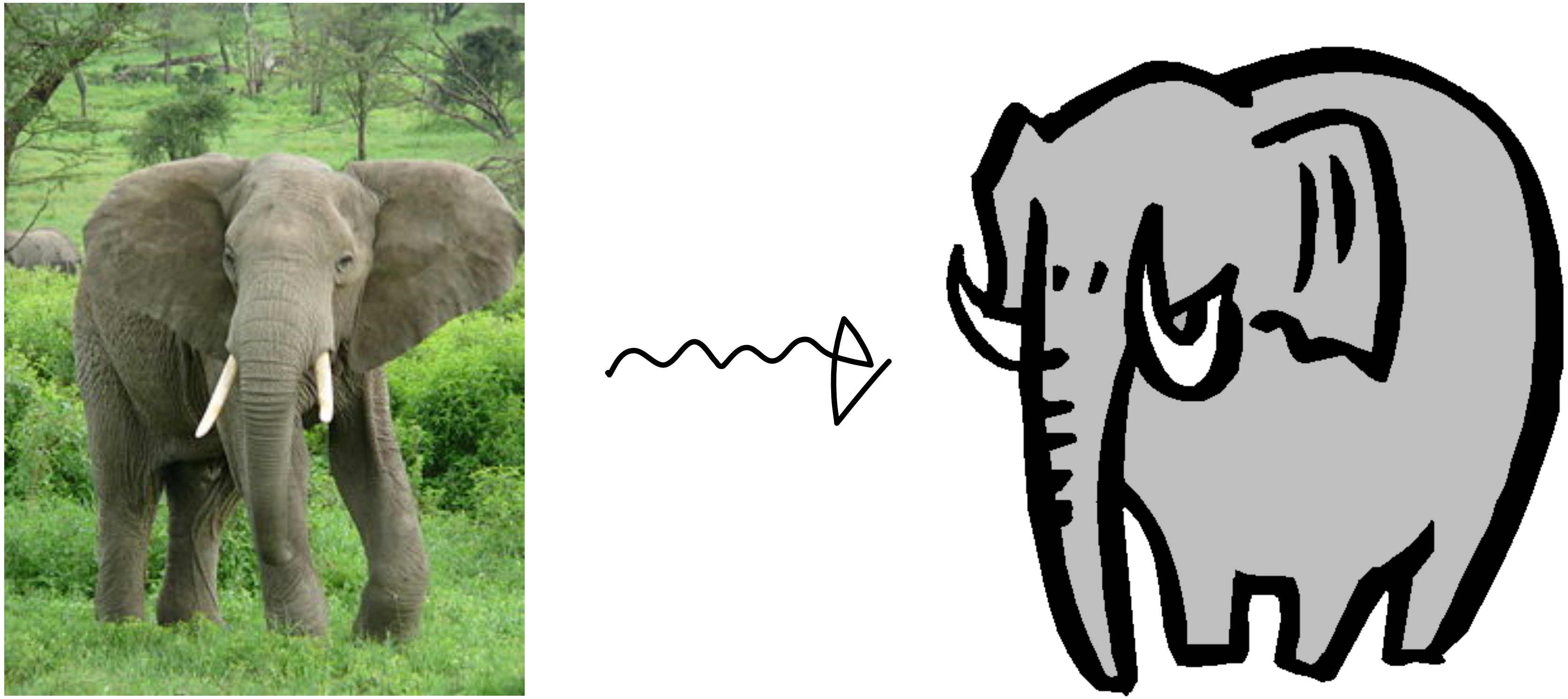
Low Distortion Embeddings Into The Line

Fedor Fomin, Daniel Lokshtanov, Saket Saurabh.

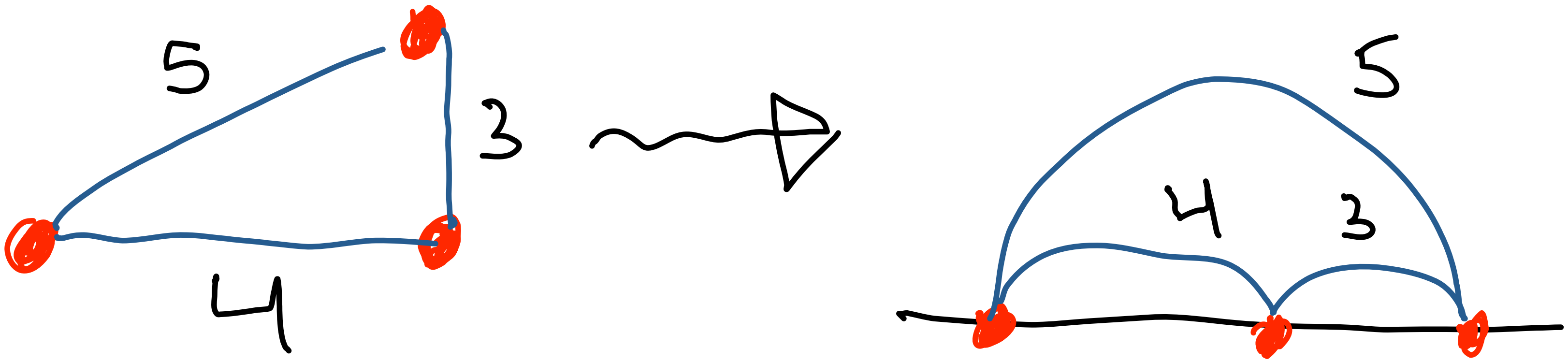
WG 2009

Intuition and Motivation

- Have a complicated object that is hard to work with.Want to approximate our object as good as possible with some “simple sketch”



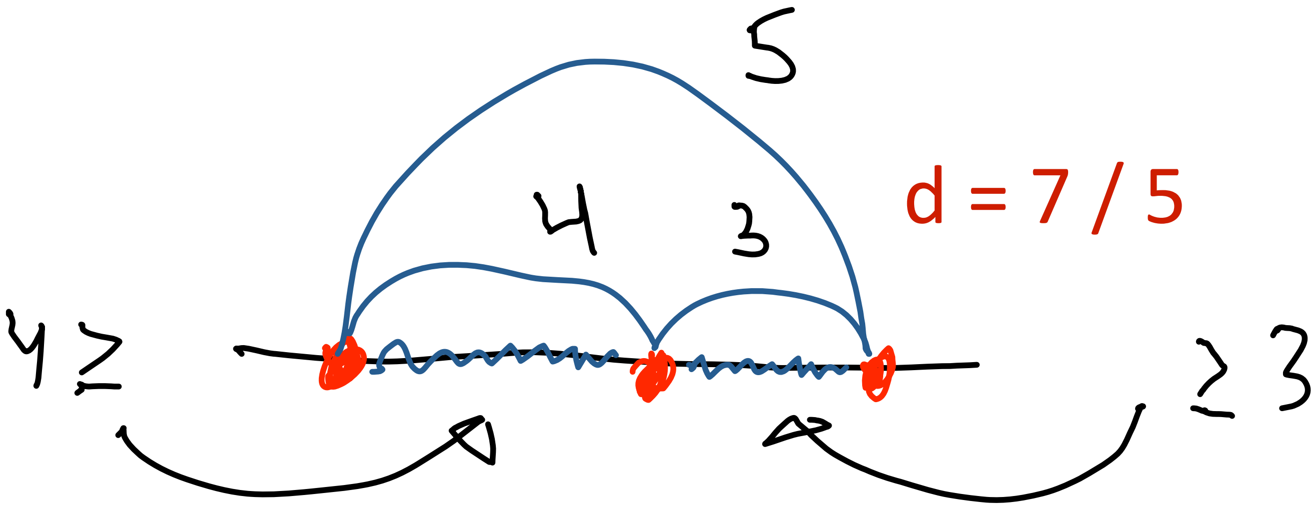
Low Distortion Embeddings



- A *metric* M is a universe V together with a distance function $D : V * V \rightarrow N$.
- An embedding of M into the the line is a function $f : V \rightarrow N$.

Problem definition

- An embedding is non-contracting if
  $D(u,v) \leq |f(u) - f(v)|$
- The *distortion* d of a non-contracting embedding is $\max[|f(u) - f(v)| / D(u,v)]$.



Problem Considered

- Low distortion means the original metric is well approximated by the new metric.
- We consider the shortest path distance metrics of unweighted graphs.

In: Graph G and integer d

Q: Is there a non-contracting embedding of G into the real line with distortion at most d?

Previous results

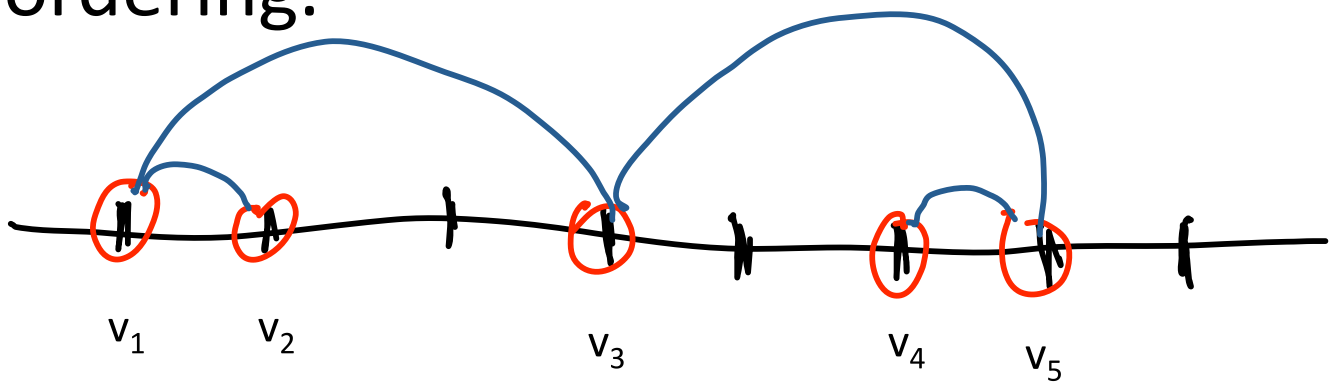
- Hard to approximate within a constant factor even for metrics generated by trees.
- Several approximation results (with approximation rations like $n^{1/2}$ and $n^{1/3}$ for special cases like trees.
- Exact algorithm with running time $O(n^{4d})$.

Our results

- Exact algorithm with running time $5^{n+o(n)}$.

An Observation

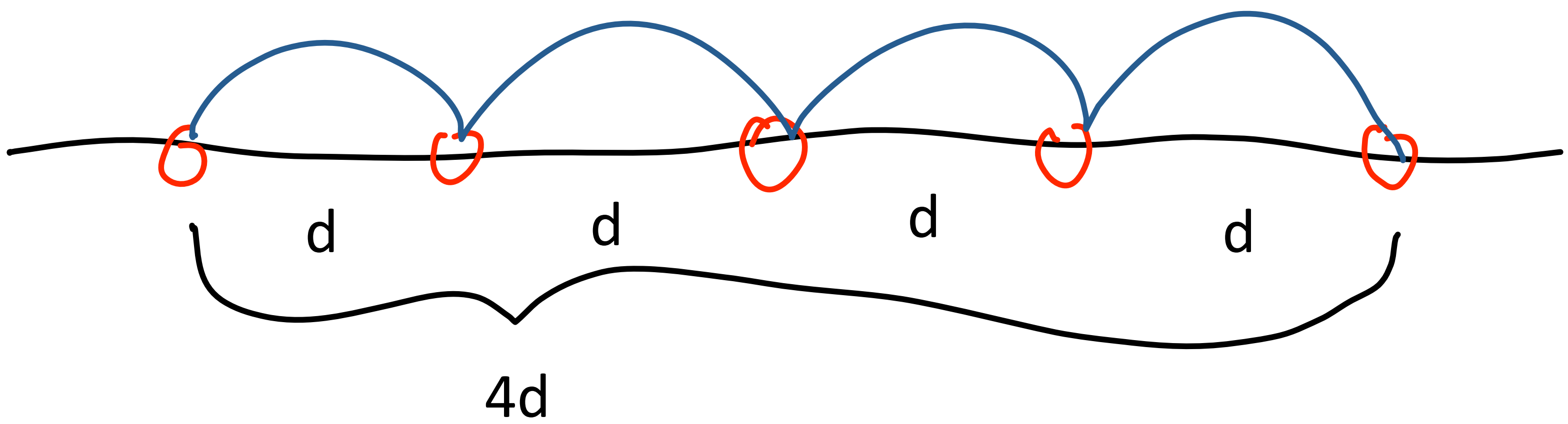
- **Observation:** Let f be an embedding of $G=(V,E)$ into the line that orders V into $v_1 \dots v_n$ from left to right. Then f is non-contracting if and only if it does not contract any consecutive pair in this ordering.



$$f(v_4) - f(v_3) \geq D(v_3, v_4)$$
$$f(v_3) - f(v_2) \geq D(v_2, v_3)$$
$$\rightsquigarrow \quad f(v_4)-f(v_2) \geq D(v_2, v_4)$$

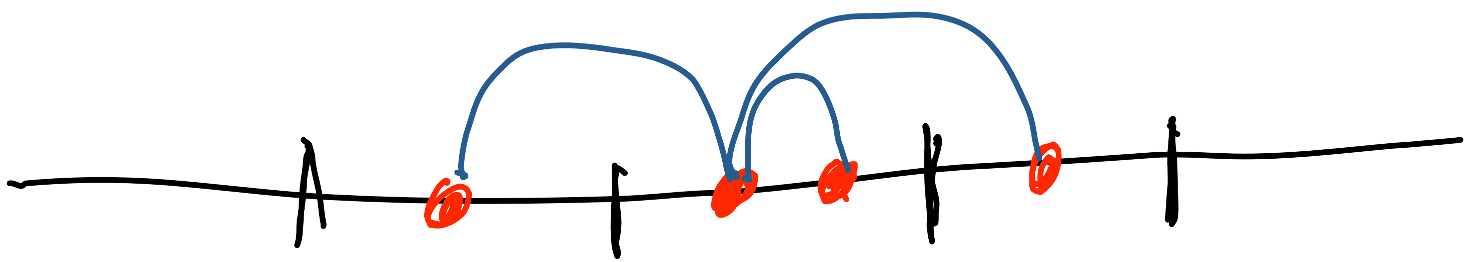
Another Observation

- The distortion of f is at most d if and only if f streches every edge by at most d.



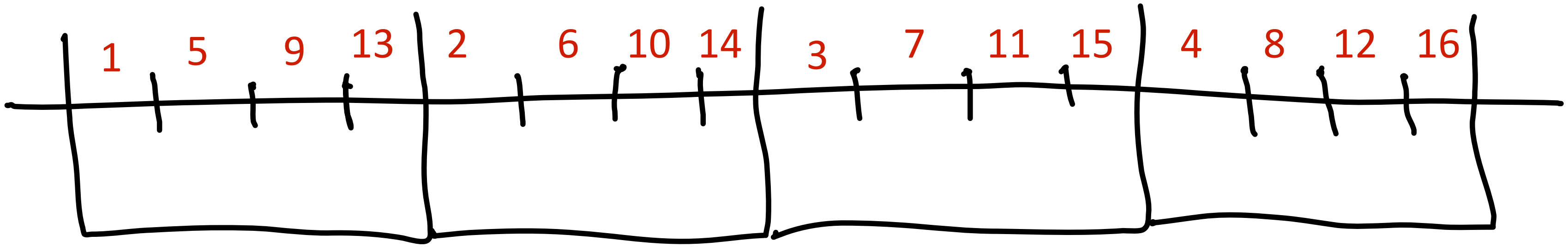
Algorithm, part I

- Divide the line into subintervals, “buckets”, of length d. Notice that neighbours of the graph must go to neighbouring buckets.
- Branch on all possible ways to distribute vertices to buckets. There are 3^n possible ways.



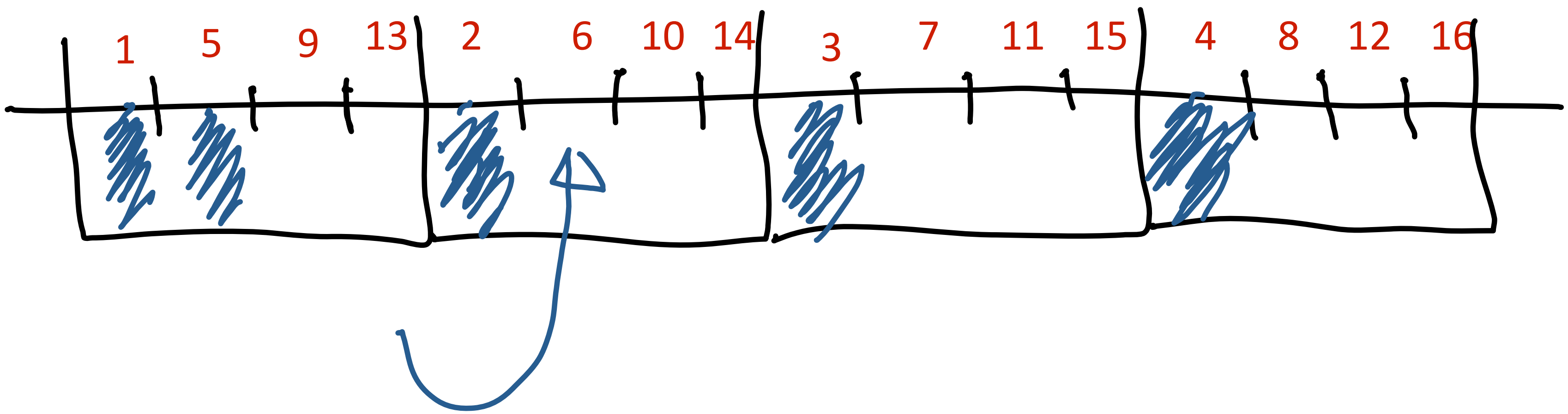
Algorithm, Part II

- Given a distribution into buckets we want to find a good embedding that has the same distribution.
- For b buckets we do it in $n^{6b}2^n$ time.
- Order the positions as shown in the figure:



Algorithm, Part II

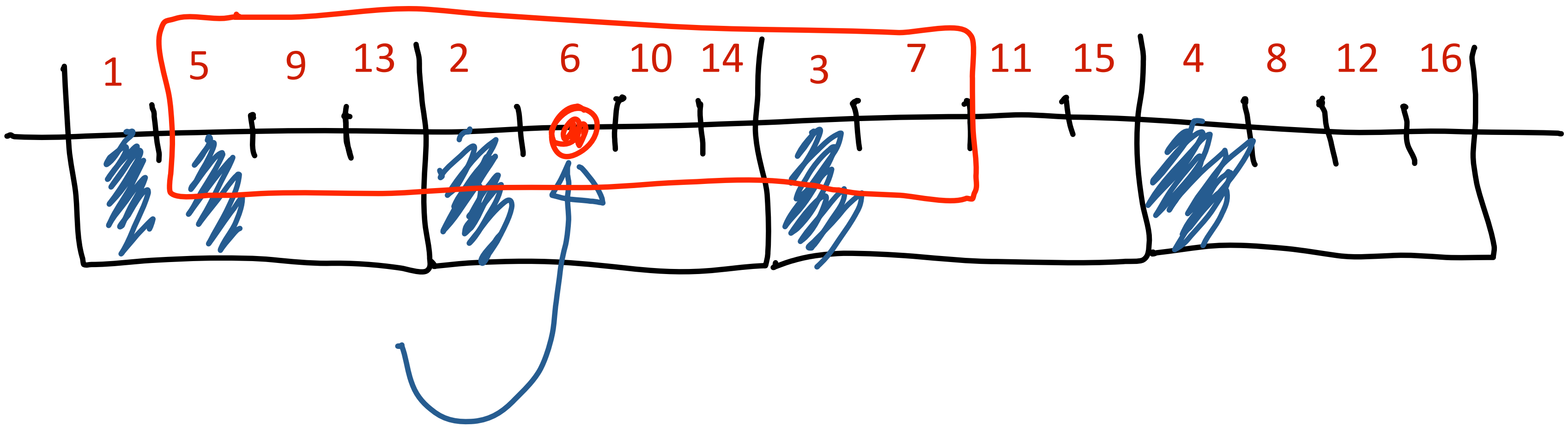
- We first decide which vertex (if any) goes into the first position, then the second position etc.



When a vertex is placed, we need to make sure that no edge is streched more than d and that no pair is contracted.

Algorithm, Part II

- No long stretch: Let A be the set of already picked, and let a_L be the vertex at position 5 (if any). Knowing A and a_L is sufficient to decide which vertices can be placed at position 6 without stretching an edge more than d.

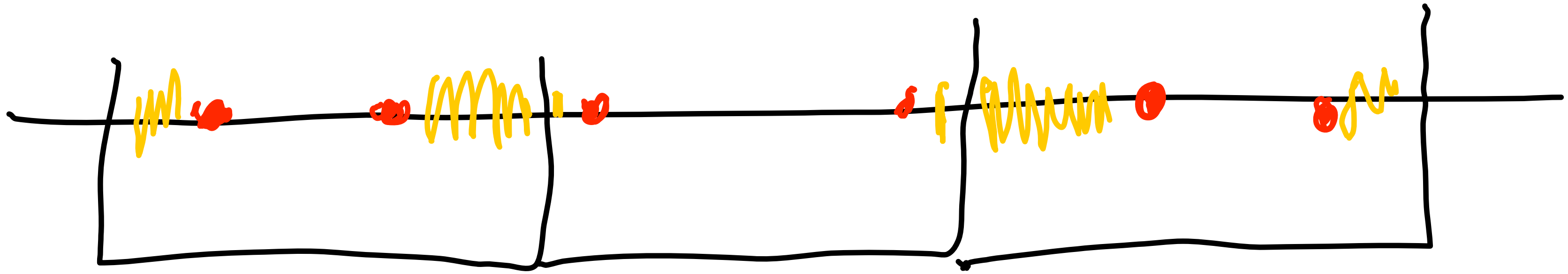


Algorithm, Part II

- We can now do Dynamic Programming, keeping track of the set A of vertices already placed, and which vertex a_L was placed at the last position.
- But, but… hey! Did we forget about non-contraction?

Algorithm, Part II

- To enforce non-contraction guess the first and the last vertex in every bucket and the positions of the first and last vertex in each bucket.
- Make sure the DP is consistent with your guesses.
- Total running time: $2^n n^{6b}$.

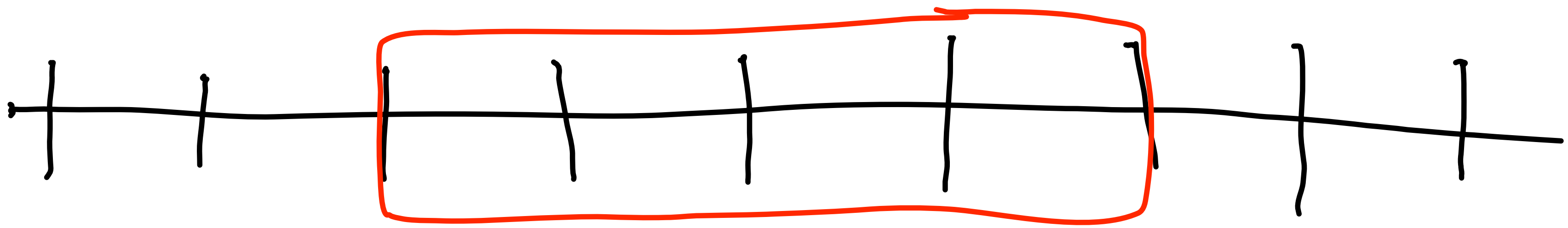


Algorithm, Part III

- Combining Part I and Part II yields an algorithm with running time $6^n n^{6b}$. The number of buckets can be n, so this is still not c^n.
- Need an extra trick to get the number of buckets to be sufficiently small.

Algorithm, Part III

- If the number of buckets, $b \geq n / \log^2 n$, then the average number of vertices in a bucket is $\log^2 n$.
- Take the "middle half" of the buckets. The average number of vertices in one of these buckets is at most $2\log^2 n$.



Algorithm, Part III

- At least one of the buckets contains at most the average number, $\log^2 n$ of vertices. Branching on the postion of these vertices splits the left and the right part into two independent subproblems.
- A recurrence bounding branching:

$$T(n,b) < d^{2\log(n)\text{^}2} * 2T(n, 3b/4)$$

$$< (2d^{2\log(n)\text{^}2})^{2\log(\log(n))}\, T(n, b / \log^2(n))$$

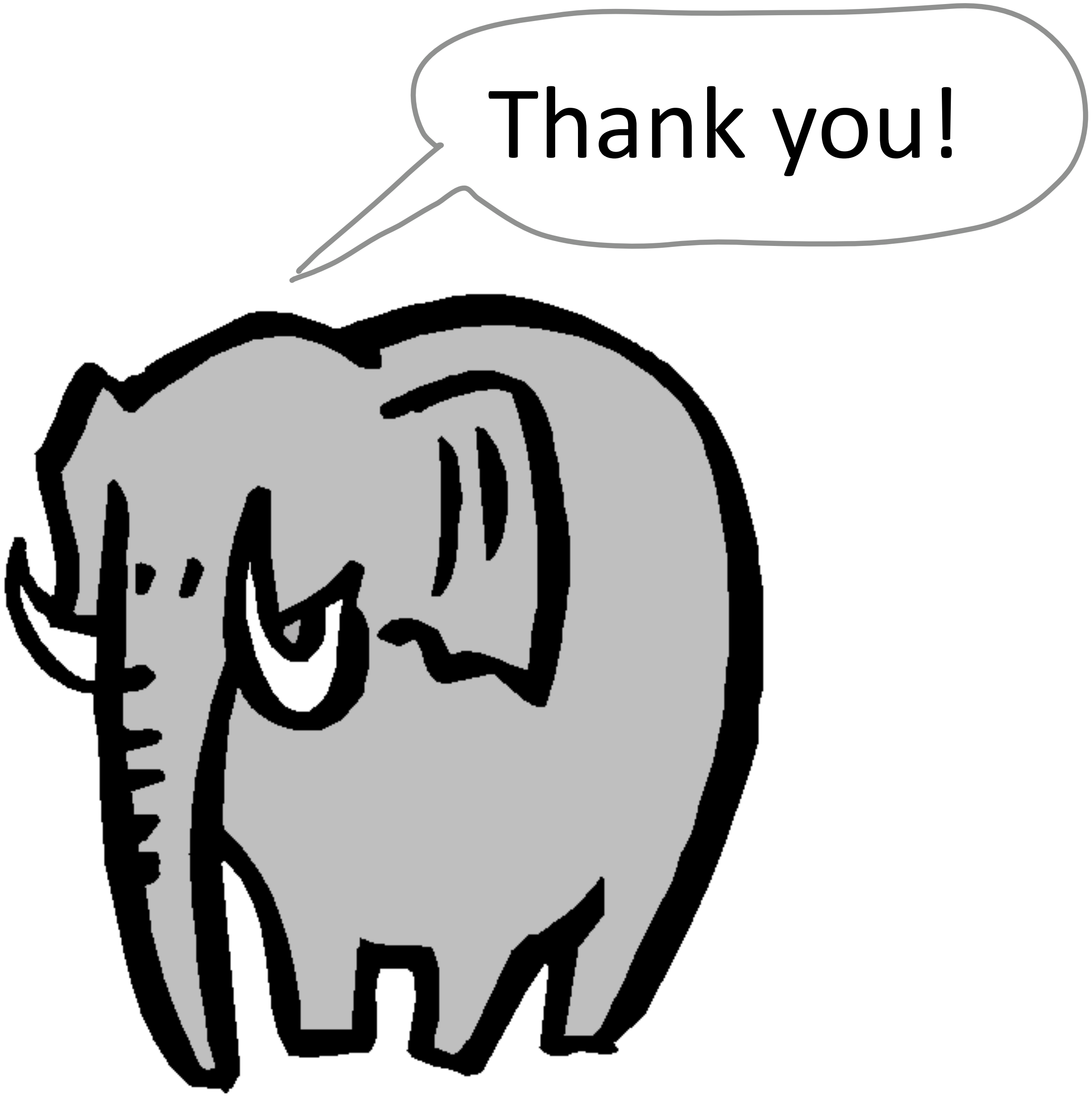
$$< 2^{o(n)}T(n, n/\log^2(n))$$

Tally up The Time

- Part I contributes a factor 3^n to the running time.
- Part III reduces the number of buckets to $n / \log^2 n$ at a cost of a factor $2^{o(n)}$ to the running time.
- Part II solves each subproblem in time $2^n n^{n / (\log(n))^2} = 2^{n+o(n)}$
- Total running time becomes $6^{n+o(n)}$.
- With some work this can be improved to a $5^{n+o(n)}$ algorithm..

Open Problem

- For general finite metrics with integer distances, nothing better than an $n!$ algorithm is known. Can a c^n algorithm be obtained for this more general version?


Thank you!