Basic Algorithms CSCI-UA.0310
Computer Science Department, New York University Final Exam

NetID:	Name:

Do not open this booklet until you are directed to do so. Read all the instructions on this page.

Instructions

- **When the exam begins, write your name on every page of this booklet.**
- **Answer each question in the provided box. If you need more space, use the extra box towards the end of the booklet, and clearly indicate it in the original question's box. Do not use the back side.**
- **The exam is 110 minutes long. The maximum possible score is 100 points.**
- **Grading is based on correctness, clarity, and conciseness.**
- **If you don't know how to answer a question, please say so, and explain what you tried. You will get significantly more points doing so.**
- **You may use any algorithm or theorem we saw in class without proof, as long as you state it correctly. For all questions asking you to give algorithms, you do *not* have to give detailed pseudo-code, or even any pseudo-code. It is enough to give a clear description of your algorithm.**
- **This exam is closed book. No calculators, phones, smartwatches, etc. are permitted.**
- **Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.**
- **Good luck!!**

1. **(15 Points)**

(a) Check the correct answer(s) and give a short justification. **There may be multiple correct answers; check all correct answers.**

If $f(n) = 4n^2$ and $g(n) = 2^{\frac{n}{4}}$:

Solution: ☐ $f = \Omega(g)$ ☐ $f = O(g)$ ☐ $f = \Theta(g)$

(b) Check the correct answer(s) and give a short justification. **There may be multiple correct answers; check all correct answers.**

If $f(n) = 4\log(n^5)$ and $g(n) = n^{\log(5n)}$:

Solution: ☐ $f = \Omega(g)$ ☐ $f = O(g)$ ☐ $f = \Theta(g)$

(c) Give a Θ-expression for the solution of the following recurrence:

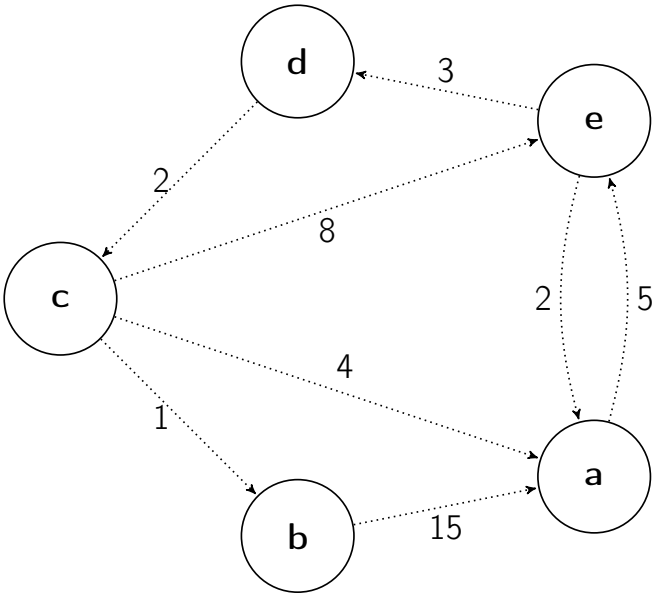
$$T(n) = 2T(n/3) + n$$

Assume $T(1) = 1$. Briefly justify your solution. You may assume that n is a power of 3.

Solution: $T(n) = \Theta($_____$)$

2. **(15 Points) Algorithmic Understanding**

(a) (8 points) Consider the following weighted directed graph $G = (V, E)$:



Run Dijkstra's single-source shortest-path algorithm with source vertex a on G. Assume that both the vertices and the adjacency lists are traversed in *alphabetic order*, i.e., when looping overall all vertices they are ordered as (a, b, c, d, e), and when looping over the adjacency list of a vertex its neighbors are ordered alphabetically (e.g., $Adj(e) = (a, d)$ and not (d, a)).

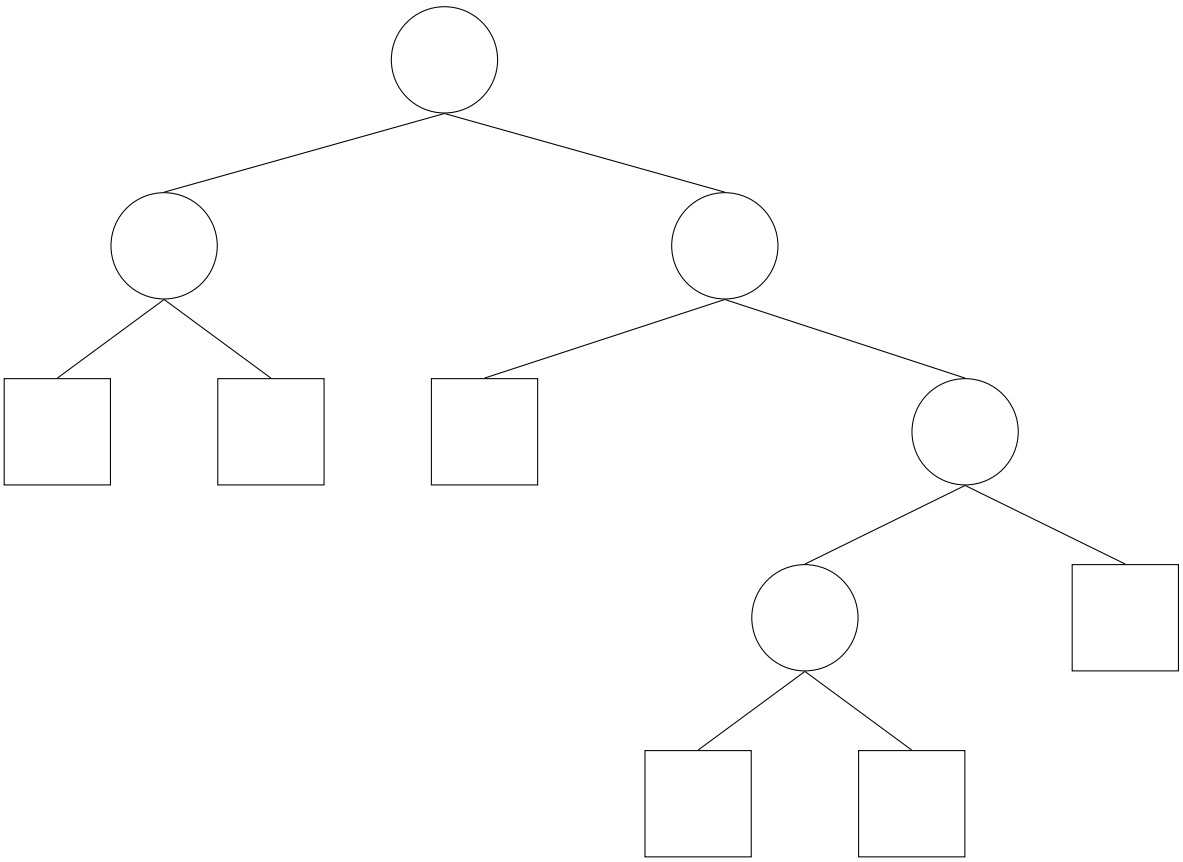
For each node v, fill all the values that $v.d$ takes during the course of the execution of Dijkstra, in the order in which these values appear. Note that the first value is ∞ for all nodes but a. Write only changes to a value, not repetitions of the same value (e.g., if $b.d = \infty$ and after the first step of Dijkstra we still have $b.d = \infty$, do not write the value ∞ twice.) Again, mind the alphabetic ordering.

	$v.d$
a	0
b	∞, _____
c	∞, _____
d	∞, _____
e	∞, _____

(b) (7 points) Assume that letters A to F are given with the following frequencies:

A	B	C	D	E	F
0.19	0.25	0.18	0.06	0.21	0.11

1. Complete the corresponding Huffman tree below by placing the letters into the respective leaves. (You do not need to write anything in the inner nodes.) Where the order is unspecified by Huffman's algorithm, place the letter with the lower frequency to the left of the letter with the higher frequency.



2. Decode the following word according to the Huffman Code that you constructed above:

$$00111011100 = \_\_\_\_\_$$

3. **(15 Points) Double Knapsack**

Frustrated with not being able to steal all the treasure, the thief from the knapsack problem returns with *two knapsacks*. Help the thief decide what items to take:

There are n items with positive integer weights $w_1, \ldots, w_n \in \mathbb{Z}_+$ and corresponding non-negative real values $s_1, \ldots, s_n \geq 0$. You have *two* knapsacks, with weight limits $W_1 \in \mathbb{Z}_+$ and $W_2 \in \mathbb{Z}_+$, respectively. You want to choose, for each knapsack, a subset of the items to pack into that knapsack. The goal is to maximize the *total value* (the sum over both knapsacks), while not exceeding the weight limit of each knapsack (total weight $\leq W_1$ for the first knapsack, and $\leq W_2$ for the second). Items cannot be duplicated; each item can be placed into at most one of the two knapsacks (not both), and items cannot be broken into two.

Let $K[i, u, v]$ denote the value of the most valuable solution using at most items $1, \ldots, i$ and weight bounds u and v for the two knapsacks, respectively.

(a) (2 points) For any u and v, the base case is $K[0, u, v] =$ _____

(b) (10 points) Complete the following recursive formula for $K[i, u, v]$, for $1 \leq i \leq n$. Note that the formula computes the maximum over three terms, where two of them are only valid under a certain condition. Write the condition as well. (If the condition does not hold, the corresponding term is not considered when taking the maximum.)

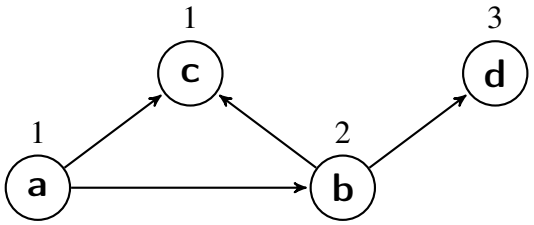
$$K[i, u, v] = \max \begin{cases} \_\_\_\_\_ & \text{(if } \_\_\_\_\_\text{)} \\ \_\_\_\_\_ & \text{(if } \_\_\_\_\_\text{)} \\ \_\_\_\_\_ & \end{cases}$$

(c) (3 points) Specify the running time of a bottom-up dynamic programming algorithm for filling the entire table K. Your run time should be a function of n, W_1, and W_2.

$$\Theta\Big(\_\_\_\_\_\Big)$$

4. **(22 Points) Reachable Colors**

(a) (7pt points) Consider a directed acyclic graph $G = (V, E)$, where each vertex has one of k colors, represented by numbers $1, \ldots, k$. Given a color $C \in \{1, \ldots, k\}$, we want to determine for each vertex $v \in V$ whether there is a vertex with color C that is reachable from v by a directed path in G. In the following example (with $k = 3$) the color 3 is reachable from vertices a, b, d, but not from vertex c.



Describe an algorithm that takes a directed acyclic graph G and a color $C \in \{1, \ldots, k\}$, and determines for each vertex v whether a vertex with color C can be reached from v, by setting a field $v.r$ to *true* or *false*. (In the example above, the output of your algorithm with $C = 3$ should be $a.r =$ *true*, $b.r =$ *true*, $c.r =$ *false*, $d.r =$ *true*.)

For full credit, your algorithm should run in $O(|V| + |E|)$ time. You do *not* need to justify the correctness of your algorithm, but do describe how it works in words.

(b) (7pt points) Now we wish to find the vertex u with the largest number of colors reachable from it (including the color of u itself). If there are multiple vertices that maximize this, the algorithm may return any arbitrary one of them. In the example from part (a), the output should be either vertex a or b, as from both vertices all three colors are reachable, and vertices c, d each have only one color that is reachable from them.

For full credit, your algorithm should run in time $O(k \cdot (|V| + |E|))$. You do *not* need to justify the correctness of your algorithm, but do describe how it works in words. (**Hint**: Extend your solution from part (a).)

(c) (8pt points) Extend your algorithm to solve the problem in part (b) when G is any directed graph (not necessarily acyclic).

5. **(18 Points) MST in Very Sparse Graphs**

In this question we show how to find a minimum-weight spanning tree (MST) in graphs where the number of edges is exactly the same as the number of vertices, with faster running time than Kruskal or Prim. Assume that $G = (V, E)$ is an undirected, *connected* graph with n vertices and n edges, and $w : V \to \mathbb{R}^+$ is weight function such that *no two edges have the same weight*.

(a) (10 points) Complete the following algorithm such that it outputs the edges of the MST in time $O(n)$.

(**Hint**: Recall that any spanning tree for a graph with n nodes has exactly $n-1$ edges. Therefore, G must have a cycle C.)

1. Find a cycle C in $O(|V| + |E|) = O(n)$ time as follows (be precise about which algorithm from the course you will run, and how you will identify the edges that belong to the cycle):

2. Delete the _____ edge on the cycle C.
3. Output the remaining edges as the MST.

(b) (8 points) Prove the correctness of your algorithm from part (a). To this end, show that for any cycle C of a graph, the edge you deleted cannot be part of the MST.

6. **(15 Points) Short Answers**

Each multiple-choice question below has exactly one correct answer. Clearly write the correct answer **in the box**. You do **not** have to justify your answers. There is no partial credit.

(a) (3 points) Assume in QuickSort, at each recursive step, we choose the middle element in the unsorted array as the pivot (ignoring rounding issues). Then, the *worst-case* running time is:

(A) $\Theta(n^2)$
(B) $\Theta(n \log n)$
(C) $\Theta(n)$
(D) none of the above

(b) (3 points) In class we saw that Dijkstra's algorithm computes shortest paths when there are no negative-weight edges, but now we want to run it on a graph that can have some edges with negative weights. Consider a directed, weighted graph G that can have negative-weight edges, and a start vertex s with no incoming edges. When is Dijkstra's algorithm *guaranteed* to correctly compute shortest paths from s to all other vertices?

(A) there is exactly one negative-weight edge.
(B) all negative-weight edges are out-edges of s.
(C) there is no negative-weight cycle.
(D) there is no cycle at all in G.
(E) none of the above.

(c) (3 points) We can topologically sort a directed acyclic graph by running DFS, and then sorting the vertices in

(A) increasing order of finish time.
(B) increasing order of discovery time.
(C) decreasing order of finish time.
(D) decreasing order of discovery time.
(E) none of the above.

(d) (3 points) Is the following problem decidable (i.e., can it be solved by an algorithm): given a program P and an input x, output "yes" if P does *not* terminate when it is executed on input x, and "no" if it *does terminate* on input x.

Circle the correct answer: Yes / No .

(e) (3 points) Is the following decision problem in NP?

$$\mathcal{CN} = \{n \in \mathbb{N} \mid n \text{ is NOT a prime number}\}$$

Circle the correct answer: Yes / No .

Extra sheet of paper #1. If you use this sheet make a note of it on the original question.

Solution:
Question number:

Extra sheet of paper #2. If you use this sheet make a note of it on the original question.

Solution:
Question number:

Scratch paper: anything written here will NOT be graded!

Scratch paper: anything written here will NOT be graded!

Cheat Sheet

Logarithm Rules

The following holds for any basis b (and c) and values X, Y.

$$\begin{aligned}
\log_b(b^X) &= X \\
b^{\log_b(X)} &= X \\
\log_b(X \cdot Y) &= \log_b(X) + \log_b(Y) \\
\log_b\left(\frac{X}{Y}\right) &= \log_b(X) - \log_b(Y) \\
\log_b(X^Y) &= Y \cdot \log_b(X) \\
\log_b(X) &= \frac{\log_c(X)}{\log_c(b)}
\end{aligned}$$

Arithmetic Series

Let d be a constant such that $a_n := a_{n-1} + d$ for all $n \geq 1$ and a_0 some initial value. We have

$$a_0 + a_1 + \ldots a_k = \sum_{i=0}^{k} a_i = \frac{k+1}{2}(a_0 + a_k).$$

Geometric Series

For a *finite* geometric series we have

$$a + a \cdot r + a \cdot r^2 + \cdots + a \cdot r^k = \sum_{i=0}^{k} a \cdot r^i = \frac{a \cdot (1 - r^{n+1})}{1 - r}$$

and in particular

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1 = \Theta(2^k).$$

If $|r| < 1$, then the *infinite* geometric series converges:

$$a + a \cdot r + a \cdot r^2 + \cdots = \sum_{i=0}^{\infty} a \cdot r^i = \frac{a}{1 - r}$$

and in particular

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 2 = \Theta(1).$$