Homework 03

[2024-1] 데이터사이언스를 위한 컴퓨팅 1 (001)

Due: 2024 년 5 월 28 일 15 시 30 분

In this homework, you will solve problems using the various algorithms presented in the class.

1. Max-heap with tree-based structure [25pts]

Implement a binary max-heap with tree-based structure which has the following properties:

- Structure property: It is a complete binary tree.
- Heap property: The value of every non-root node is equal to or smaller than the value of its parent.

You need to complete the `enqueue` and `dequeue` method in `MaxHeap` class provided in the skeleton code.

- The `enqueue` method takes an integer k as input. A new node is created with the value k and then appended to the heap as the last node. After the new node is appended, the heap property is restored.
- The `dequeue` method takes no input. It removes the value at root node(which includes the maximum value among all nodes in the heap). Then the value at root node is replaced with the last node. The last node is then removed from the heap. Finally, the heap property is restored.
- Make sure each operation of `enqueue` and `dequeue` method are completed in $O(\log n)$ time.

Some helper methos are provided. The description of each method is as following:

- `getMax()`: This function returns a pointer of root node.
- `printHeap()`: This function prints the values of nodes included in the heap, from the root node to leaves, from left to right at the same level.
- `swap(Node * a, Node * b)`: This function swaps the values of two input nodes.

Example:

```
Maxheap h;
h.enqueue(10);
h.enqueue(30);
h.enqueue(40);
h.enqueue(500);
h.enqueue(80);
h.dequeue();

h.getMax()->val;
>> 80
```

2. Minimum distance [25pts]

Given an array of points representing integer coordinates on a 2D-plane, where each point is represented as $[x_i, y_i]$, Your goal is to find the minimum distance to connect all points. The distance between two points$[x_i, y_i]$ and $[x_j, y_j]$ is based on the Manhattan distance: $|x_i - x_j| + |y_i - y_j|$.

Implement the `minDistance` fucntion in `functions.cpp` according to the following instructions.

2.1 Input

The function takes one input argument: A vector of integer vectors of length 2, `points`.

2.2 Output

The function returns the minimum distance to connect all points.

2.3 Constraint

$$1 \leq \text{points.size}() \leq 1000$$
$$-10^6 \leq x_i, y_i \leq 10^6$$
All points $[x_i, y_i]$ are distinct.

2.4 Example

```
minDistance (vector({vector({3,12}), vector({-2,5}), vector({-4,1})}))
>> 18
```

3. Flight routes [25pts]

You are going to travel from San Francisco to New York by plane. You would like to find answers to the following questions:
• What is the minimum price of such a route?
• How many minimum-price routes are there? (modulo `INT_MAX (or 2147483647)`)
• What is the minimum number of flights in a minimum-price route?
• What is the maximum number of flights in a minimum-price route?

Implement `flightRoute` function in `functions.cpp` following the instruction below.

3.1 Input

The function takes 3 input arguments: the number of cities `n`, the number of flights `m`, and a vector `route_info` of length `m` which inlcudes the flight route information. The cities are numbered 1,2,. . . ,n. City 1 is San Francisco, and city n is New York. Each elemet of `route_info` has three integers a, b, and c: there is a flight from city a to city b with price c. All flights are one-way flights. You may assume that there is a route from San Francisco to New York.

3.2 Output

Print four integers according to the problem statement. Separate each integer by space(“ “). Do not include any new line character or `std::endl` at the end of the `stdout`. The function does not return anything.

3.3 Constraints

$$1 \le n \le 10^5$$

$$1 \le m \le 2 \cdot 10^5$$

$$1 \le a, b \le n$$

$$1 \le c \le 10^9$$

3.4 Example

```
flightRoute(4, 5, vector({{1,4,5}, {1,2,4}, {2,4,5}, {1,3,2}, {3,4,3}}))
>> 5 2 1 2
```

4. Coin combinations[25pts]

Consider a money system consisting of `n` coins. Each coin has a positive integer value. Write a function `coinCombinations` that calculates the number of distinct ways you can produce a money sum x using the available coins. For example, if the coins are {1,2,3} and the desired sum is 4, there are 7 ways:

- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

Implement `coinCombinations` function in `functions.cpp` following the instruction below.

4.1 Input

The functtion takes 3 input arguments: the number of coins n, the desired sum of money x, and a vector that includes n distinct integers $c_1, c_2, c_3, \ldots c_n$: the value of each coin.

4.2 Output

The function returns an integer: the number of ways modulo `INT_MAX (or 2147483647)`

4.3 Constraints

$$1 \leq n \leq 100$$

$$1 \leq \mathrm{x} \leq 10^6$$

$$1 \leq c_i \leq 10^6$$

4.4 Example

```
coinCombinations (3, 4, vector({1, 2, 3}))
>> 7
```

5. Submission Instructions

- For each problem, write your functions in the `functions.cpp` file prepared for that problem, and submit this file to Gradescope for grading. You can modify other files for your own testing, but keep in mind that **only functions.cpp will be graded using the original header files and main.cpp.**
- Grading will be conducted using the C++11 standard. Our compilation process is as follows:
  - `$ g++ -c main.cpp -o main.o -std=c++11` (using the original main.cpp)
  - `$ g++ -c functions.cpp -o functions.o -std=c++11` (using your functions.cpp)
  - `$ g++ main.o functions.cpp -o main`
- When submitting your codes, ensure that all test codes printing out `stdout` are removed from `functions.cpp`, unless explicitly instructed otherwise. Grading relies solely on the `stdout`, and including any unrelated codes may result in inaccurate grading.
- You should NOT share your code with other students. Any student found to have a high level of code similarity, as determined by our similarity detection process, may face severe penalties.
- If you submit late, grace days will be automatically deducted. You can utilize 5 grace days throughout the semester for homework submissions. Grace days are counted based on 24-hour increments from the original due time. For instance, if an assignment is due on 5/28 at 3:30pm, submitting it 30 minutes late will count as using one grace day, while submitting it on 5/29 at 9pm will count as using two grace days. Late submissions after exhausting all grace days will NOT be accepted.
- For questions about this homework, use the “Homework 3” discussion forum on eTL.
- Failing to adhere to the submission guidelines may result in penalties applied without exception.