GSCP
Habib University

Developing a Course on Competitive Programming

Authors

Areesha Amir
CS, DSSE, HU

Ali Muhammad Asad
CS, DSSE, HU

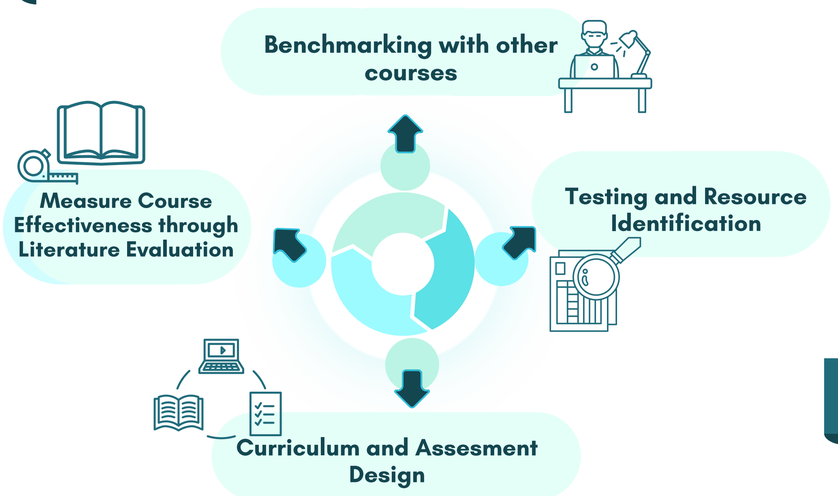
Iqra Ahmed
CS, DSSE, HU

Waqar Saleem
CS, DSSE, HU (Supervisor)

Research Objective

Developing a foundational course to introduce competitive programming to undergraduate students.

Methodology



Literature Review

- **Advantages of Dedicated Module:**
  - Identification of and Training for Talented Students.
  - Develop team dynamics and formation of strong teams through skill identification
  - Allocation of resources.
- **Key Challenge Areas:**
  - Manpower, Students, Assessment, Course Materials.
- **Methods to Determine Course Efficacy:**
  - Student Surveys, Edumetric tests.

Findings

- Competitive programming courses emphasize **traditional lectures** and slide-based learning.
- **Design-based assessments** are used in some courses to promote project-based learning.
- **Mini contests** help students adapt to competitive programming conditions and improve time management.
- Platforms:
  - **Codeforces** is renowned for its extensive collection of problems and is a top **contest-hosting** platform.
  - **Kattis** offers a user-friendly interface, making it an excellent choice for **practice problems** in course assessments.

Discussion

Assessment Design:
- Usual focus: Contest problem-solving and algorithmic practice.
- Our twist: Adding **design challenges**, **platform rankings**, and **bonus points** for holistic skill advancement.

Teaching Strategy:
- Conventional: Lecture-based instruction in competitive programming courses.
- Our shift: **Flipped classroom model** with interactive worksheets and in-class mini-contests for heightened engagement and critical thinking.

Platform Selection:
- Common practice: One platform for assessments in Competitive Programming courses.
- Our approach: Incorporating **two platforms** to leverage diverse benefits.

Conclusion

The first course in competitive programming was designed, incorporating active learning through the **flipped classroom model**, with the objective of teaching students skills in **problem-solving**, **algorithmic mastery**, **efficient code implementation**, **time-sensitive development**, and **collaborative aptitude**.

Figure 1: Topic Division

Week No.	Topics
1	Introduction: Input/Output Techniques + Ad Hoc Simulation
2	Elementary Data Structures (arrays, lists, vectors) and Libraries in python, and C++
3	Data Structures and sublinear complexity structures - Sorted associative sets, maps, i
4	Search and Sorting + Problem Solving Paradgim: Divide and Conquer
5	Greedy Algorithms
6	Dynamic Programming
7	Midterm Exam
8	Graphs(including unweighted), graph traversal and graph algs including BFS and DFS
9	Intermediate graph algorithms + Trees
10	Shortest path algorithms
11	Network Flow
12	Computational Geometry and Geometry Algorithms
13	Strings, string matching, suffix tree, prefix tree
14	Mathematics and Number theory
15	Combinatorics
16	Final Exam

QR code linking to syllabus n report folder

Figure 2: Course Books

References

[1] T. Di Mascio, L. Laura and M. Temperini, "A Framework for Personalized Competitive Programming Training," 2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET), Olhao, Portugal, 2018, pp. 1-8, doi: 10.1109/ITHET.2018.8424620.
[2] Geurrerio, P. and Rebeiro, P.Early Introduction of Competitive Programming [Preprint].
[3] Halim, Steven. Competitive Programming 4. 2017.
[4] Lamsal, Subarna. "Competitive Programming Journey." Data Insight Online. 10 May 2020.