# Lower Bounds via the Cell-Sampling Method

Omri Weinstein
Columbia

# Locality in TCS

- Locality/Sparsity is central to TCS and Math:
  - PCP Theorems
  - Locally-Decodable Codes (LDCs)
  - Data Structures
  - Derandomization (expanders, k-wise independence)
  - Matrix Rigidity
  - Compressed sensing
  - Graph decompositions (LLL)
  - ...

# Limits of Local Computation ?

- Typically, locality comes at price (e.g. blowup in size of input)

- How can we prove lower bounds on this tradeoff?

- This Tutorial:

  "Cell Sampling": A simple & unified technique. Proves highest known unconditional lower bounds in various computational models.

# Plan

- Cell-Sampling technique
- Applications:

  I) Time-Space Tradeoffs in Data Structures (near-neighbor search)

  II) LB for Locally Decodable Codes (rate vs. locality)

  III) Matrix Rigidity (sparsity vs. rank)

- Limits of cell-sampling method

# Information Theory 101

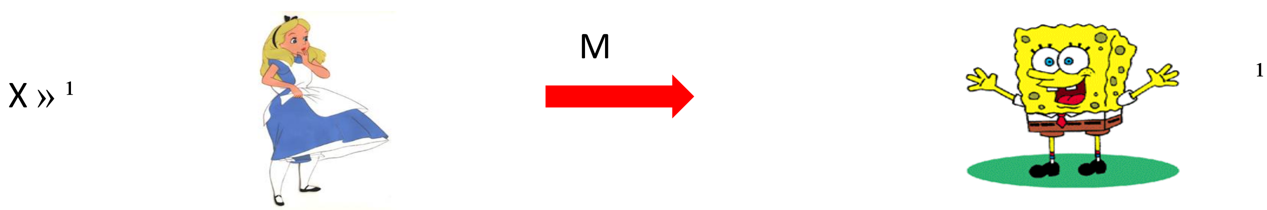
- Entropy : For random variable X » ¹ ,

$$H_{¹}(X) := \sum_{x2\,X} ¹(x) \lg(1/¹(x)) = \mathbf{E}_X[\lg 1/¹(X)]$$

Captures how “unpredictable” X is – E.g., H(Ber(½)) = 1 bit, $H(\mathrm{Unif}_n) = \lg(n)$ bits .

- Conditional Entropy : $H_{¹}(X|Y) := \mathbf{E}_y[H_{¹}(X\,|Y=y)]$

Thm (Shannon ’48): $\mathbf{E}_{¹}$[cost of sending X] ¸ $H_{¹}(X)$ bits . (tight by Huffman code)



## Cell Sampling

- LBs on "locality" via compression argument.

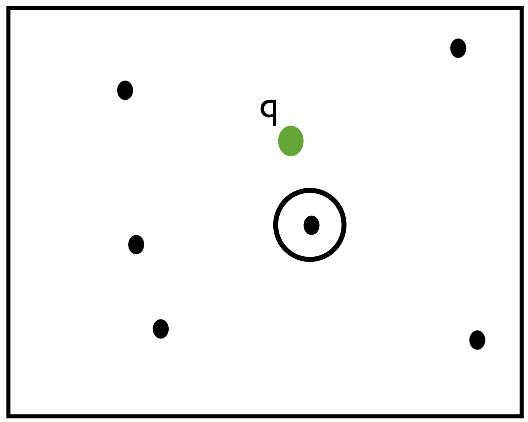
- High-level idea:
  Too-good-to-be-true "local" Algorithm ➔ impossible compression of input.

  (Typically not enough by itself – need to combine argument with extra features/ structure of problem, e.g., geometric/ combinatorial etc – more on this soon)

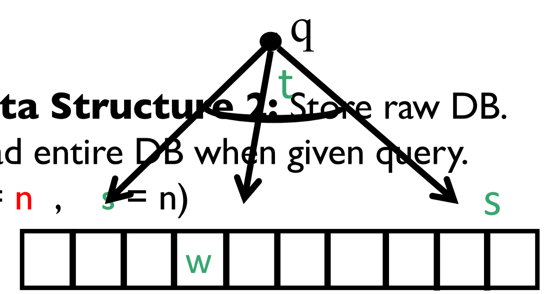
- Let's exemplify this method by proving time-space tradeoffs for data structures.

# Data Structures LBs (“Cell-Probe” model)

- DS = “compact” representation of info in database, so that queries about data can be answered quickly.
- Static Data Structures: Given data X of n elements in advance (e.g., graph, string, set of pts, etc.), *preprocess* it into small memory s so that 8 query q 2 Q can be computed fast with t memory accesses (computations free of charge!).
- *NNS* : Data = n pts in $\{0,1\}^d$




**Data Structure 1**: Precompute and store all answers in lookup table.
($t = 1$ , $s = 2^d$ )

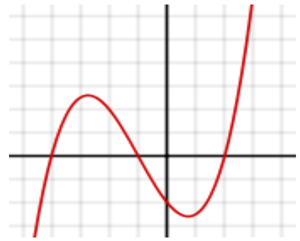



**Data Structure 2:** Store raw DB. Read entire DB when given query.
($t = n$ , $s = n$)

- Data Structure LBs: Is there anything in between ? Study time-space tradeoffs (s vs. t).

# Ex: Polynomial Evaluation

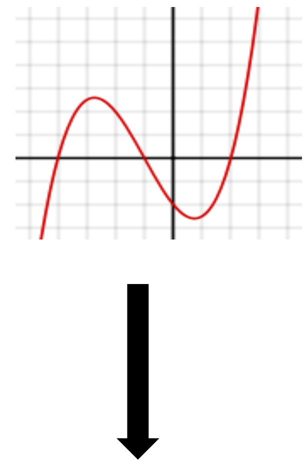
- PolyEval:
  - Input: Random degree-$n$ polynomial $\mathrm{P}\ 2_{\mathrm{R}}\ \mathbf{F}_m$ ($m = n^2$).
  - Query: Element $x\ 2\ \mathbf{F}_m$ ➔ Return $\mathrm{P}(x)$.
  - $\mathrm{H}(\mathrm{P}) = (n+1)\lg(m)$ (n+1 random coeff $2\ \mathbf{F}_m$ , word size w=lg m)



- Trivial: s= n+1 , t = n+1 (read all coefficients)

Thm : Any **D** with space s=O(n) must have query time $t$ ¸ $\Omega(\lg n)$.

# Proof (via cell-sampling)



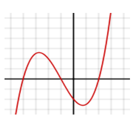
- Assume toward contradiction $\exists$ **D** with space $s=10n$ and query time $t=o(\lg n)$. (recall word-size w=lg m).

$\Rightarrow$ Use data structure to encode P using less than $(n+1)\lg m$ bits (!)

$|\mathbf{D}(P)| = 10n$ (words)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 222 | j41 | If… | Else | 389 | j4# | $y | j13 | |

- Alice Encodes P :
  - Build DS **D** on P.
  - Alice picks a *random sample* $C$ of mem cells:
    - Include each cell w.p $p := 1/100$.
  - Sends Bob $C$ (addresses + contents).
    - $\mathbf{E}$[message length] $= (s/100)\, 2w < 10n\ 3\lg n/100 < (n+1)\lg(m)$ bits $= H(P)$ !



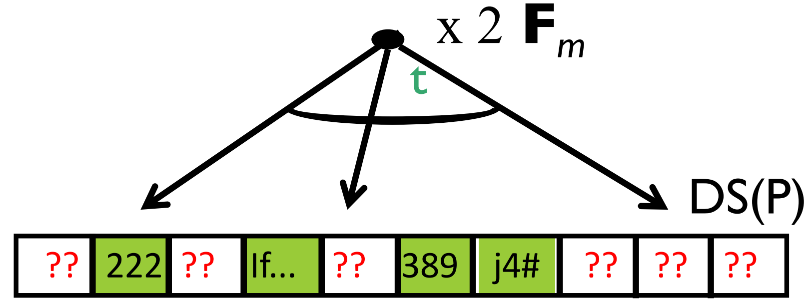
P

$C$

- Expected cost < H(P) bits.

- Decoding P :
  - Bob iterates through all $x$ 2 $\mathbf{F}_m$:
    - Run query algorithm of DS on $x$:
      - If read outside $C$, discard $x$.
  - Probability recover answer to fixed $x$ = $p^t$ = $(1/100)^t$
    - $E_C$[# surviving queries x] = $m$¢ $(1/100)^t$= £($n^2$ ¢ $2^{-t}$) = $n^{2-o(1)}$

    But every n+1 queries determine P, hence Bob learns H(P) ~n¢lg$m$ bits of info from <n¢lg$m$ bits of CC (!)




$t$=o(lg $n$).

) $t$=Ω(lg $n$).  [ More generally : t ¸ Ω(lg(n)/lg(s/n)) ]

- Special property of polynomials: Any *large enough* set of answers recovers entire input ("n-wise independence"). Most natural problems don't have this feature..

# Time-Space Lower Bounds for Near-Neighbor Search

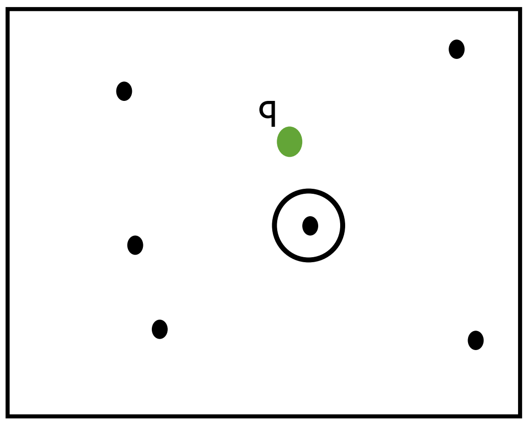
# Nearest-Neighbor Search

- **NNS**: Preprocess dataset $X = x_1, \ldots, x_n$ in metric space (say $R^d$ with $l_1$ norm), s.t given a query q 2 $R^d$, closest point in X to q can be retrieved as fast as possible.

$\{0,1\}^d$ (d = 100¢lg(n) )

**Data Structure 1**:
Precompute and store all answers in lookup table.

($t = 1$ , $s = 2^d = n^{100}$)




**Data Structure 2:**
Store DB (graph) as is. Read entire DB when given query (linear scan).

($t = n$ , $s = n$)

- Better time-space tradeoffs (s vs. t) ?
- Probably not… ("Curse of dimensionality")

# Approximate NNS

- **(c,r)-ANN**: Relaxed requirement: Given *radius* r and apx parameter c > 1, if
  9 $x_i$ s.t $|q - x_i| < r$, return $x_j$ s.t $|q - x_i|$ · c¢r.

- "Robust" version has dramatic consequences (LSH) :
  $s = n^{1+²}$, $t = O(n^{²})$ for c=(1/²)-apx. ($l_1$, $l_2$ [IM98, Pan06, AR15..])

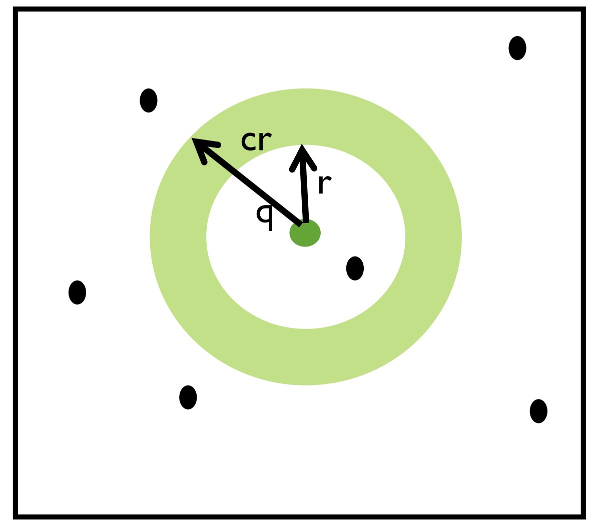
- Is this optimal? Can we get near-linear space an $t = n^{o(1)}$?

- Thm [PTW'10, LMWY'19] :
  8 DS **D** for (1/²)-ANN over d-dim Hamming space,

  t ¸ $\Omega(d / \lg(dw))$ for $s = O(n)$ space.

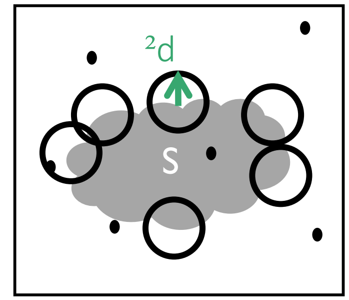
- For d = £(lg n) → $\Omega(\lg n/\lg\lg n)$.

$\{0,1\}^d$



# Proof

$2^d = n^{10}$

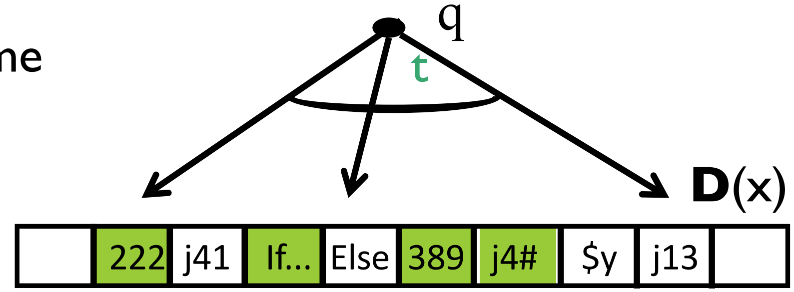



- Consider X = $x_1,\ldots x_n$ » U($2^d$) , d = 10¢lg(n) (whp, $B_{2^2 d}(x_i)$ are all unique)
- Isoperimetric Fact : 8 |S| = $2^{(1-^2\,^2)d}$ ) ¡₂(S) ¸ $2^{d-1}$
  (*Harper's Inequality*: least-expanding subset of hypercube = ball)

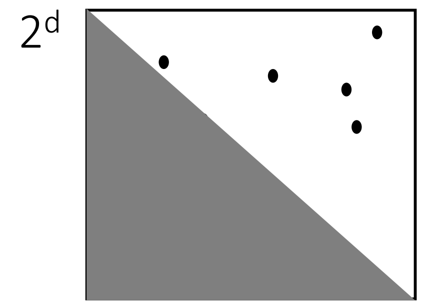
(*) Cor: 8 fixed subset of |S| & $2^{(1-^2\,^2)d}$ r-ANN queries (r=²d), Pr $_{x\_i\ »\ U}$ [$x_i$ 2 ¡₂(S)] ¸ ½

) n/4 data points $x_i$ fall into any such S whp.

- Consider (0-err) **D** solving ANN with s=10n space (say), t = o($^{22}$d / lg w) query time.
- **D** ➔ too good to be true (rand) compression scheme for encoding n/8 $x_i$'s using o(n¢d) = o(n lg n) bits !
- Alice samples 8 cell c 2 **D**(x) iid wp p := 1/100w.



- Alice sends Bob contents + addresses of sampled cells
  $\mathbf{E}[|C| ¢ w] = 2psw < n/10$ bits (recall $p = 1/100w$)

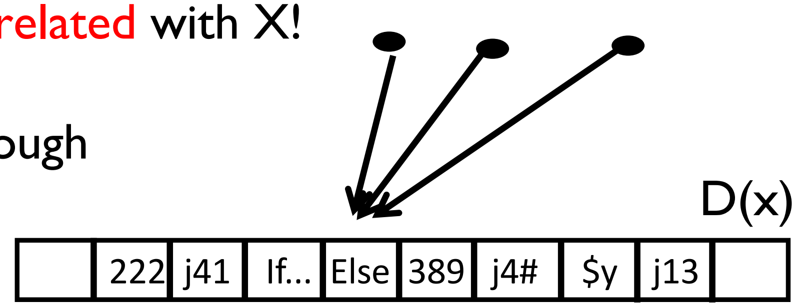



- $\mathbf{E}_{C,X}$[# *surviving queries* Q] = $2^d ¢ \Pr_{C,X}[\mathbf{D}(q)$ ½ $C] = 2^d ¢ p^t$
  & $2^d ¢ (1/100w)^{o(^{22} d/\lg w)} > 2^{d - o(^{22} d)} > 2^{(1- ^{22})d}$ (p = 1/100w)

- If it were the case that Q ? X (i.e., (X|Q) » U($2^d$)) ) By (*) (¡₂(Q) ¸ $2^{d-1}$),
  Bob would have been able to recover n/4 (say) $x_i$'s just from C → contradiction!

- But **D** is adaptive → surviving queries heavily depend on *content* of cells (function of X) ➔ Surviving set Q = Q(X) correlated with X!

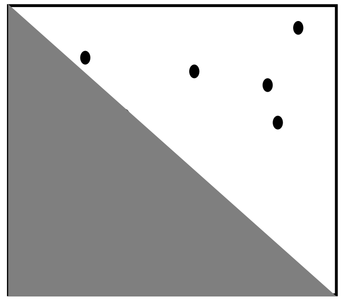
- In principle, all $x_i$'s could all fall into :¡₂(Q) (even though it has covers ½ the space).



- Obs: Q(X) *determined* by only |C|w < o(n) bits (actually n/10 but good enough)

(DPI) H(X|¡$_2$(Q(X))) > nd – o(n) bits : X is still "close" to U($2^d$)…

$2^d$



- Formalize this using simple "geometric packing" argument: Suppose fsoc that > n/4 $x_i$'s fall outside ¡$_2$ (Q) ➔ these pts are (essentially) (d-1)-dim.

➔ Can save 1 bit for their encoding, already gives impossible compression…

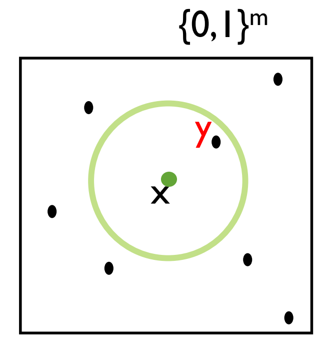
- So may assume > n/4 $x_i$'s indeed fall into ¡$_2$(Q) as desired, in which case prev "naiive" analysis goes through.

- Cell-Sampling also used in highest (~lg^2 n) dynamic data structure lower bounds… [Lar12, LWY18]

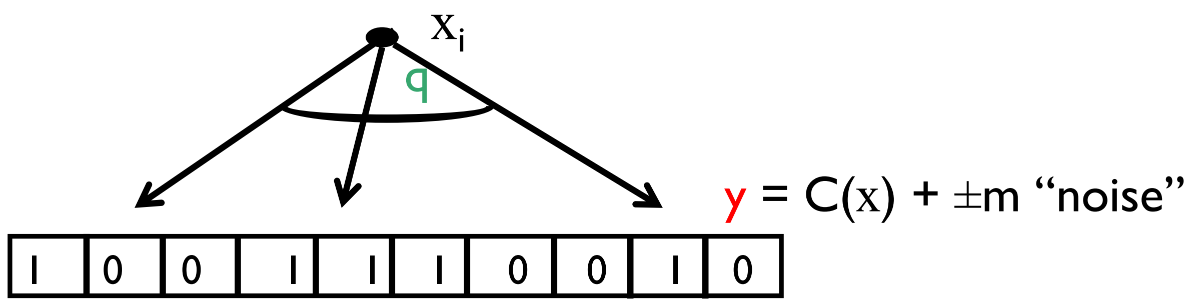
# Lower Bounds on Locally Decodable Codes

# Error Correcting Codes

± = (frac) distance of C



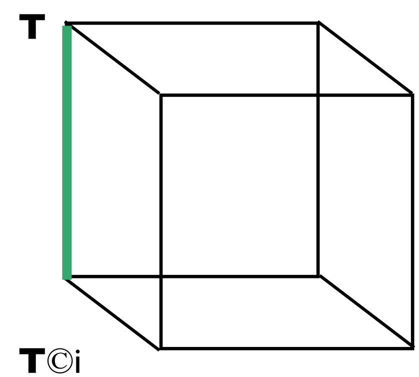

- ECC C : $\mathbf{F}^n \mapsto \mathbf{F}^m$ (m > n) s.t 8x,y |C(x) – y| < ± ) x recovered from y.
- For ± = ¼ (say), 9 ECCs with constant rate m = O(n).
- But decoding requires reading entire codeword C(x), even if just want $x_i$ .
- If interested in decoding only $x_i$ , can hope to read few (ideally O(1)) bits of C(x) ?
- q-LDC C : $\{0,1\}^n \mapsto \{0,1\}^m$ s.t 8x,y |C(x) – y| < ± ) recover $x_i$ by reading only q bits of y.



# Locally-Decodable Codes

LDCs must randomize!

- q-LDC $C : \{0,1\}^n \mapsto \{0,1\}^m$ s.t 8x, $d(C(x),y) < 1/4$ ) recover $x_i$ by reading q bits (whp).
- Tradeoff b/w q and m ? Is *q=O(1)* possible with m=O(n) ??
- Claim: for q=1, <u>impossible</u> (intuition: some bit j 2 [m] must convey info on $\Omega_{\pm}$(n) $x_i$'s)
- q=2 ? Possible with m = $2^n$ :
- To Encode x 2 $\{0,1\}^n$, store 8 T µ [n] $C(x)_T$ := $©_{i2T}\, x_i$ (m = $2^n$)
- To Decode $x_i$ from y, pick T$2_R$[n] & query $y_T$ © $y_{T©i}$
- $Pr_{random\ T}$[both $y_T$ & $y_{T©i}$ uncorrupted] ¼ 1-2± ☺
- LB on tradeoff b/w q and m ? Is *q=O(1)* possible with m=O(n) ??



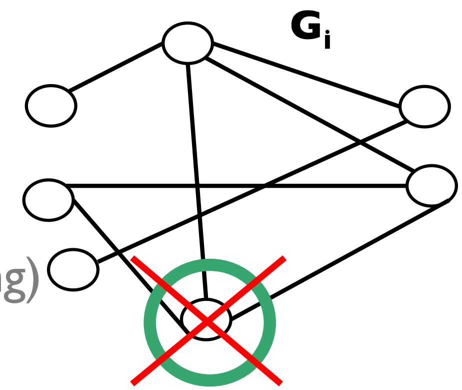
**Thm** [KatzTrevisan'00] : 8 q-LDC , m ¸ ~$\Omega_{\pm}(n^{1+1/q})$

- **Proof**: For q-LDC C, the query graph $\mathbf{G_i}$ of C is the *q-hypergraph* containing possible q-tuples from which $x_i$ can be recovered ($|\mathbf{G_i}|$=m) .
- "Smoothness": Intuitively, q-edges of $\mathbf{G_i}$ are ¼ uniformly distributed: No vertex j 2 $\mathbf{G}_i$ has (weighted) <u>degree</u> ¢ > q/±m (o.w adversary can corrupt it. Avg deg = q/m (Markov)).
- <u>Corollary</u>: 8 q-LDC, $\mathbf{G}_i$ contains Matching $|M_i|$ ¸ ±m/$q^2$ .
- Proof : Max |Matching($\mathbf{G}_i$)| in q-hypergraph

  ¸ Min |VC($\mathbf{G}_i$)| / q (any VC must pick 1 v from max matching)

  ¸ 1/¢ ¢ (1/q) (each vertex covers · ¢ "mass")

  ¸ 1 / (q¢q/±m) > ±m/$q^2$ (max-deg ¢ <q/±m)



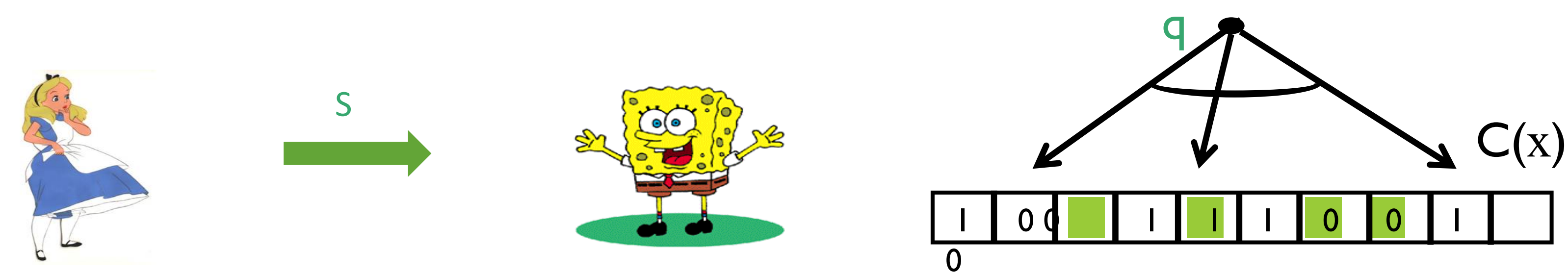
- Use LDC to compress input (via Cell-Sampling) :
- Alice builds C(x), samples 8 j 2 [m] w.p p:= n/10m ➔ **E**[S] = m*p = n/10 bits.
- 8 i 2 [n] $Pr_S$[Bob can recover $x_i$] ¸ $Pr_S$ [ $\bigcup_{e\ 2\ M_i}$ "e survives"]

= $\sum_{e\ 2\ M_i}$ $Pr_S$ ["e survives"] (disjoint events since $M_i$ = matching !)

= $|M_i|$ ¢ $p^q$ = $\boxed{\pm m/q^2 ¢ (n/10m)^q < 3/4}$ for >n/2 i's (o.w. recover n/2 $x_i$'s from < n/10 bits!)

, $m^{q-1}$ & $n^q / q^2$ , $\Omega_\pm(n^{1+1/q})$ ■

x $2_R$ $\{0,1\}^n$



# Matrix Rigidity

## Matrix Rigidity

$$\underbrace{M \neq A}_{t\text{-sparse}} + \underbrace{B}_{\mathrm{rk} \leq \varepsilon n}$$

- **Def:** A matrix M 2 $\mathbf{F}^{m\mathrm{x}n}$ is t-Rigid if decreasing its rank n → n/2 requires modifying ¸ t entries in some row.

- M "t-far" from any low-rank matrix . (assume m=poly(n))

- **Thm:** $n^2$£n Vandermonde matrix is $\Omega$(lg n)-Rigid.

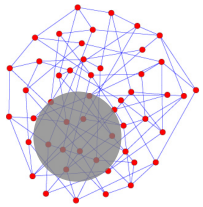
$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & \vdots & \vdots & & \\ \vdots & \vdots & \vdots & & \\ 1 & x_n & x_n^2 & & x_n^n \end{bmatrix}$$

- Cell-Sampling + Subadditivity of Rank :

- <u>Sketch</u>: Sample 8 column of A w.p 1/10 (call it S) ➔ If every row is < lg(n)/100 sparse ➔ 9 n rows R s.t |S|=n/10 "covers" **all** 1's in these rows ➔ rk($A_R$ + $B_R$) · rk($A_R$) + rk($B_R$) · n/10 + n/2 <n. But every submatrix of V is also full rank (n) !

# Limits of Cell Sampling

- Cell Sampling relies on simple fact : In *every* graph of size n with m edges, there is a small set (~pn) containing “nontrivial” (~m/$2^p$) edges.

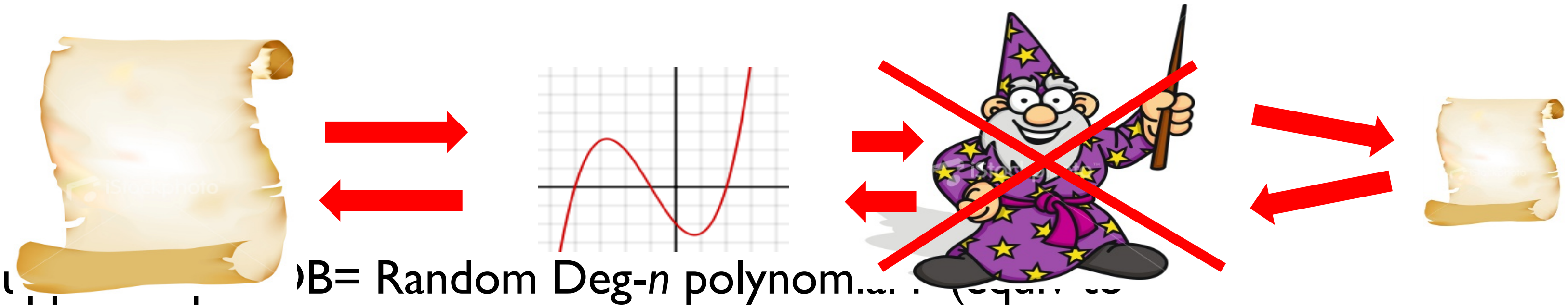


- Tight for *expanders*… (any o(lg n) subset contains o(n) edges)
- ) log(n) is a fundamental limit of cell sampling ☹
- Still very useful technique, that unifies/explains current barrier in complexity holy-grails (LDC/Rigidity/DS…)

# Thanks!

# Cell Sampling: Time-Space LBs via Compression

- Assume for contradiction that a "too-good-to-be-true" DS exists for PolyEval with $t$ < lg $n$ and linear space (s=O(n)).



- Su… B= Random Deg-$n$ polynom… (equiv to n-letter text T w. random symbols : P(i) = $T_i$).

- Use magic data structure to encode and decode the input set using less than $y$ symbols. A contradiction!