Implementation

You will implement several algorithms without using any external libraries.

MAX-HEAPIFY Procedure [10 points]

Implement the MAX-HEAPIFY procedure. This procedure is crucial for maintaining the max-heap property. Your code should ensure that the max-heap property is preserved and the procedure should run in $O(\log n)$ time. Provide clear comments explaining each step of your implementation.

BUILD-MAX-HEAP Procedure [10 points]

Extend your implementation to include the BUILD-MAX-HEAP procedure. The goal is to produce a max-heap from an unordered input array. This procedure should run in linear time. Clearly document your code and provide explanations for the logic behind each step.

HEAPSORT Procedure [15 points]

Implement the HEAPSORT procedure. This procedure should use the MAX-HEAPIFY and BUILD-MAX-HEAP procedures to sort an array in place. Provide a detailed explanation of each step in the HEAPSORT procedure. Discuss the time complexity and any optimizations you have made.

Priority Queue Operations [15 points]

Integrate priority queue operations into your heap implementation. Implement the following procedures:

- MAX-HEAP-INSERT: Inserts a new element into the max-heap.
- HEAP-EXTRACT-MAX: Removes and returns the maximum element from the max-heap.
- HEAP-INCREASE-KEY: Increases the key of a specified element in the max-heap.
- HEAP-MAXIMUM: Returns the maximum element from the max-heap without removing it.

Provide a script with clear explanations for each procedure. Include examples demonstrating the usage of these procedures in a real-world scenario.

Implementation of d-ary Heap Operations [20 points]

Consider a d-ary heap, a variation of the binary heap where non-leaf nodes have d children.

(a) Height Calculation
Implement a function dary_calculate_height that takes the number of elements n and the degree d as input and returns the height of the corresponding d-ary heap. Provide comments for clarity.

(b) EXTRACT-MAX Implementation
Implement a function dary_extract_max that performs the EXTRACT-MAX operation in a d-ary max-heap. Analyze the running time of your implementation in terms of d and n. Provide comments for each step of your code.

(c) INSERT Implementation
Implement a function dary_insert_element that performs the INSERT operation in a d-ary max-heap. Analyze the running time of your implementation in terms of d and n. Provide comments for each step of your code.

(d) INCREASE-KEY Implementation
Implement a function dary_increase_key that performs the INCREASE-KEY(A, i, k) operation, where A is a d-ary max-heap, i is the index, and k is the new

key. Your implementation should first set A[i] to max(A[i], k) and then update the d-ary max-heap structure appropriately. Analyze the running time of your implementation in terms of d and n. Provide comments for each step of your code.

Note: Ensure that your code is well-documented and follows good coding practices. Analyze the time complexity of your implementations, considering the parameters d and n. Present your analysis in the report.

DataSet

For this assignment, you are given a simplified version of a DataSet containing information about cities and their populations. The DataSet includes the following columns:

- City [String]
- Population [Integer]

Multiple permutations of the same simplified DataSet are provided as ”population1.csv,” ”population2.csv,” ”population3.csv,” and ”population4.csv” for testing your code and reporting results.

Deliverables

Code [70 points]

Submit a C++ code file titled ”Heapsort.cpp.” Your code should implement the specified procedures and accept the following command line arguments:

1. The name of the DataSet file.
2. The name of the fuction, which is one of the below (You will implement these 12 functions with these names):
   max_heapify, build_max_heap, heapsort, max_heap_insert, heap_extract_max, heap_increase_key, heap_maximum, dary_array_representation, dary_calculate_height, dary_extract_max, dary_insert_element, dary_increase_key

3. The name of the output file. The sorted output should be written to a CSV file with the same format as the input.

4. The optional arguments ’d’, ’i’ and ’k’ which are the number of children of non-leaf nodes, the index of a node and the key value to update an element, respectively. They are optional and the interpretation of given values depend on the name of the function. When providing any of these three parameters as argument, their values are concatenated to the letter, e.g. ’d5’, ’i125’ or ’k43985’. (Note that indices begin with 1 in the context of heap. Therefore if 1 is given as index, it corresponds to the root of the heap.)

Compile the code using the shell command:

```
g++ Heapsort.cpp -o Heapsort
```

Three example shell commands to run the compiled executable are:

```
./Heapsort population2.csv heapsort out.csv
./Heapsort population3.csv heap_increase_key out.csv i435 k2749
./Heapsort population1.csv dary_calculate_height out.csv d5
```