COMS 5110 Final Exam Guide

Based on Confirmed Topics

Question 1: Amortized Analysis (Aggregate Method)

Topic: Amortized cost / Counting.
Source: HW3, Problem 1(a).
Problem: Consider a dynamic array where we perform n `Append` operations. If the array is full, we double its size and copy all elements. Use **Aggregate Analysis** to find the amortized cost per operation.

Solution

In Aggregate Analysis, we calculate the total cost $T(n)$ for the entire sequence of n operations and divide by n.

- **Insertion Cost:** Every operation performs exactly 1 insertion. Total $= n$.
- **Copying Cost:** Copying occurs only when size doubles (at sizes $1, 2, 4, \ldots, 2^k$).

$$\text{Total Copy Cost} = \sum_{i=0}^{\lfloor \log n \rfloor} 2^i = 1 + 2 + 4 + \cdots + n = 2n - 1$$

- **Total Cost:**

$$T(n) = (\text{Insertions}) + (\text{Copies}) = n + (2n - 1) \approx 3n = O(n)$$

- **Amortized Cost:**

$$\frac{T(n)}{n} \approx \frac{3n}{n} = O(1)$$

Question 2: Amortized Analysis (Accounting Method)

Topic: Amortized cost / Counting.
Source: HW3, Problem 1(b).
Problem: Use the **Accounting Method** to prove the amortized cost of `Append` is $O(1)$.

Solution

We assign a specific amortized cost (charge) $\hat{c}_i$ to each operation.

- **The Charge:** Set $\hat{c}_i = \$3$.
  - \$1 pays for the actual insertion.
  - \$2 is stored as **Credit** on the inserted element.
- **The Expensive Event:** When array doubles from n to $2n$, we must copy n existing elements.
- **The Payment:**
  - Actual cost to copy n items is \$n.
  - We have \$2 credit stored on each of the n items.
  - We consume \$1 from each item's credit to pay for the copy (leaving \$1 still in the bank).
- **Conclusion:** Since the accumulated credit is always sufficient to pay for doubling, the bank balance never drops below zero. Thus, amortized cost is $O(1)$.

Question 3: Dynamic Programming (Linear Space)

Topic: Linear space sequence alignment / S,D,I Equations.
Source: HW4.
Problem: Write the recurrence equations for Sequence Alignment with Affine Gap Penalties (S, D, I) and the pseudocode for Linear Space Alignment.

Part A: The Recurrence Equations

- $S[i, j]$: Optimal score ending with a **Match/Mismatch**.
- $D[i, j]$: Optimal score ending with a **Deletion** (gap in B).
- $I[i, j]$: Optimal score ending with an **Insertion** (gap in A).

$$D[i,j] = \max \begin{cases} D[i-1,j] - r & \text{(Extend existing deletion)} \\ S[i-1,j] - (q+r) & \text{(Start new deletion)} \end{cases}$$

$$I[i,j] = \max \begin{cases} I[i,j-1] - r & \text{(Extend existing insertion)} \\ S[i,j-1] - (q+r) & \text{(Start new insertion)} \end{cases}$$

$$S[i,j] = \max \begin{cases} S[i-1,j-1] + \sigma(a_i, b_j) & \text{(Match/Mismatch)} \\ D[i,j] & \text{(End a deletion)} \\ I[i,j] & \text{(End an insertion)} \end{cases}$$

Part B: Pseudocode (Midpoint Algorithm)

Goal: Find optimal alignment score in $O(mn)$ time but $O(m+n)$ space.

Algorithm 1 LinearSpaceAlignment(A, B)

$m \leftarrow \text{length}(A)$, $n \leftarrow \text{length}(B)$
if $m \leq 1$ **then return** Standard_Alignment(A, B)
end if

Step 1: Find Midpoint Split
$imid \leftarrow \lfloor m/2 \rfloor$
$(S_{mid}, D_{mid}) \leftarrow \text{ForwardScore}(A[0 \ldots imid], B)$
$(S_{rev}, D_{rev}) \leftarrow \text{BackwardScore}(A[imid \ldots m], B)$

Step 2: Find Best Split Column (*jmid*)
$Max \leftarrow -\infty$, $jmid \leftarrow -1$
for $j \leftarrow 0$ to n **do** ▷ Add q to deletion case to refund double-charged penalty
    $Score \leftarrow \max(S_{mid}[j] + S_{rev}[j], \quad D_{mid}[j] + D_{rev}[j] + q)$
    if $Score > Max$ **then**
        $Max \leftarrow Score$; $jmid \leftarrow j$
    end if
end for

Step 3: Recurse and Concatenate
$Left \leftarrow \text{LinearSpaceAlignment}(A[0 \ldots imid], B[0 \ldots jmid])$
$Right \leftarrow \text{LinearSpaceAlignment}(A[imid \ldots m], B[jmid \ldots n])$
return Concatenate$(Left, Right)$

Question 4: NP-Completeness (Membership + Reduction)

Topic: Standard NP Reduction.
Source: HW5, Problem 5.
Problem: Prove **INDEPENDENT-SET** is NP-Complete. Assume **CLIQUE** is NP-Complete.

Part A: Membership (In NP)

- **Certificate:** A subset of vertices S.
- **Verifier:** Check if $|S| = k$. Then iterate through all pairs $(u, v) \in S$ and verify $(u, v) \notin E$. Time is $O(k^2)$, which is polynomial.

Part B: Reduction (CLIQUE $\leq_p$ INDEPENDENT-SET)

- **Construction:** Given instance $\langle G, k \rangle$ for CLIQUE:
  - Construct the **Complement Graph** $\bar{G} = (V, \bar{E})$.
  - An edge $(u, v) \in \bar{E}$ exists if and only if $(u, v) \notin E$ (edge was missing in original).
- **Proof:**
  - If G has a **Clique** (all connected), those vertices in $\bar{G}$ have **no edges** between them.
  - Thus, they form an **Independent Set** in $\bar{G}$.

– G has Clique $\iff$ $\bar{G}$ has Independent Set.

Question 5: Approximation Algorithm (Cost Analysis)

Topic: Approx cost algo.
Source: HW5, Problem 4.
Problem: Describe the 2-approximation algorithm for **Metric TSP** and prove $c(H) \leq 2c(H^*)$.

The Algorithm

1. Compute the **Minimum Spanning Tree (MST)** of the graph.
2. Create a path by walking around the MST (Full Walk / DFS).
3. Create the tour H by **short-cutting** (skipping repeated vertices).

The Proof ($c(H) \leq 2c(H^*)$)

- **Step 1 (Lower Bound):** Removing one edge from the Optimal Tour H^* creates a spanning tree. Therefore, the cost of the MST is less than the optimal tour.

$$c(T) \leq c(H^*)$$

- **Step 2 (The Walk):** A full walk of the MST traverses every edge exactly twice (once down, once up).

$$c(Walk) = 2 \times c(T)$$

- **Step 3 (Triangle Inequality):** Short-cutting allows us to skip nodes we've already visited. By the Triangle Inequality (direct path is shorter than detour), this reduces cost.

$$c(H) \leq c(Walk)$$

- **Conclusion:**

$$c(H) \leq 2 \times c(T) \leq 2 \times c(H^*)$$