23su CSE332 Final 1

Full Name: Solution

Email Address (UW NetID): This_better_not_be_a_number **@uw.edu**

Instructions:

- The allotted time is 1 hour.
- Do not turn the page until the staff says to do so.
- Read the directions carefully, especially for problems that require you to show work or provide an explanation.
- This is a closed-book and closed-notes exam.
- You are NOT permitted to access electronic devices including calculators.
- You must put your final answer inside the box.
    - If you run out of space, indicate where the answer continues.
    - Try to avoid writing on the very edges of the pages as we scan the exams.
- Unless otherwise noted, any bounds must be the **worst-case**, **simplified** and **tight**.
- Unless otherwise noted, $\log$s are base 2.
- Unless otherwise noted, all material is assumed as in lecture.
- For answers that involve bubbling in a ○ or □, fill in the shape completely.
- A formula sheet has been included at the end of the exam.

Advice:

- If you feel like you're stuck on a problem, you may want to skip it and come back at the end if you have time.
- Look at the question titles on the cover page to see if you want to start somewhere other than problem 1.
- **Relax and take a few deep breaths. You've got this! :-).**

Q1: Short Answer Questions (17 pts)

- O, Ω, or Θ bound must be **simplified** and **tight**. This means that, for example, $7n + 3 \in O(7n + 3)$ (not simplified) or $n \in O\left(2^{n!}\right)$ (not tight enough) are unlikely to get points.
- Unless otherwise specified, all $\log$s are base 2.
- For questions with a mathematical answer, you may leave your answer as an unsimplified formula (e.g., $7 \cdot 103$).
- Points are all or nothing (i.e., we will only grade what is inside the box).

a) (2 pts) True or False: Even when a `QuadraticProbingHashTable` has a prime number TableSize, it's still possible for cycling to occur.

[✓ True] False

b) (2 pts) Give the worst-case runtime of a delete operation on a `ChainingHashTable` containing N elements, where each bucket is an unsorted linked list. The load factor of this table is $\log M$.

$O(\boxed{N})$

c) (2 pts) Give the worst-case runtime of inserting N elements in an empty `ChainingHashTable` where each bucket is an unsorted linked list.

$O(\boxed{N^2})$

d) (1 pt) As in lecture, give a stable, **not** in-place, $n \log n$ sorting algorithm.

Merge Sort

e) (2 pts) True or False: A `Queue` (instead of a `Stack`) can be used to implement `DFS`.

True [✓ False]

f) (2 pts) If 30% of your program must be run sequentially (and the rest can be parallelized), what is the maximum speedup you would expect to get with infinite processors?

$\boxed{\frac{10}{3}}$ times

Bonus Question (0 pts): Who is credited with this law?

(Gene) Amdahl

g) (1 pt) Give the worst-case span of parallel merge sort (as in lecture, using parallel sort and merge).

$O(\boxed{\log^3 N})$

h) (2 pts) True or False: Using a reentrant lock for read-only operations will eliminate all possibilities of potential data race situations.

True [✓ False]

i) (2 pts) True or False: The topological sort algorithm can be run on a **cyclic** graph without giving an incorrect topological ordering.

True [✓ False]

The intent of this question was meant to be **True** at first as the standard algorithm for topological sorting can detect cycles and not output any topological ordering. This is why it was worded with a double negative. However, the topological sort algorithm presented in lecture cannot detect cycles.

j) (1 pt) True or False: For Kruskal's algorithm, does processing the edges in ascending order of weight produce the correct results?

[✓ True] False

Q2: Hashing (12 pts)

a) (6 pts) Double Hashing

Consider the following hash table of `TableSize = 10` using double hashing with the following hash functions:

$$h(key) = key \,\%\, 10$$
$$g(key) = 7 - (key \,\%\, 7)$$

i) (1 pt) Insert the numbers `23, 45, 13, 29, 37, 9, 19` into the table. It is possible that some elements cannot be inserted.

0	
1	19
2	
3	23
4	13
5	45
6	
7	37
8	
9	29

ii) (1 pt) If any elements cannot be inserted, write them here:

$\boxed{9}$

iii) (1 pt) What is the load factor of the above hash table? Write your answer as a precise fraction without simplifying as per the definition.

$$\boxed{\frac{6}{10}}$$

b) (6 pts) Quadratic Probing

Consider the following hash table of `TableSize = 10` using quadratic probing and the hash function $h(key) = key \% 10$

i) (2 pts) Using lazy deletion, delete the numbers `91, 3, 31, 66` from the hash table on the left (then put the new hash table after deletion on the right):

0	11
1	91
2	12
3	3
4	23
5	31
6	66
7	
8	
9	

0	11
1	~~91~~
2	12
3	~~3~~
4	23
5	~~31~~
6	~~66~~
7	
8	
9	

ii) (2 pts) Suppose that we make the following change to lazy deletion within a table using quadratic probing. When deleting an element, if the next slot in the probe sequence is empty, then we can mark the slot as 'empty' instead of 'deleted'.

True or False: Future search operations will work.

○ **True** [✓ **False**]

No, future search operations may not work. Consider the following hash table with hash function $h(key) = key \% 7$

0	**1**	**2**	**3**	**4**	**5**	**6**
		2	3			9

Now suppose that we remove 3. The initial hash location is given by 3 % 7 = 3 and the next hash location is given by $(3 + 1^2)\%7 = 4$. Since the next slot in the probe sequence (i.e. slot 4) is empty, we would mark slot 3 as 'empty'.

However, when we search for 9, the initial hash location is given by $9\%7 = 2$. Since this is occupied, we keep searching. The next hash location is given by $(9 + 1^2)\ \%\ 7 = 3$. Since this is now marked empty, we would stop searching. This is incorrect since 9 is actually present in the hash table.

iii) (2 pts) Consider Device X, a small computer with limited memory. As the designer of Device X, you are faced with the challenge of incorporating a hash table to manage key-value pairs efficiently. Device X must handle numerous key-value pairs reliably, but high-speed performance is not a critical requirement.

Given this scenario, choose the best hash table type.

Separate [✓ **Linear**] **Quadratic**

Linear probing. Since Device X has limited memory, we are unable to use additional data structures such as linked lists. This eliminates separate chaining.

Since Device X is likely to handle numerous key-value pairs while having limited memory, the hash table is likely to be at a high load factor (i.e. > 0.5). This means that quadratic probing is not guaranteed to find a slot such that it will likely be unreliable.

In contrast, linear probing is guaranteed to find an empty slot, even at very high load factors. Since performance is not a critical requirement, it is fine for linear probing to make multiple probes.

Q3: Sorting (12 pts)

You are given an initial array: `[50, 42`$_1$`, 42`$_2$`, 37, 34]`.

For each of the arrays shown below, indicate whether they could represent **one or more** of the given sorting algorithms **at any point** (after an iteration has completed) during the sorting algorithm's execution. If none of the sorting algorithms could have the given state, select **None**.

Assume all algorithms work exactly as in lecture.

Note: For Radix sort, if you have following items in buckets 0-9:
`0:[a], 1:[b, c], 2:[d], 3:[], 4:[], 5:[e, f], 6:[], 7:[g], 8:[], 9:[]`
we would write the resulting array as: `[a, b, c, d, e, f, g]`.

a) (3 pts) `[34, 42`$_1$`, 42`$_2$`, 37, 50]`

○ **Insertion Sort**	[✓ **Selection sort**]	○ **Radix sort**	○ **None**

b) (3 pts) `[34, 37, 42`$_1$`, 42`$_2$`, 50]`

[✓ **Insertion Sort**]	○ **Selection sort**	[✓ **Radix sort**]	○ **None**

c) (3 pts) `[50, 42`$_1$`, 42`$_2$`, 34, 37]`

○ **Insertion Sort**	○ **Selection sort**	[✓ **Radix sort**]	○ **None**

d) (3 pts) `[42`$_1$`, 42`$_2$`, 50, 37, 34]`

[✓ **Insertion Sort**]	○ **Selection sort**	○ **Radix sort**	○ **None**

Q4: Graphs (18 pts)

a) (2 pts) Given a **Dense**, **Directed** Graph, which graph implementation would be preferable for computing the reverse graph (i.e. a graph where all edges are reversed)?

$\boxed{\text{Adjacency Matrix}}$

b) (2 pts) Give a simplified, tight bound for the runtime of computing the reverse graph. Express your answer in terms of V (number of vertices) only.

In an adjacency matrix or List, $E \in O\left(V^2\right)$. This is not technically the tightest overall, but is the tightest bound if you can only answer in terms of V.

$O(\boxed{V^2})$

c) (2 pts) Give an exact number for the maximum number of edges in an **Undirected**, **Acyclic** graph. Express your answer in terms of V (number of vertices) only.

$\boxed{V - 1}$

d) (2 pts) A WinstonHeap has these worst-case runtimes:

`insert` - $O(1)$
`deleteMin` - $O(1)$
`decreaseKey` - $O(1)$

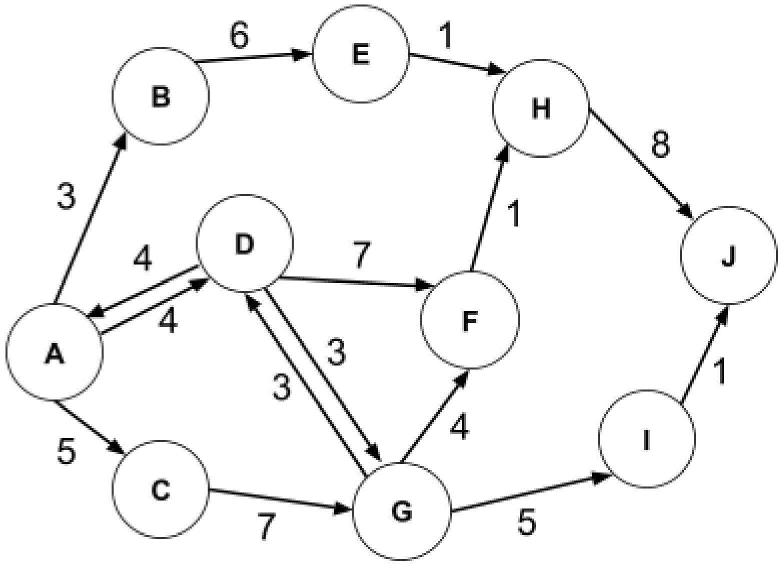
Give the worst-case runtime for Dijkstra's algorithm using a WinstonHeap. Express your answer in terms of V (number of vertices) only.

Dijkstra's $\log V$ bottleneck is due to `deleteMin` and `decreaseKey` (`changePriority`) operations. With WinstonHeap, Dijkstra's runs in $O(E + V)$.

$O(\boxed{V^2})$

This page has been intentionally left blank.

e) Dijkstra's



i) With the graph above, step through Dijkstra's Algorithm to calculate the **<u>single source shortest path from A</u>** to every other vertex. Break ties by choosing the lexicographically (alphabetically) smallest letter first (i.e., if B and C were tied, you would explore B first). Note that later on, you will need to recall what order vertices were declared *visited*. For full credit, you should only show the final values (use scratch paper or erase old values).

Node	Visited?	Cost	Predecessor
A	Y	0	-
B	Y	3	A
C	Y	5	A
D	Y	4	A
E	Y	9	B
F	Y	11	D
G	Y	7	D
H	Y	10	E
I	Y	12	G
J	Y	13	I

ii) In what order would Dijkstra's algorithm mark each node as *visited*?

$\boxed{A} \rightarrow \boxed{B} \rightarrow \boxed{D} \rightarrow \boxed{C} \rightarrow \boxed{G} \rightarrow \boxed{E} \rightarrow \boxed{H} \rightarrow \boxed{F} \rightarrow \boxed{I} \rightarrow \boxed{J}$

iii) List the shortest path from A to J (Give the actual path, not the cost).

A ➜ $\boxed{D➜G➜I}$ ➜ J

CSE 332: Data Structures and Parallelism

Useful Math Identities

Summations

1. $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$ for $|x| < 1$
2. $\sum_{i=0}^{n-1} 1 = \sum_{i=1}^{n} 1 = n$
3. $\sum_{i=0}^{n} i = 0 + \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
4. $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$
5. $\sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2 = \frac{n^4}{4} + \frac{n^3}{2} + \frac{n^2}{4}$
6. $\sum_{i=0}^{n-1} x^i = \frac{1-x^n}{1-x}$
7. $\sum_{i=0}^{n-1} \frac{1}{2^i} = 2 - \frac{1}{2^{n-1}}$

Logs

1. $x^{\log_x n} = n$
2. $a^{\log_b c} = c^{\log_b a}$
3. $\log_b a = \frac{\log_d a}{\log_d b}$

This page has been intentionally left blank.