Algorithms Projects'17

Requirements Summary

**Group Count:** 3 – 5 members

**Group Registration:** [**Online form**](https://docs.google.com/forms/d/e/1FAIpQLSeUG9dzeOgmdI5mpsp8Is9a-EBIwBkD94HqYBAPeTh2ymxzfw/viewform) due to **SUN 26-Nov-2017**, (**After deadline:** Groups will be **Manually Assigned** to a Project)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Project** | **Allowed** | **Inputs & Outputs** | **Deliverables** | **Grades[[1]](#footnote-1)** | **Bonus** |
| [**Image Encryption and Compression**](%5b1%5d%20Image%20Encryption%20and%20Compression/Image%20encryption%20and%20compression.docx) | **Priority Queue**  **C# Code** to   1. Open image & load it in 2D array 2. Calculate the edge-strength G between two pixels 3. Display image. | **Inputs:**   1. Image (2D array of pixels) 2. Encryption Key (Initial seed + Tap position)   **Outputs:**  Binary file contains:   1. Key (seed value & tap position) 2. Huffman Tree 3. Compressed image | **Document contains ONLY:**   1. Code of image encryption. 2. Code for constructing the Huffman Tree. 3. Code for image compression. 4. Code for image decompression. 5. Detailed analysis of the above codes. 6. Compression ratio of complete test cases **WITH** and **WITHOUT** encryption. | 40% | * 1. alphanumeric password   2. Break the encryption |
| **Implementation**:   1. Image encryption using Linear Feedback Shift Register 2. Image decryption using Linear Feedback Shift Register 3. Image compression using Huffman Coding with priority queue 4. Image decompression using Huffman Coding 5. Save and load the binary encoded and compressed file | 60% |
| [**Sounds Packing**](%5b2%5d%20Sounds%20Packing/Sounds%20Packing.docx) | **Priority Queue**  **Sorting** | **Inputs:**   1. folder contains audio files. 2. A text file contains the following for each file:    1. File name.    2. File duration.   **Outputs:**   1. Folders with audio files 2. Text file per folder: 3. folder name. 4. files names & durations. 5. total duration. | **Document contains ONLY:**   1. Code of the following algorithms    1. **Allocation Strategies (Heuristics)**       1. Worst-fit decreasing (Using Linear Search and Priority Queue).       2. First- fit decreasing    2. **Folder Filling Algorithm.** 2. Detailed analysis of the above codes. 3. Comparison between all methods in term of **number of folders** and **execution time** | 40% | 1. Best fit strategy 2. Any other efficient recommendations are welcomed |
| **Implementation**:   1. Worst-fit algorithm using Linear Search. 2. Worst-fit algorithm using Priority queue. 3. Worst-fit decreasing algorithm using Linear Search. 4. Worst-fit decreasing algorithm using Priority queue. 5. First-fit decreasing algorithm using Linear search. 6. Folder Filling algorithm | 60% |

|  |  |  |
| --- | --- | --- |
| **Delivery** | [**Image Encryption and Compression**](file:///C:\Users\Dell\Desktop\%5bALG'17%5d%20Lab8%20-%20%5bPROJECT%5d%20+%20Priority%20Queue\PROJECT%20RELEASE\%5b1%5d%20Image%20Encryption%20and%20Compression\Image%20encryption%20and%20compression.docx) | [**Sounds Packing**](%5b2%5d%20Sounds%20Packing/Sounds%20Packing.docx) |
| **Milestone1**  Week13: start at SUN 10-12-2017  During LABS | 1. Image encryption using Linear Feedback Shift Register 2. Image decryption using Linear Feedback Shift Register 3. Construction of the Huffman Tree using priority queue 4. Documentation I | 1. Worst-fit algorithm using Linear Search. 2. Worst-fit algorithm using Priority queue. 3. Worst-fit decreasing algorithm using Linear Search. 4. Worst-fit decreasing algorithm using Priority queue. 5. Documentation I |
| **Milestone2**  Final Delivery  Lab Exam week | 1. Image compression using Huffman Coding with priority queue 2. Image decompression using Huffman Coding 3. Save and load the binary encoded and compressed file 4. Documentation II | 1. First-fit decreasing algorithm using Linear search. 2. Folder Filling algorithm 3. Documentation II |
| **For Milestone1:**   * + **MUST** deliver the required tasks and **ENSURE** it’s worked correctly   + **MUST** deliver the **part of the documentation** that is related to the Milestone (printed document)   + **MUST** deliver in the **section time of the majority members** (e.g. if a group consists of 3 members from sec.1 and two members from sec.2, they should deliver in the time of sec.1) | | |

1. Grades distribution is subject to change without prior announcement [↑](#footnote-ref-1)