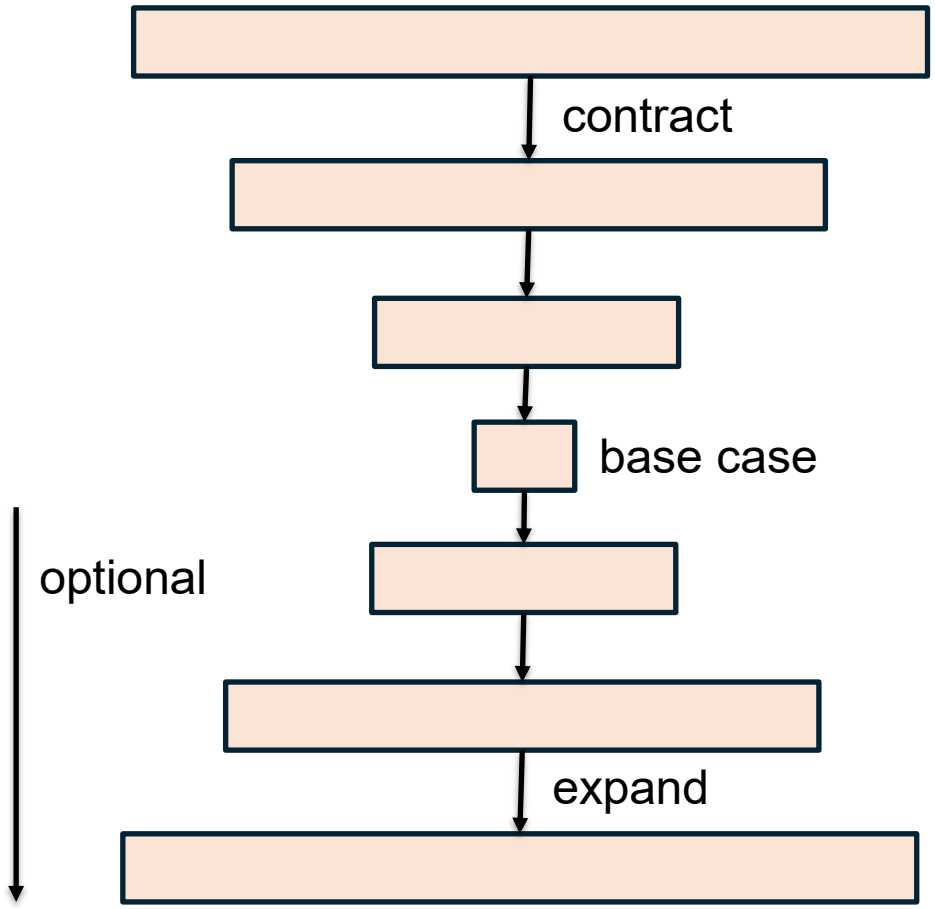
Parallel And Sequential Data Structures and Algorithms

Graph Contraction, Part I

Learning Objectives

- Introduce **Graph Contraction**, a way to achieve polylogarithmic span algorithms on graphs
- Define two different types of Graph Contraction: **Edge Contraction** and **Star Contraction**, and analyze their performance

Contraction: Review

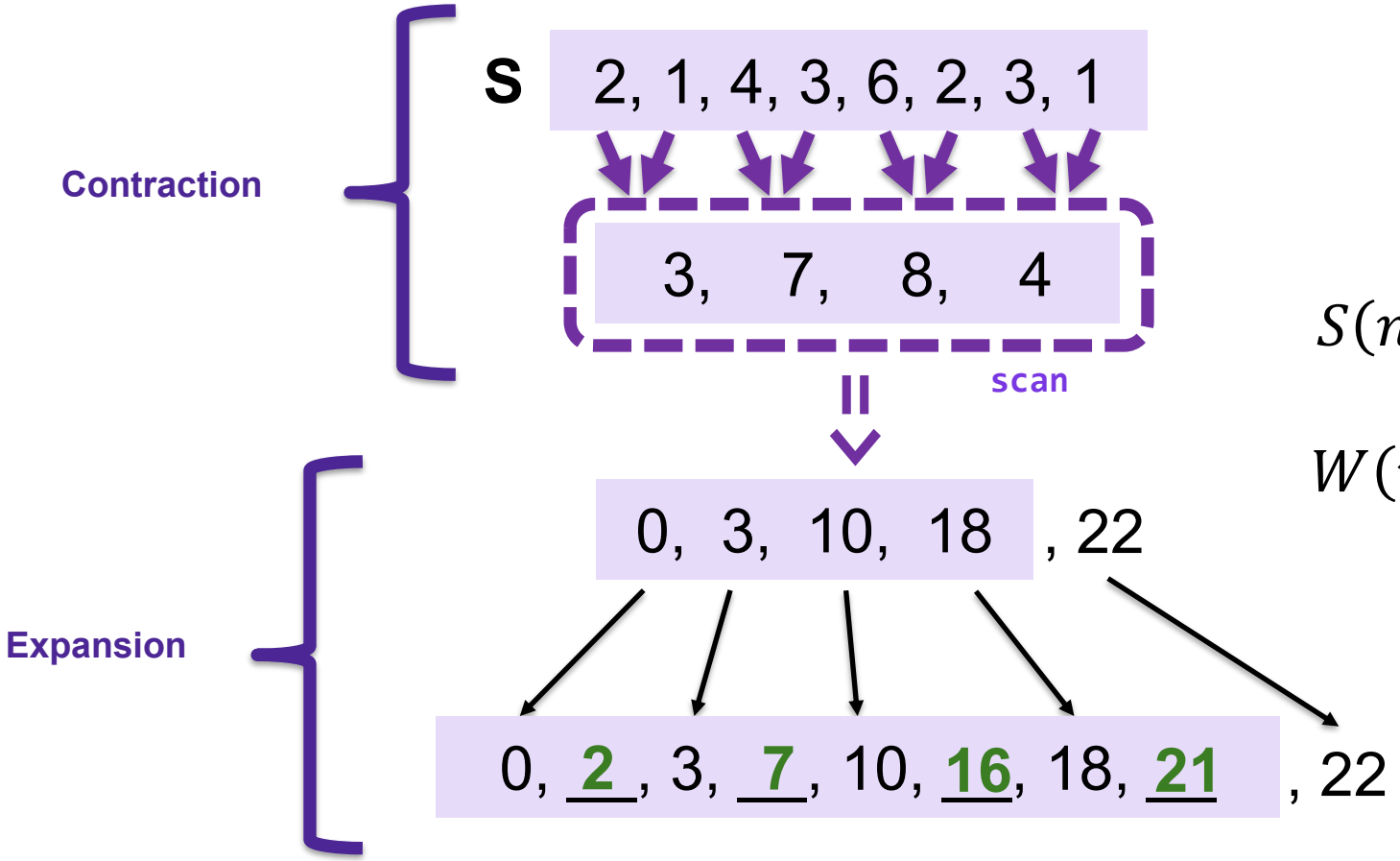



```
fun contractionAlgorithm(A) :
  if base case : return base(A)
  else :
    C = contract(A)
    R = contractionAlgorithm(C)
    E = expand(R)          // optional
    return E
```

Advantages:

- Correctness by induction
- If shrinks by constant factor
  - # levels is logarithmic
  - Work is root dominated

Contraction-based Scan



$$S(n) = S\left(\frac{n}{2}\right) + O(1) \in O(\log n)$$

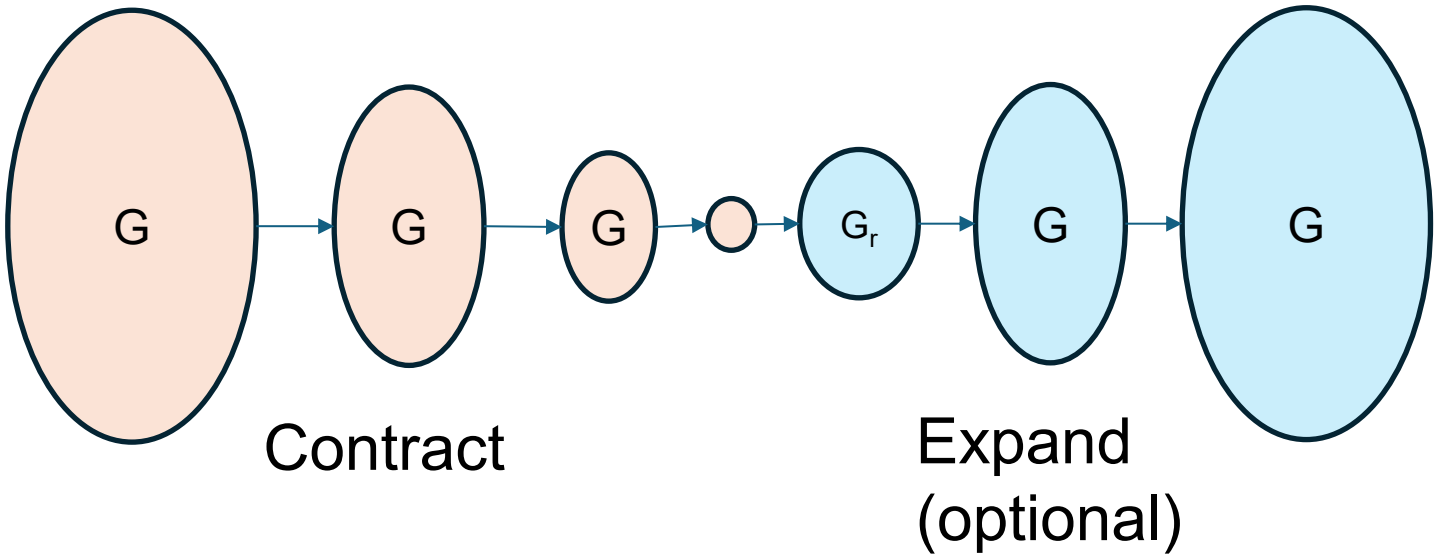
$$W(n) = W\left(\frac{n}{2}\right) + O(n) \in O(n)$$

Other contraction-based algorithms?

- reduce
- QuickSelect (also ***randomized***)

What is Graph Contraction?

Graph contraction:




```
fun graphContraction(G) :
  if base : return base(G)
  else :
    Gc = contractGraph(G)
    Gr = graphContraction(Gc)
    Ge = expandGraph(Gr)
    return Ge
```

Mostly used on **undirected** graphs

Goals:

- Reduce size by constant factor
- Maintain connectivity

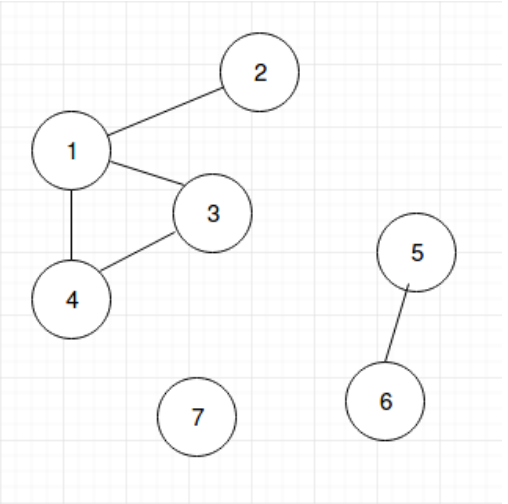
Why Graph Contraction:

Goal: **get graph algorithms with polylogarithmic span** (for any graph).

- We have not achieved this for **any** graph problems discussed so far.
  - DFS and Dijkstra were not parallel at all
  - BFS is parallel, but span is $O\left(D \log^2(n)\right)$, where D is the diameter
  - Bellman-Ford and Floyd-Warshall are highly parallel, but still have span that is at least $O(n)$

Applications of Graph Contraction

- Graph connectivity (undirected graphs)



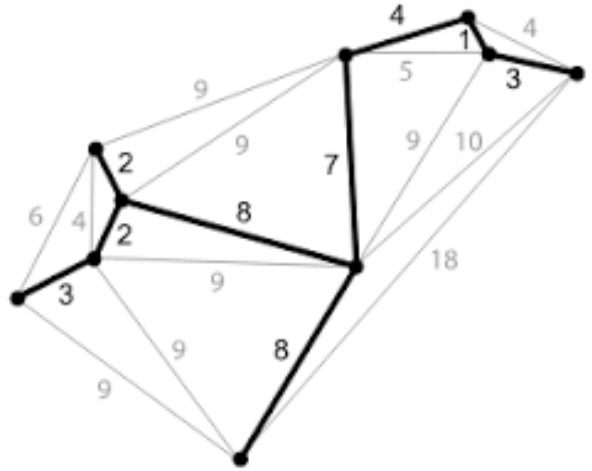
Definition (Connected Components): The maximal subgraphs that are connected.

Problem (Component Count): Return the number of connected components.

Problem (Component Labeling): Give all vertices a label such that two vertices have the same label if and only if they are in the same component.

Applications of Graph Contraction

- Minimum Spanning Trees (MSTs), *foreshadowing… 2 lectures from now!*



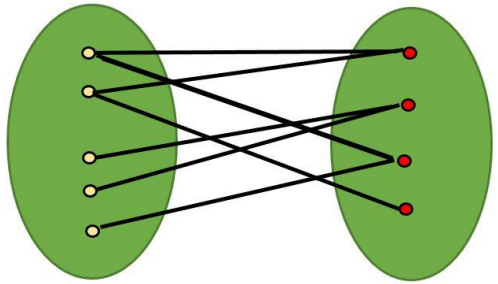
Definition (Tree): a connected acyclic graph

Definition (Spanning Tree): for a graph G = (V,E), a tree T = (V,E') with $E' \subset E$ i.e., a subgraph that is a tree.

Definition (Minimum Spanning Tree): For a weighted graph, a spanning tree with minimum sum of weights.

Applications of Graph Contraction

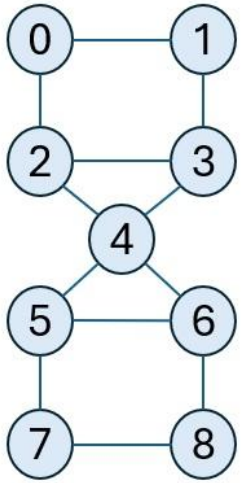
- Connected Components
- Minimum Spanning Trees (MSTs),
- Check if a graph is bipartite



Definition (Bipartite): a graph $G = (V, E)$ is bipartite if V can be partitioned into two sets X and Y such that for all edges (u, v) in E, exactly one of u and v is in X, and therefore the other one is in Y.

Applications of Graph Contraction

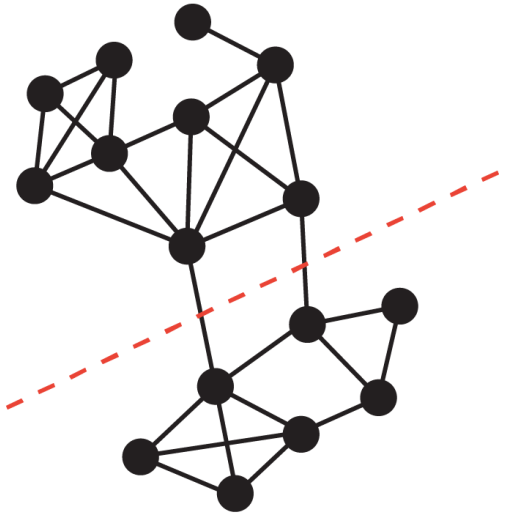
- Connected Components
- Minimum Spanning Trees (MSTs),
- Check if a graph is bipartite
- Find undirected bi-connected components



Definition (Bi-Connected Components): A graph G is biconnected if when any vertex is removed, the graph remains connected.

Applications of Graph Contraction

- Connected Components
- Minimum Spanning Trees (MSTs),
- Check if a graph is bipartite
- Find undirected connected components
- Min Cut in a graph



Definition (Cut): A partition of a graph's vertices into S and $V - S$.

A **Min Cut** of a graph *G* is a cut that minimizes the sum of the weights of the edges that cross the cut, i.e., edges (u, v) such that exactly one of u and v is in *S*, and the other is not in *S*.

Types of Graph Contraction

One Step of Graph Contraction

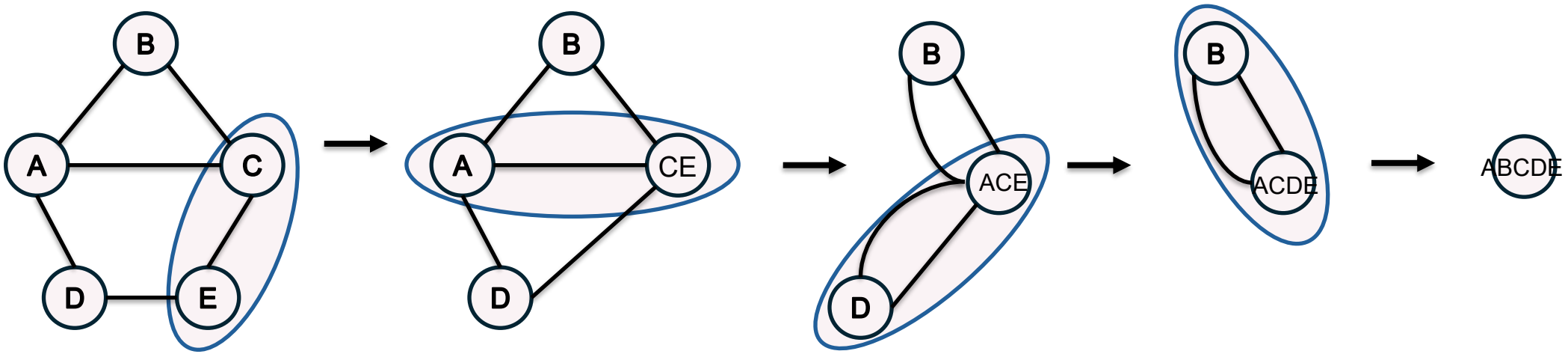
Generic Algorithm (Contraction):

- Take a connected set of edges and vertices, and contract them into a new **“super vertex”**
- Can find multiple connected sets **in parallel** as long as the vertices in each set are **disjoint**.

Edge Contraction

Sequential Edge Contraction

- The “connected sets of edges and vertices” is just one edge and its two endpoints.



Parallel Edge Contractions

Pick multiple edges, but must be edges that do not share a vertex.

Definition (Matching): a **matching** in a graph $G = (V, E)$ is a subset of edges $E' \subseteq E$ s.t. no two edges share an endpoint.

Important in graph theory

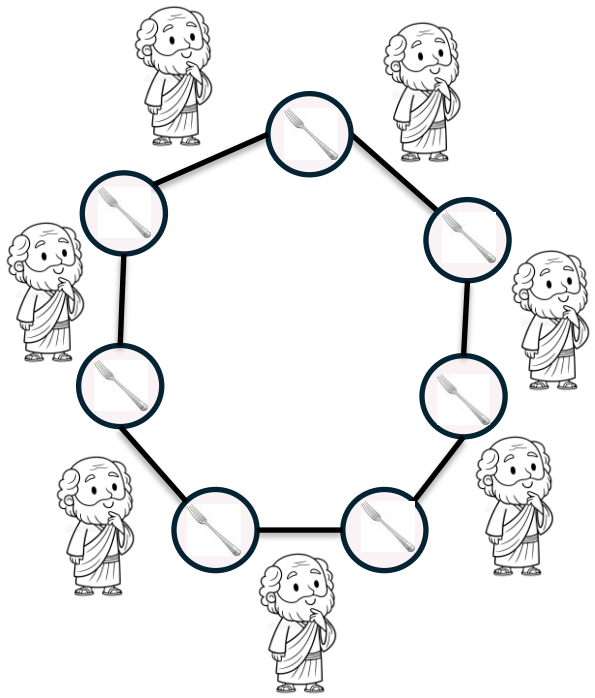
Definition (Maximal Matching): a **matching** s.t. no more edges can be added without sharing an endpoint.

Definition (Maximum Matching): a **matching** that has maximizes $|E'|$

We only need a matching

Parallel Edge Contraction

- How do we find a matching in parallel that covers many edges?

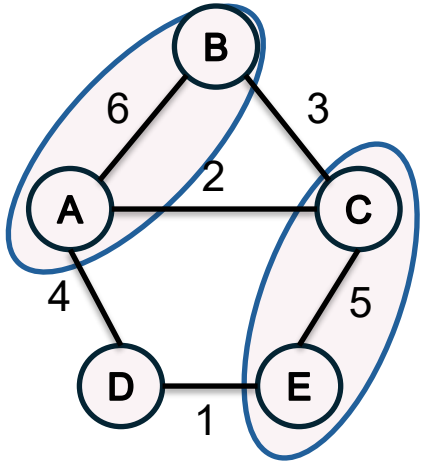


Problem (Dining philosophers):
- They need both forks to eat, but they keep conflicting with their neighbors.
- How do we break this “symmetry”?

Randomization: e.g.
- Flip a coin and wait if tails
- Flip a coin and pick from left if tails, then right
- …

Parallel Edge Contraction

? **What if we give each edge a random priority, how would we select edges?**

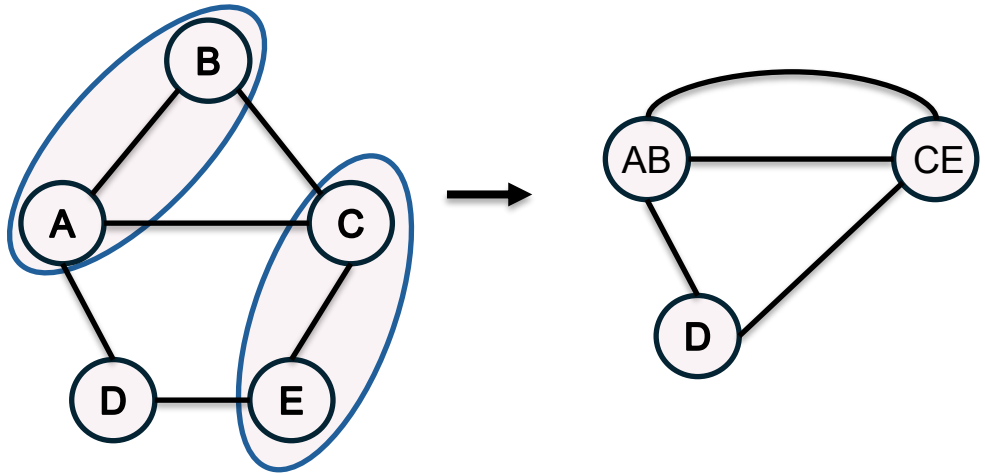


Pick the local max?

- i.e., largest among neighboring edges
- Which would we pick?

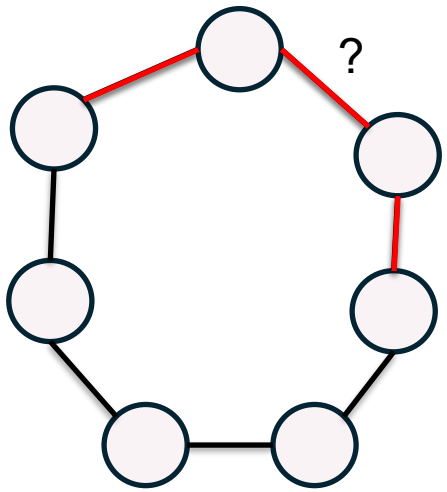
Parallel Edge Contraction

Algorithm (Parallel Edge Contraction): given each edge has random priority, select an edge for contraction if its priority is higher than all its incident edges. Contract all such edges.



Parallel Edge Contraction

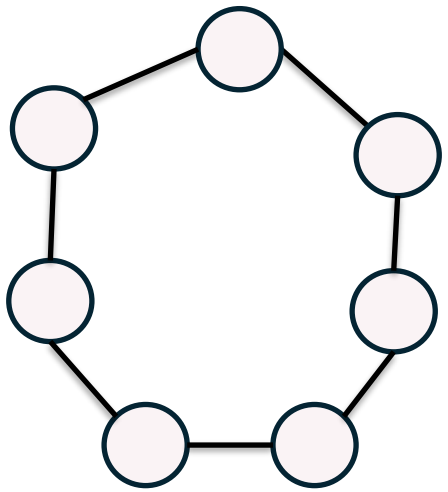
? Consider the cycle graph. What is the probability an edge will be selected?



Every edge has two neighboring edges, so 1 out of 3

Parallel Edge Contraction

? What is the expected number of edges removed?



$$X_e = \begin{cases} 1 & \text{if selected} \\ 0 & \text{otherwise} \end{cases}$$

$$X = \sum_{e \in E} X_e \quad \text{(total number selected)}$$

$$\mathrm{E}[X] = \sum_{e \in E} \mathrm{E}[X_e]$$

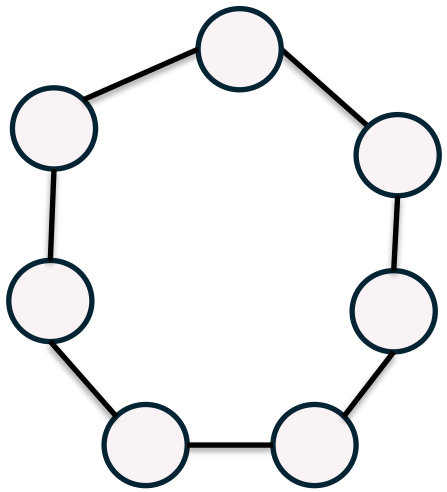
$$E[X_e] = \text{probability e is selected} = 1/3$$

$$\text{Hence } \mathrm{E}[X] = \frac{|E|}{3}$$

Parallel Edge Contraction

? How many rounds of contraction?





By the “Skittles Lemma” since each round is reducing the size in expectation by a constant factor, the number of rounds is $O(\log n)$ with high probability.

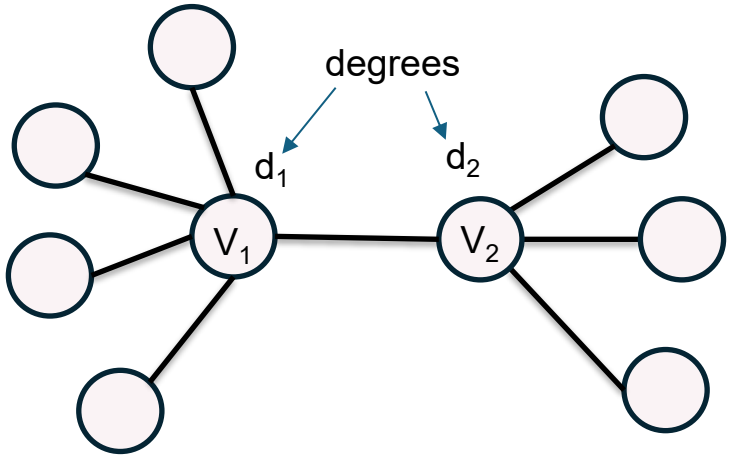
Furthermore, if each round takes linear work, the total work is $O(n)$ in expectation.

? What is the limitation of this analysis?

We only showed it for the cycle graph

Parallel Edge Contraction

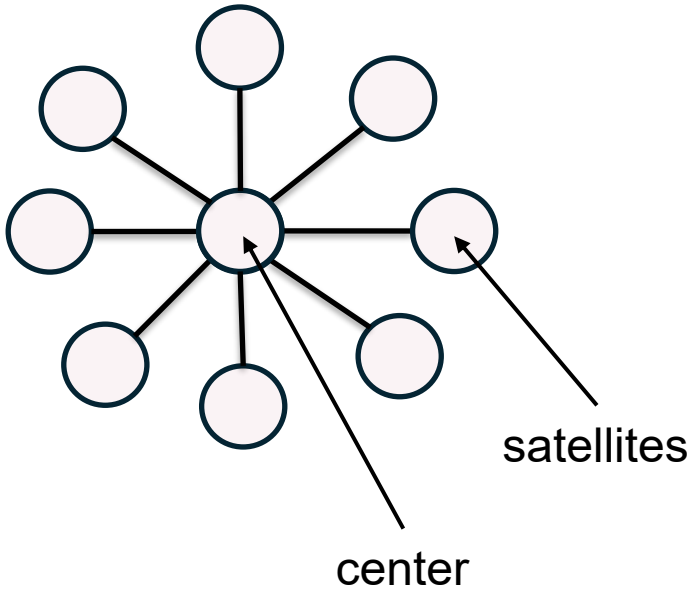
? What is the probability the edge (v_1, v_2) is selected?



$$\mathbf{Pr}[\text{selected}] = \frac{1}{d_1 + d_2 - 1}$$

Parallel Edge Contraction

? What is the worst case?



The star graph: a set of $n - 1$ edges incident on a single vertex.

Can only select one edge per round.
Total rounds = $n - 1$

Star Contraction

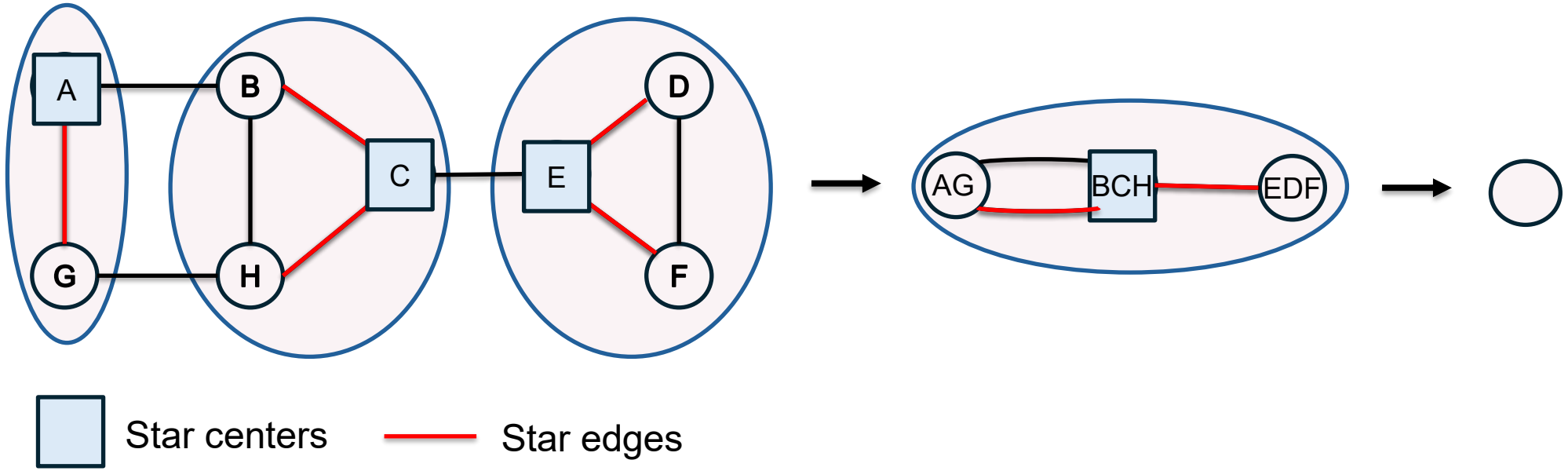
Star Contraction

Let's get rid of the star

Star Idea:
1. Select a set of edges that form **independent stars**
2. Contract each star

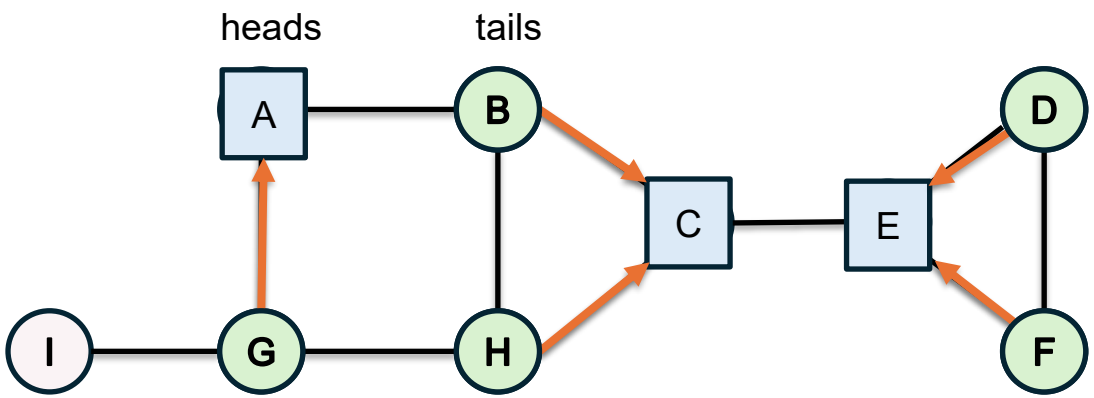
Star Contraction

Star Idea:
1. Select a set of edges that form **independent stars**
2. Contract each star



Star Contraction

But how do we pick **independent stars** in **parallel**?



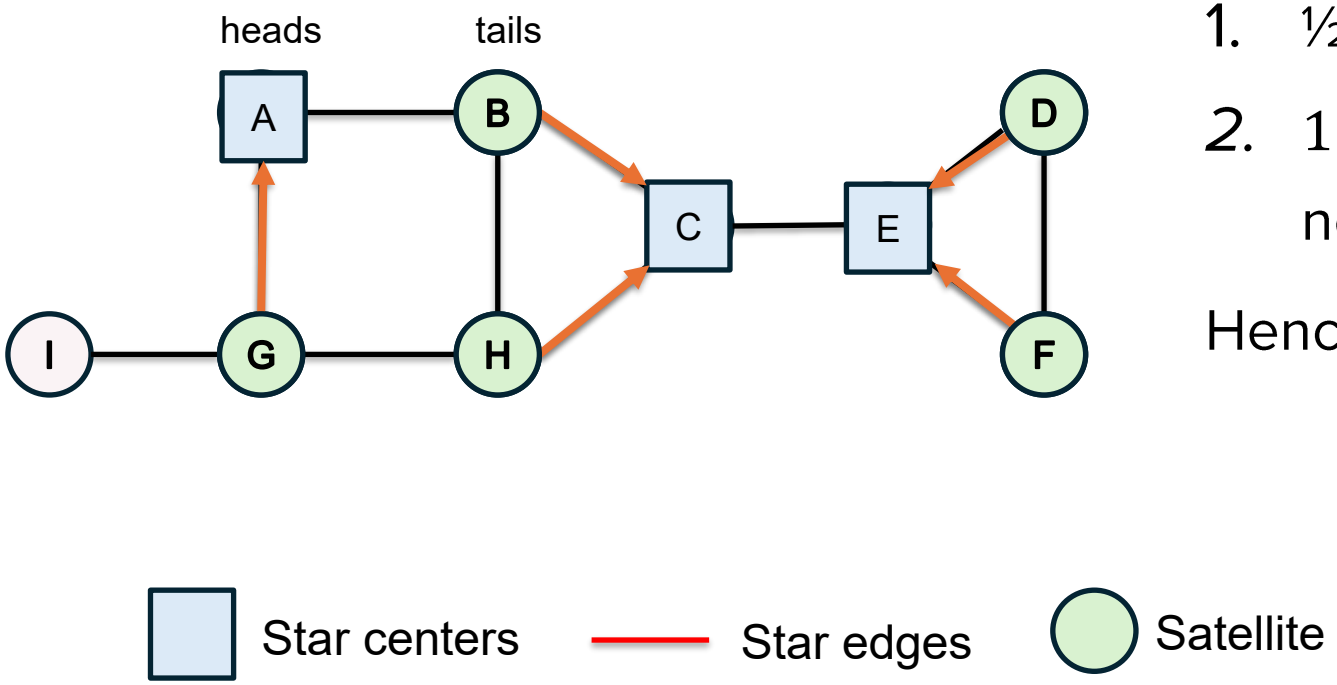
1. Flip a coin on each vertex. A vertex is a **star center** if heads.
2. For each vertex that flips tails pick a neighboring heads (center) if there is one.
3. All tails that found a head are **satellites**

Star centers Star edges Satellite

Note: each satellite will be removed in contraction.

Star Contraction

? What is probability a vertex v is a satellite?



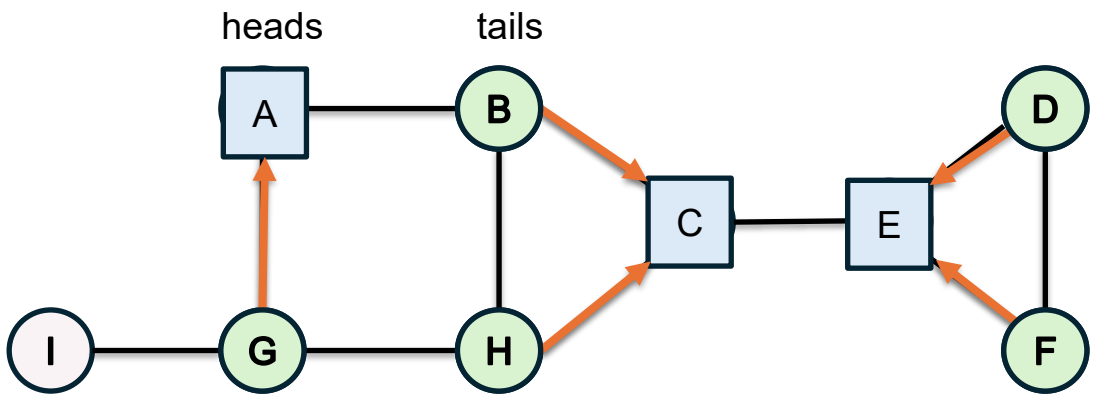
1. ½ probability that v flips tails
2. $1 - \left(\frac{1}{2}\right)^{\mathrm{d}}$ probability that v has a neighboring head

Hence probability that v is a satellite is:

$$\frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^{\mathrm{d}}\right) \geq \frac{1}{4}$$

Star Contraction

? How many vertices are removed on each round?



Importantly: this holds for any graph!!

$$X_v = \begin{cases} 1 & \text{if satellite} \\ 0 & \text{otherwise} \end{cases}$$

$$X = \sum_{v \in V} X_v \text{ (number of vertices removed)}$$

$$\boldsymbol{E}[X] = \sum_{v \in V} \mathrm{E}[X_v]$$

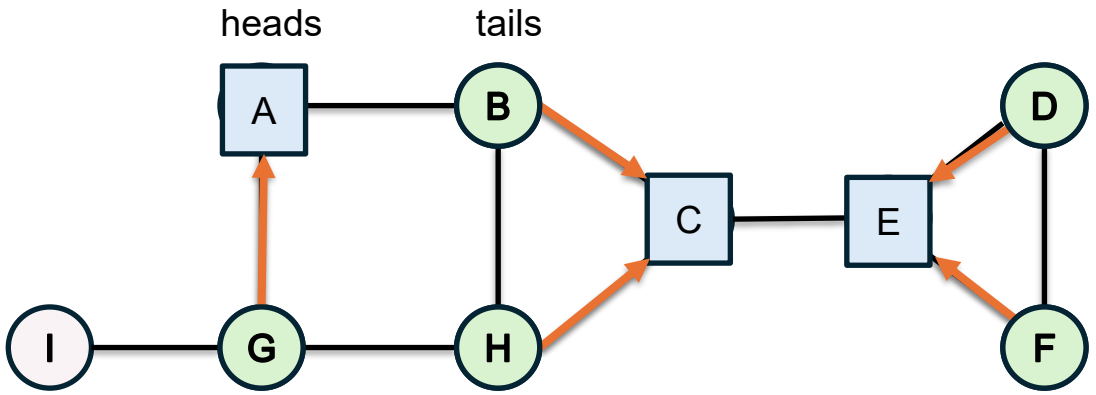
$$\mathbf{E}[X_v] = \text{ probability v is satellite } \geq 1/4$$

Hence $\mathbf{E}[X] = \frac{|V|}{4}$

Star Contraction

? How many rounds of contraction?





By the “Skittles Lemma” since each round is reducing the expected number of vertices by a constant factor, the number of rounds is $O(\log n)$ with high probability.

Star Contraction :

Algorithm: Generic Star Contraction without expansion

```
fun starContraction(G = (V,E)) :
  if |E| = 0 : return
  else :
    each vertex in V flips a coin
    each tail picks neighboring head:
       if there is one it is a satellite
    V’ = all but the satellites
    E’ = relabel edges to point to centers and remove self edges
    return starContraction(G’ = (V’,E’))
```