Filtering: A Method for Solving Graph Problems in MapReduce

Benjamin Moseley
UIUC

Silvio Lattanzi
Google

Siddharth Suri
Yahoo!

Sergei Vassilvitski
Yahoo!

Large Scale Distributed Computation, Jan 2012

Overview

- Introduction to MapReduce model
- Our settings
- Our results
- Open questions

Introduction to the MapReduce model

XXL Data

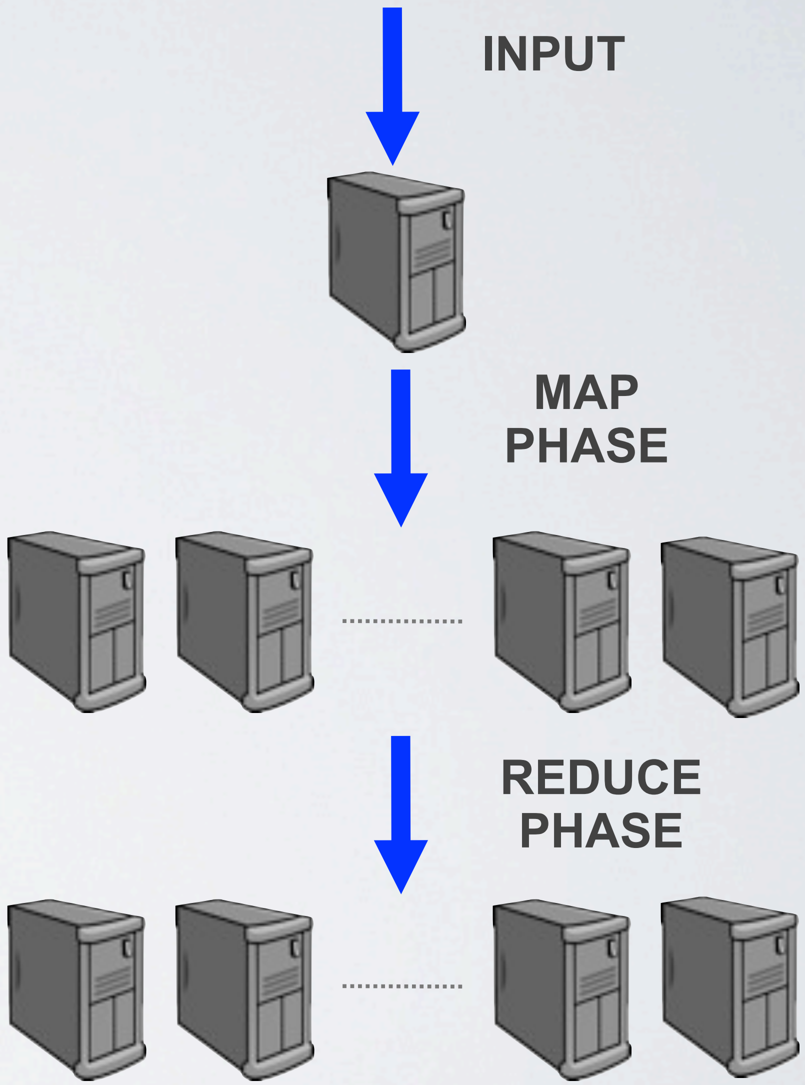
- Huge amount of data
- Main problem is to analyze information quickly

XXL Data

- Huge amount of data
- Main problem is to analyze information quickly
- New tools
- Suitable efficient algorithms

MapReduce

- MapReduce is the platform of choice for processing massive data

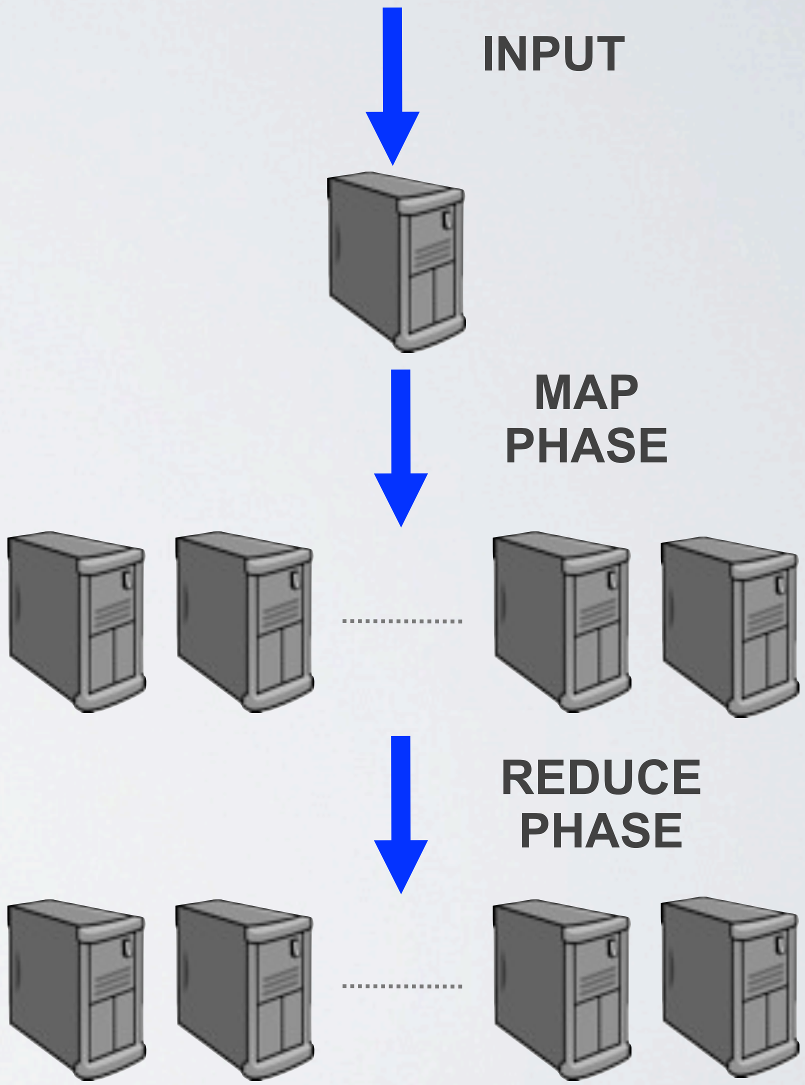


MapReduce

- MapReduce is the platform of choice for processing massive data

- Data are represented as tuple

  < *key, value* >



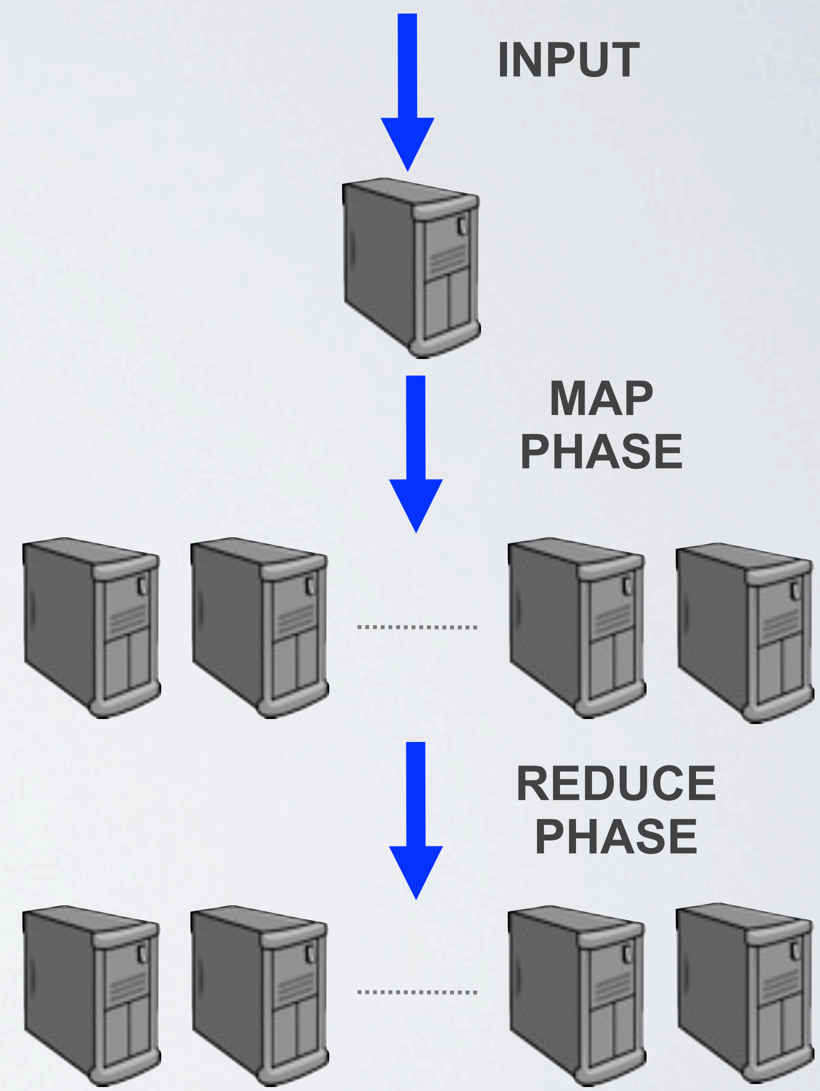
MapReduce

- MapReduce is the platform of choice for processing massive data

- Data are represented as tuple

  < *key, value* >

- Mapper decides how data is distributed

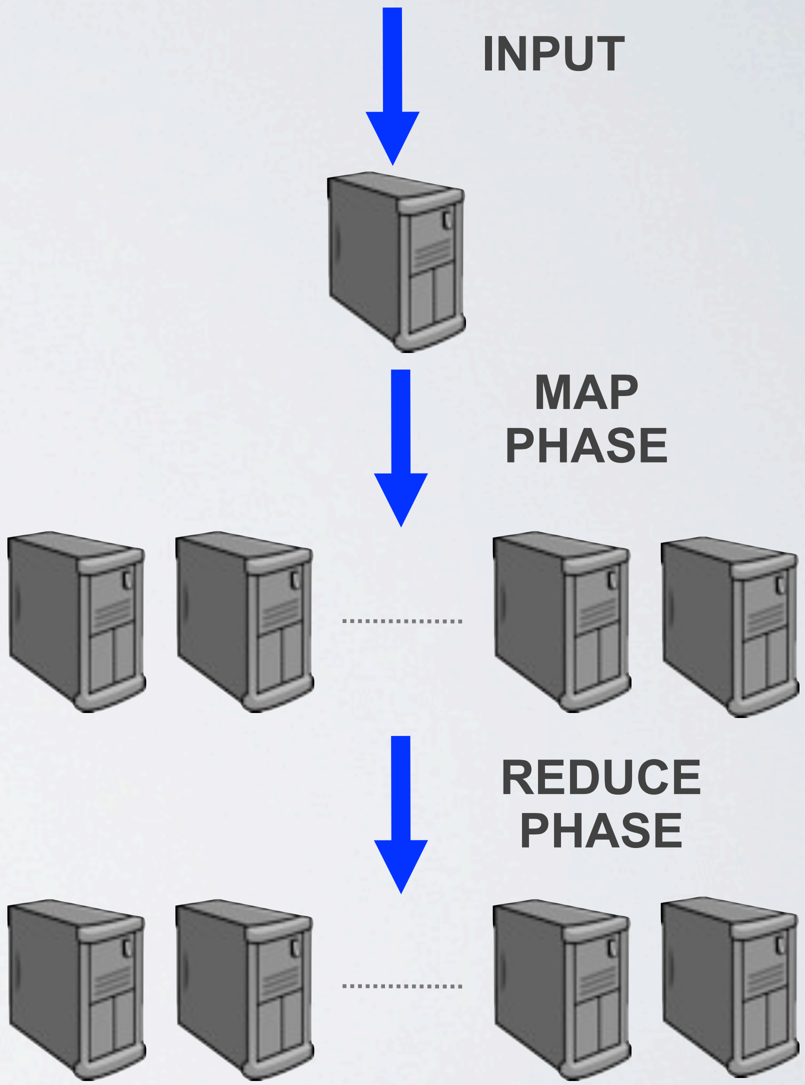


MapReduce

- MapReduce is the platform of choice for processing massive data
- Data are represented as tuple

  < *key, value* >

- Mapper decides how data is distributed
- Reducer performs non-trivial computation locally



How can we model MapReduce?

PRAM

- No limit on the number of processors
- Memory is uniformly accessible from any processor
- No limit on the memory available

How can we model MapReduce?

STREAMING

- Just one processor
- There is a limited amount of memory
- No parallelization

MapReduce model

[Karloff, Suri and Vassilvitskii]

- N is the input size and $\epsilon > 0$ is some fixed constant

MapReduce model

[Karloff, Suri and Vassilvitskii]

- N is the input size and $\epsilon > 0$ is some fixed constant
- Less than $N^{1-\epsilon}$ machines

MapReduce model

[Karloff, Suri and Vassilvitskii]

- N is the input size and $\epsilon > 0$ is some fixed constant
- Less than $N^{1-\epsilon}$ machines
- Less than $N^{1-\epsilon}$ memory on each machine

MapReduce model

[Karloff, Suri and Vassilvitskii]

- N is the input size and $\epsilon > 0$ is some fixed constant
- Less than $N^{1-\epsilon}$ machines
- Less than $N^{1-\epsilon}$ memory on each machine

$\mathcal{MRC}^i$: problem that can be solved in $O(\log^i N)$ rounds

Combining map and reduce phase

- Mapper and Reducer work only on a subgraph

Combining map and reduce phase

- Mapper and Reducer work only on a subgraph
- Keep time in each round polynomial

Combining map and reduce phase

- Mapper and Reducer work only on a subgraph
- Keep time in each round polynomial
- Time is constrained by the number of rounds

Algorithmic challenges

- No machine can see the entire input

Algorithmic challenges

- No machine can see the entire input
- No communication between machines during each phase

Algorithmic challenges

- No machine can see the entire input

- No communication between machines during each phase

- Total memory is $N^{2-2\epsilon}$

MapReduce vs MUD algorithms

- In MUD framework each reducer operates on a stream of data.
- In MUD, each reducer is restricted to only using polylogarithmic space.

Our settings

Our settings

- We study the Karloff, Suri and Vassilvitskii model

Our settings

- We study the Karloff, Suri and Vassilvitskii model

- We focus on class $\mathcal{MRC}^0$

Our settings

- We assume to work with dense graph $m = n^{1+c}$, for some constant $c > 0$

Our settings

- We assume to work with dense graph $m = n^{1+c}$, for some constant $c > 0$

- Empirical evidences that social networks are dense graphs
[Leskovec, Kleinberg and Faloutsos]

Dense graph motivation

[Leskovec, Kleinberg and Faloutsos]

- They study 9 different social networks

Dense graph motivation

[Leskovec, Kleinberg and Faloutsos]

- They study 9 different social networks
- They show that several graphs from different domains have n^{1+c} edges

Dense graph motivation

[Leskovec, Kleinberg and Faloutsos]

- They study 9 different social networks
- They show that several graphs from different domains have n^{1+c} edges
- Lowest value of c founded $.08$ and four graphs have $c > .5$

Our results

Results

Constant rounds algorithms for

- Maximal matching
- Minimum cut
- 8-approx for maximum weighted matching
- 2-approx for vertex cover
- 3/2-approx for edge cover

Notation

- $G = (V, E)$: Input graph
- n: number of nodes
- m : number of edges
- η: memory available on each machine
- N : input size

Filtering

- Part of the input is dropped or filtered on the first stage in parallel

Filtering

- Part of the input is dropped or filtered on the first stage in parallel
- Next some computation is performed on the filtered input

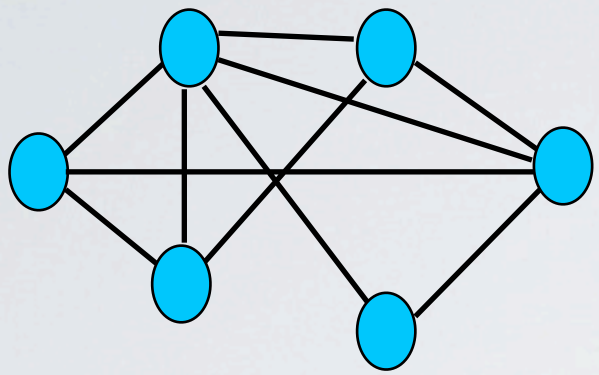
Filtering

- Part of the input is dropped or filtered on the first stage in parallel

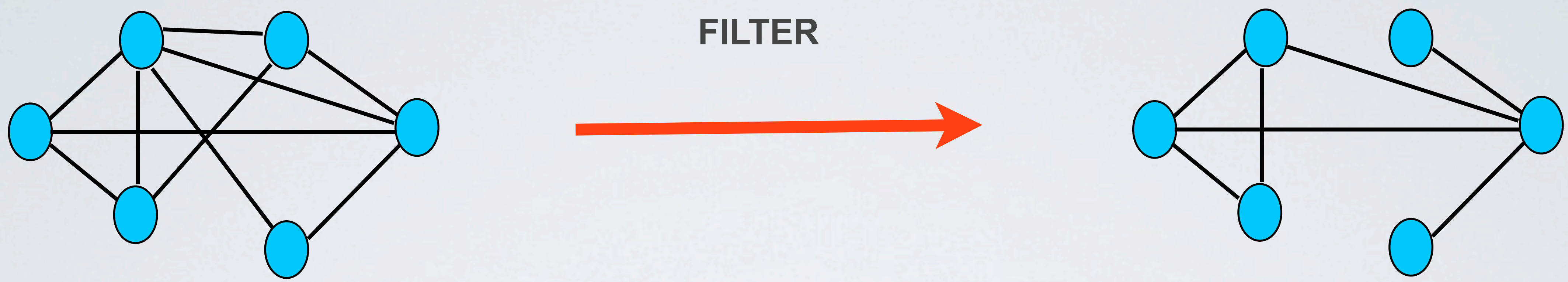
- Next some computation is performed on the filtered input

- Finally some patchwork is done to ensure a proper solution

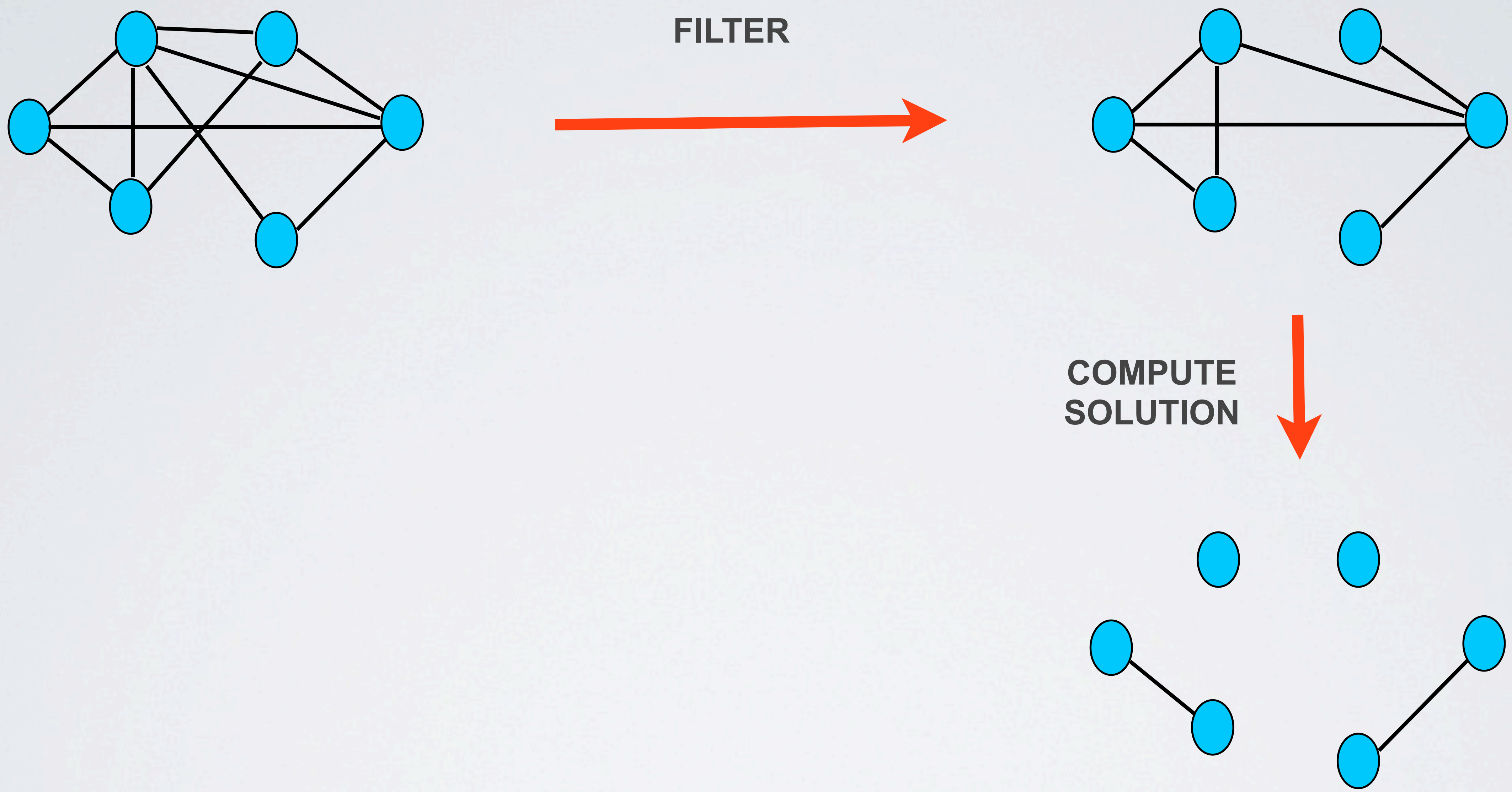
Filtering



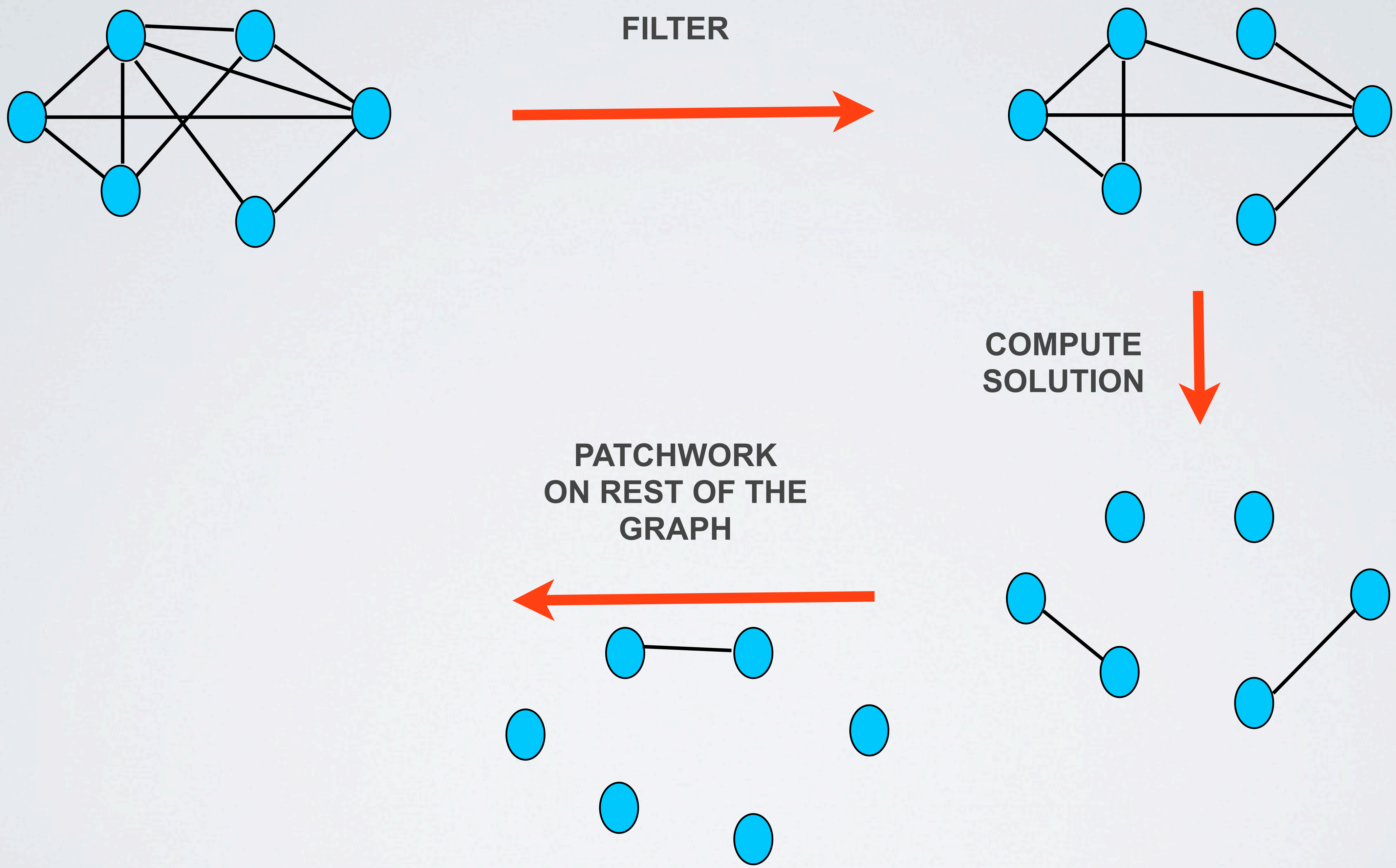
Filtering



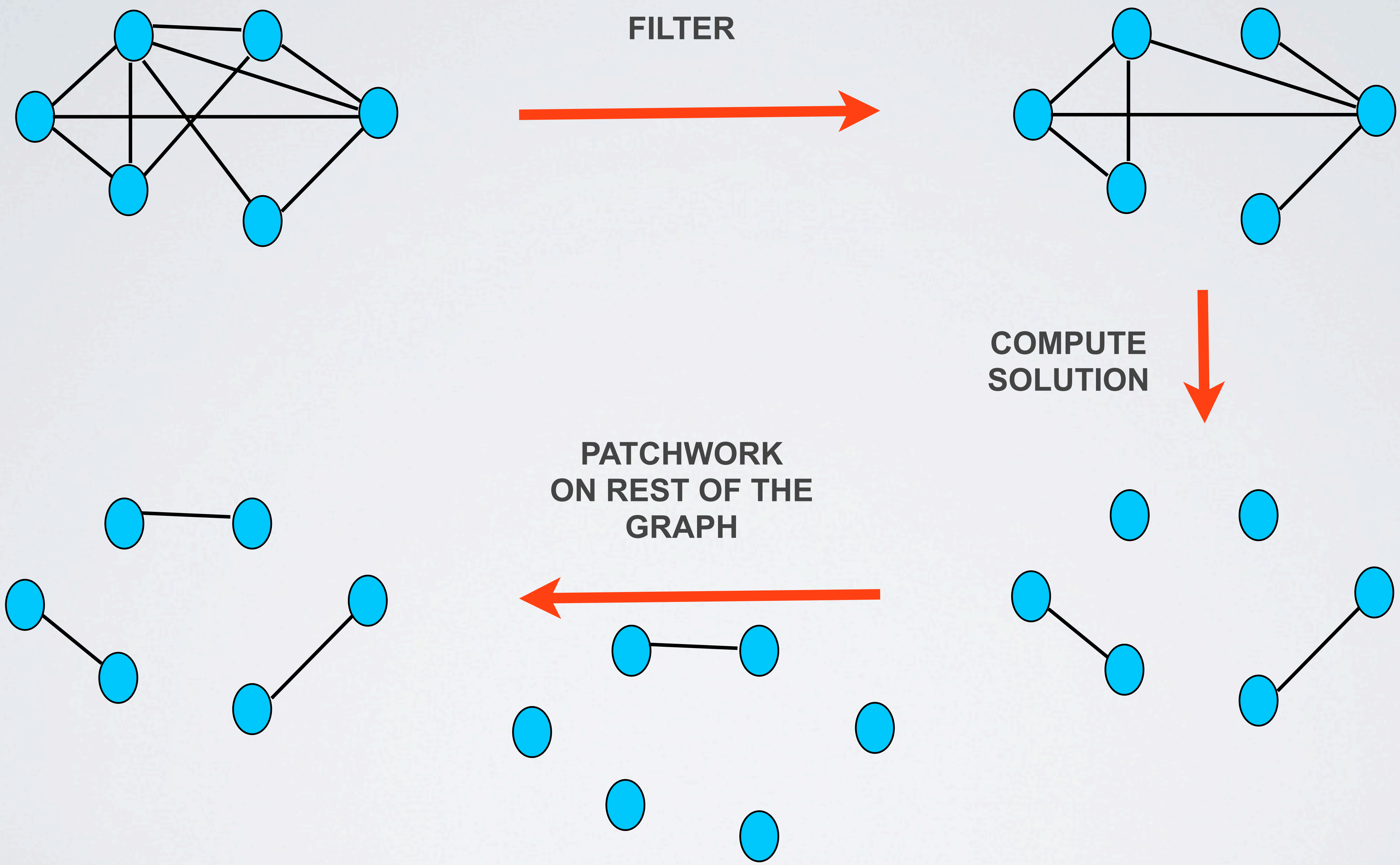
Filtering



Filtering

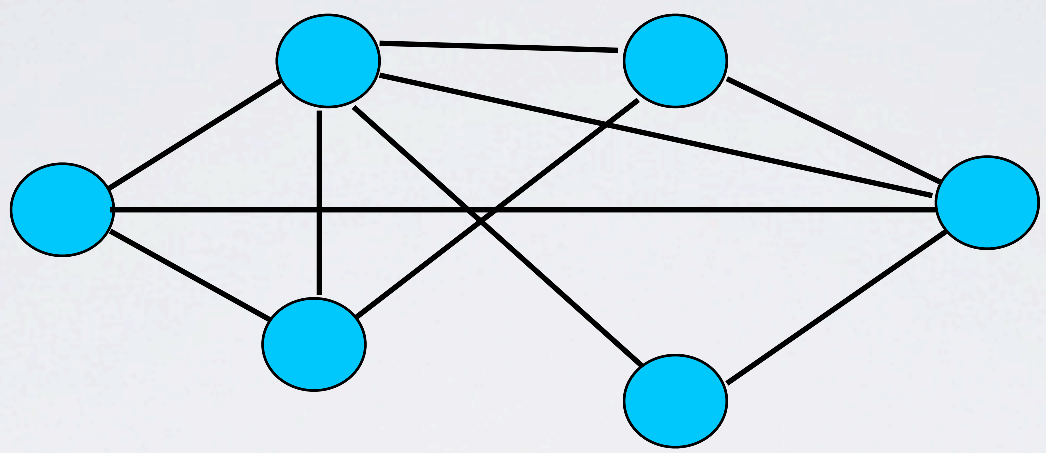


Filtering



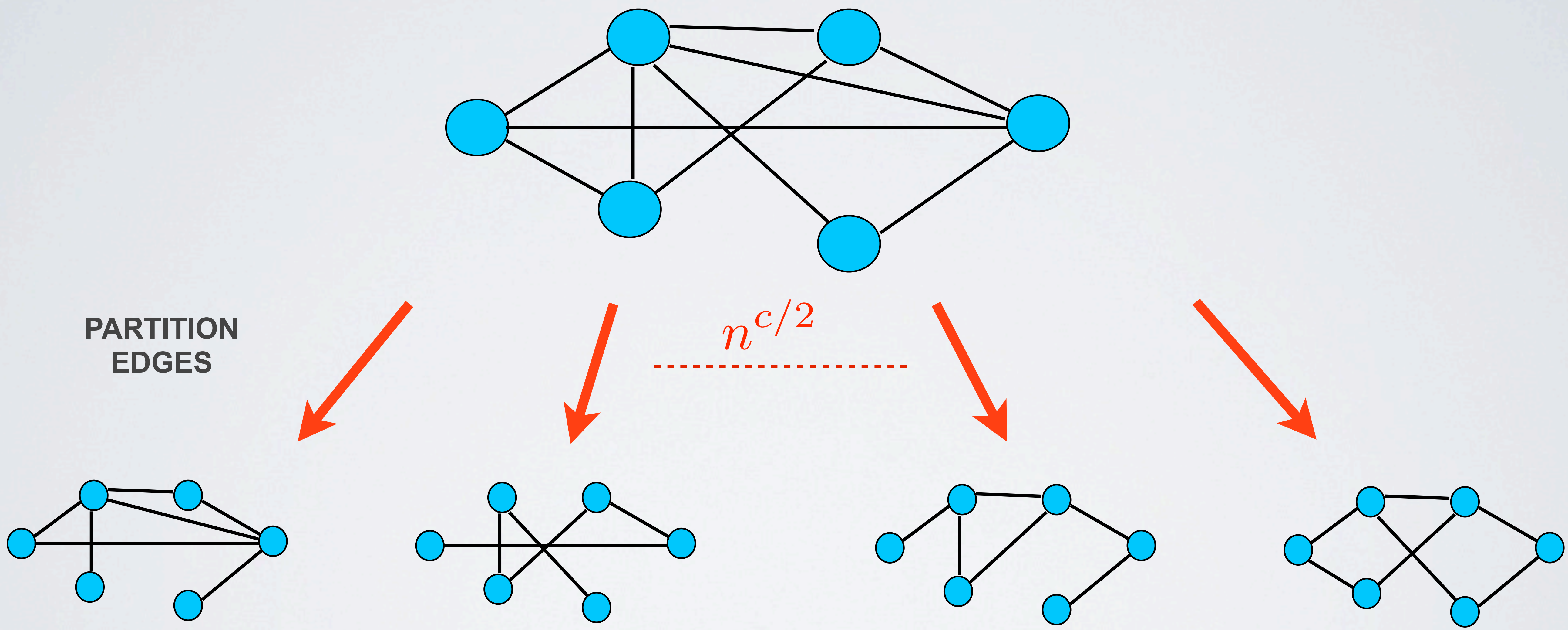
Warm-up: Compute the minimum spanning tree

$m = n^{1+c}$



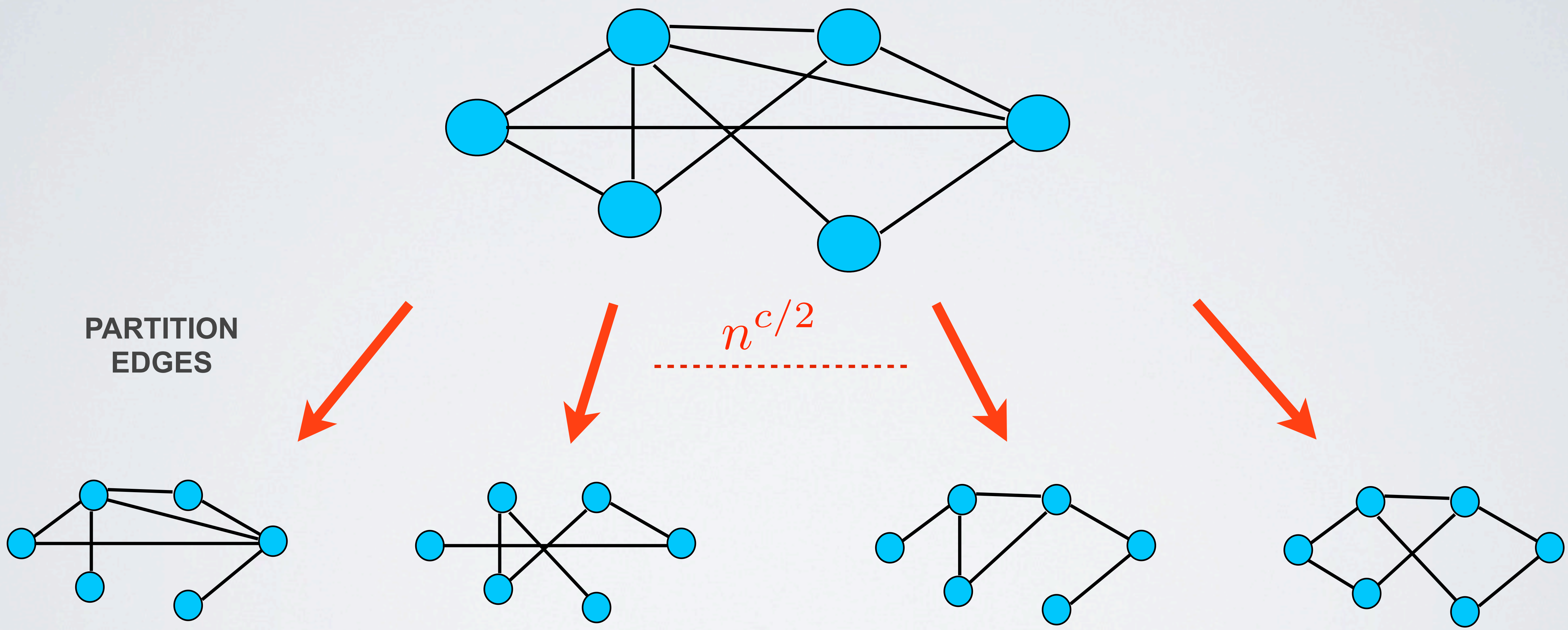
Warm-up: Compute the minimum spanning tree

$$m = n^{1+c}$$



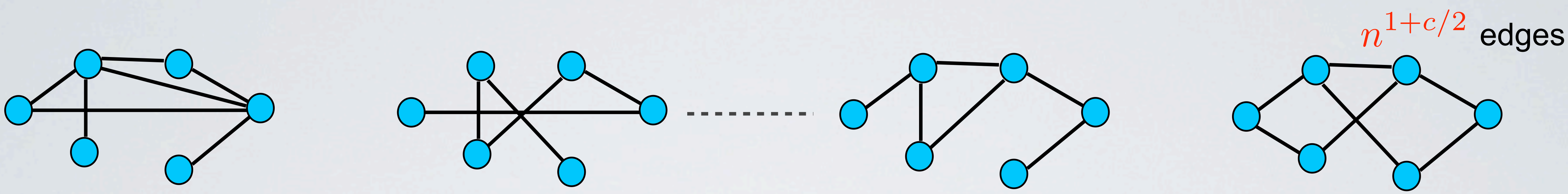
Warm-up: Compute the minimum spanning tree

$$m = n^{1+c}$$

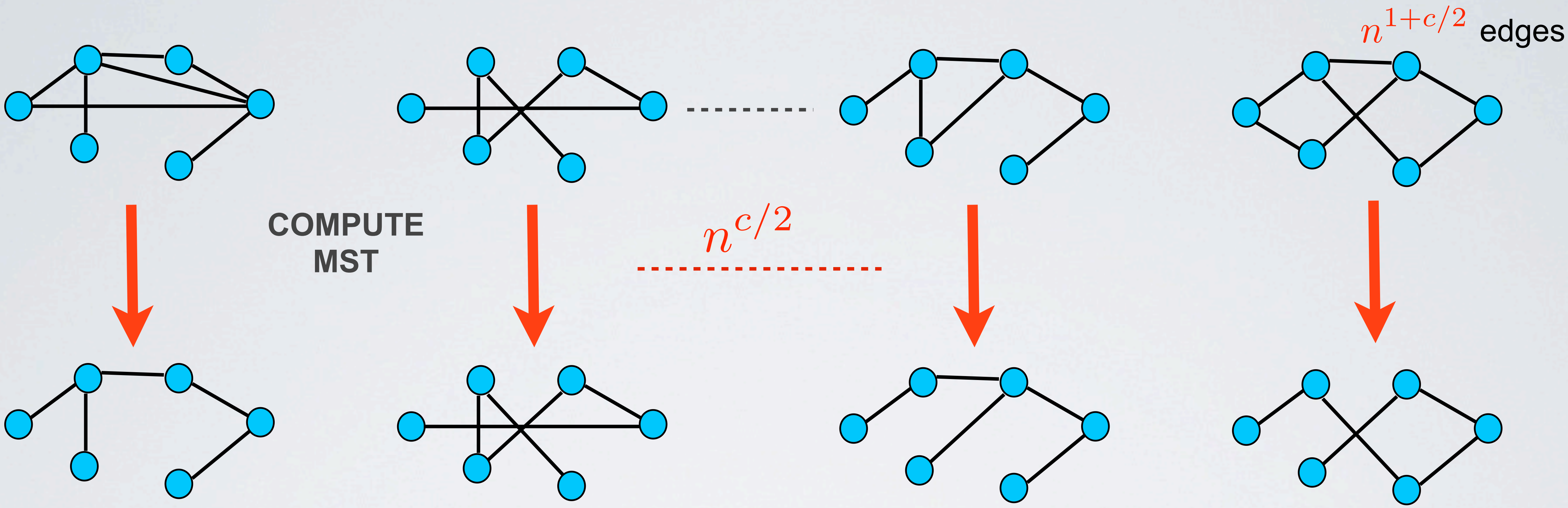


$n^{1+c/2}$ edges

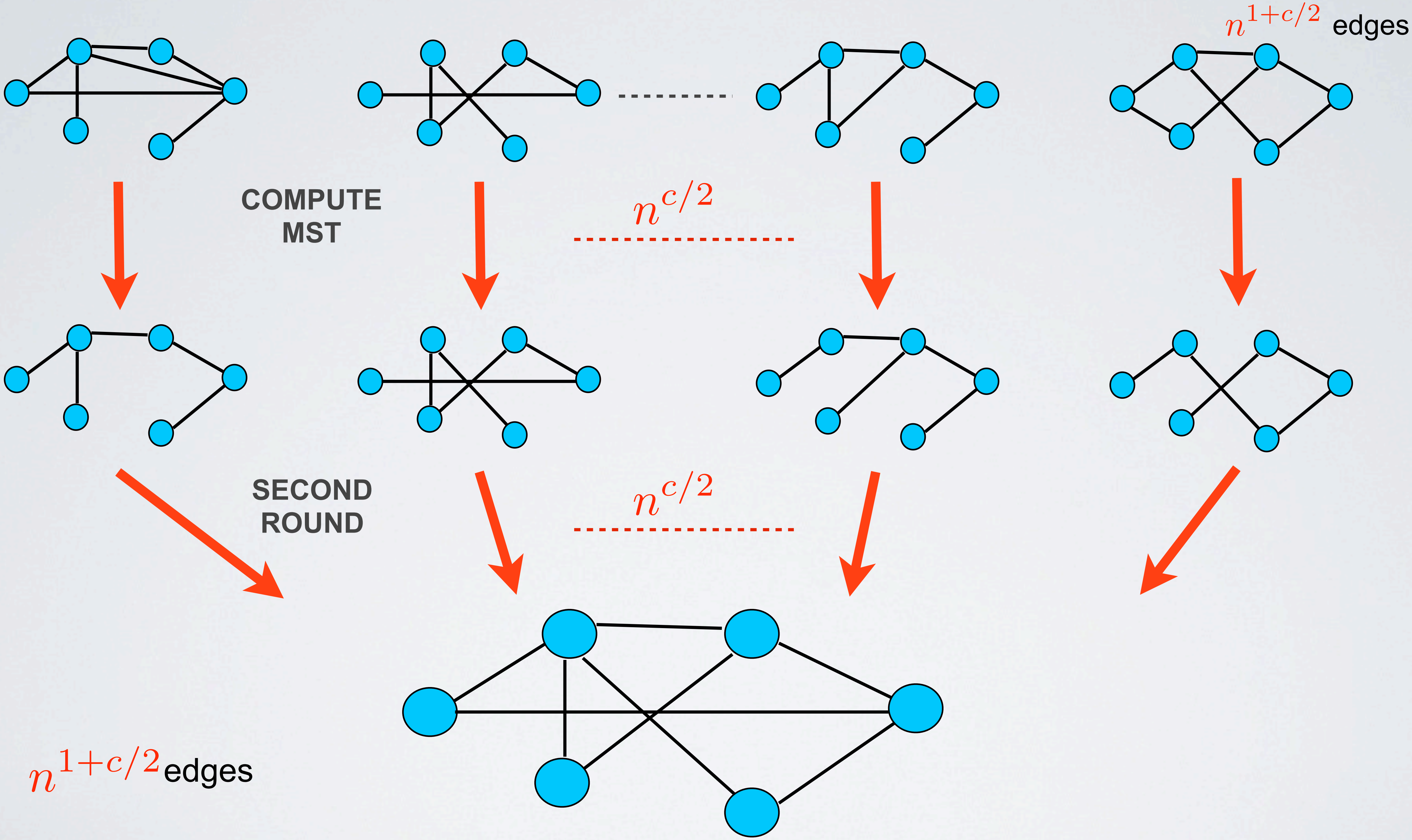
Warm-up: Compute the minimum spanning tree



Warm-up: Compute the minimum spanning tree



Warm-up: Compute the minimum spanning tree



Show that the algorithm is in MRC^0

- The algorithm is correct

Show that the algorithm is in MRC^0

- The algorithm is correct
- No edge in the final solution is discarded in partial solution

Show that the algorithm is in MRC^0

- The algorithm is correct
- No edge in the final solution is discarded in partial solution
- The algorithm runs in two rounds

Show that the algorithm is in MRC^0

- The algorithm is correct
- No edge in the final solution is discarded in partial solution
- The algorithm runs in two rounds
- No more than $O\left(n^{\frac{c}{2}}\right)$ machines are used

Show that the algorithm is in MRC^0

- The algorithm is correct
- No edge in the final solution is discarded in partial solution
- The algorithm runs in two rounds
- No more than $O\left(n^{\frac{c}{2}}\right)$ machines are used
- No machine uses memory greater than $O(n^{1+c/2})$

Maximal matching

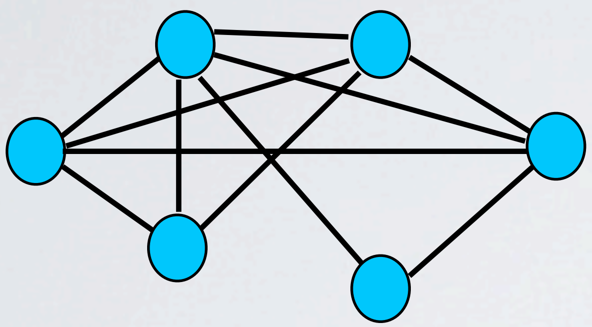
- Algorithmic difficulty is that each machine can only see edges assigned to the machine

Maximal matching

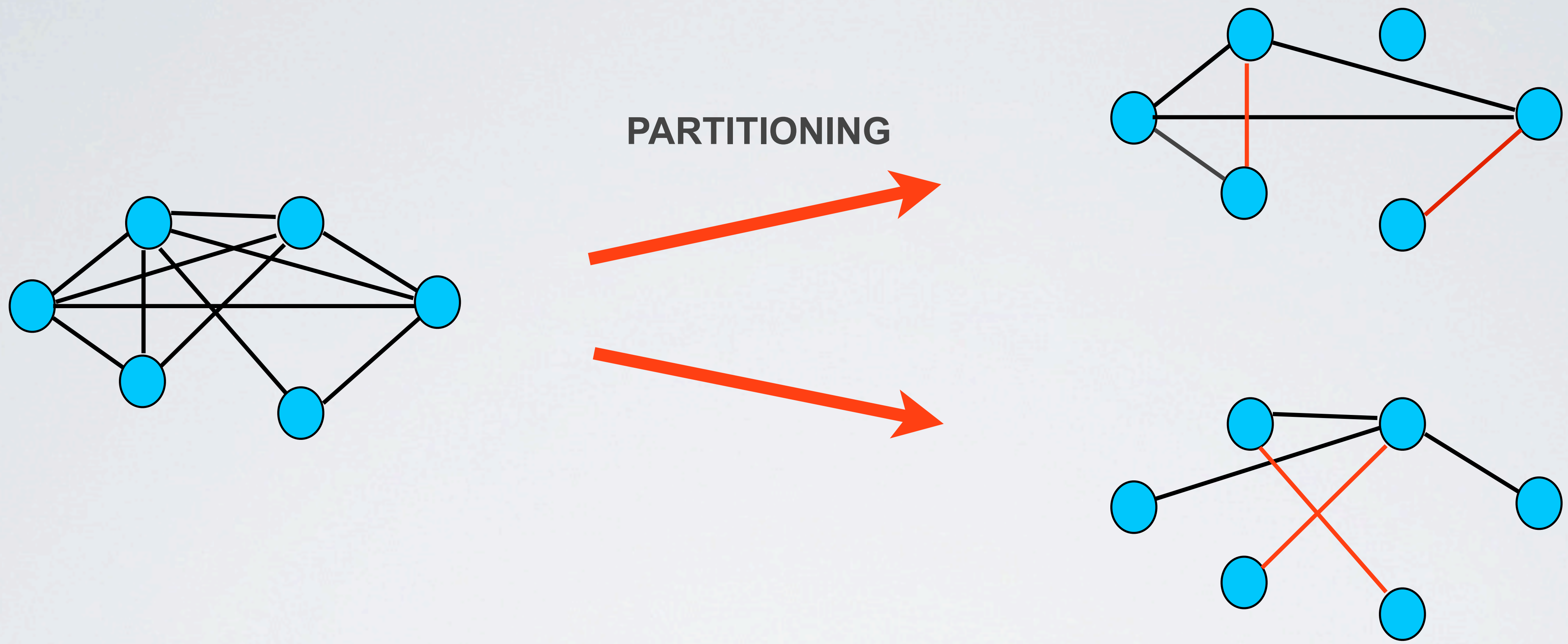
- Algorithmic difficulty is that each machine can only see edges assigned to the machine

- Is a partitioning based algorithm feasible?

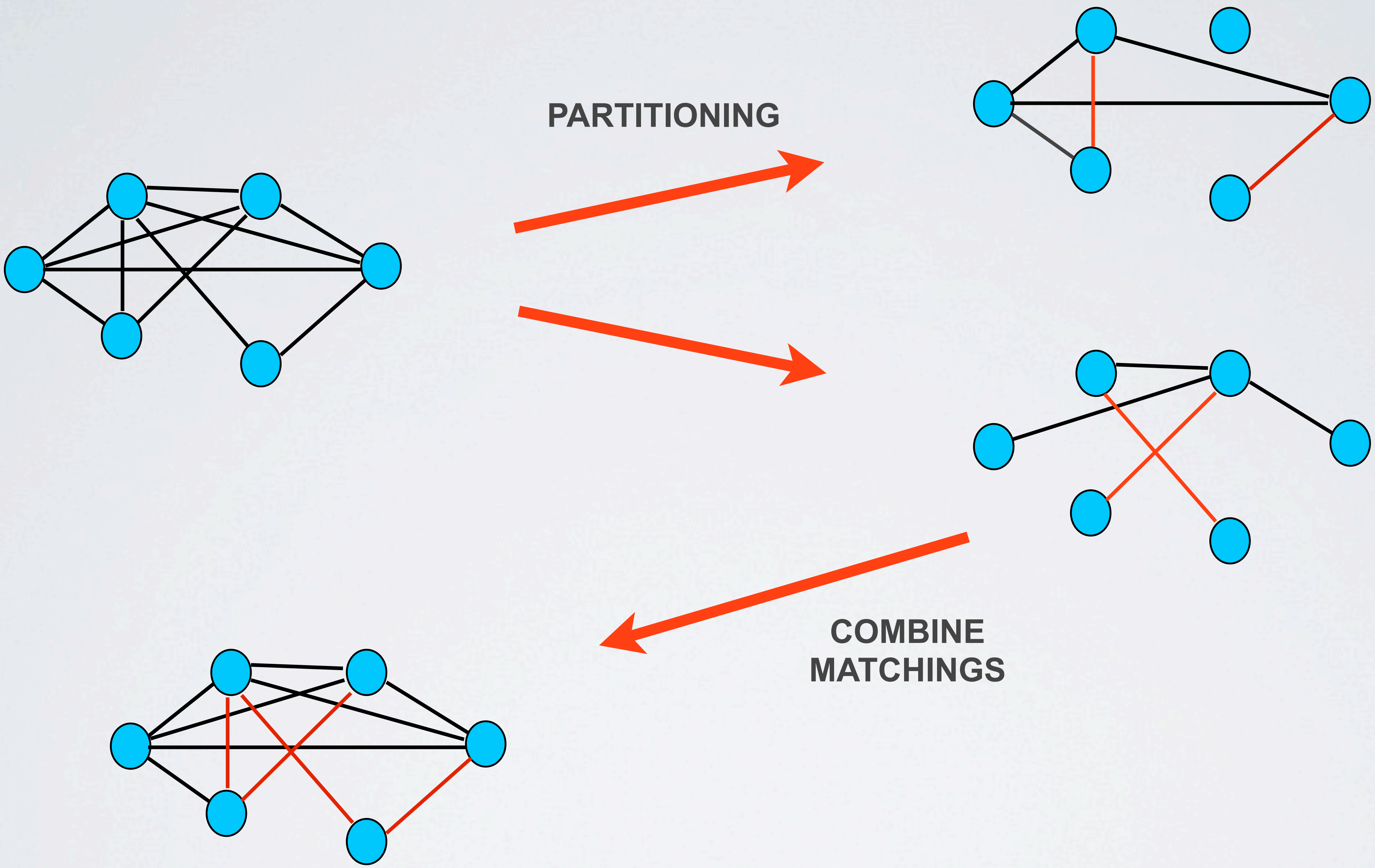
Partitioning algorithm



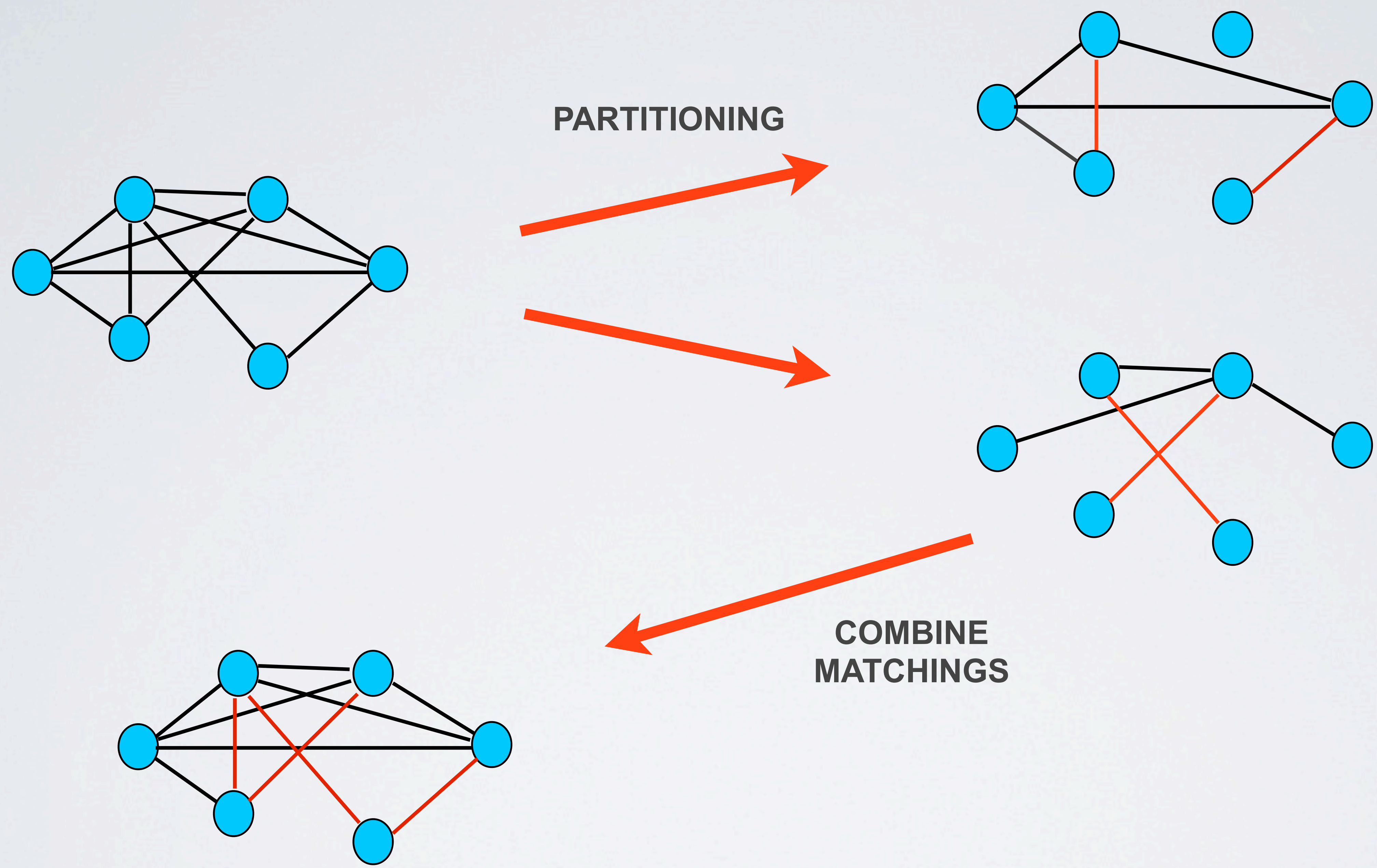
Partitioning algorithm



Partitioning algorithm



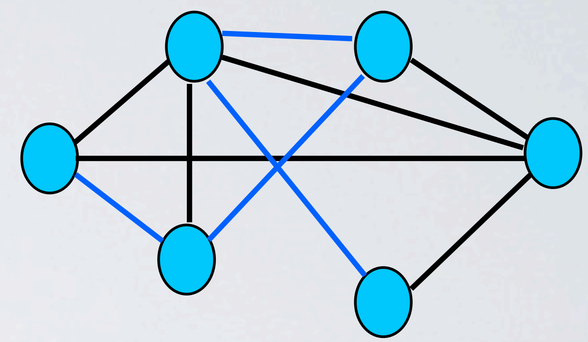
Partitioning algorithm



It is impossible to build a maximal matching using only red edges!!

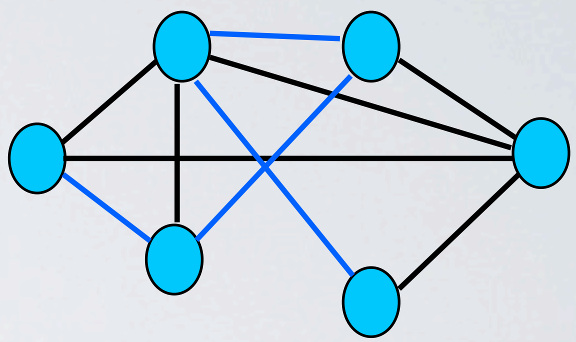
Algorithmic insight

- Consider any subset of the edges E'

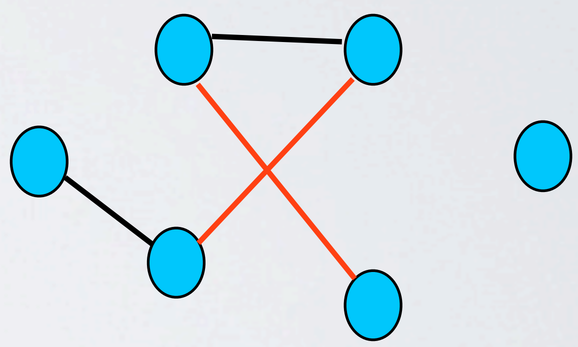


Algorithmic insight

- Consider any subset of the edges E'

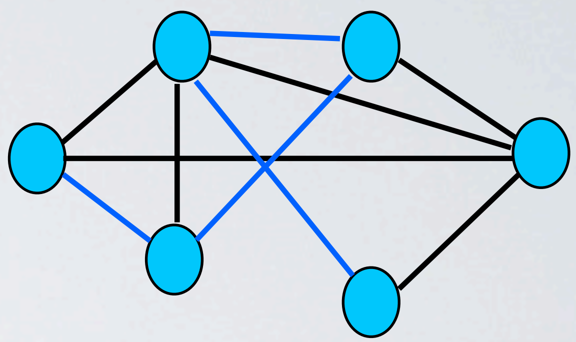


- Let M' be a maximal matching on $G[E']$

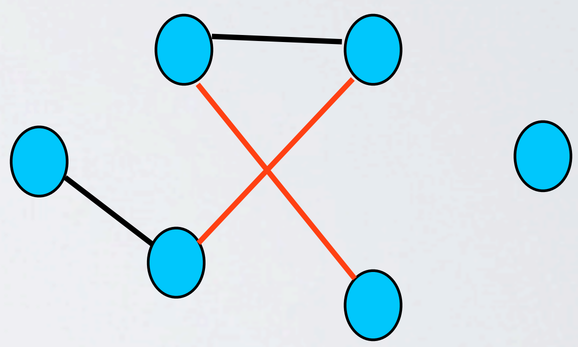


Algorithmic insight

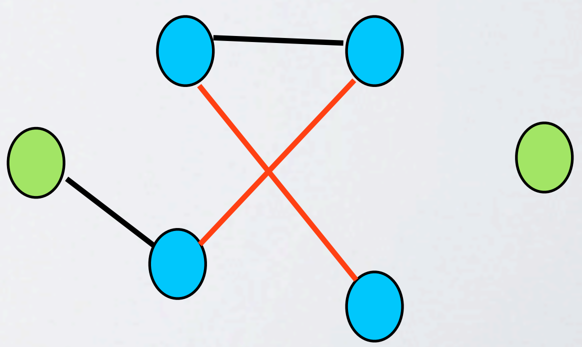
- Consider any subset of the edges E'



- Let M' be a maximal matching on $G[E']$



- The unmatched vertices form a *independent set*



Algorithmic insight

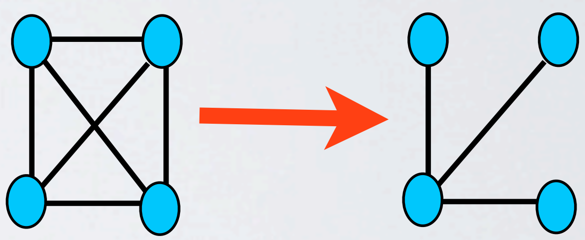
- We pick each edge with probability p

Algorithmic insight

- We pick each edge with probability p
- Find a matching on a sample and then find a matching on unmatched vertices

Algorithmic insight

- We pick each edge with probability p
- Find a matching on a sample and then find a matching on unmatched vertices
- For dense portions of the graph, many edges should be sampled

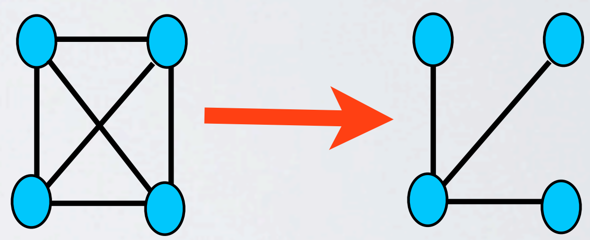


Algorithmic insight

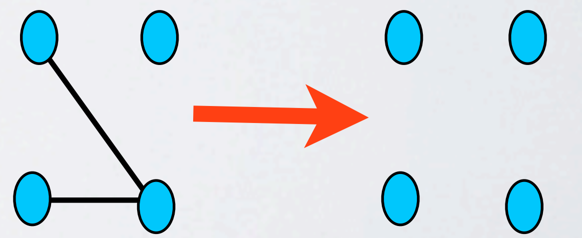
- We pick each edge with probability p

- Find a matching on a sample and then find a matching on unmatched vertices

- For dense portions of the graph, many edges should be sampled



- Sparse portions of the graph are small and can be placed on a single machine



Algorithm

- Sample each edge independently with probability

$$p = \frac{10\log n}{n^{c/2}}$$

Algorithm

- Sample each edge independently with probability

$$p = \frac{10 \log n}{n^{c/2}}$$

- Find a matching on the sample

Algorithm

- Sample each edge independently with probability

$$p = \frac{10 \log n}{n^{c/2}}$$

- Find a matching on the sample

- Consider the induced subgraph on unmatched vertices

Algorithm

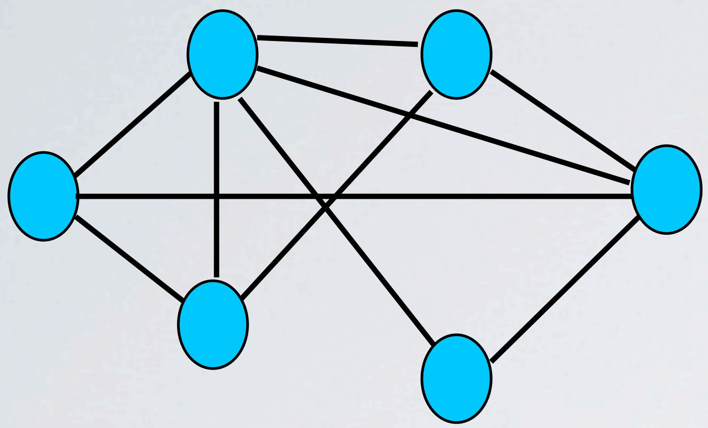
- Sample each edge independently with probability
$$p = \frac{10 \log n}{n^{c/2}}$$

- Find a matching on the sample

- Consider the induced subgraph on unmatched vertices

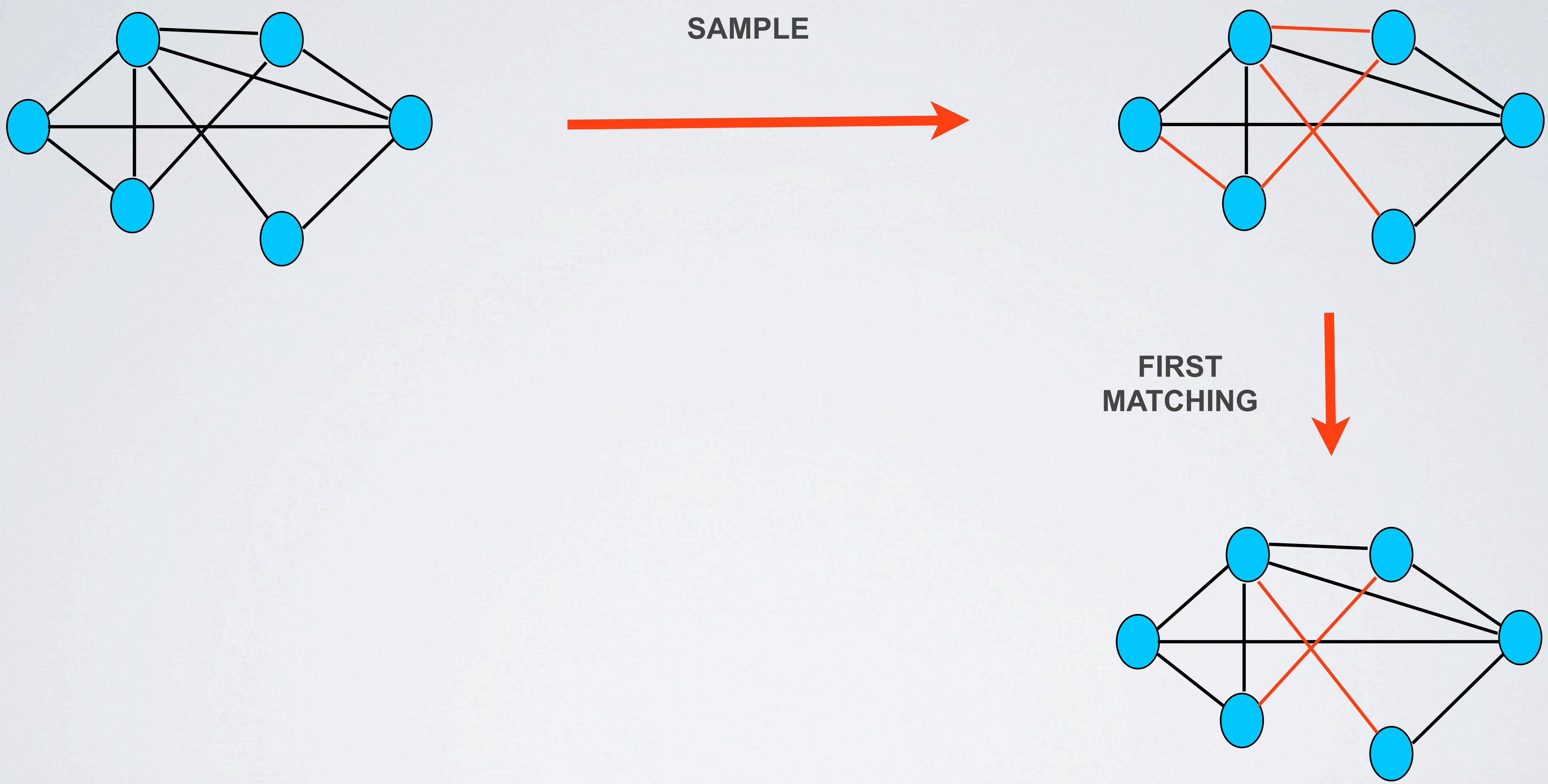
- Find a matching on this graph and output the union

Algorithm

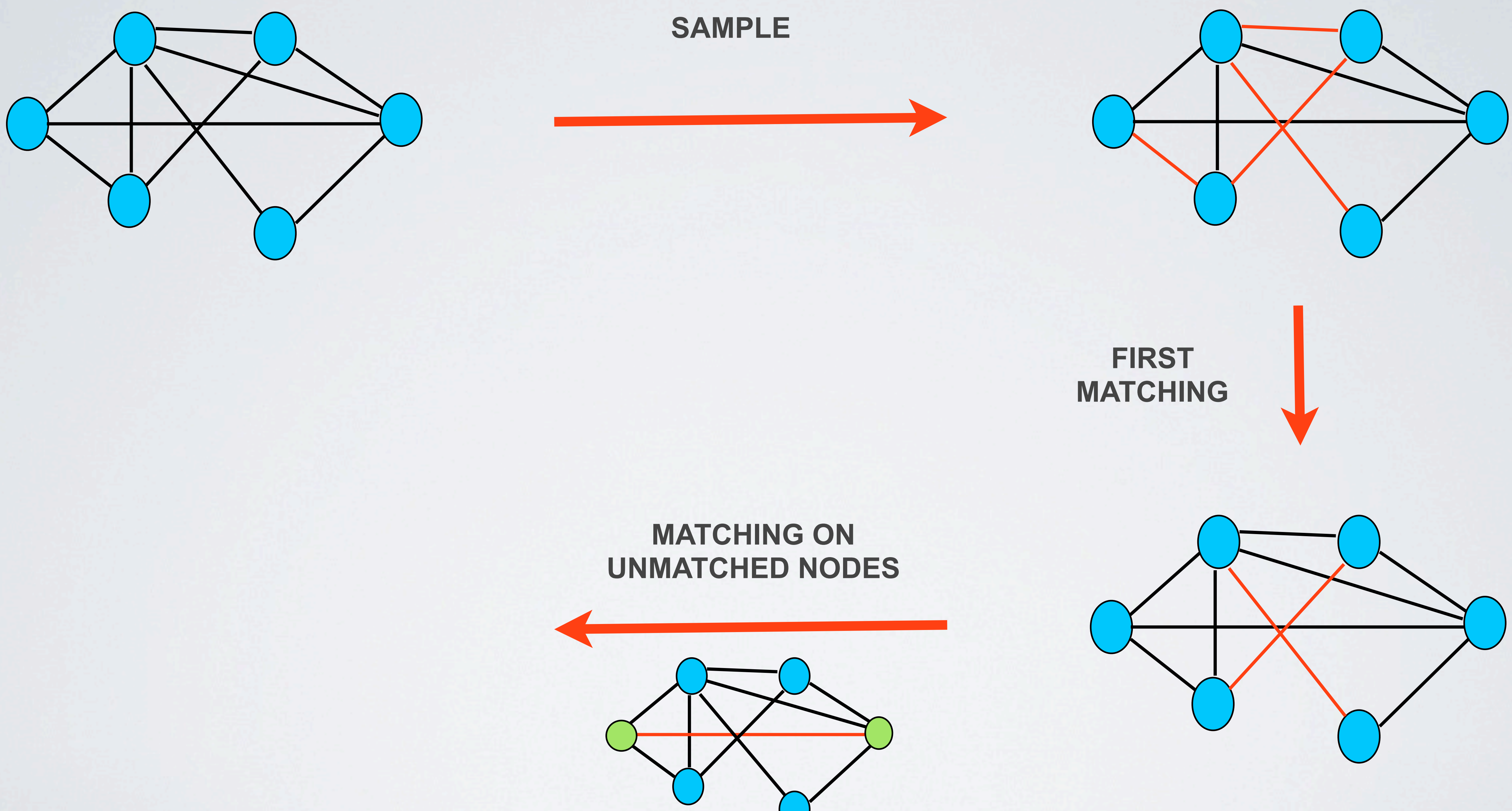


Algorithm

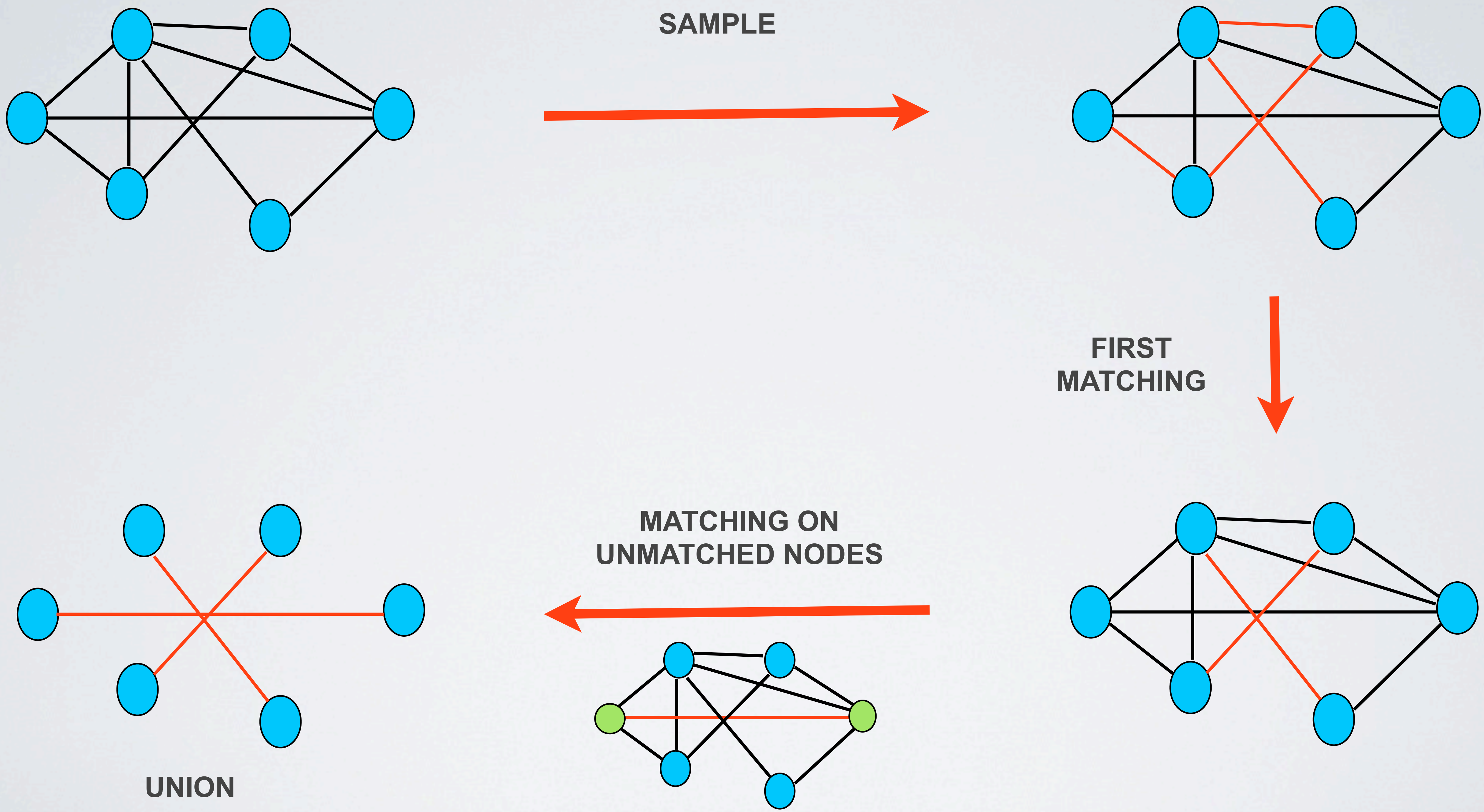
Algorithm



Algorithm



Algorithm



Correctness

- Consider the last step of the algorithm

Correctness

- Consider the last step of the algorithm

- All unmatched vertices are placed on a machine

Correctness

- Consider the last step of the algorithm
- All unmatched vertices are placed on a machine
- All unmatched vertices or are matched at the last step or have only matched neighbors

Bounding the rounds

Three rounds:

- Sampling the edges

Bounding the rounds

Three rounds:

- Sampling the edges
- Find a matching on a single machine for the sampled edges

Bounding the rounds

Three rounds:

- Sampling the edges
- Find a matching on a single machine for the sampled edges
- Find a matching for the unmatched vertices

Bounding the memory

- Each edge was sampled with probability $p = \dfrac{10\log n}{n^{c/2}}$

Bounding the memory

- Each edge was sampled with probability $p = \dfrac{10 \log n}{n^{c/2}}$
- Using Chernoff the sampled graph has size $\tilde{O}(n^{1+c/2})$ with high probability

Bounding the memory

- Each edge was sampled with probability $p = \dfrac{10 \log n}{n^{c/2}}$
- Using Chernoff the sampled graph has size $\tilde{O}(n^{1+c/2})$ with high probability
- Can we bound the size of the induced subgraph on the unmatched vertices?

Bounding the memory

- For a fixed induced subgraph with at least $n^{1+c/2}$ edges the probability an edge is not sampled is:

$$\left(1 - \frac{10\log n}{n^{c/2}}\right)^{n^{1+c/2}} \leq exp(-10n\log n)$$

Bounding the memory

- For a fixed induced subgraph with at least $n^{1+c/2}$ edges the probability an edge is not sampled is:

$$\left(1 - \frac{10\log n}{n^{c/2}}\right)^{n^{1+c/2}} \leq exp(-10n\log n)$$

- Union bounding over all 2^n induced subgraphs shows that at least one edge is sampled from every dense subgraph with probability

$$1 - exp(-10\log n) \geq 1 - \frac{1}{n^{10}}$$

Using less memory

- Can we run the algorithm with less then $\tilde{O}(n^{1+c/2})$ memory?

Using less memory

- Can we run the algorithm with less then $\tilde{O}(n^{1+c/2})$ memory?

- We can amplify our sampling technique!

Using less memory

- Sample as many edges as possible that fits in memory

Using less memory

- Sample as many edges as possible that fits in memory
- Find a matching

Using less memory

- Sample as many edges as possible that fits in memory
- Find a matching
- If the edges between unmatched vertices fit onto a single machine, find a matching on those vertices

Using less memory

- Sample as many edges as possible that fits in memory
- Find a matching
- If the edges between unmatched vertices fit onto a single machine, find a matching on those vertices
- Otherwise, recurse on the unmatched nodes

Using less memory

- Sample as many edges as possible that fits in memory
- Find a matching
- If the edges between unmatched vertices fit onto a single machine, find a matching on those vertices
- Otherwise, recurse on the unmatched nodes
- With $n^{1+\epsilon}$ memory each iteration a factor of n^{ϵ} edges are removed resulting in $O(c/\epsilon)$ rounds

Parallel computation power

- Maximal matching algorithm does not use parallelization

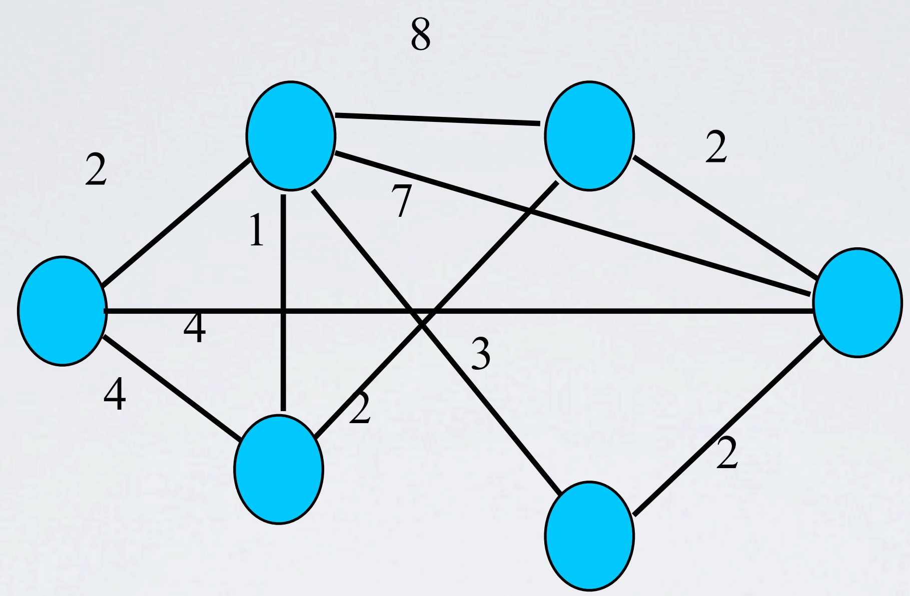
Parallel computation power

- Maximal matching algorithm does not use parallelization
- We use a single machine in every step

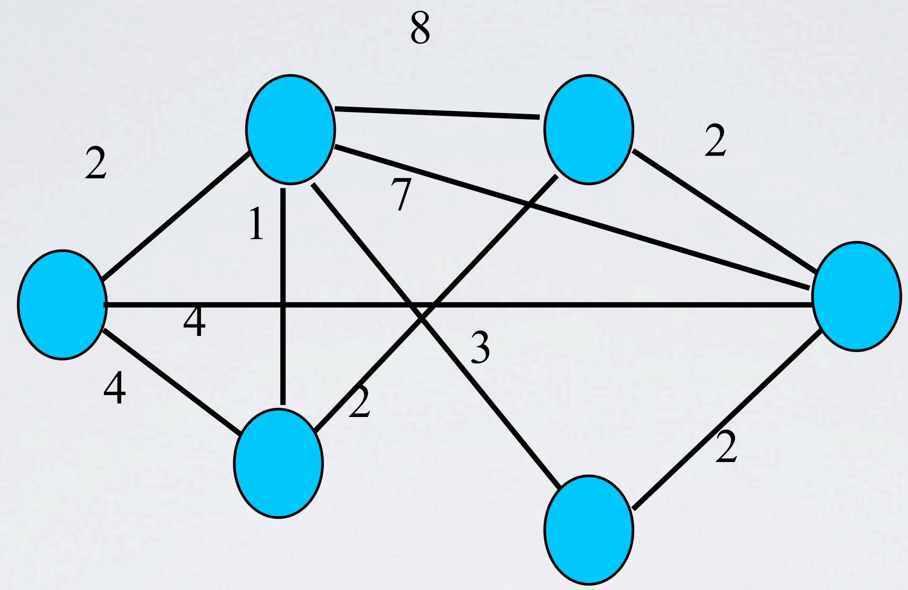
Parallel computation power

- Maximal matching algorithm does not use parallelization
- We use a single machine in every step
- When parallelization is used?

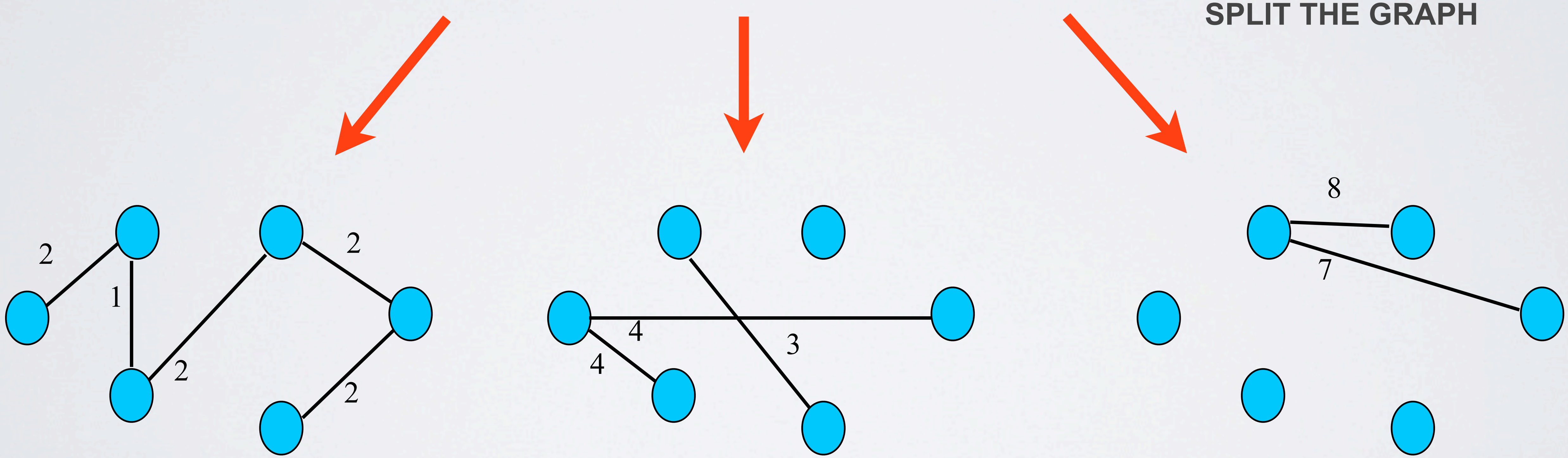
Maximum weighted matching



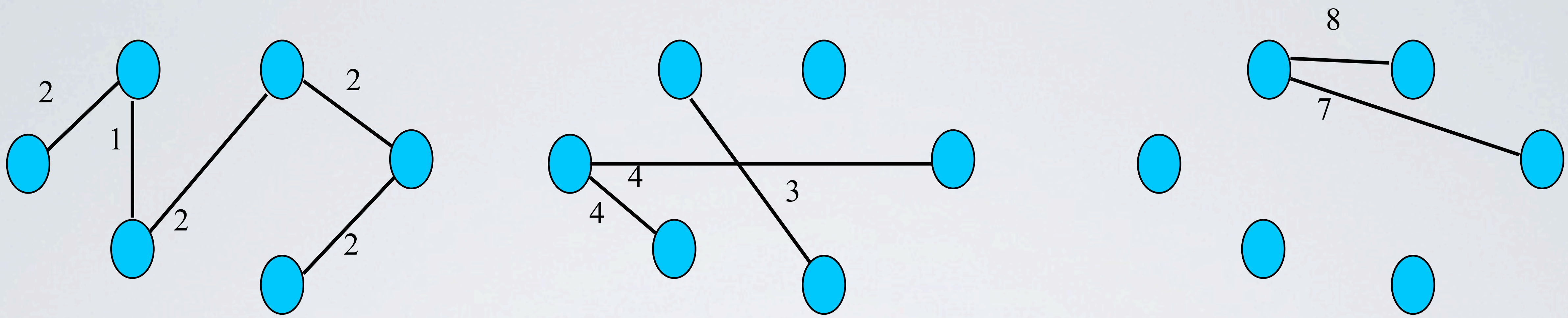
Maximum weighted matching



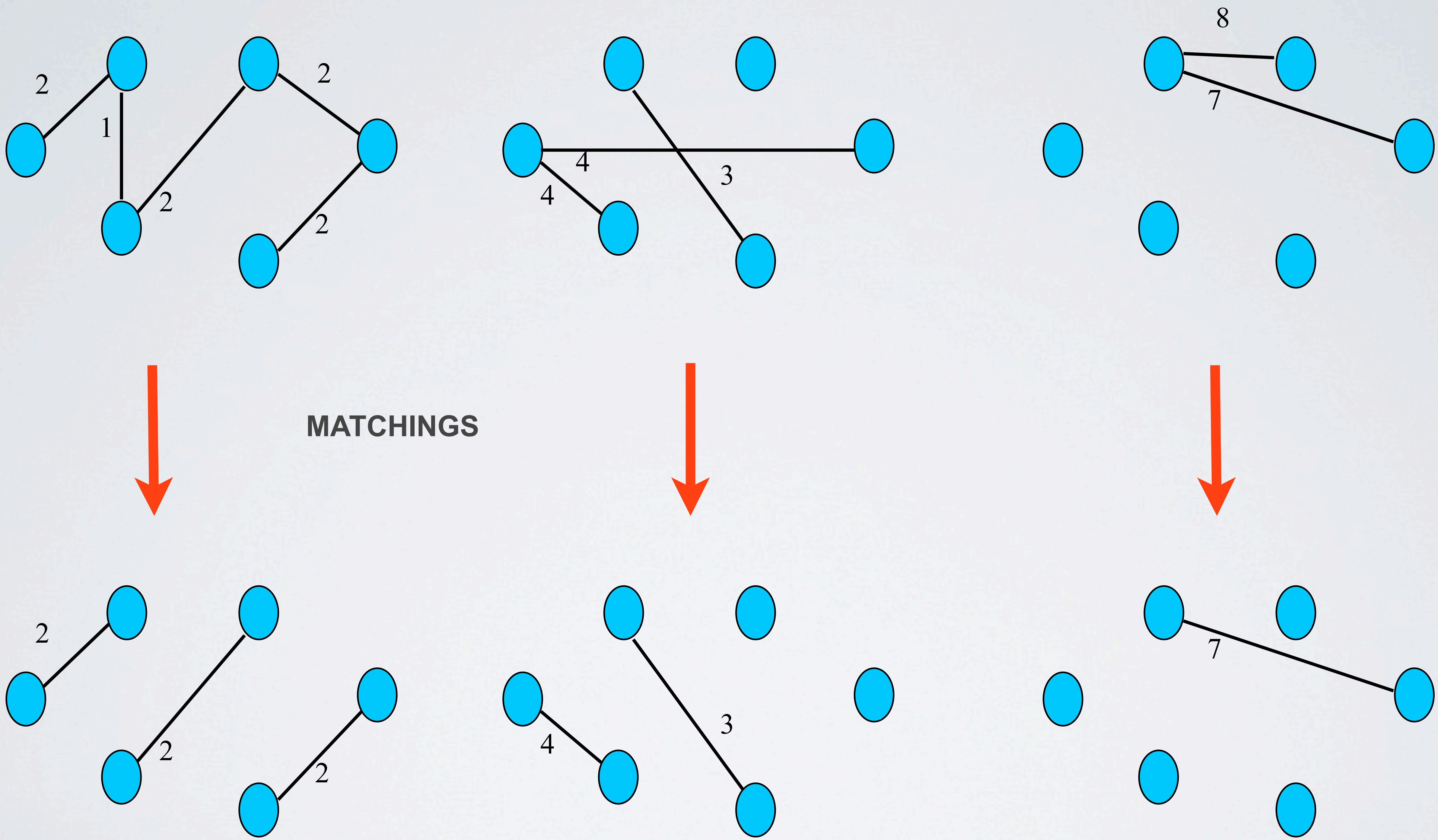
SPLIT THE GRAPH



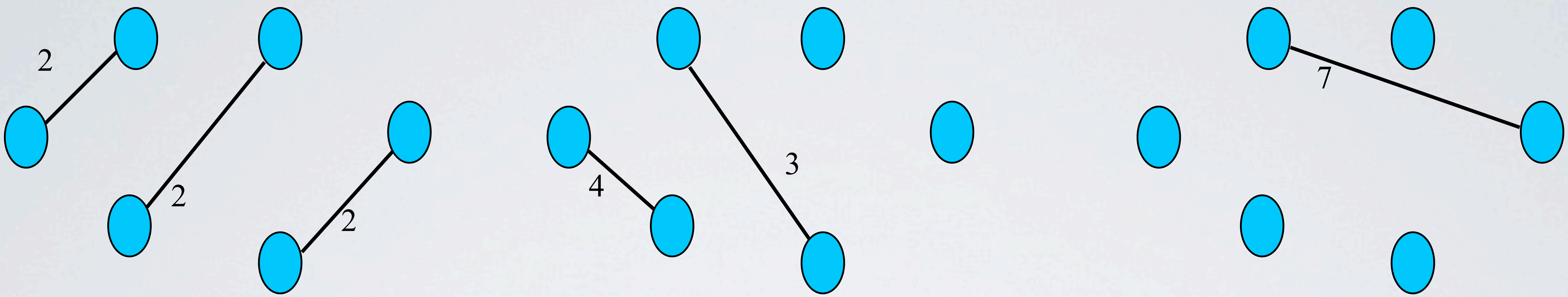
Maximum weighted matching



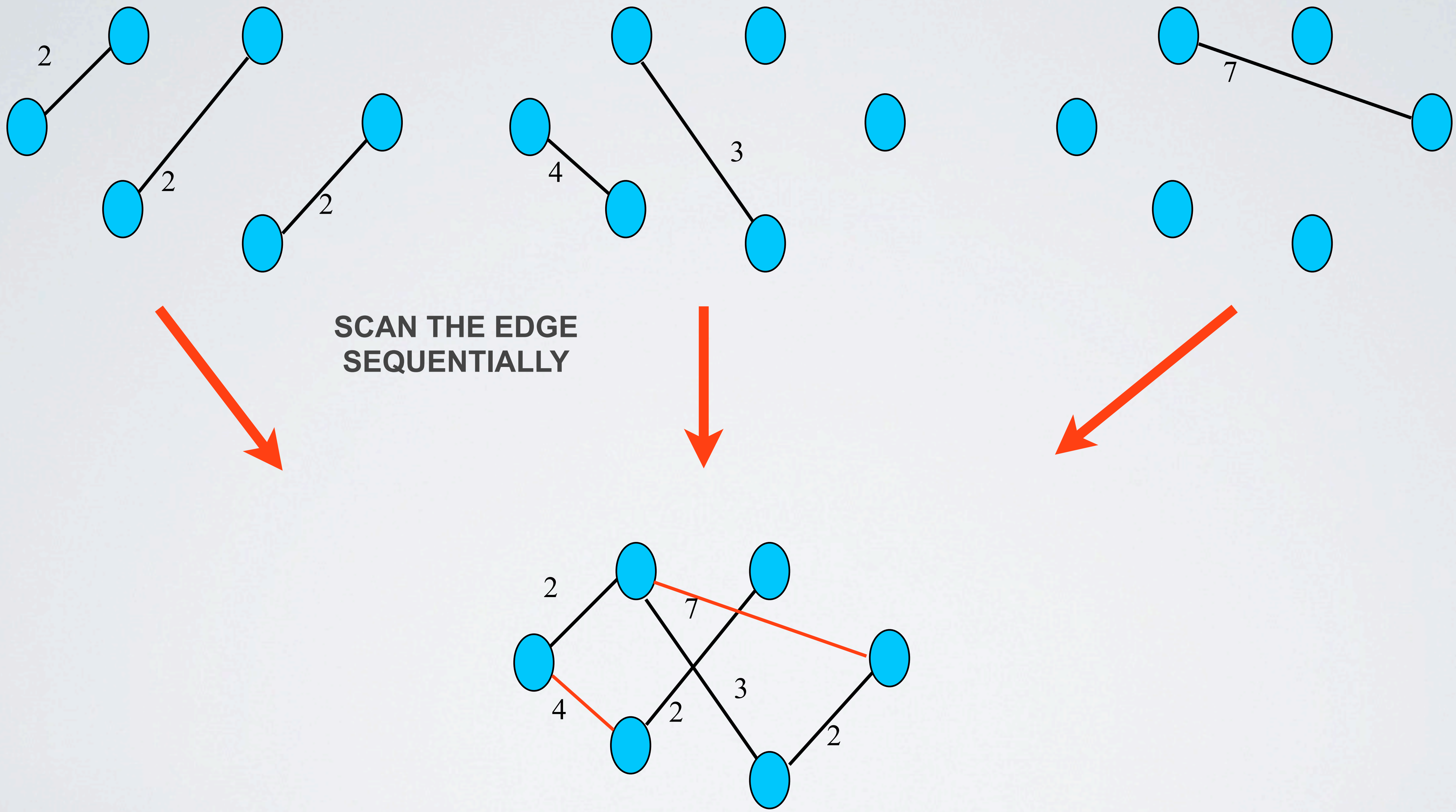
Maximum weighted matching



Maximum weighted matching



Maximum weighted matching



Bounding the rounds and memory

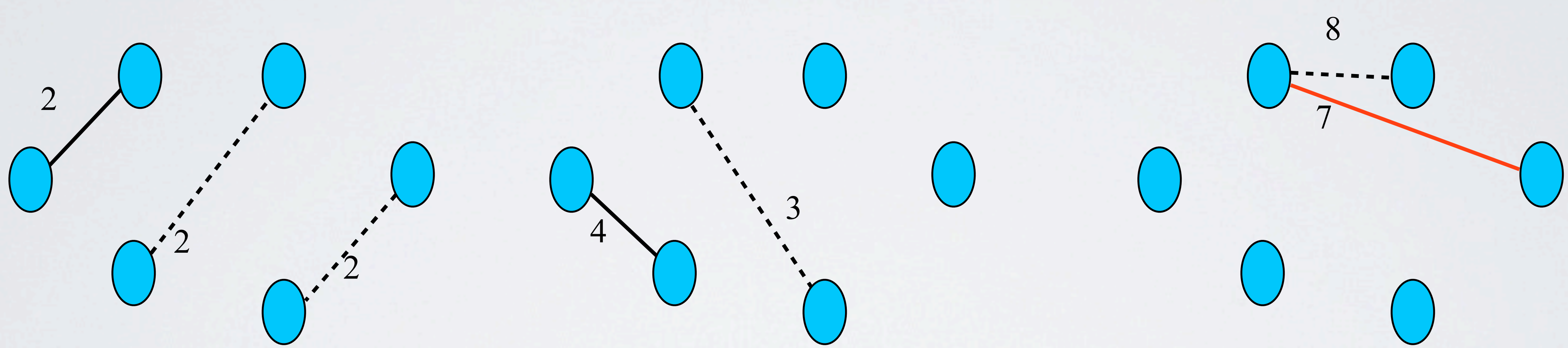
Four rounds:

- Splitting the graph and running the maximal matching:
  3 rounds and $\tilde{O}(n^{1+c/2})$ memory

- Compute the final solution:
  1 round and $O(n \log n)$ memory

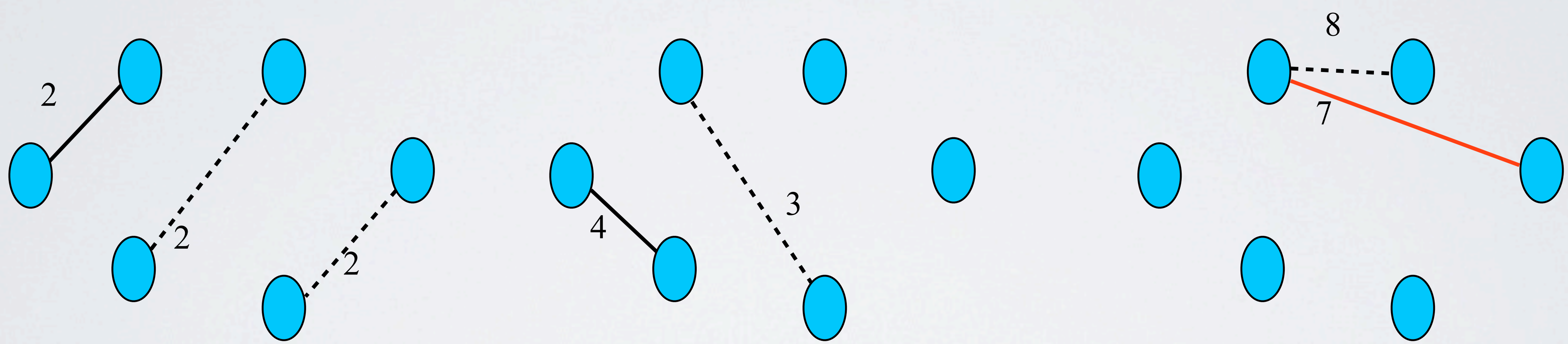
Approximation guarantee (sketch)

- An edge in the solution can block at most 2 edges in each subgraph of smaller weight



Approximation guarantee (sketch)

- An edge in the solution can block at most 2 edges in each subgraph of smaller weight



- We loose a factor or 2 because we do not consider the weight

Other algorithms

Based on the same intuition:

- 2-approx for vertex cover

- 3/2-approx for edge cover

Minimum cut

- Partition does not work, because we loose structural informations

Minimum cut

- Partition does not work, because we loose structural informations
- Sampling does not seem to work either

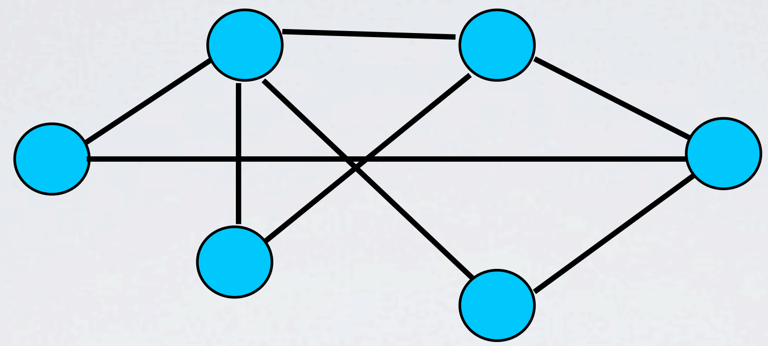
Minimum cut

- Partition does not work, because we loose structural informations
- Sampling does not seem to work either
- We can use the first steps of Karger algorithm as a filtering technique

Minimum cut

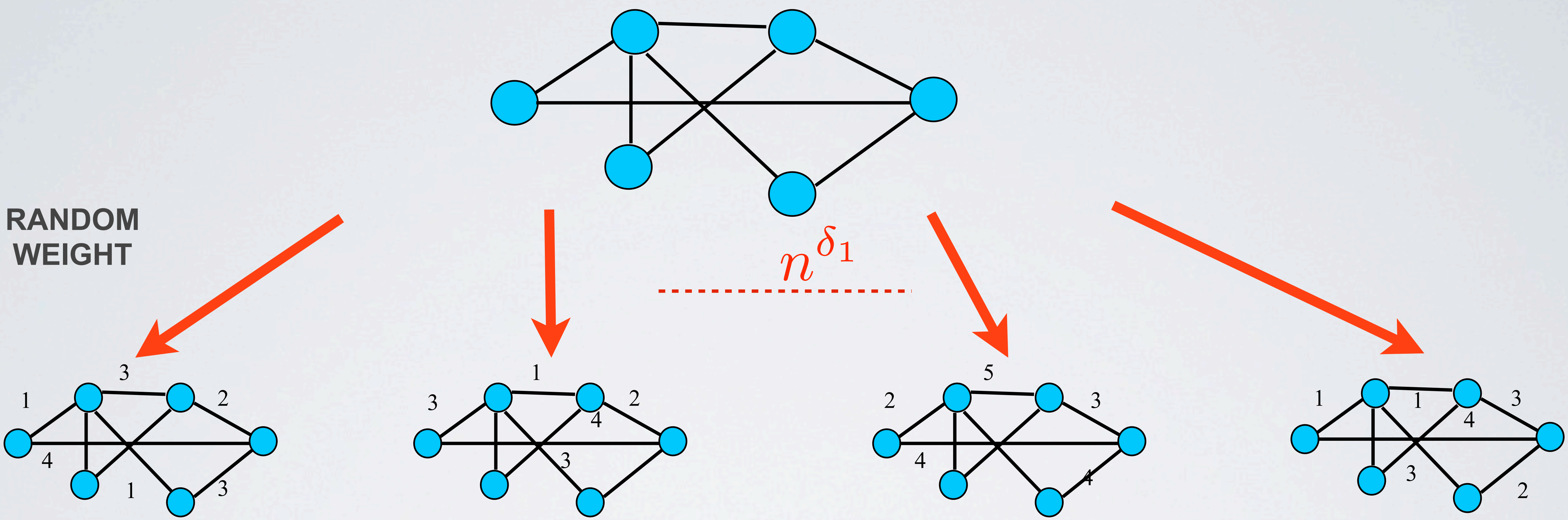
- Partition does not work, because we loose structural informations
- Sampling does not seem to work either
- We can use the first steps of Karger algorithm as a filtering technique
- The random choices made in the early rounds succeed with high probability, whereas the later rounds have a much lower probability of success

Minimum cut

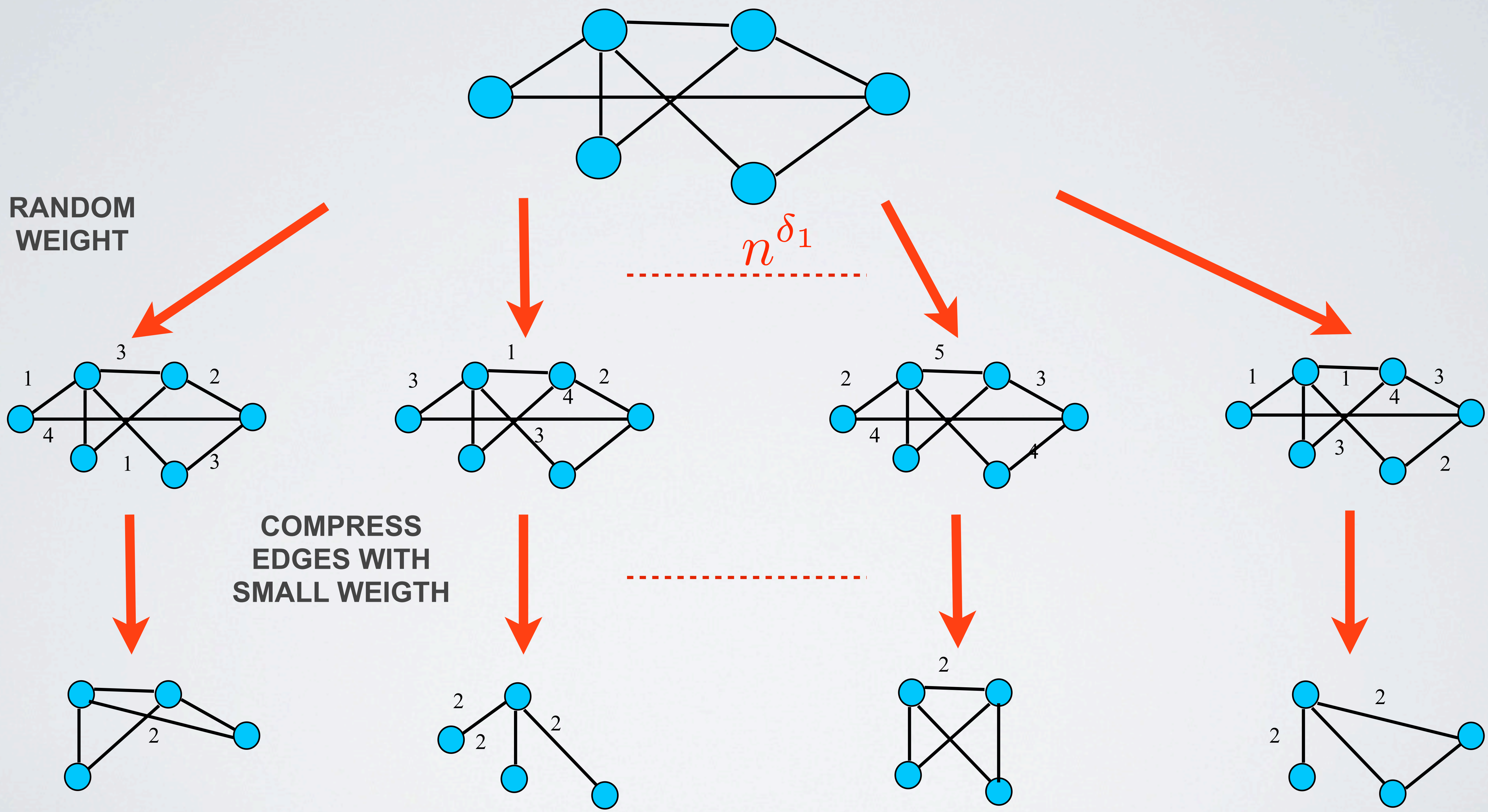


1

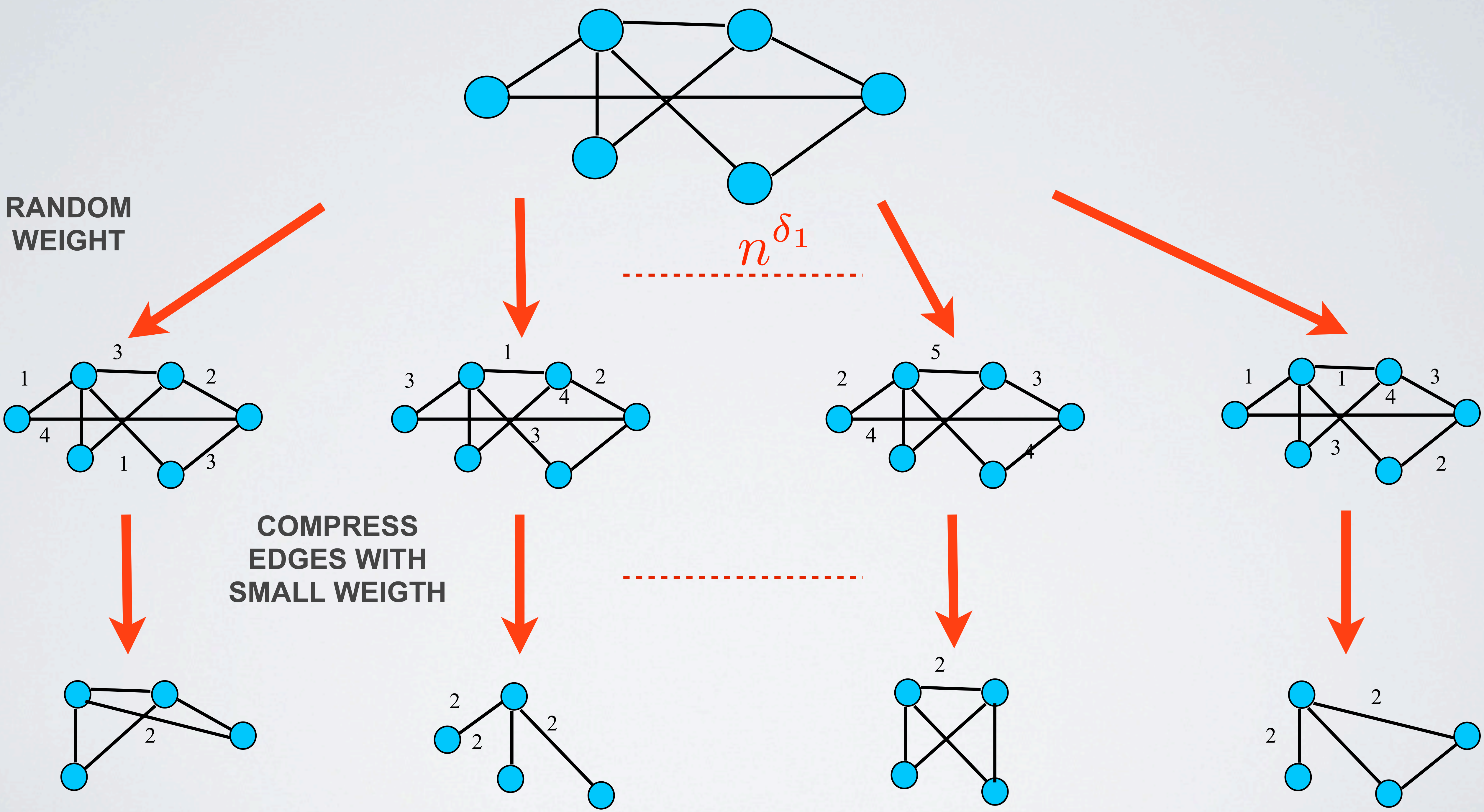
Minimum cut



Minimum cut

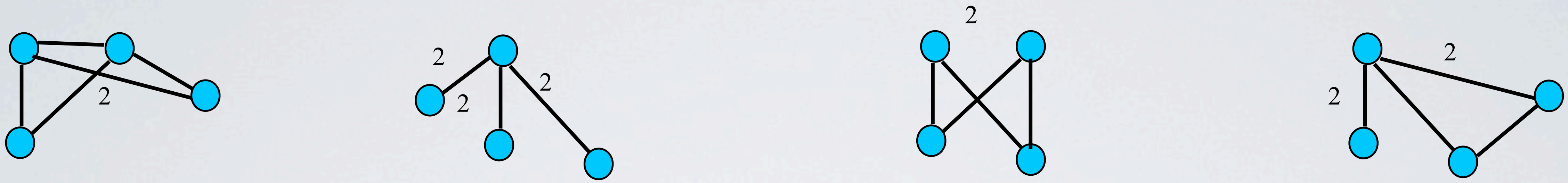


Minimum cut

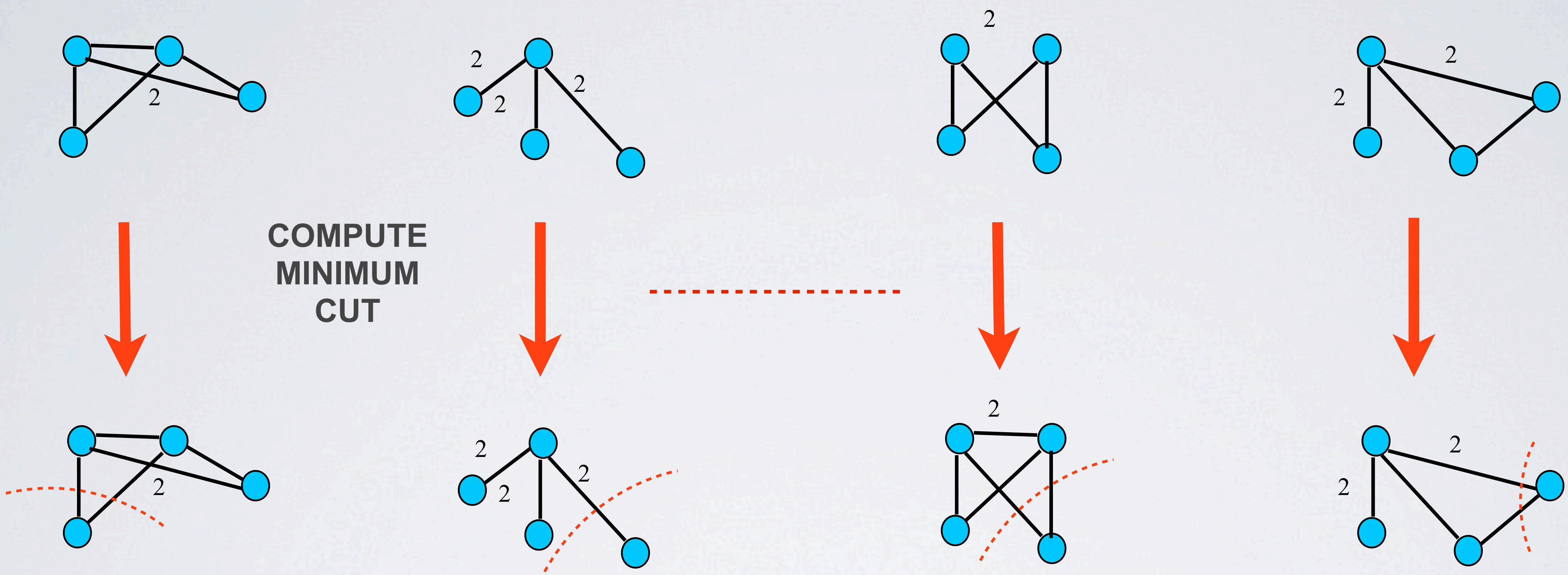


From Karger at least one graph will contain a min cut w.h.p.

Minimum cut

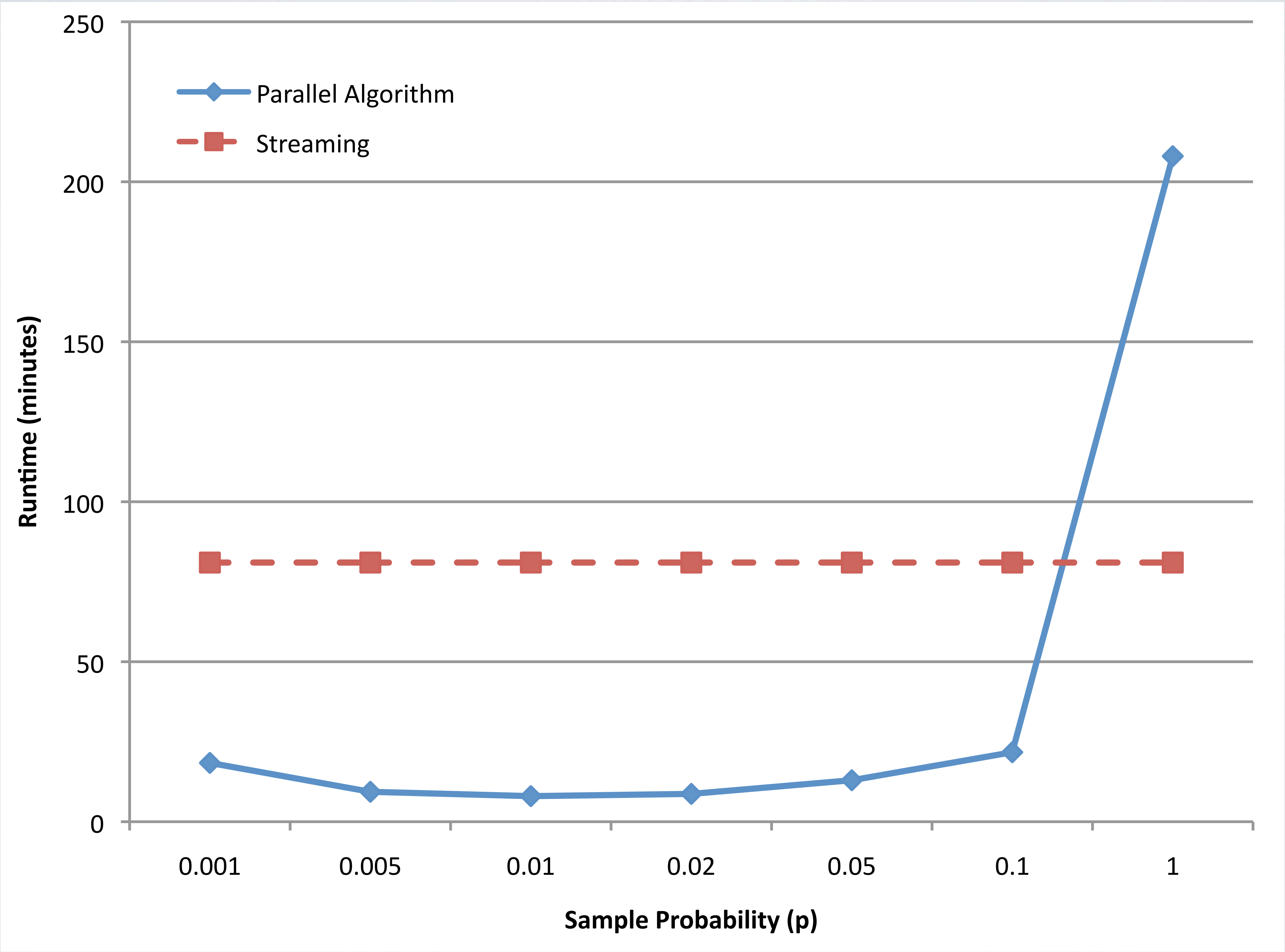


Minimum cut



We will find the minimum cut w.h.p.

Empirical result (matching)



Open problems

Open problems 1

- Maximum matching

- Shortest path

- Dynamic programming

Open problems 2

- Algorithms for sparse graph
- Does connected components require more than 2 rounds?
- Lower bounds

Thank you!