)$
$Q(n,d) = n^{1-\varepsilon}\,\text{poly}(d)$

then OV can be solved in $O(nd + n^{2-\varepsilon}\,\text{poly}(d) + n \cdot n^{1-\varepsilon}\,\text{poly}(d))$

$= O(n^{2-\varepsilon}\,\text{poly}(d))$

Assuming the OV conjecture

then there is no NN data structure with subquadratic initialization time and sublinear query time.