Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

The Arcadia Engine

CAPSTONE IN ALGORITHMS & DATA STRUCTURES

Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

1. Background / Real-World Context

Scenario: Your team has been hired by *Arcadia Games* to build the backend engine for their new MMORPG (Massively Multiplayer Online Role-Playing Game). The game requires a high-performance system to handle player data, inventory management, world navigation, and server task scheduling.

Instead of writing isolated scripts, you will build the **ArcadiaEngine** library. The engine consists of four subsystems:

1. **The Registry:** Handles player lookups and leaderboards (Hashing & Skip Lists).
2. **The Inventory System:** Manages loot distribution and storage (Dynamic Programming).
3. **The Navigator:** Manages travel routes and safe paths (Graphs).
4. **The Kernel:** Manages server CPU tasks (Greedy Algorithms).

2. Integrated Tasks

Part A: The Registry (Data Structures)

Context: The game needs to store player profiles and maintain a live leaderboard.

1. **Player Lookup (Hashing):** Implement a *PlayerTable* using a **Hash Table**. You must implement a custom hash function (Mid-Square or Multiplication method) and handle collisions using **Double Hashing**.
2. **Live Leaderboard (Skip List):** Implement a Leaderboard class using a **Skip List**. This must support $O(log\ n)$ insertion, deletion, and searching of player scores. It must also support a function *getTopN(int n)* to retrieve the highest-ranked players.
3. **Auction House (Red-Black Tree):** Implement an *AuctionTree* using a **Red-Black Tree** to store items currently for sale, ordered by price. You must implement insert and delete maintaining the RB properties.

Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

Part B: The Inventory System (Dynamic Programming)

Context: Players find loot and need to organize it.

1. **Loot Splitting (Partition Problem):** Two guild leaders want to split a bag of gold coins as evenly as possible. Given a list of coin values, minimize the difference between the two sums.

Input	Output	Explain
n=3 coins = {1,2,4}	1	Best split: {4} vs {1,2}. Difference: 4-3 = 1
n=2 coins = {2,2}	0	Perfect split: {2} vs {2}. Difference: 0
n=4 coins = {1,5,11,5}	0	Best split: {11} vs {1,5,5}. Both sum to 11

2. **Inventory Packer (Knapsack):** A player has a backpack with weight capacity W. Given a list of items with weight w_i and value v_i, maximize the total value they can carry. (Note: This combines the logic of the "Bank Robber" and "Diver" problems).

Input	Output	Explain
capacity=3 items = {{1,10}, {2,15}, {3,40}}	40	Select item 3 (weight=3, value=40) only
capacity=5 items = {{1,10}, {2,15}, {3,40}}	55	Select items 1 and 3 (total weight=4, value=50) OR items 2 and 3 (total weight=5, value=55)
capacity=10 items= {{1,10}, {2,20}, {3,30}}	60	All items fit (total weight=6). Total value=60

3. **Chat Autocorrect (String DP):** The chat system is buggy. Given a received string (where 'w' might be 'uu' and 'm' might be 'nn'), calculate the number of possible original strings modulo $10^9 + 7$.

Input	Output	Explain
"uu"	2	Possible originals: "uu" (no substitution) or "w" (reverse substitution)
"uunn"	4	Possible: "uunn", "wnn", "uum", "wm"

Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

Part C: The Navigator (Graph Algorithms)

Context: Players need to traverse the game world consisting of N cities and M roads.

1. **Safe Passage (Path Existence):** Determine if a valid path exists between a source city and a destination city given a set of bidirectional edges.

Input	Output	Explain
n=3 edges = {{0,1}, {1,2}} source=0 dest=2	true	Path exists: 0→1→2
n=4 edges= {{0,1}, {2,3}} source=0 dest=3	false	No path connects 0 and 3 (disconnected components)
n=1 edges = {} source=0 dest=0	true	Source equals destination

2. **The Bribe (Min Cost Path/MST variant):** Roads are guarded by bandits requiring specific gold/silver payments. Given the exchange rate between gold/silver and "Olympian Tugriks," find the minimum cost to ensure connectivity or specific passage (Variation of the "Kingdom of Olympia" problem).

Input	Output	Explain
n=3 m=3 goldRate=1 silverRate=1	15	Roads: {{0,1,10,0}, {1,2,5,0}, {0,2,20,0}} Costs: Road1=10, Road2=5, Road3=20 MST picks roads 2,1 → Total: 15

3. **The Teleporter Network (All-Pairs Shortest Path):** The world has road lengths that are powers of two. Calculate the sum of minimum distances between all pairs of cities (binary representation required).

Input	Output	Explain
n=3	"110"	Distances: 0-1=1, 1-2=2, 0-2=3.

roads= {{0,1,1}, {1,2,2}}		Sum=6. Binary: 110
n=2 roads = {{0,1,4}}	"100"	Distance: 0-1=4. Sum=4. Binary: 100
n=3 roads = {{0,1,2}, {0,2,8}}	"1010"	Distances: 0-1=2, 0-2=8, 1-2=∞ (disconnected). Sum=10. Binary: 1010

Part D: The Kernel (Greedy Algorithms) Server Job Scheduler: The server processes tasks 'A'-'Z'. Tasks of the same type require a cooling interval n. Calculate the minimum CPU intervals to finish all tasks.

Constraints:

- 1 <= tasks.length <= 104
- tasks[i] is an uppercase English letter.
- 0 <= n <= 100

Input	Output	Explain
tasks={'A','A','B'} n=2	4	Schedule: A → B → idle → A. Total: 4 intervals
tasks={'A','A','A'} n=2	7	Schedule: A → _ → _ → A → _ → _ → A. Total: 7 intervals
tasks={'A','B','C'} n=2	3	All unique tasks, no cooling needed. Total: 3 intervals
tasks={'A','A','A','B','B','B'} n=2	8	Schedule: A → B → idle → A → B → idle → A → B. Total: 8 intervals

3. FAQ

Q: In Part A, Can I use STL containers (set, map, unordered_map, unordered_set etc.) internally?

A: No, You can't

Q: How should getTopN handle ties?

A: Return tied players in any order, but ensure all top-scoring players appear before lower-scoring ones.

Q: What does minBribeCost calculate?

A: The minimum total cost to connect ALL cities (Minimum Spanning Tree problem).

Q: What format are items in for maximizeCarryValue?

A: Each item is {weight, value}. Example: {{1,10}, {2,15}}

means item 1 has weight=1, value=10.

Q: What should optimizeLootSplit return?

A: Return the minimum difference as an integer.

Example: coins={1,2,4} → return 1 (split: {4} vs {1,2}=3)

Q: What are the expected time complexities?

A: Hash table O(1) avg, Skip list O(log n), RB tree O(log n),

Knapsack O(n×W), Partition O(n×sum), String O(n),

Path O(V+E), MST O(E log V), **All-Pairs Shortest Path** - APSP $O(V^3)$, Scheduler O(n log n)

Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

4. Submission Guidelines:

- Teams must consist of 4 to 5 members.

    - A penalty will be imposed for groups with fewer than 4 or more than 5 members will incur point deductions.

- Beside pdf there are two files *ArcadiaEngine.h* and *ArcadiaEngine.cpp*.

- Do not modify *ArcadiaEngine.h* under any circumstances.

    - Any alterations to this file will result in automatic test case **failures and a grade of zero**.

- Implement TODO and empty function in *ArcadiaEngine.cpp*.

- You may add private helper functions or struct in *ArcadiaEngine.cpp*.

- All team members must understand all parts of the project.

    - if there are team member didn't understand his/her work and his/her teammate work He / She will lose points

- No late submissions are allowed.

- Cheating is NOT tolerated by any means.

- Only one team member has to submit one compressed file (ZIP or RAR format) following the naming convention:

Assignment02_ID1_ID2_ID3_ID4 _ID5.zip/rar

    - if naming convention is wrong you will lose points

- A penalty will be imposed for violating any of the assignment rules or missing any deliverable.

- Cheaters will get ZERO and no excuses will be accepted as per the attached "Plagiarism Scope" document.

- TAs will grade the assignment out of 100, but this score may be adjusted later for scaling purposes.

- **Deadline 2025-12-18**

5. Deliverables

Team Submission (Compressed File): Assignment02_ID1_ID2_ID3_ID4_ID5.zip/rar

The submission must include:

1. ArcadiaEngine.cpp: Complete implementation of all required classes
2. TeamInfo.txt: Full names and student id of all team members

Cairo University

Faculty of Computing and Artificial Intelligence

CS321 - Algorithms Analysis and Design

Contents