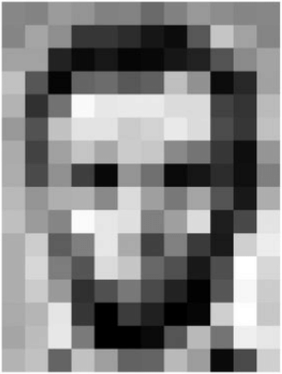
Sublinear Time Algorithms

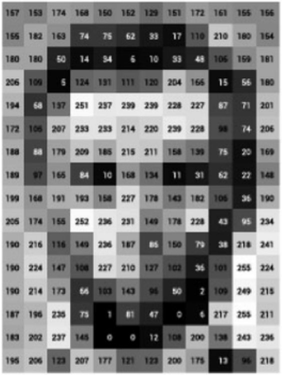
Motivation

- Algorithm takes input of size n
  - Pixels of an image
  - Entries in an adjacency matrix of a graph.
- n is large
- Assumed to be too slow to look at entire input

Pixels of an Image

Input size n is the number of pixels

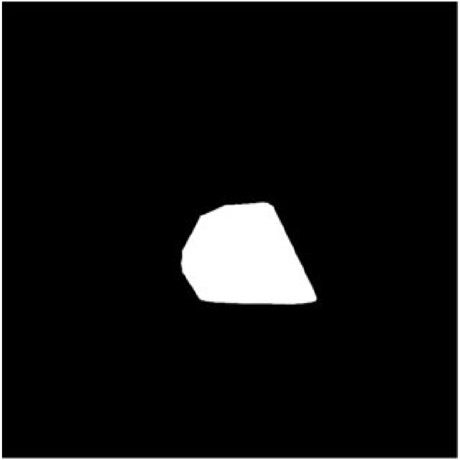




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

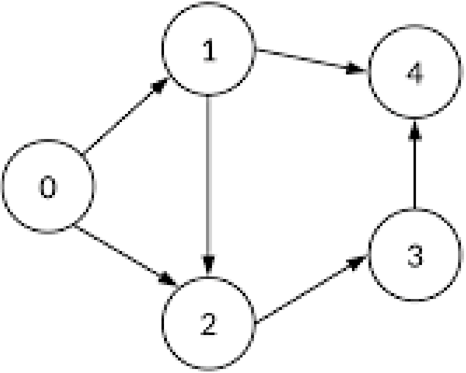
Example Problem on Image

Is the image a convex shape?



Graph Adjacency Matrix

Input size n is the number of matrix entries

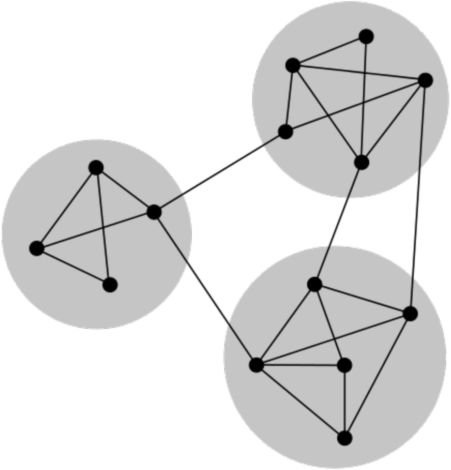


Adjacency Matrix

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Example Problem on Graph

Is the graph connected?



Sublinear Time Algorithm

Definition

A sublinear time algorithm is an algorithm whose execution time, $T(n)$, grows slower than the size of the problem, n, but only gives an approximate or probably correct answer. So $T(n) = o(n)$.

Examples of sublinear time:

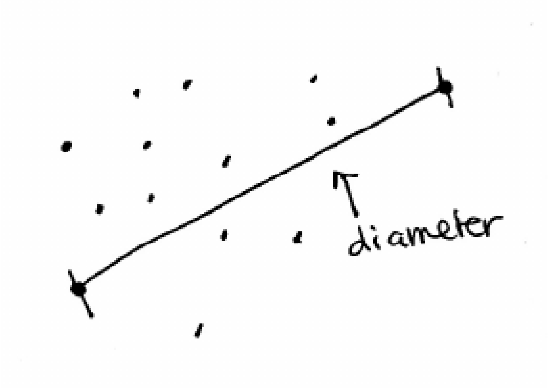
- $O(\log(n))$
- $O(1/\epsilon)$ for constant ϵ
- $O(\sqrt{n})$

Sublinear Time Algorithms

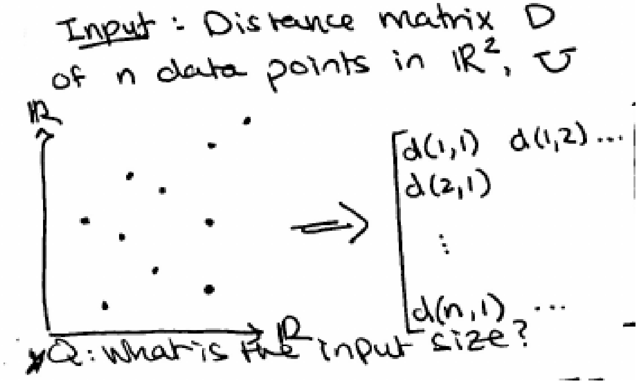
- Can't answer "exactly" types of statements.
- Approximate answers
- Random samples

Diameter of a Metric Space

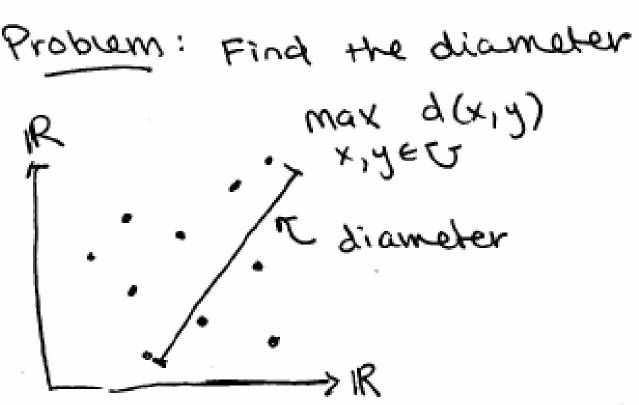
In this problem we wish to find the largest distance between any pair of points in a finite subset of $\mathbb{R}^2$.



Diameter of a Metric Space



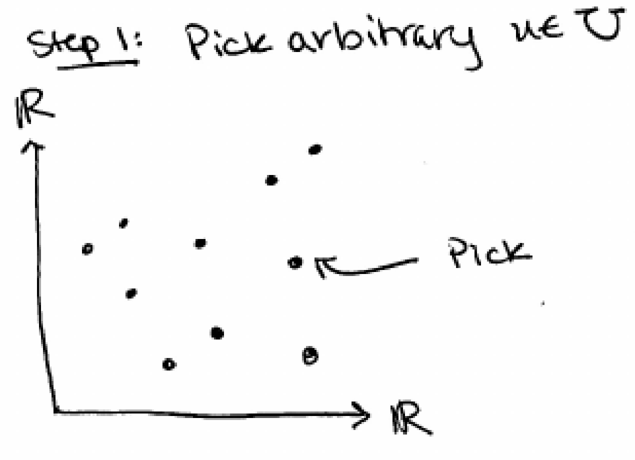
Diameter of a Metric Space



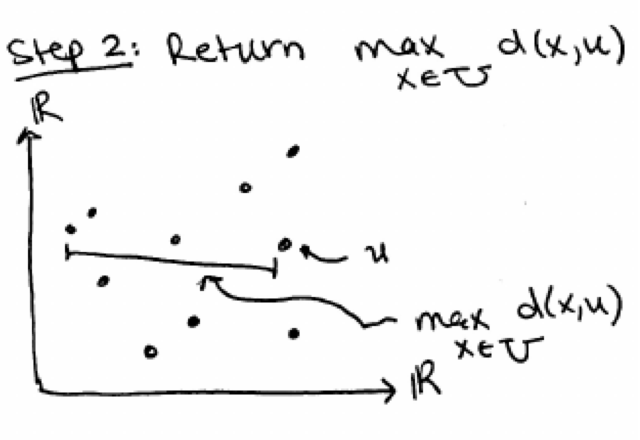
Diameter of a Metric Space

- Input is of size n^2
- Looking at each distance and finding max is linear time
- Instead propose algorithm that gives approximation

Algorithm for Diameter of a Metric Space



Algorithm for Diameter of a Metric Space



What is the run time of this algorithm?

Diameter of a Metric Space

What is the run time?

Input size = n^2

Run time = $O(n)$

Algorithm is sublinear!

How does it perform?

$$\max_{x \in V} d(x,u) \geq \frac{1}{2} \max_{x,y \in V} d(x,y)$$

diameter

Q: How to prove this?

Diameter of a Metric Space

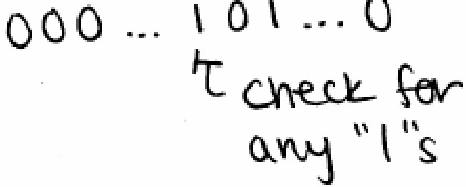
<u>Pf.</u> Let u', v' be the elements of U such that $d(u', v') = \max_{x,y \in U} d(x, y)$. Then the Δ property states that

$$d(u', v') \le d(u', u) + d(u, v')$$

$$\le 2 \max_{x \in U} d(x, u). \quad \boxtimes$$

Testing for an all “0” String

We now look at the problem of testing whether a binary string is all “0”s.



Testing for an all "0" String

Input : A string $w \in \{0,1\}^n$

Problem: Determine whether
w is all "0", return "YES"
if it is, "No" otherwise.
* Can you do this in sublinear
time?

Testing for an all "0" String

Can you answer this exactly in sublinear time?

NO. Anywhere in the input you don't look could be a "1".

Testing for an all "0" String

Alternative Problem: parameters $\epsilon, p \in (0,1)$

If all "0"s $\Rightarrow$ return "YES"

If at least ϵ fraction of w are "1"s (ϵn "1"s) $\Rightarrow$ return "NO" with probability $\geq p$.

Testing for an all "0" String

<u>Algorithm</u>

We will take a random sampling approach.

Testing for an all "0" String

Input: string $w \in \{0,1\}^n$
suppose that ϵ fraction
of w are "1"s.

So bigger $\varepsilon \Rightarrow$ more "1"s
smaller $\varepsilon \Rightarrow$ less "1"s

* If we uniformly randomly sample a spot in w, what is the probability it's a "0"?

Testing for an all "0" String

Probability of sampling a "0" is $1-\varepsilon$.

~~Tak~~ Suppose we take s independent, uniformly random samples.

* Probability of getting all "0"s?

Testing for an all "0" String

Probability of getting all "0"s is $(1-\varepsilon)^s$.

This is because if events A and B are independent then $P(A \cap B) = P(A)P(B)$.

Testing for an all "0" String

Witness Lemma

If a test catches a witness with probability $\geq \varepsilon$, then $s = \frac{\ln\left(\frac{1}{1-p}\right)}{\varepsilon}$ iterations of the test catches a witness with probability $\geq p$.

Testing for an all "0" String

Pf.

$P[\text{don't catch witness}] \le (1-\varepsilon)^s$

$\le e^{-\varepsilon s}$ (Identity $1-x \le e^{-x}$)

$= e^{-\ln\left(\frac{1}{1-p}\right)} = e^{\ln(1-p)} = 1-p.$

$\therefore P[\text{catch witness}] \ge p. \quad \blacksquare$

Testing for an all "0" String

Algorithm for approximate all "0" tester

Step 1: Take $s = \frac{\ln\left(\frac{1}{1-p}\right)}{\varepsilon}$ random samples from string w.

Step 2: If all "0" $\Rightarrow$ return "YES"

Otherwise $\Rightarrow$ return "NO".

Testing for an all "0" String

* If all "0"s, what will the algorithm return?

* If at least ε fraction of w are "1"s?

Testing for an all "0" String

Apply the witness lemma.

*witness = A "1"

Witness lemma states that if $\geq \epsilon$ fraction of w are "1", then "NO" is returned w/ prob $\geq \rho$.

* What about if $< \epsilon$ fraction?

Approximate Algorithm for All "0" String

Suppose that $\epsilon = 0.2$ and $p = 0.9999$.

- What is s?
- Why does it not depend on the input size n?
- What will the algorithm do on input:
  - "0000000000000000000000000000000000000000"
  - "1001110001000111100111000111011001011001"
  - "0000000000000010000000000000000000001000"

ϵ-far

Input x is ϵ-far from property P if ϵ fraction of x has to be changed in order for x to have property P. E.g. P is whether x is an all "0" string.

What is the ϵ for the below input for the property of being all "0"?

- "0100000000"
- "0000000000"
- "0010101101"
- "1111111111"

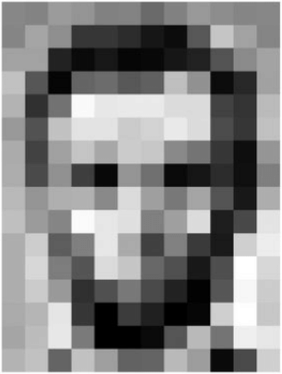
ϵ-tester

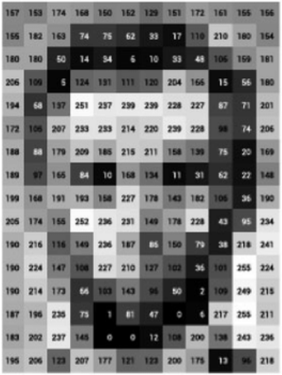
An algorithm with parameters $\epsilon, p \in (0.1)$, is an ϵ-tester of property P if:

- If input x has property P it returns "YES" with probability at least p
- If input x is ϵ-far from property P it returns "NO" with probability at least p.

Pixels of an Image

Input size n is the number of pixels

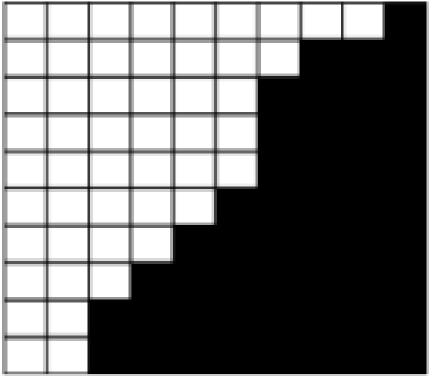




157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

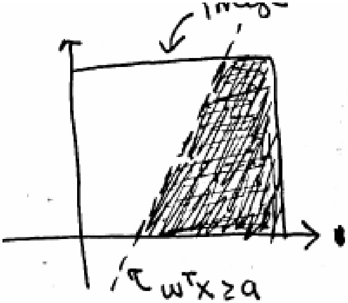
Half Plane Tester

The half plane tester problem is, given an image, return “YES” if the image is a half plane, and “NO” otherwise.



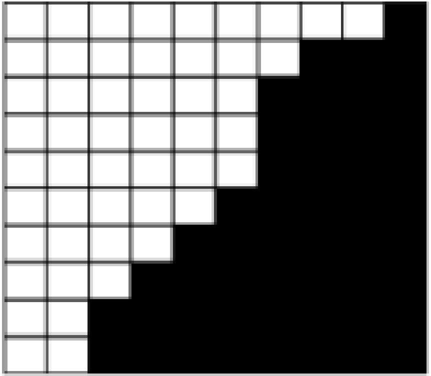
Half Plane Tester

An image is a half plane if there exists a $w \in \mathbb{R}^2$ and an $a \in \mathbb{R}$ such that pixel x is black (white) if and only if $w^T x \geq a$.



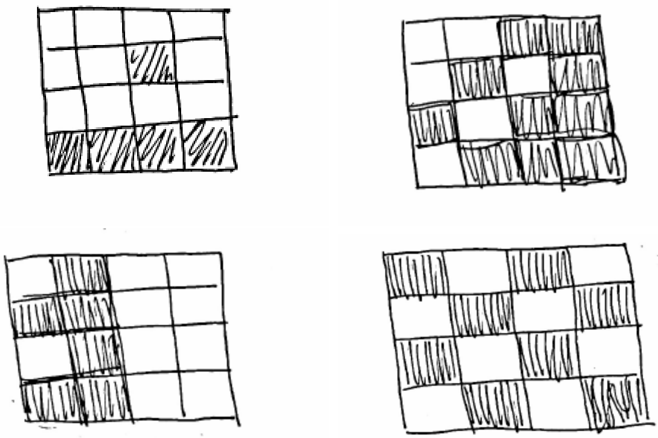
Half Plane Tester

The ϵ-tester version of the half plane tester problem is, given an image, return "YES" if the image is a half plane, and if the image is ϵ-far from being a half plane return "NO" with probability at least p.



ϵ-far from a half plane

What is the ϵ in the examples below?

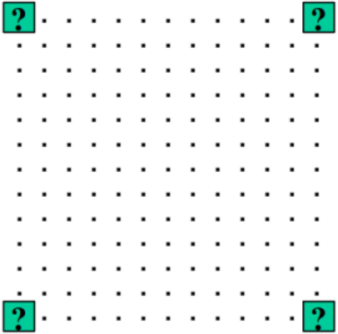


ϵ-tester for Half Plane

- Step 1: Determine what type of half plane this could be
- Step 2: Use random sampling in order to confirm, or reject

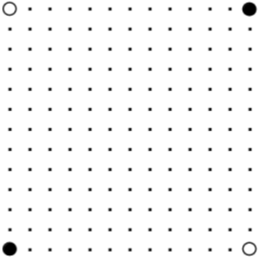
ϵ-tester for Half Plane

Step 1: Look at the four sides of the image, count how many have different color endpoints.



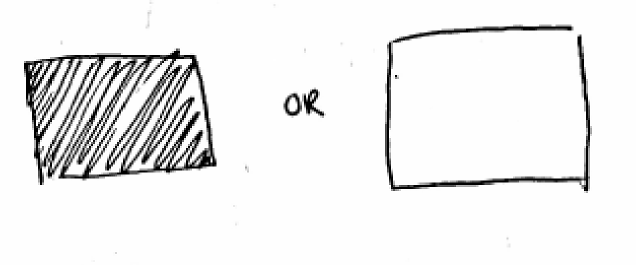
Case 1 of Algorithm

Case 1: All four sides have different color endpoint $\Longrightarrow$ return "NO"



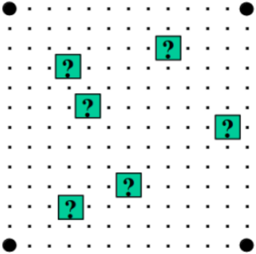
Case 2 of Algorithm

Case 2: No sides have different color endpoints



Case 2 of Algorithm

Do random sampling to decide with high probability if all one color



Witness Lemma

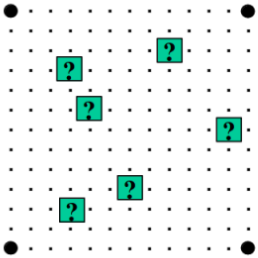
If a test catches a witness with probability $\geq \epsilon$, then

$$s = \frac{\ln\left(\frac{1}{1-p}\right)}{\epsilon}$$

iterations of the test catches a witness with probability $\geq p$.

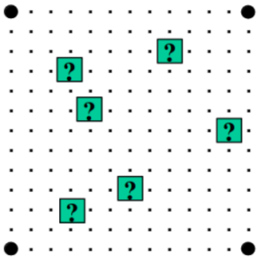
Case 2 of Algorithm

If ϵ-far from half plane $\implies$ all black (white) but with ϵ fraction of white (black) pixels.



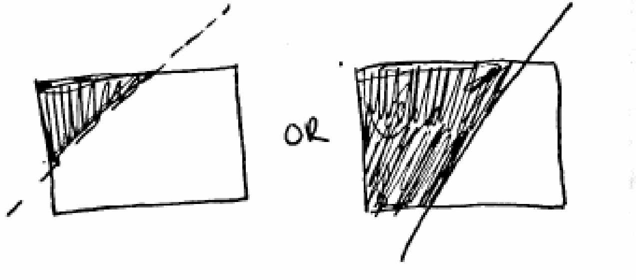
Case 2 of Algorithm

Take $s = \ln\left(\frac{1}{1-p}\right)/\epsilon$ random samples. If all black/white $\implies$ return "YES". Otherwise $\implies$ return "NO".



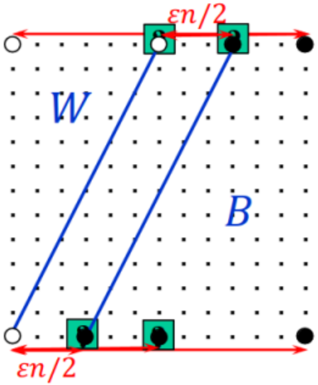
Case 3 of Algorithm

Case 3: Two sides have different color endpoints.



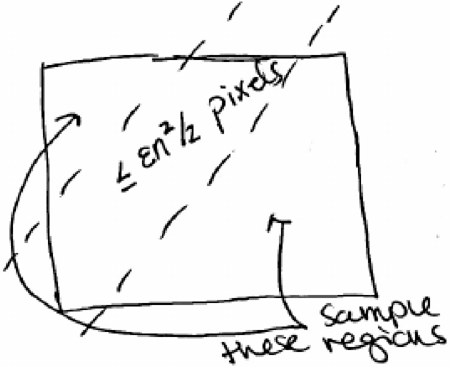
Case 3 of Algorithm

First, do a binary search of those sides to find approximately where the flip occurs.



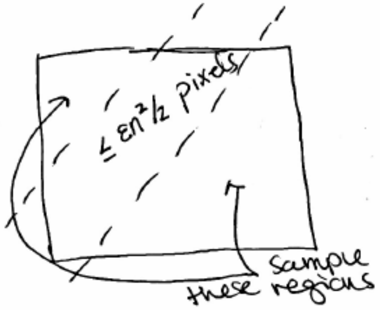
Case 3 of Algorithm

Then sample on either side of the region to test if each side is one color.



Case 3 of Algorithm

The border region contains at most $\epsilon n^2/2$ pixels, therefore if a picture is ϵ far from a half plane there has to be at least $\epsilon n^2/2$ "wrong" pixels in the sampled regions. So the probability of witnessing one is at least $\epsilon/2$.



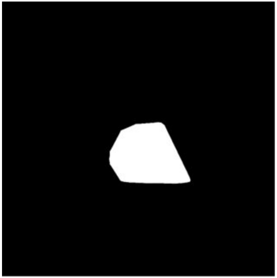
Analysis of ϵ-tester for Half Plane

- $\triangleright$ If half plane $\implies$ always returns "YES"
- $\triangleright$ If ϵ-far from half plane $\implies$ returns "NO" with probability $\geq p$
- $\triangleright$ Run time is looks at $O(\ln(1/\epsilon) + \ln(1/(1-p))/\epsilon)$ pixels

Other Problems on Images

Is the image a convex shape? Is the image a connected shape?

Raskhodnikova, Sofya. "Approximate testing of visual properties." International Workshop on Randomization and Approximation Techniques in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.



ϵ-far

Input x is ϵ-far from property P if ϵ fraction of x has to be changed in order for x to have property P. E.g. P is whether x is an all "0" string.

ϵ-tester

An algorithm with parameters $\epsilon, p \in (0.1)$, is an ϵ-tester of property P if:

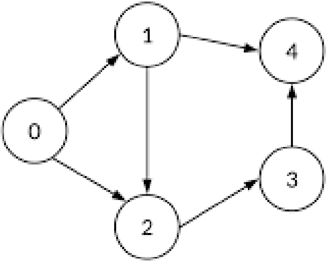
- If input x has property P it returns "YES" with probability at least p
- If input x is ϵ-far from property P it returns "NO" with probability at least p.

Large Graph

Consider algorithms run on large input undirected graph $G = (V, E)$, where V are n vertices and E are pairs of vertices representing an edge.

Adjacency Matrix Representation

$G = (V, E)$ can be represented as an adjacency matrix. If $|V| = n$, then the size of the adjacency matrix is n^2 entries. Good for dense graphs, wasteful for sparse graphs.

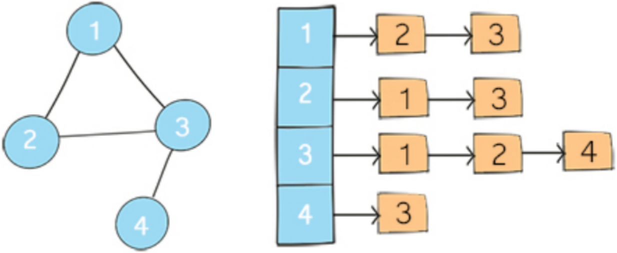


Adjacency Matrix

	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Adjacency List Representation

$G = (V, E)$ can be represented as an adjacency list. Good for sparse graphs. If G has bounded degree d, adjacency list is of size nd entries.



Graphs of Bounded Degree

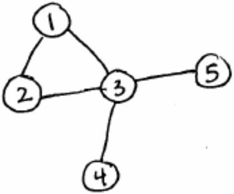
The degree of a node if the number of edges incident with it. A graph has bounded degree d if no node in its graph has degree more than d.

Adjacency List Distance

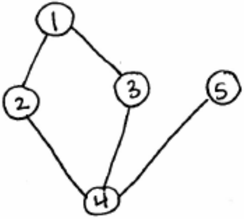
Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be graphs of bounded degree d, and $|V| = n$. Then the distance between G_1 and G_2 is

$$\frac{\#\text{ entries in adjacency lists that are different}}{dn}$$

Example of Distance Between Adjacency Lists



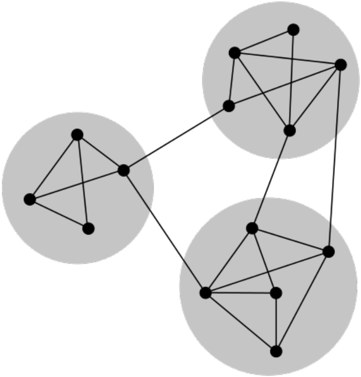
1	2	3	ϕ	ϕ
2	1	3	ϕ	ϕ
3	1	2	4	5
4	3	ϕ	ϕ	ϕ
5	3	ϕ	ϕ	ϕ



1	2	3	ϕ	ϕ
2	1	4	ϕ	ϕ
3	1	4	ϕ	ϕ
4	2	3	5	ϕ
5	4	ϕ	ϕ	ϕ

Connected Graph

A graph $G = (V, E)$ is connected if for every $u, v \in V$ there exists a path from u to v in G.

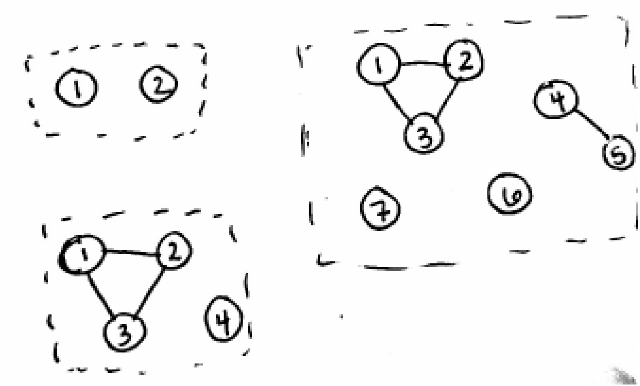


ϵ-far from Connected

Consider only graphs of bounded degree d. Graph $G = (V, E)$ is ϵ-far from connected if ϵ fraction of the adjacency list has to be changed in order for G to be a connected graph.

ϵ-far from Connected

Let $d = 3$.



ϵ-tester for Connectedness

Want algorithm with parameters $\epsilon, p \in (0.1)$, that takes in any graph $G = (V, E)$ of bounded degree d, and:

- If G is connected it returns "YES";
- If G is ϵ-far from connected it returns "NO" with probability at least p.

Algorithm Overview

- Step 1: Randomly sample some number of nodes
- Step 2: Do a small breadth first search to see if we are in a small, connected component of the graph that is disconnected from the rest of the graph.
- Step 3: If we detect any disconnected component, return “NO”. Otherwise, return “YES”.

Algorithm

Intuitively, ϵ-far from connected $\iff$ Many connected components that are not connected from the rest of the graph $\iff$ Each of those components is small

Connected Component Lemma

Lemma

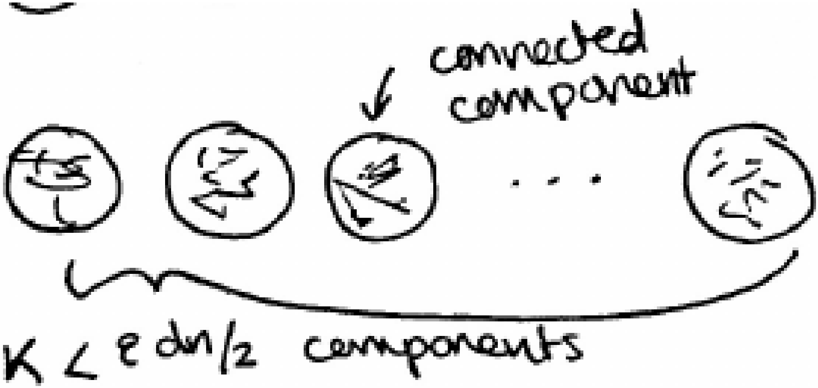
If graph $G = (V, E)$ is ϵ-far from connected, then G has at least $\epsilon dn/2$ connected components that are not connected to each other.

ε-far from connected

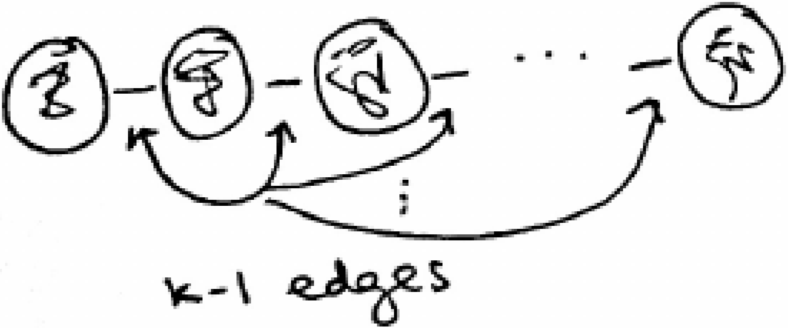
$\Rightarrow \ \geq \varepsilon dn/2$ components

Connected Component Lemma

Proof by the contrapositive. Suppose that G has less than $\epsilon dn/2$ connected components that are not connected to each other. Let k be the number of such components in G.



Then it is possible to connect these components with no more than $k-1$ edges, and therefore we can change the adjacency matrix representation of G in at most $2k < \epsilon dn$ spots and make G connected. Therefore G is less than ϵ-far from connected. $\square$



Average Number of Nodes in a Component

Lemma

If graph $G = (V, E)$ is ϵ-far from connected, then G has an average of $2/(\epsilon d)$ nodes in each of its at least $\epsilon dn/2$ components.

average # nodes $\leq \frac{2}{\epsilon d}$

Average Number of Nodes in a Component

Proof.

From the previous Lemma, G has at least $\epsilon dn/2$ components. Therefore the average number of nodes is at most

$$\frac{n}{\epsilon dn/2} = \frac{2}{\epsilon d}.$$

□

Markov's Inequality

Theorem (Markov's Inequality)

Let X be a non-negative random variable, and let $a > 0$. Then

$$P(X \geq a\mathbb{E}(X)) \leq \frac{1}{a}.$$

Number of Small Components

Lemma

If G is ϵ-far from connected it has at least $\epsilon dn/4$ connected components of size at most $4/(\epsilon d)$.

Proof.

Suppose we uniformly randomly choose a component of G, and let the random variable X be the number of nodes in this connected component. Then by applying Markov's and the previous lemma, we have that

$$P(X \geq 2\mathbb{E}[X]) \leq \frac{1}{2}.$$

Therefore at least half of G's components have the number of nodes at most $4/(\epsilon d)$. □

Probability of Finding a Small Component

Since each small connected component has at least a single element in it, this means that if we uniformly randomly sample an element then we have at least a $\epsilon d/4$ chance of being in a small component!

Witness Lemma

If a test catches a witness with probability $\geq \epsilon$, then

$$s = \frac{\ln\left(\frac{1}{1-p}\right)}{\epsilon}$$

iterations of the test catches a witness with probability $\geq p$.

Algorithm Overview

- Step 1: Randomly sample $s = 4\ln(1/(1-p))/(\epsilon d)$ number of nodes
- Step 2: Do a breadth first search of size at most $4/(\epsilon d)$ for each of the nodes.
- Step 3: If we detect any disconnected component, return "NO". Otherwise, return "YES".