sublinear algorithms

how to compute if there is not enough time even to look at each object once?

why sublinear algorithms?

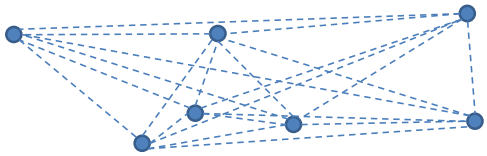
- sales logs
- data originated by scientific experiments
- satellite data and pictures
- genome project
- outbreak of diseases
- social networks
- network traffic
- click patterns
- ….

points on the Earth and distance measures

sometimes approximating is fine

approximate diameter of a point set

- a satellite is asked to compute all the distances between all the pair of points of a huge set of points on the Earth
- the satellite computes the distances and stores them into the cloud



approximate diameter of a point set

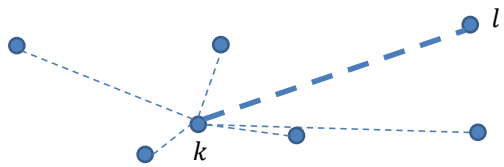
- we are interested in computing the distance between the furthest points (*diameter* of the point set)
- we do not have time for reading all the measures
- we would be happy to get an approximation of the exact result

approximate diameter of a point set

- let D_{ij} be the distance between points i and j
- maximum D_{ij} is the *diameter* of the point set

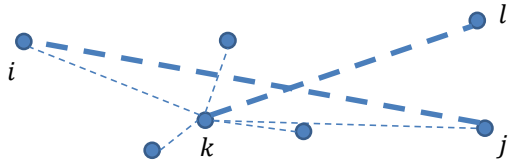
an approximate algorithm

- pick point k of the point set arbitrarily
- scan all the measures involving k and pick l to maximize D_{kl}
- output D_{kl}



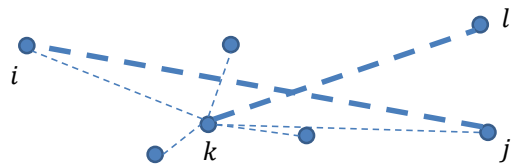
approximation guarantee

- let D_{ij} be the correct diameter
- by the triangle inequality $D_{ij} \leq D_{ik} + D_{kj}$
- because of the choice of l and because of the symmetry of the measures we have $D_{ij} \leq D_{lk} + D_{kl} \leq D_{kl} + D_{kl} \leq 2D_{kl}$



approximation guarantee

- hence our output could be wrong but it is at least half of the true diameter $D_{ij} \leq 2D_{kl}$



how many measures we check?

- if the points are n, then the measures taken by the satellite are $m = (n-1) + (n-2) + \cdots + 3 + 2 + 1 = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} = O(n^2)$
  - Gauss formula
- to get the approximate diameter we check only $n-1$ measures, i.e. $O(\sqrt{m})$ measures
  - sublinear algorithm

property testing

Q: has the input a given property?

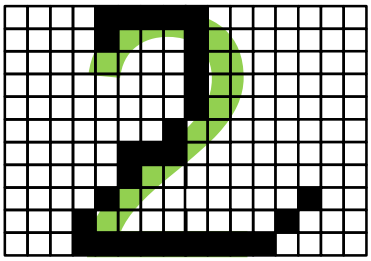
A: yes or no

property testing

- if the input has the property, then answer yes with probability $\geq \frac{2}{3}$
- if the input is ε–far from having the property, then answer no with probability $\geq \frac{2}{3}$
- if the input is less than ε–far from having the property, then don't care

ε–far means that the input does not have the property for more than a fraction ε of its *features*

test – is this number 2 or is ε-far?



property testing – is w a string of 0's?

- input: string w consisting of n numbers 0 or 1
- question: is $w = 0000 \cdots 0$?

requires reading all the numbers (linear time)

- approximate version: is $w = 0000 \cdots 0$ or does it have $\geq \varepsilon n$ 1's (is ε–far from being a string of 0's)

property testing – is w a string of 0's?

- example:
  - we can select $\varepsilon = 0.1$
  - in this case ε–far means that more than 10% of the digits of w are 1's; i.e., suppose $n = 1000$, we have that w contains at least 100 1's

property testing – is w a string of 0's?

randomized algorithm

- sample $s = {}^{2}/_{\varepsilon}$ positions uniformly and independently at random
- if at least one 1 is found in the sample then answer no, otherwise answer yes

- example: if $\varepsilon = 0.1$ we have $s = {}^{2}/_{0.1} = 20$

property testing – is w a string of 0's?

analysis of the algorithm

- if $w = 0000 \cdots 0$ then algorithm answers yes
- suppose w contains exactly εn 1's, then the probability to find a 1 in a sample is $\frac{\varepsilon n}{n} = \varepsilon$
- hence, the probability to find a 0 in a sample is $1 - \varepsilon$
- if w contains more than εn 1's, then the probability to find a a 0 in a sample is $\leq 1 - \varepsilon$

property testing – is w a string of 0's?

analysis of the algorithm

- hence, the probability that algorithm answers yes even if w is ε–far from being a string of 0's, that is the probability of finding a 0 for each sample, is $\leq (1 - \varepsilon)^s$
- exploiting that $1 - x \leq e^{-x}$ we have

  $(1 - \varepsilon)^s \leq e^{-\varepsilon s} = e^{-\varepsilon \frac{2}{\varepsilon}} = e^{-2} = \frac{1}{e^2} < \frac{1}{3}$
- hence, the probability that algorithm answers no if w is ε–far from being a string of 0's is $> \frac{2}{3}$

witness lemma

- if a sample finds a witness with probability $\geq p$, then $s = \frac{p}{2}$ iterations of the test find a witness with probability $\geq \frac{2}{3}$

sublinear algorithms – conclusions

- sublinear time possible in many contexts
  - new area, many interesting techniques
- pervasive applicability
- algorithms are usually simple, analysis is much more complex