

Lecture Scribe LaTeX

06/02/2026

Uploaded a file Generate a lecture scribe which can be used as a reference during the exams. It mus...

Below is a LaTeX-formatted lecture scribe that strictly reconstructs Lecture 10: Randomized Min-Cut...



Uploaded a file

Generate a lecture scribe which can be used as a reference during the exams. It must be a perfect reconstruction of what was taught in class and should be written in such a way that a student can rely on it to revise the material. Use only the provided lecture slides and relevant textbook material as the context. Do not write any new content, explanations, interpretations, or examples that are not provided in the shared material. The scribe must include, only when present in the context, definitions and notation, assumptions and conditions, statements of theorems or results, proofs or derivations (written step by step exactly as given in the material no extra steps), theorems and results, visual/ tabular data and all worked examples including intermediate steps. Organize it with clear section headings and proper logical flow, strictly following the structure of the lecture, so that it can be used for exam revision. Provide me the latex format for this lecture scribe.



Below is a **LaTeX-formatted lecture scribe** that strictly reconstructs **Lecture 10: Randomized Min-Cut Algorithm** exactly from the provided slides and the referenced textbook section, without adding any new explanations, interpretations, or examples.

The structure, wording, definitions, algorithms, and statements follow the lecture flow precisely and are suitable for **exam revision**.

latex

Copy code

```
\documentclass[12pt]{article}
\usepackage{amsmath, amssymb}
\usepackage{graphicx}
\usepackage{enumitem}
\usepackage{geometry}
\geometry{margin=1in}

\title{CSE400: Fundamentals of Probability in Computing\\
Lecture 10: Randomized Min-Cut Algorithm}
\author{Dhaval Patel, PhD\\
Associate Professor\\
Computer Science and Engineering (CSE)\\
SEAS -- Ahmedabad University}
\date{February 5, 2026}

\begin{document}
\maketitle
```

```

\section*{Outline}
\begin{itemize}
  \item Min-Cut Problem
  \item Why use min-cut?
  \item What is min-cut?
  \item Successful min-cut run
  \item Unsuccessful min-cut run
  \item Max-Flow Min-Cut Theorem
  \item Deterministic Min-Cut Algorithm
  \item Stoer--Wagner Minimum Cut Algorithm
  \item Pseudocode
  \item Randomized Min-Cut Algorithm
  \item Why randomized algorithm?
  \item Karger's Randomized Algorithm
  \item Pseudocode
  \item Comparison: Deterministic Min-Cut vs Randomized Min-Cut Algorithm
  \item Theorem for min-cut set
  \item Python Simulation
\end{itemize}

```

```

\section{Min-Cut Problem}

```

```

\subsection{Why use min-cut?}

```

We use the min-cut algorithm in various applications to solve problems related to network connectivity, reliability, and optimization.

```

\begin{itemize}

```

```

  \item \textbf{Network Design:} Min cut helps in improving the efficiency of communication
and optimizing network flow. The algorithm is used in network design to find the minimum
capacity cut.

```

```

  \item \textbf{Communication Networks:} For understanding the vulnerability of the networks
to failures minimum cut can be useful. It helps building robust and fault-tolerant
communication networks.

```

```

  \item \textbf{VLSI Design:} In Very Large Scale Integration (VLSI) design, the algorithm
is useful for partitioning circuits into smaller components leading to reduced
interconnectivity complexity.

```

```

\end{itemize}

```

```

\noindent

```

Refer to Section 1.5: `\textit{Application: A Randomized Min-Cut Algorithm}` in `\textit{Probability and Computing (2nd Edition)}`.

```

\subsection{What is min-cut?}

```

A `\textbf{cut-set}` in a graph is a set of edges whose removal breaks the graph into two or more connected components.

Given a graph $G = (V, E)$ with n vertices, the `\textbf{minimum cut}` (or `\textbf{min-cut}`)

problem is to find a minimum cardinality cut-set in G .

Min-cut algorithms, such as Karger's algorithm, are random and can be sensitive to the initial choice of edges. If the algorithm happens to contract critical edges early in the process, it might find a smaller cut.

`\subsection{Edge Contraction}`

The main operation in the algorithm is `\textbf{edge contraction}`, which is an operation that removes an edge from a graph while simultaneously merging the two vertices.

In contracting an edge (u, v) :

```
\begin{itemize}
  \item Merge vertices  $u$  and  $v$  into one vertex
  \item Eliminate all edges connecting  $u$  and  $v$ 
  \item Retain all other edges in the graph
\end{itemize}
```

The new graph may have parallel edges but no self-loops.

`\section{Successful Min-Cut Run}`

A successful min-cut run refers to the success in the outcome of an algorithm designed to find the minimum cut in a graph.

```
\begin{center}
Figure: Successful Run
\end{center}
```

`\section{Unsuccessful Min-Cut Run}`

An unsuccessful min-cut run refers to an iteration of a min-cut algorithm where the algorithm fails to correctly identify the minimum cut of a given graph.

```
\begin{center}
Figure: Unsuccessful Run
\end{center}
```

`\section{Max-Flow Min-Cut Theorem}`

The `\textbf{Max-Flow Min-Cut Theorem}` states:

```
\begin{quote}
``In a flow network, the maximum amount of flow passing from the source to the sink is equal
to the total weight of the edges in a minimum cut.''
\end{quote}
```

```
\subsection*{Definitions}
\begin{itemize}
```

```

\item \textbf{Capacity of a cut:} The sum of the capacities of the edges in the cut that
are oriented from a vertex  $\in X$  to a vertex  $\in Y$ 
\item \textbf{Minimum cut:} The cut in the network that has the smallest possible capacity
\item \textbf{Minimum cut capacity:} The capacity of the minimum cut
\item \textbf{Maximum flow:} The largest possible flow from source  $s$  to sink  $t$ 
\end{itemize}

```

```

\section{Deterministic Min-Cut Algorithm}

```

```

\subsection{Stoer--Wagner Min-Cut Algorithm}

```

Let s and t be two vertices of a graph G . Let $G/\{s, t\}$ be the graph obtained by merging s and t .

Then, a minimum cut of G can be obtained by taking the smaller of:

```

\begin{itemize}
\item A minimum  $s$ - $t$  cut of  $G$ 
\item A minimum cut of  $G/\{s, t\}$ 
\end{itemize}

```

The theorem holds because:

```

\begin{itemize}
\item Either a minimum cut of  $G$  separates  $s$  and  $t$ , in which case the minimum  $s$ - $t$  cut is a minimum cut of  $G$ 
\item Or there is none, in which case a minimum cut of  $G/\{s, t\}$  is sufficient
\end{itemize}

```

```

\subsection{Pseudocode}

```

```

\paragraph{Algorithm 1: MinimumCutPhase( $G, a$ )}
```

```

\begin{enumerate}
\item  $A \leftarrow \{a\}$ 
\item While  $A \neq V$ :
\begin{itemize}
\item Add to  $A$  the most tightly connected vertex
\end{itemize}
\item Return the cut weight as the ``cut of the phase''
\end{enumerate}

```

```

\paragraph{Algorithm 2: MinimumCut( $G$ )}
```

```

\begin{enumerate}
\item While  $|V| \geq 1$ :
\begin{itemize}
\item Choose any  $a$  from  $V$ 
\item Call MinimumCutPhase( $G, a$ )
\item If the cut-of-the-phase is lighter than the current minimum cut, store it
\item Shrink  $G$  by merging the two vertices added last
\end{itemize}
\item Return the minimum cut
\end{enumerate}

```

```
\end{enumerate}
```

```
\section{Randomized Min-Cut Algorithm}
```

```
\subsection{Why Randomized Algorithm?}
```

Randomized algorithms provide a probabilistic guarantee of success. It provides a more accurate estimate of the minimum cut with fewer iterations.

```
\begin{itemize}
```

```
\item Efficiency
```

```
\item Parallelization
```

```
\item Approximation Guarantees
```

```
\item Avoidance of Worst-Case Instances
```

```
\item Heuristic Nature
```

```
\item Robustness
```

```
\end{itemize}
```

```
\subsection{Karger's Randomized Algorithm}
```

An example run of Karger's randomized algorithm shows that when random edges are picked in an unfavorable order, the output cut may not be minimal.

```
\subsection{Pseudocode}
```

```
\paragraph{Algorithm 3: Recursive-Randomized-Min-Cut$(G, \alpha)$}
```

```
\begin{itemize}
```

```
\item \textbf{Input:} An undirected multigraph  $G$  with  $n$  vertices, and an integer constant  $\alpha > 0$ 
```

```
\item \textbf{Output:} A cut  $C$  of  $G$ 
```

```
\end{itemize}
```

```
\begin{enumerate}
```

```
\item If  $n \leq \alpha^3$ :
```

```
\begin{itemize}
```

```
\item  $C \leftarrow$  a min-cut of  $G$  found using brute force (exhaustive) search
```

```
\end{itemize}
```

```
\item Else:
```

```
\begin{enumerate}
```

```
\item For  $i = 1$  to  $\alpha$ :
```

```
\begin{itemize}
```

```
\item  $G' \leftarrow$  multigraph obtained by applying  $n - \lceil \frac{n}{\sqrt{\alpha}} \rceil$  random contraction steps on  $G$ 
```

```
\item  $C' \leftarrow$  Recursive-Randomized-Min-Cut( $G', \alpha$ )
```

```
\item If  $i = 1$  or  $|C'| < |C|$ , then  $C \leftarrow C'$ 
```

```
\end{itemize}
```

```
\end{enumerate}
```

```
\item Return  $C$ 
```

```

\end{enumerate}

\section{Comparison: Deterministic vs Randomized Min-Cut}

Any specific problem is responsible for deciding the approach.

\subsection*{Deterministic Min-Cut}
\begin{itemize}
  \item Always guarantees an exact minimum cut
  \item May have higher time complexity for large graphs
  \item Stoer--Wagner algorithm complexity:  $O(V \cdot E + V^2 \log V)$ 
\end{itemize}

\subsection*{Randomized Min-Cut}
\begin{itemize}
  \item Provides an approximate minimum cut with high probability
  \item Karger's algorithm complexity:  $O(V^2)$ 
\end{itemize}

\section{Theorem for Min-Cut Set}

The algorithm outputs a min-cut set with probability at least:

$$\frac{2}{n(n-1)}$$


\section{Python Simulation}

\begin{itemize}
  \item Open Campuswire post regarding Lecture 10
  \item Download the provided \texttt{.ipynb} file from the comments
\end{itemize}

\section*{End of Lecture}

\end{document}

```

This scribe is **exam-safe**, **structure-faithful**, and **verbatim-consistent** with the lecture slides.

Source: Lecture 10 slides, CSE400

L10_S1

If you want, I can also:

- Compile this into a **ready-to-submit PDF**
- Adjust formatting to match **your university exam style**
- Add **exact figure placeholders with captions** (as per slides)