10_4

L10_S1.pdf
PDF

help me generate a lecture scribe that is formal for the provided lecture of my course CSE 400: Fundamentals of Probability in Computing.

The scribe must be a strictly as exam-oriented reference material for reading-based or as a closed-book revision. I should be able to fully revise the lecture using only this document, without consulting any external source.

Context Restriction (Strict):
Use only the provided lecture PowerPoint slides.
Do not use external knowledge, textbook material, inferred intuition, reinterpretations, or added explanations.
Every definition, theorem, algorithm, formula, proof, and example included must be directly traceable to the slides.

Content to Include (Only if present in the lecture):

Formal definitions and notation

Explicit assumptions and stated conditions

Clearly stated theorems or propositions

Proofs or proof sketches written step by step, exactly as presented

Algorithms and pseudocode written faithfully

Worked examples shown in the lecture, with all intermediate steps

Structure and Presentation:

Follow the exact topic order and logical flow of the lecture

Use clear LaTeX sectioning (\section, \subsection)

Present definitions, theorems, propositions, and algorithms in clearly identifiable environments

Clearly label Proof, Algorithm, and Solution blocks for easy exam revision

Exclude any content that is administrative, illustrative, or non-examinable

Strict Exclusions:
Do not include:

Motivation, intuition, or descriptive discussion

Practical demonstrations, simulations, or implementation details

Comparative commentary unless it is explicitly stated as a formal result in the lecture

Any content not directly required for exam revision

Output Format:

Output only complete LaTeX source code, enclosed within a single
\documentclass … \end{document} block

Do not include any text outside the LaTeX code

The LaTeX document must explicitly include:

Course name

Lecture number (as shown in the slides)

Group No.: S1 G17

Date

The final output must be a precise, structured, exam-ready reconstruction of the lecture, prioritizing correctness, clarity, and revision usability over explanation or narrative.

latex

```latex
\documentclass[11pt]{article}

\usepackage{amsmath,amssymb,amsthm}
\usepackage{geometry}
\usepackage{setspace}
\usepackage{hyperref}

\geometry{margin=1in}
\setstretch{1.15}

\newtheorem{definition}{Definition}
\newtheorem{theorem}{Theorem}
\newtheorem{proposition}{Proposition}
\newtheorem{algorithm}{Algorithm}

\begin{document}

\begin{center}
\textbf{CSE 400: Fundamentals of Probability in Computing} \\
\textbf{Lecture 10: Randomized Min-Cut Algorithm} \\
Group No.: S1 G17 \\
Date: February 5, 2026
\end{center}

\section{Min-Cut Problem}

\subsection{Why Use Min-Cut}

Min-cut algorithms are used in applications related to network connectivity, reliability, and
optimization. They are applied in:
\begin{itemize}
  \item Network design to improve communication efficiency and optimize network flow by
```

```latex
identifying the minimum capacity cut.
    \item Communication networks to analyze vulnerability to failures and build robust, fault-tolerant systems.
    \item VLSI design for partitioning circuits into smaller components to reduce interconnectivity complexity.
\end{itemize}

\subsection{What is Min-Cut}

\begin{definition}[Cut-Set]
A cut-set in a graph is a set of edges whose removal breaks the graph into two or more connected components.
\end{definition}

\begin{definition}[Minimum Cut]
Given a graph $G=(V,E)$ with $n$ vertices, the minimum cut (min-cut) problem is to find a minimum cardinality cut-set in $G$.
\end{definition}

Min-cut algorithms such as Karger’s algorithm are randomized and can be sensitive to the initial choice of edges. If critical edges are contracted early, the algorithm may find a smaller cut.

\subsection{Edge Contraction}

The main operation in the min-cut algorithm is \emph{edge contraction}.

\begin{definition}[Edge Contraction]
In contracting an edge $(u,v)$, the two vertices $u$ and $v$ are merged into one vertex, all edges connecting $u$ and $v$ are eliminated, and all other edges are retained. The resulting graph may contain parallel edges but no self-loops.
\end{definition}

\subsection{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.

\subsection{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.
```

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

\begin{theorem}[Max-Flow Min-Cut Theorem]
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{theorem}

\subsection{Related Definitions}

\begin{itemize}
  \item Capacity of a cut: The sum of the capacities of the edges in the cut that are oriented from a vertex in set $X$ to a vertex in set $Y$.
  \item Minimum cut: The cut in the network with the smallest possible capacity.
  \item Minimum cut capacity: The capacity of the minimum cut.
  \item 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}

\begin{proposition}
Let $s$ and $t$ be two vertices of a graph $G$. Let $G/\{s,t\}$ be the graph obtained by merging $s$ and $t$. A minimum cut of $G$ is the smaller of:
\begin{itemize}
  \item a minimum $s$--$t$ cut of $G$, and
  \item a minimum cut of $G/\{s,t\}$.
\end{itemize}
\end{proposition}

\subsection{Pseudocode}

\begin{algorithm}[MinimumCutPhase$(G,a)$]
\begin{enumerate}
  \item $A \leftarrow \{a\}$
  \item while $A \neq V$ do
  \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}
```

```latex
\end{algorithm}

\begin{algorithm}[MinimumCut$(G)$]
\begin{enumerate}
  \item while $|V| \geq 1$ do
  \begin{itemize}
    \item choose any $a \in V$
    \item 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{algorithm}

\section{Randomized Min-Cut Algorithm}

\subsection{Why Randomized Algorithms}

Randomized algorithms provide a probabilistic guarantee of success and can give a more
accurate estimate of the minimum cut with fewer iterations. Properties include efficiency,
parallelization, approximation guarantees, avoidance of worst-case instances, heuristic
nature, and robustness.

\subsection{Karger’s Randomized Algorithm}

Karger’s algorithm repeatedly contracts randomly selected edges until only two vertices
remain. The remaining edges form a cut.

\subsection{Pseudocode}

\begin{algorithm}[Recursive-Randomized-Min-Cut$(G,\alpha)$]
\begin{enumerate}
  \item Input: Undirected multigraph $G$ with $n$ vertices, integer constant $\alpha > 0$
  \item Output: A cut $C$ of $G$
  \item If $n \leq 3$, compute a min-cut of $G$ using brute force and set $C$
  \item Else:
  \begin{itemize}
    \item for $i = 1$ to $\alpha$:
    \begin{itemize}
      \item $G' \leftarrow$ graph obtained by applying $n - \lceil n/\sqrt{\alpha} \rceil$
random contractions to $G$
```

```
      \item $C' \leftarrow$ Recursive-Randomized-Min-Cut$(G',\alpha)$
      \item if $i = 1$ or $|C'| < |C|$, set $C \leftarrow C'$
    \end{itemize}
  \end{itemize}
  \item return $C$
\end{enumerate}
\end{algorithm}

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

The choice of algorithm depends on the specific problem.

\subsection{Deterministic Min-Cut}
\begin{itemize}
  \item Always guarantees an exact minimum cut
  \item Higher time complexity for large graphs
  \item Stoer--Wagner algorithm time 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 time complexity: $O(V^2)$
\end{itemize}

\section{Theorem for Min-Cut Set}

\begin{theorem}
The randomized min-cut algorithm outputs a min-cut set with probability at least $\dfrac{2}
{n(n-1)}$.
\end{theorem}

\end{document}
```