

Lecture 5: Algorithms in Retrieval-Augmented Generation (RAG)

Instructors: *Alex Andoni, Edo Liberty*Lecturer: *Edo Liberty*

Lecture Overview & Pedagogical Roadmap

Modern Large Language Models (LLMs) are trained on trillions of tokens, yet their parametric weights $\theta \in \mathbb{R}^d$ represent a static, lossy compression of the world up to a fixed pre-training cutoff. When deployed on tasks requiring private enterprise data, real-time events, verifiable citations, or long-tail factual precision, pure parametric generation falls short.

1. **Part I (Foundations & System Economics):** How LLMs access post-training knowledge, why *retrieval for LLMs* fundamentally differs from *retrieval for humans* (set selection vs. ranked lists, order-agnosticism, token budgets), and how RAG generalizes to multi-step agentic search.
2. **Part II (Information Maximization & Subset Selection):** Why selecting k chunks under a context window budget is governed by marginal information gain, the mathematical duality between *submodular information coverage* and *supermodular synergy/redundancy*, and a complete proof of the $(1 - 1/e)$ greedy approximation guarantee.
3. **Part III (Aggregating Inconsistent Preferences & Reranking):** How to rank candidate passages when pairwise LLM judges or heterogeneous retrievers produce noisy, non-transitive (cyclic) preferences. We present and prove the randomized pivot approximation algorithm of **Ailon, Charikar, and Newman (STOC 2005)**.

1 Context, Search, Agentic Access, and RAG

1.1 Parametric vs. Non-Parametric Memory

- **Parametric Knowledge (θ):** Information stored implicitly within the MLP weights and attention projections during pre-training and post-training. Accessing parametric memory requires $\mathcal{O}(1)$ context tokens per generated token, but updating θ requires gradient-based optimization, incurs catastrophic forgetting risks, and provides no provenance guarantees (hallucination).
- **Non-Parametric Knowledge ($\mathcal{D}$):** Information stored explicitly in an external corpus $\mathcal{D} = \{d_1, \dots, d_N\}$ (documents, databases, codebases, web pages) and injected into the active **context window**.

1.2 Taxonomy of Accessing Information Unavailable at Training Time

When an LLM must answer a query x whose ground truth depends on information absent from pre-training, four primary paradigms exist, summarized in Table 1.

Paradigm	Mathematical Mechanism	Update Cost	Inference Token Cost	Failure Modes & Trade-offs
1. Weight Updates (Continual Pre-training / SFT / LoRA)	$\theta' = \arg \min_{\theta} \mathcal{L}(\theta; \mathcal{D}_{\text{new}})$	High (GPU hours, gradient steps)	Zero context overhead ($\mathcal{O}(x)$)	Poor factual memorization of long-tail facts; catastrophic forgetting; stale immediately after training.
2. Long-Context Stuffing (In-Context Learning)	Condition on full corpus: $p_{\theta}(y \mid x, \mathcal{D})$	Zero	$\mathcal{O}(\mathcal{D})$ tokens per query	Prefill attention scales as $\mathcal{O}(L^2)$; KV-cache memory scales as $\mathcal{O}(L)$; high monetary cost & TTFT latency; attention dilution (<i>“lost in the middle”</i>).
3. Static / Pipeline RAG (Retrieve-then-Read)	Select subset $S \subseteq \mathcal{D}$, $ S \leq k$; generate $p_{\theta}(y \mid x, S)$	Low (re-index embeddings $\mathcal{O}(\mathcal{D}_{\text{new}})$)	$\mathcal{O}(\sum_{d \in S} d) \ll \mathcal{D} $	Retriever recall ceiling: if relevant evidence is missed in single-shot retrieval, generator cannot recover.
4. Agentic Access (Iterative Search / Tool Use)	Multi-step policy $\pi_{\theta}(a_t \mid x, h_t)$ over search/SQL/API tools	Low / Zero (live index or API)	Variable ($\sum_{t=1}^T o_t $ across T turns)	Higher multi-turn latency and token expenditure, but can inspect intermediate observations, reformulate queries, and resolve multi-hop dependencies.

Table 1: Taxonomy of paradigms for accessing post-training knowledge in Large Language Models.

1.3 Traditional Human Search vs. RAG Retrieval

A foundational mistake in early RAG systems was directly reusing classical Information Retrieval (IR) pipelines designed for human web search (e.g., standard BM25 or top- k cosine similarity without diversification). Classical IR and RAG differ across four fundamental axes:

1. The Probability Ranking Principle vs. Joint Set Utility. Classical IR is built on Robertson’s **Probability Ranking Principle (PRP, 1977)**: if documents are examined sequentially and independently by a human user, sorting documents in decreasing order of individual probability of relevance $P(\text{rel} \mid q, d)$ maximizes expected utility under standard metrics (Precision@ k , MRR, NDCG).

In RAG, **the consumer is a self-attention network that reads the entire retrieved set $S = \{d_1, \dots, d_k\}$ simultaneously**. If $d_1, d_2, \dots, d_5$ are five near-identical paraphrases of the same Wikipedia paragraph, a human search engine achieves Precision@5 = 1.0. For an LLM, chunks $d_2, \dots, d_5$ contribute **zero marginal information** while consuming 80% of the context budget and crowding out complementary evidence needed to answer the query.

2. Order-Agnosticism (and Positional Attention Biases).

- **Human Search is Strictly Order-Sensitive:** Click-through rates decay exponentially with rank (Position 1 receives $\sim 30\text{--}40\%$ of clicks; Position 10 receives $< 2\%$).

- **LLM Generation is Approximately Order-Agnostic:** Mathematically, self-attention without causal masking across prompt segments is equivariant to permutation of document blocks, and even with causal masking and Rotary Position Embeddings (RoPE), modern LLMs can attend to relevant facts anywhere in the retrieved context window.
- **Caveat — The “Lost in the Middle” Effect:** Empirically (Liu et al., 2024), causal attention exhibits a U-shaped positional bias: tokens at the very beginning (primacy bias / attention sinks) and very end (recency bias, adjacent to the generation cursor) receive higher effective attention weights than tokens buried in the middle. Nevertheless, whether a document is placed at rank #2 vs. rank #4 matters vastly less to an LLM than **whether the document made the top- k cutoff at all**.
- **Algorithmic Consequence:** Stage-1 retrieval in RAG is primarily a **subset selection problem** (maximizing total information content of the unordered set S), whereas Stage-2 reranking determines which elements survive a tighter token budget $k' < k$ and places the highest-confidence evidence at the boundaries of the prompt.

3. Query Formulation & Generality.

- **Human Queries:** Short, telegraphic keyword bags (e.g., "coms e6998 columbia syllabus"), typically 2–5 tokens long, optimized for lexical matching.
- **RAG & Agentic Queries:**
 - Can be full conversational histories or rich semantic embeddings.
 - Can use **Hypothetical Document Embeddings (HyDE)** (Gao et al., 2023): the LLM first generates a hypothetical (possibly hallucinated) answer passage $\hat{d} \sim p_\theta(\cdot | x)$, and embeds $\hat{d}$ to search for real documents in document-embedding space.
 - In **agentic search**, queries are often intentionally *more general or exploratory* in early turns (e.g., retrieving a high-level directory, schema, or table of contents) before issuing targeted follow-up queries.

4. Token Costs, Prefill Latency, and Knapsack Constraints. In classical IR, the “cost” of showing a result snippet is constant visual screen real estate (10 blue links). In RAG, every retrieved document $d \in \mathcal{D}$ has an explicit **token cost** $c(d) = |\text{tokenize}(d)| \in \mathbb{N}$:

- **Monetary Cost:** API pricing scales directly with input token count $\sum_{d \in S} c(d)$.
- **Compute & Latency (TTFT):** Transformer prefill FLOPs scale as $\mathcal{O}\left(\left(\sum_{d \in S} c(d)\right)^2\right)$, and KV-cache memory scales linearly with token count.
- Therefore, RAG retrieval is naturally constrained by a **token budget** $\sum_{d \in S} c(d) \leq B$ rather than a naive document count $|S| \leq k$.

2 Maximizing Information as Submodular Optimization

2.1 Formulating Retrieval as Combinatorial Subset Selection

Let $\mathcal{U} = \{1, 2, \dots, n\}$ be the ground set of candidate document chunks (e.g., the top $n = 200$ candidates returned by a fast vector/lexical index). We wish to select a subset $S \subseteq \mathcal{U}$ that maximizes an **information utility function** $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}_{\geq 0}$ subject to either:

- a **cardinality constraint**: $|S| \leq k$, or
- a **token budget (knapsack) constraint**: $c(S) = \sum_{e \in S} c(e) \leq B$.

We define the discrete derivative, or **marginal gain**, of adding element $e \in \mathcal{U}$ to an existing context set $S \subseteq \mathcal{U}$ as:

$$\Delta_f(e \mid S) \triangleq f(S \cup \{e\}) - f(S). \quad (1)$$

A set function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}$ is called:

1. **Normalized**: $f(\emptyset) = 0$.
2. **Monotone (Non-decreasing)**: For all $S \subseteq T \subseteq \mathcal{U}$, $f(S) \leq f(T)$ (equivalently, $\Delta_f(e \mid S) \geq 0$ for all S, e).

2.2 Submodularity in RAG

Suppose each chunk $e \in \mathcal{U}$ covers a set of atomic facts or query sub-aspects $\text{facts}(e) \subseteq \mathcal{F}$. The set coverage function

$$f_{\text{cov}}(S) = \left| \bigcup_{e \in S} \text{facts}(e) \right| \quad (2)$$

is monotone submodular.

Definition 2.1 (Submodularity — Diminishing Marginal Returns). *A set function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}$ is **submodular** if for all subsets $S \subseteq T \subseteq \mathcal{U}$ and every element $e \in \mathcal{U} \setminus T$:*

$$\Delta_f(e \mid S) \geq \Delta_f(e \mid T), \quad \text{i.e.,} \quad f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T). \quad (3)$$

Equivalently, for all $A, B \subseteq \mathcal{U}$:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B). \quad (4)$$

Facility Location (Embedding Space Coverage). Given query/topic embeddings V and chunk embeddings S , the facility location objective

$$f_{\text{fac}}(S) = \sum_{v \in V} \max_{e \in S} \text{sim}(v, e) \quad (5)$$

is also monotone submodular.

2.3 The Greedy Algorithm and the $(1 - 1/e)$ Approximation Proof

Consider the canonical problem of maximizing a normalized, monotone, submodular information function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}_{\geq 0}$ subject to a cardinality constraint $|S| \leq k$:

$$\max_{S \subseteq \mathcal{U}, |S| \leq k} f(S). \quad (6)$$

Algorithm GREEDY-SELECT($f, \mathcal{U}, k$)

1. Initialize $S_0 \leftarrow \emptyset$.

2. **For** $i = 1, 2, \dots, k$:

$$e_i \leftarrow \arg \max_{e \in \mathcal{U} \setminus S_{i-1}} (f(S_{i-1} \cup \{e\}) - f(S_{i-1})), \quad S_i \leftarrow S_{i-1} \cup \{e_i\}.$$

3. **Return** S_k .

Theorem 2.2 (Nemhauser, Wolsey, & Fisher, 1978). *For any normalized ($f(\emptyset) = 0$), monotone, submodular function $f : 2^{\mathcal{U}} \rightarrow \mathbb{R}_{\geq 0}$, the set S_k returned by **GREEDY-SELECT** satisfies:*

$$f(S_k) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) f(S^*) \geq \left(1 - \frac{1}{e}\right) f(S^*) \approx 0.632 f(S^*), \quad (7)$$

where $S^* = \arg \max_{|S| \leq k} f(S)$ is an optimal subset of size at most k .

Proof. Let $S^* = \{o_1, o_2, \dots, o_k\}$ be an optimal solution of size k (by monotonicity, we may assume $|S^*| = k$). Fix any iteration $i \in \{0, 1, \dots, k-1\}$, and let S_i be the greedy set after i steps.

Step 1 (Monotonicity). Since f is monotone and $S_i \subseteq S_i \cup S^*$,

$$f(S^*) \leq f(S_i \cup S^*).$$

Step 2 (Telescoping Sum). Write $f(S_i \cup S^*) - f(S_i)$ as a telescoping sum by adding the elements of $S^* = \{o_1, \dots, o_k\}$ one by one:

$$f(S_i \cup S^*) - f(S_i) = \sum_{j=1}^k \left(f(S_i \cup \{o_1, \dots, o_j\}) - f(S_i \cup \{o_1, \dots, o_{j-1}\}) \right) = \sum_{j=1}^k \Delta_f(o_j \mid S_i \cup \{o_1, \dots, o_{j-1}\}).$$

Step 3 (Submodularity / Diminishing Returns). Because $S_i \subseteq S_i \cup \{o_1, \dots, o_{j-1}\}$ for every $j \in \{1, \dots, k\}$, submodularity gives:

$$\Delta_f(o_j \mid S_i \cup \{o_1, \dots, o_{j-1}\}) \leq \Delta_f(o_j \mid S_i).$$

Summing over $j = 1, \dots, k$ and combining with Steps 1 and 2:

$$\begin{aligned} f(S^*) - f(S_i) &\leq f(S_i \cup S^*) - f(S_i) \leq \sum_{j=1}^k \Delta_f(o_j \mid S_i) \\ &\leq k \max_{1 \leq j \leq k} \Delta_f(o_j \mid S_i) \leq k \max_{e \in \mathcal{U} \setminus S_i} \Delta_f(e \mid S_i) = k(f(S_{i+1}) - f(S_i)). \end{aligned}$$

Step 4 (One-Step Progress Inequality). Rearranging gives the fundamental one-step progress inequality:

$$f(S_{i+1}) - f(S_i) \geq \frac{1}{k} (f(S^*) - f(S_i)). \quad (8)$$

Step 5 (Unrolling the Recurrence). Let $\delta_i \triangleq f(S^*) - f(S_i)$ be the optimality gap after step i . Subtracting both sides of (8) from $f(S^*) - f(S_i)$ yields:

$$\delta_{i+1} = f(S^*) - f(S_{i+1}) = (f(S^*) - f(S_i)) - (f(S_{i+1}) - f(S_i)) \leq \left(1 - \frac{1}{k}\right) \delta_i.$$

By induction from $i = 0$ to k , using $\delta_0 = f(S^*) - f(\emptyset) = f(S^*)$:

$$\delta_k = f(S^*) - f(S_k) \leq \left(1 - \frac{1}{k}\right)^k f(S^*).$$

Finally, using the standard analytic inequality $1 - x \leq e^{-x}$ for $x = 1/k$:

$$\left(1 - \frac{1}{k}\right)^k \leq \left(e^{-1/k}\right)^k = \frac{1}{e},$$

which yields

$$f(S_k) \geq \left(1 - \left(1 - \frac{1}{k}\right)^k\right) f(S^*) \geq \left(1 - \frac{1}{e}\right) f(S^*). \quad \square$$

Remark 2.3 (Tightness & Token Budget Extension).

1. **Optimality:** Feige (1998) proved that no polynomial-time algorithm can achieve a $(1 - 1/e + \varepsilon)$ -approximation for Max- k -Cover unless $P = NP$.
2. **Variable Token Costs (Knapsack Constraint):** When chunks have non-uniform token lengths $c(e) \in \mathbb{N}$ and total budget $\sum_{e \in S} c(e) \leq B$, the density-greedy rule $e_{i+1} = \arg \max_e \frac{\Delta_f(e \mid S_i)}{c(e)}$ alone can be arbitrarily bad (e.g., picking one tiny 1-token chunk with value 2ε that blocks a B -token chunk with value B). However, **taking the better of** (i) the best single element $\max_{e: c(e) \leq B} f(\{e\})$ and (ii) the density-greedy solution achieves a provable $\frac{1}{2}(1 - 1/e)$ -approximation (and partial enumeration of 3 elements plus density greedy restores the full $(1 - 1/e)$ -approximation; Sviridenko, 2004).

3 Efficient Ranking with Inconsistent Information

3.1 Pointwise vs. Pairwise LLM Reranking

Once a candidate subset $V = \{1, \dots, n\}$ is retrieved, we often wish to rank the n candidates by relevance to the query q .

A natural approach is **Pointwise Scoring**: ask an LLM to independently assign a scalar relevance score $s(i) \in \mathbb{R}$ to each passage $i \in V$ in $\mathcal{O}(n)$ calls and sort by $s(i)$. However, LLMs are notoriously poorly calibrated across independent calls. Scores saturate, drift with passage length or style, and fail to capture fine-grained relative superiority without attending to both passages simultaneously.

Pairwise LLM Reranking (e.g., Qin et al., 2024) is much more robust to these effects. We prompt an LLM judge with a pair of passages (i, j) :

“Given query q , which passage provides better evidence: i or j ?”

These comparator models output a single relative preference $i \succ j$ or $j \succ i$. They are far more robust to calibration drift and can be trained efficiently using contrastive learning.

The Fundamental Algorithmic Obstacle is Intransitivity. Because an LLM judge attends to different features depending on the pair being compared, its pairwise preferences are **not transitive** (or at least, are not guaranteed to be). We frequently encounter directed cycles:

$$i \succ j, \quad j \succ k, \quad \text{and} \quad k \succ i.$$

3.2 Formal Problem Definition: Minimum Feedback Arc Set in Tournaments

We model the pairwise comparison outcomes on $V = \{1, \dots, n\}$ as a **tournament**.

Definition 3.1 (Tournament & Minimum Feedback Arc Set — FAS-TOURNAMENT).

- A **tournament** $G = (V, A)$ is a directed graph on $n = |V|$ vertices such that for every distinct pair $i, j \in V$, **either** $(i, j) \in A$ (meaning i beats j , written $i \succ j$) **or** $(j, i) \in A$ (meaning j beats i), with $(i, j) \in A \iff (j, i) \notin A$.
- Given a linear ordering (permutation) π of V , we write $i <_\pi j$ if π ranks i ahead of (to the left of) j . Think of π as ordering the elements in descending order of relevance/utility.
- A directed edge $e = (j, i) \in A$ is called a **backward edge** (or **feedback arc** / **upset**) with respect to π if $i <_\pi j$ (i is ranked ahead of j , even though j beat i in the pairwise comparison).
- The **FAS-TOURNAMENT problem** asks for a permutation π of V that minimizes the total number of backward edges:

$$\text{cost}_G(\pi) = |\{(i, j) \in V \times V : i <_\pi j \text{ and } (j, i) \in A\}|. \quad (9)$$

We denote the minimum possible cost by $C^{\text{OPT}} \triangleq \min_\pi \text{cost}_G(\pi)$ and an optimal permutation by π^* .

- **If the LLM comparator is perfectly consistent (transitive):** G is a DAG, and there is a unique topological sort π^* with $C^{\text{OPT}} = 0$.

- **In general:** Computing C^{OPT} exactly is **NP-hard** (conjectured by Bang-Jensen & Thomassen, 1992; proved via randomized reduction by Ailon, Charikar, & Newman, 2005 and derandomized by Alon, 2006).

Naive Solution 1: Compare All Pairs and Sort by Number of “Wins” (Copeland’s Method)

The most intuitive heuristic is to run a full round-robin tournament: query the LLM on **all** $\binom{n}{2}$ pairs, count the number of “wins” (out-degree) for each candidate

$$d^+(i) \triangleq |\{j \in V \setminus \{i\} : (i, j) \in A\}|,$$

and sort the vertices in decreasing order of $d^+(i)$ (Copeland’s method).

- **Copeland’s method gives a 5-approximation:** A celebrated theorem of **Coppersmith, Fleischer, and Rudra (SODA 2006)** proves that sorting by number of wins is a **5-approximation** for FAS-TOURNAMENT (and this factor of 5 is tight when ties are broken arbitrarily).
- **A Simple $3\times$ Backward-Arcs Example:** Suppose the true underlying ranking is $1 < 2 < \dots < n$, where $i \rightarrow j$ for all $i < j$, *except for a single noisy LLM comparison* where the worst candidate n upsets the best candidate 1 (so $(n, 1) \in A$, giving $C^{\text{OPT}} = 1$). In the uncorrupted transitive tournament, the win counts are $(n-1, n-2, \dots, 1, 0)$. Flipping the single edge $(1, n)$ to $(n, 1)$ reduces $d^+(1)$ from $n-1$ to $n-2$ (tying vertex 1 with vertex 2) and raises $d^+(n)$ from 0 to 1 (tying vertex n with vertex $n-1$). Breaking those two ties adversely yields the ordering $(2, 1, 3, 4, \dots, n, n-1)$, which incurs **3 backward edges**—namely $(1, 2)$, $(n-1, n)$, and $(n, 1)$. Since OPT has a single backward arc and this solution has three, $\text{cost} = 3C^{\text{OPT}}$.
- **How expensive is it?** It requires querying all $\binom{n}{2} = \Theta(n^2)$ pairs. Because every edge query requires an expensive LLM inference call, $\Theta(n^2)$ complexity is prohibitive even for modest candidate sets (e.g., for $n = 200$ chunks, $\binom{200}{2} = 19,900 \approx 20,000$ LLM calls for a single user query!).

Naive Solution 2: Standard Deterministic Sorting (MergeSort or Fixed-Pivot QuickSort)

To avoid $\Theta(n^2)$ LLM calls, why not pass our pairwise LLM oracle directly as the comparator $\text{cmp}(i, j)$ into a standard $\mathcal{O}(n \log n)$ comparison sort such as **MergeSort** or deterministic **QuickSort** (e.g., always choosing the first element as pivot)?

- **Cost:** Only $\mathcal{O}(n \log n)$ LLM calls.
- **Approximation guarantee:** Can be arbitrarily bad—up to an $\mathcal{O}(n^2)$ approximation factor, as an adversary can choose the input tournament to maximize cascading errors of any fixed deterministic pivot schedule.

3.3 The FAS-PIVOT Algorithm

The key insight of Ailon, Charikar, and Newman (2005) is that **randomizing the pivot choice in QuickSort** completely eliminates this deterministic cascading failure mode: in the counterexample above where $(n, 1) \in A$ is the only corrupted edge, FAS-PIVOT only makes a mistake if either 1 or n is chosen as the pivot *before* any of the $n-2$ vertices in $\{2, \dots, n-1\}$ separates them.

Algorithm FAS-PIVOT($G = (V, A)$)

1. **If** $|V| \leq 1$, **return** V .
 2. Initialize $V_L \leftarrow \emptyset$, $V_R \leftarrow \emptyset$.
 3. Pick a pivot vertex $k \in V$ uniformly at random.
 4. **For all** vertices $j \in V \setminus \{k\}$:
 - Query the pairwise oracle on $\{j, k\}$:
 - **If** $(j, k) \in A$ **then** add j to V_L $(j \text{ beat } k \implies \text{place } j \text{ left of } k)$
 - **Else** $((k, j) \in A)$ add j to V_R $(k \text{ beat } j \implies \text{place } j \text{ right of } k)$
 5. Let $G_L = (V_L, A_L)$ and $G_R = (V_R, A_R)$ be the induced sub-tournaments.
 6. **Return** the concatenated ordering: **FAS-PIVOT**(G_L), k , **FAS-PIVOT**(G_R).
-

Oracle Query & Running Time Complexity. Because the pivot k is chosen uniformly at random from the current subproblem of size m , the recursion tree has expected depth $\mathcal{O}(\log n)$ and performs $\mathcal{O}(n \log n)$ expected pairwise comparisons—identical to standard Randomized QuickSort, even though the underlying comparator is non-transitive.

3.4 Complete Proof of the 3-Approximation Guarantee

Theorem 3.2 (Ailon, Charikar, & Newman, STOC 2005). *For any tournament $G = (V, A)$, let C^{PIV} be the number of backward edges in the ordering produced by **FAS-PIVOT**(G), and let C^{OPT} be the minimum number of backward edges over all permutations of V . Then:*

$$\mathbb{E}[C^{PIV}] \leq 3C^{OPT}. \quad (10)$$

Step 1: Structural Characterization — When Does an Edge Become Backward?

Lemma 3.3 (Backward Edges Only Arise from Directed Triangles). *An edge $e = (i, j) \in A$ (where $i \rightarrow j$) becomes a backward edge in **FAS-PIVOT** (i.e., j is placed to the left of i) **if and only if** there exists a third vertex $k \in V \setminus \{i, j\}$ such that:*

1. $t = (i, j, k)$ forms a **directed triangle** in G :

$$i \rightarrow j \rightarrow k \rightarrow i \quad (\text{i.e., } (i, j) \in A, (j, k) \in A, (k, i) \in A), \quad \text{and}$$

2. k is chosen as the pivot in a recursive call of **FAS-PIVOT** that contains all three vertices $\{i, j, k\}$.

Proof. Consider the unique recursive call of **FAS-PIVOT** in which i and j are separated into different subproblems (or one of them is chosen as pivot). Let k be the pivot chosen in this call:

- **Case A** ($k = i$ or $k = j$): No edge directly incident to the pivot k can ever become backward. Indeed, if $k = i$, then since $(i, j) \in A$, Step 4 places $j \in V_R$ (to the right of i), so $i \rightarrow j$ is oriented

forward. Similarly, if $k = j$, since $(i, j) \in A$, Step 4 places $i \in V_L$ (to the left of j), so $i \rightarrow j$ is again oriented forward.

- **Case B** ($k \notin \{i, j\}$): The vertices i and j are separated by pivot k in the wrong order ($j \in V_L$ to the left of k , and $i \in V_R$ to the right of k) **if and only if** j beats k ($(j, k) \in A$) and k beats i ($(k, i) \in A$). Together with $(i, j) \in A$, this means $i \rightarrow j \rightarrow k \rightarrow i$ is a directed triangle $t = (i, j, k)$, and k was chosen as pivot while $\{i, j, k\}$ were all present in the same recursive call. $\square$

Step 2: Charging Backward Edges to Directed Triangles

Let T denote the set of all directed 3-cycles (directed triangles) $t = (i, j, k)$ in G . For each directed triangle $t \in T$:

- Define A_t to be the event that **all three vertices of t reside in the same recursive subproblem at the moment the first vertex among the three is chosen as a pivot.**
- Let $p_t \triangleq \Pr[A_t]$ denote the probability of event A_t .

Observe what happens when event A_t occurs for triangle $t = (i \rightarrow j \rightarrow k \rightarrow i)$:

- If k is chosen as pivot, $j \in V_L$ and $i \in V_R$, so edge (i, j) becomes backward (and edges $(j, k), (k, i)$ incident to pivot k become forward).
- By cyclic symmetry, whichever of the three vertices $\{i, j, k\}$ is chosen as pivot causes **exactly one** of the three edges of t to become backward, while the other two edges become forward.
- Moreover, after this pivot step, the three vertices of t are split across $(V_L, \{k\}, V_R)$, so no other edge of t can ever be separated by a third vertex of t again.

We **charge** the unit cost of the backward edge $e = (i, j)$ to the unique directed triangle $t = (i, j, k)$ whose pivot k separated i and j . By Lemma 3.3, every backward edge in FAS-PIVOT is charged to a unique triangle $t \in T$ for which A_t occurred, and every triangle $t \in T$ for which A_t occurs is charged a cost of **exactly** 1. By linearity of expectation:

$$\mathbb{E}[C^{\text{PIV}}] = \sum_{t \in T} \Pr[A_t] = \sum_{t \in T} p_t. \quad (11)$$

Step 3: Mutual Disjointness & The Fractional Packing Inequality

For any directed edge $e = (i, j) \in A$, let B_e be the event that e becomes a backward edge in FAS-PIVOT. For any directed triangle $t = (i, j, k) \in T$ containing edge $e = (i, j)$, let $B_{e,t} \triangleq (B_e \wedge A_t)$ be the event that e becomes backward **and** is charged to triangle t (i.e., A_t occurs and k is chosen as pivot among $\{i, j, k\}$).

Conditioned on event A_t (all three vertices $\{i, j, k\}$ are present in the current subproblem, and the random pivot falls in $\{i, j, k\}$), each of the three vertices is equally likely to be the chosen pivot with conditional probability $\frac{1}{3}$. Since $e = (i, j)$ becomes backward iff k is the chosen pivot:

$$\Pr[B_{e,t}] = \Pr[B_e \wedge A_t] = \Pr[B_e \mid A_t] \cdot \Pr[A_t] = \frac{1}{3} p_t. \quad (12)$$

Crucial Disjointness Property. Fix any edge $e = (i, j) \in A$, and consider all directed triangles $t \in T$ that contain edge e . The events $\{B_{e,t} : t \in T, e \in t\}$ are **mutually disjoint**: an edge $e = (i, j)$ can be separated into V_L and V_R **at most once** during the entire execution of FAS-PIVOT. If $B_{e,t}$ occurs (pivot k separates i and j), then i and j are placed into disjoint recursive subproblems V_R and V_L , so no other third vertex k' from another triangle $t' = (i, j, k')$ can ever separate i and j .

Because the events $\{B_{e,t}\}_{t \ni e}$ are mutually exclusive and their union is precisely the event B_e that edge e becomes backward:

$$\Pr[B_e] = \sum_{t \in T: e \in t} \Pr[B_{e,t}] = \sum_{t \in T: e \in t} \frac{1}{3} p_t \leq 1 \quad \forall e \in A. \quad (13)$$

Step 4: Lower Bounding C^{OPT} via Triangle Packing / LP Duality

Now consider the optimal permutation π^* , and for each edge $e \in A$, let $x_e^* \in \{0, 1\}$ be the indicator that e is a backward edge in π^* , so that

$$C^{\text{OPT}} = \sum_{e \in A} x_e^*.$$

Because π^* defines a valid linear order (which is acyclic), π^* **cannot keep all three edges of any directed cycle $t = (e_1, e_2, e_3) \in T$ forward**. Therefore, every directed triangle $t \in T$ must contain **at least one** backward edge in π^* :

$$\sum_{e \in t} x_e^* = x_{e_1}^* + x_{e_2}^* + x_{e_3}^* \geq 1 \quad \forall t \in T. \quad (14)$$

Combining (11), (13), and (14) via a double-counting (LP duality) swap of summation:

$$\begin{aligned} C^{\text{OPT}} &= \sum_{e \in A} 1 \cdot x_e^* \\ &\geq \sum_{e \in A} \left(\sum_{t \in T: e \in t} \frac{1}{3} p_t \right) x_e^* && \text{(by (13): } \sum_{t \ni e} \frac{1}{3} p_t \leq 1 \text{ and } x_e^* \geq 0) \\ &= \sum_{t \in T} \frac{1}{3} p_t \left(\sum_{e \in t} x_e^* \right) && \text{(interchanging summation over pairs } (e, t) \text{ with } e \in t) \\ &\geq \sum_{t \in T} \frac{1}{3} p_t \cdot 1 && \text{(since } \sum_{e \in t} x_e^* \geq 1 \text{ for all } t \in T \text{ by (14))} \\ &= \frac{1}{3} \sum_{t \in T} p_t = \frac{1}{3} \mathbb{E}[C^{\text{PIV}}] && \text{(by (11)).} \end{aligned}$$

Multiplying both sides by 3 yields $\mathbb{E}[C^{\text{PIV}}] \leq 3 C^{\text{OPT}}$, completing the proof of Theorem 3.2. ■

Remark 3.4 (Why LP Duality Explains the Magic of FAS-PIVOT). If we knew the entire tournament G in advance and found a maximal collection $\mathcal{P} \subseteq T$ of **edge-disjoint directed triangles**, then trivially $C^{\text{OPT}} \geq |\mathcal{P}|$ (since each edge-disjoint triangle requires at least one edge reversal). More generally, any **fractional triangle packing** assigning weights $\beta_t \geq 0$ to triangles $t \in T$ such that $\sum_{t: e \in t} \beta_t \leq 1$ for every edge $e \in A$ is a dual feasible solution to the LP relaxation of FAS-TOURNAMENT, proving $C^{\text{OPT}} \geq \sum_{t \in T} \beta_t$. Equation (13) shows that the probabilities $\beta_t \triangleq \frac{1}{3} p_t$ induced by FAS-PIVOT automatically form a **valid**

fractional triangle packing whose value $\sum_{t \in T} \beta_t = \frac{1}{3} \mathbb{E}[C^{\text{PIV}}]$ is within a factor of 3 of the algorithm’s own expected cost—and **FAS-PIVOT** achieves this in $\mathcal{O}(n \log n)$ time **without ever explicitly searching for triangles or inspecting all $\binom{n}{2}$ edges**.

References

- [1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 684–693, 2005. Journal version in *Journal of the ACM (JACM)*, 55(5):1–27, 2008.
- [2] Noga Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006.
- [3] Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 776–782, 2006.
- [4] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.
- [5] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels (HyDE). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1762–1777, 2023.
- [6] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 12:157–173, 2024.
- [7] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1):265–294, 1978.
- [8] Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL*, pages 1504–1518, 2024.
- [9] Stephen E. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.
- [10] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.