COMP 582 Final Exam, Fall 2022

MCS@Rice Program
Time Limit: 5 Hours

Instructions

For the exam, you may use or refer to videos, video slide sets, problem sets, live session slides, and any of your personal notes or materials. You may use any result or theorem that was stated in a course problem set. You may also assume that any problem solved with an algorithm from the problem sets is correct. (You do not need to have gotten the problem right). You may *NOT*, however, collaborate with any other people.

You may also use the Internet in a limited way. You may search for something on the internet, but you may *NOT* ask an exam question on any question answering site (like Stack Overflow). Also, if you use something from the internet, you may **not copy it verbatim.** You must paraphrase into your own words. For code, please change a variable name and/or rearrange some of the serial lines**.** Furthermore, if you use something you found on the internet, please cite whatever you found.

For a "short answer" question, you may just put the answer.

For a "long answer" question, please show *all* relevant work, and *only* relevant work.

If you turn in spurious work, or random discarded approaches, I will assume that you think whatever you wrote should contribute to the solution. Anything you turn in will be assumed to be part of your answer.

As with your problem sets, you may turn in PDF or plain text. You must, however, turn in **only 1 file**

Also, do not forget to put the Honor Code Pledge on your last age of your exam (you may use a separate page if necessary). Please be sure to include your start time and end time as part of the Pledge.

When you have finished your exam, please upload to Canvas (just like the problem sets).

General Guidelines

- Asking for the "complexity" of an algorithm means the worst-case *time* complexity of the algorithm.

  ✦You might be asked specifically for the "space complexity", or "time complexity", but absent the qualifier, "time complexity" is the default

- If you are asked to give an "efficient" algorithm, then you should produce an algorithm with the best worst-case time complexity.

- By default, complexity and solutions to recurrences may be given in Big Oh notation.

  ✦You may be asked for *exact* solutions in some cases. For those cases, Big Oh notation is ***not*** acceptable.

- Algorithms should be implemented in Python. All of Python language features and standard library can be used. In addition, NumPy module is also usable.

  No other 3rd party modules are allowed.

  NOTE: In some problems, *Restricted* Python (i.e. Python without dictionaries or sets) might be called for. Absent the explicit

Short Answer Questions, Part 1

For each of the following algorithms, give:
a) the worst-case complexity
b) the average case complexity.of these algorithms

Give your answer in Big Oh notation.
If there is no difference in the worst case and the average case, then you may write "Same" for part b), the average case

Each question is worth 2 points, 1 point each for a) and b) answers

1. Selection Sort of N items

2. Insertion Sort of N items

3. A sequence of M union-find operations on N disjoint sets.
   Assume that the union operation is weighted quick union, and all operations use path compression.

4. The total copying cost when using the doubling algorithm for a sequence of N push operations. The initial size of stack is 1 element.

5. Quicksort, where the pivot is always 1st element, of N items

6. Merge sort of N items

7. Quickselect, where pivot is always 1st element, to find the median of N items (assume N is an arbitrary *odd* number)

8. Quickselect, where pivot is always 1st element, to find the 4th smallest element of N items ($N \geq 4$).

9. Delete 1 element from a hash table that has N elements.

10. Find an item in a hash table that has N items.

Short Answer Questions, cont

Give the **worst case** performance (in Big Oh notation) for the following algorithmic problems

Each question is worth 1pt each

11. A LSB radix sort over an alphabet of size A, for fixed-length strings of length L. There are N such strings.

12. Build a binary heap from an array of size *N*.

13. Find an item in a left-leaning red-black tree of size N.

14. The **1st** find operation in sequence of *M* Union/Find operations on *N* disjoint sets.

15. For a graph G = (V,E) with no negative edge weights, how long does Dijkstra's algorithm take to find the shortest path between distinguished vertices s and t?

16. Strassen's algorithm to multiply 2 square matrices of size NxN?

17. Given a sample string of length N, and a fixed pattern (NOT a regular expression) of length M, what is the complexity of searching the sample string for the pattern?

18. Given a sample string of length N, and a (fully parenthesized) regular expression of length M, what is the complexity of searching the sample string for the pattern?

19. Given a graph G = (V,E) with negative edge weights, but no negative cycles, what is the best known asymptotic complexity for finding the shortest path from 1 distinguished vertex s to *all* other vertices in the graph?

20. Given a graph G = (V,E), with weighted edges given by the function w(e) for e $\in$E, what is the complexity to find the minimum cost spanning tree for G?

"Long" Answer Questions

1. [4 pts] What is the complexity of the following program? (Big Oh notation is sufficient). Assume that `X[i][j]` references the `(i,j)` component of matrix `X`. Also, assume that `N` parameter describing the size of `X` as `X[N,N]`, size of `v` as `v[N]`, and the size of b as `b[N]`.

```
for i in range(N+1):
    v[i] = 0
    for j in range(i+1,N+1,2):
        v[i] += A[i][j]*b[j]
```

   Be sure to give your reasons for your complexity calculation.

2. [6 pts] Suppose someone you know implemented Kruskal's algorithm for the minimum spanning tree of a graph G = {V, E}. Unfortunately, your acquaintance did ***not*** implement weighted union or path compression for the union/find operations. What is the complexity of this variant of Kruskal?

3. [10 pts] Suppose you have an implementation of an A-heap. The A-heap is a heap-like data structure that has $O(1)$ complexity for the decrease-key operation. All other operations of the data structure have the same complexity as a traditional binary heap. What is the complexity of Dijkstra's algorithm using the A-heap data structure?

4. [6 pts] Using any method you like, find the Big Oh complexity of the recurrence

$$T(n) = 5T(\tfrac{n}{4}) + \frac{n^3}{\log^2(n)}$$

5. [10 pts] You are given a complete, undirected graph $G = (V, E)$ with edge weights. A complete graph is a graph s.t. for any 2 distinct vertices v_1, v_2, there is an edge $< v_1, v_2 > \in E$. You construct a minimum cost spanning tree (MST) for G. Now suppose that 1 of the edges of the MST is deleted.

   Give an efficient algorithm to construct a new MST from the old 1 without using the deleted edge.
   In addition to your algorithm, give the complexity of your algorithm.

6. [4 pts] In Dijkstra's algorithm, all distances are initialized to "infinity". Suppose, however, that you cannot represent infinity in your programming language. What actual number could be used in place of "infinity"?

7. [10 pts] You have just won a shopping spree on a game show. You are allowed to fill your shopping cart with everything the cart can hold. The cart has a weight capacity of W. All of the items in the store are marked with both the price and the weight. You are allowed to take multiple items.

   7.1. [4 pts] Devise an efficient algorithm to maximize your profit subject to the shopping cart constraints.

   7.2. [2 pts] Apply your algorithm to this problem instance:

   W = 20 lbs
   Store items: item A: ($160, 7 lbs). item B: ($90, 3 lbs), item C:($15, 2 lbs)

   7.3. [4 pt] For capacity W, and store item list of length N, give the complexity of your algorithm in terms of both W and N.

8. [5 pts] You are given a list of meeting times as pairs of numbers. For example:

   #1 (7:30, 9) = from 7:30 AM to 9 AM
   #2 (15, 16:30) = from 3 PM to 4;30 PM
   #3. (8, 11) = from 8 AM to 11 AM

   In the list, some meetings will overlap. For example meeting #1 and meeting #3 overlap.
   For this problem, you should find the most efficient possible algorithm to merge all overlapping meetings.
   Your output should be a list of new meeting times that do not overlap.

   In the given example, your final output would be:

   #1 (7:30, 11) = 7:30 AM to 11 AM, merge #1 & #3 above
   #2 (15, 16:30). = 3 PM to 4:30 PM (nothing to merge)

   Note that consecutive meetings should be merged. (2, 3) and (3, 4) should be merged into (2,4)

   Similarly, meetings that are subsumed should be merged into the inclusive meeting. (1, 5) and (2, 4) should be merged into (1, 5).

   The input to your merging algorithm will be an array of N pairs of starting and ending times for meetings. Furthermore, meeting start and end times are guaranteed to be on half hour boundaries, between 8:00 AM and 5:30. PM. Finally, the times will be given in military time. 8:00 AM is 800; 5:30 PM is 1730.

   In addition to showing your algorithm, analyze the complexity of your algorithm in terms of N (the number of input pairs).

9. [5 pts] Write an algorithm that computes nonnegative integer powers of arbitrary 3x3 matrices:

   `def pow3x3(M,n)` = M^n, where M is an arbitrary 3 x 3 matrix.

10. [10 pts] A cycle in a directed graph. G = (V,E) is defined as a path $<v_1,v_2,\ldots v_l>$, where the last vertex is the same as the first vertex.

    Given a directed graph G=(V,E), determine if G has a directed cycle.

    Hint: Use Depth First Search