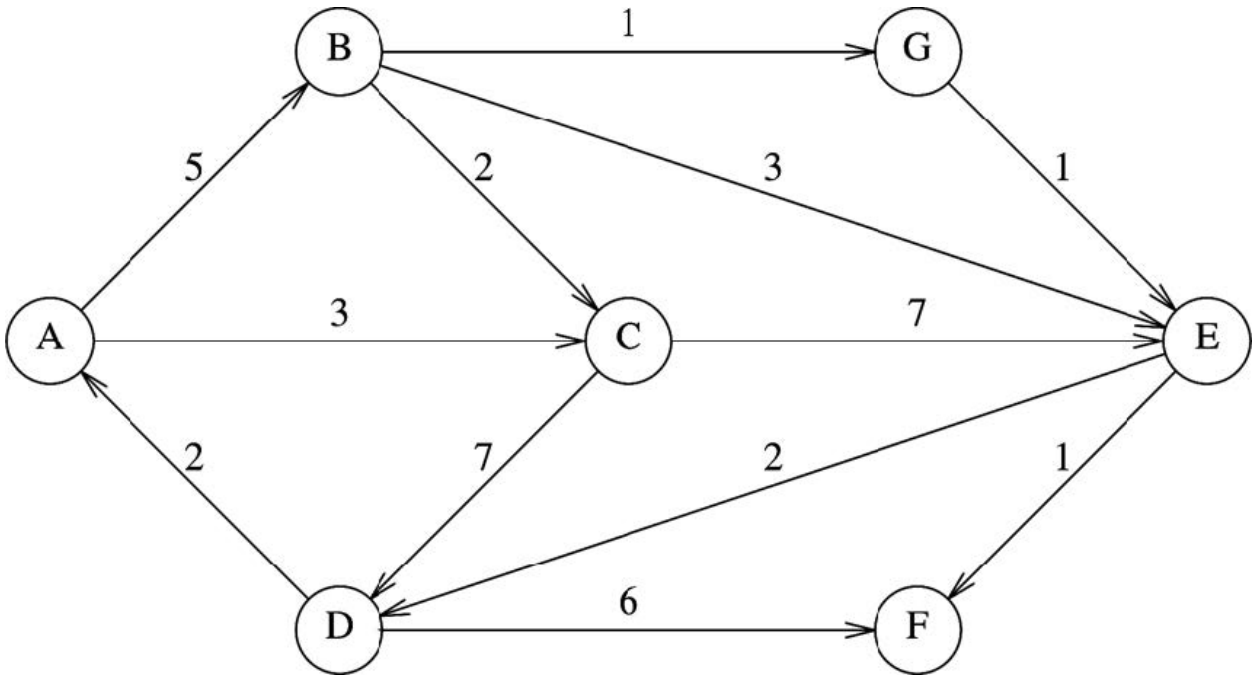
Written and Programming Assignment 4
EECS 233 Introduction to Data Structures

Due: December 9, 2019 11:59 PM EST on Canvas

Written Exercises

1. Consider a **connected undirected** graph G with N nodes. [10 pts]
   (a) What is the **smallest** possbile number of edges in G? In this scenario, is this **acyclic or not**? Why?
   (b) If G is a complete graph. How many possible spanning trees G can have? Why?
2. Run the Dijkstra algorithm with heap in following graph. For each step, write down the cost from the source to each vertex as well as the updated binary heap. [20 pts]



3. Consider the bucket sort algorithm. The pseudocode[1] for bucket sort is given below.

```
BUCKET-SORT(A)
 1  n = A.length
 2  let B[0..n-1] be an empty array
 3  for i = 0 to n-1
 4        make B[i] an empty list
 5  for i = 0 to n-1
 6        insert A[i] into list B[hash(A[i])]
 7        // Hashing function determines which bucket element goes into
 8  for i = 0 to n-1
 9        sort list B[i] with insertion sort
10  concatenate the lists B[0], B[1], ..., B[n-1] together in order
```

   (a) Argue that the worst-case running time for bucket sort is $O(n^2)$. Provide a scenario in which this happens. [5 pts]

   (b) By replacing insertion sort with a more efficient mergesort or heapsort in line 9 of the pseudocode, bucketsort can achieve a worst-case running time of $O(nlogn)$ instead of $O(n^2)$. Argue why switching insertion sort to a more "efficient" sort is not a good idea. *Hint: When would you use bucket sort in the first place?* [5pts]

4. The asymptotic running time of quicksort vastly outperforms the asymptotic running time of insertion sort. However, for small array sizes, insertion sort is usually more efficient because there is less overhead. Consider an improved version of quicksort that runs the classic recursive quicksort algorithm, but *stops* when the subarrays have fewer than k elements. At this point, insertion sort is called *once* on the *entire array*, which finishes the sorting. Show that this sorting algorithm has an $O(nk + nlog(n/k))$ running time. [10pts]

Programming Exercises

1. **Implement Mergesort, Quicksort, and Select** [30 pts]
   In a file called `Sort.java`, implement the following static methods:

   (a) `void mergesort(int[] input)`: Takes an input array of integers. Uses the mergesort algorithm[2] to sort the input array into ascending order.

   (b) `void quicksort(int[] input)`: Takes an input array of integers. Uses the quicksort algorithm to sort the input array into ascending order. Has an average O(n logn) running time in the average case.

   (c) `int select(int[] input, int k)`: Takes an input array of integers. Returns the kth smallest integer in the array in average O(n) time. The algorithm is allowed to reorder array elements. Example: `[5, 3, -2, 9, 8]`. Selecting $k = 0$ should return -2, the smallest element. Selecting $k = 2$ should return 5.

[1] Adapted from Cormen et al. *Introduction to Algorithms* 3ed.

[2] This does not have to be an in-place implementation, as long as the input array is sorted at the end of the method.

You may use any additional helper methods and functions as necessary. Keep your code clean and abstracted; avoid having to repeat large segments of code in different methods.

In the main method of the `Sort.java`, test out all three methods on some sample input arrays. For the sorting methods, print out the arrays before and after sorting. For the selection method, print out the returned value. Be sure to test your methods on sorted ascending, sorted descending, and unsorted arrays.

2. **Minimum campus distance** [20 pts]
   There are N buildings in Case Western Reserve University campus. David doesn't have a good memory, so he wants to choose $N-1$ roads that, from any building in the campus, he can get to any building he wants. However, David doesn't want to walk too much, so he also wants total length of all roads to be minimum. Your task is to help David to find those $N-1$ roads. For the simplicity of the problem, instead of using building's name, we will use the building's number, so there are N buildings from 0 to $N-1$. In `CWRUMap.java`, please implement the following method:

   `int minDistance(int[][] dists)`: Takes an input of a 2-D integers array where dists[i][j] is the distance of a single road from building i to building j (note that $dists[i][i]$ is always 0 and $dists[i][j] = dists[j][i] > 0$ when $i \neq j$). Returns the minimum total distance of $N-1$ roads. For example, giving this 2-D array:

0	1	2	10
1	0	3	4
2	3	0	3
10	4	3	0

   Your method should return 6 because the roads David should choose are $0-1, 0-2, 2-3$. Please also include the `main` method in `CWRUMap.java` to test at least 3 scenarios with $N > 4$

Submission

The assignment is due December 9, 2019 at 11:59 PM EST on Canvas. Please submit the written and programming exercises *separately.*

1. Submit a PDF with your answers to the written exercises to "Written Assignment 4" on Canvas.

2. Submit a zip file that contains your source code for the programming exercises `Sort.java` and `CWRUMap.java` to "Programming Assignment 4" on Canvas. *Please make sure to get rid of all package statements so that the grader can run and compile your code on the spot.*