

- All problems are worth 15 points. Unless states otherwise, subproblems are weighted equally.
- Please write down name and student ID in all pages.

Problem 1

Determine whether each statement is True or False and briefly justify your answer for each statement. Assume that every given function here is positive. If the statement is false, providing a counter-example is enough.

(a) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$.

False. Counterexample: $f(n) = n$, $g(n) = n^2$

Grading Criteria:

(+1 points) If you got the correct answer (True or False).

(+2 points) If you got the correct justification.

(b) If $f(n) = \Omega(g(n))$, then $f(f(n)) = \Omega(g(g(n)))$.

False. Counterexample: $f(n) = 2^n$, $g(n) = 2^{n+1}$.

Grading Criteria:

(+1 points) If you got the correct answer (True or False).

(+2 points) If you got the correct justification.

(c) If $f(n) = \Theta(g(n))$ and $g(n) = \Omega(h(n))$, then $f(n) = \Theta(h(n))$.

False. Counterexample: $f(n) = n^2$, $g(n) = n^2$, $h(n) = n$.

Grading Criteria:

(+1 points) If you got the correct answer (True or False).

(+2 points) If you got the correct justification.

(d) $f(n) + o(f(n)) = \Theta(f(n))$.

True. Let g be any function such that $g(n) = o(f(n))$. Since g is asymptotically positive, let n_0 be such that $n \geq n_0$ implies $g(n) \geq 0$. Then $f(n) + g(n) \geq f(n)$ so $f(n) + o(f(n)) = \Omega(f(n))$. Next, choose n_1 such that $n \geq n_1$ implies $g(n) \leq f(n)$. Then $f(n) + g(n) \leq f(n) + f(n) = 2f(n)$, so $f(n) + o(f(n)) = O(f(n))$. This implies $f(n) + o(f(n)) = \Theta(f(n))$.

Grading Criteria:

(+1 points) If you got the correct answer (True or False).

(+2 points) If you got the correct justification.

(e) $f(n) = \Theta(g(n))$ if and only if $f(n) = o(g(n))$ and $f(n) = \omega(g(n))$.

False. Counterexample: $f(n) = n$, $g(n) = n$

Grading Criteria:

(+1 points) If you got the correct answer (True or False).

(+2 points) If you got the correct justification.

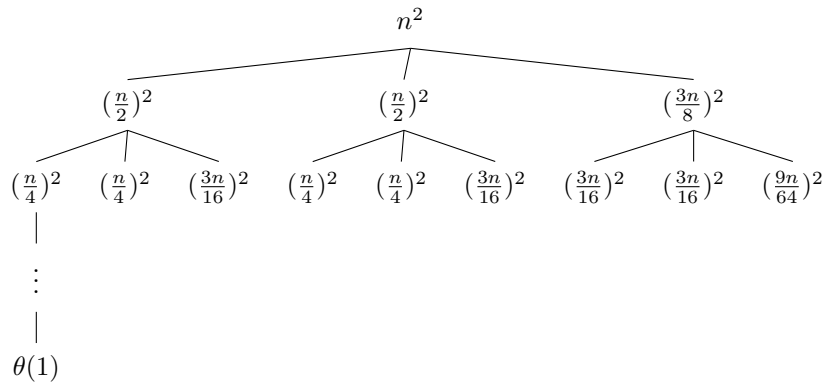
Problem 2

Using the indicated method, find a tight asymptotic bound on the recursion formula. The answer should contain the process of finding the tight asymptotic bound.

- (a) (5 points) Using the recursion tree method, find a tight asymptotic bound on the recursion formula.

$$T(n) = 2T\left(\frac{n}{2}\right) + T\left(\frac{3n}{8}\right) + n^2$$

Solution:



The k -th per-level cost of recursion tree is $\left(\frac{41}{64}\right)^k n^2$. The shortest depth is $\log_{\frac{8}{3}} n$ and longest depth is $\log_2 n$. Then the lower bound is the cost of full tree with longest depth, and upper bound is the cost of full tree with shortest depth.

The leaves number of full tree with shortest depth is $3^{\log_{\frac{8}{3}} n} = n^{\log_{\frac{8}{3}} 3}$.

$$n^2 \left(1 + \frac{41}{64} + \left(\frac{41}{64}\right)^2 + \dots + \left(\frac{41}{64}\right)^{\log_{\frac{8}{3}} 3 - 1}\right) + \theta(1) \times n^{\log_{\frac{8}{3}} 3} \leq T(n)$$

$$T(n) = \Omega(n^2)$$

Geometry series is constant. So, upper bound of $T(n) = \Omega(n^2)$.

The leaves number of full tree with longest depth is $3^{\log_2 n} = n^{\log_2 3}$.

$$T(n) \leq n^2 \left(1 + \frac{41}{64} + \left(\frac{41}{64}\right)^2 + \dots + \left(\frac{41}{64}\right)^{\log_2 3 - 1}\right) + \theta(1) \times n^{\log_2 3}$$

$$T(n) = O(n^2)$$

The lower bound of $T(n) = O(n^2)$. So the tight asymptotic bound is $\Theta(n^2)$.

Grading Criteria:

(+2 points) If your solution is $\Theta(n^2)$.

(+1 point) If you draw write recursion tree.

(+2 points) If you write the correct lower, upper bounds.

(-1 points) If you did not separate the leaves cost.

(-1 points) If you write wrong leaves number.

(-2 points) If you did not use recursion method. (If you write correct answer but not using recursion method, then you will get $+3 - 2 = +1$ point.)

(Note) If you use wrong values in process, each mistake deducts 1 point.

- (b) Using the substitution method, find a tight asymptotic bound on the recursion formula.

$$T(n) = 4T\left(\frac{n}{2}\right) + \sqrt{n}$$

Solution:

Guess that $T(n) = \Omega(n^2)$, and prove that there exists c and n_0 such that,

$$T(n) \geq cn^2 \quad \forall n > n_0 \text{ and } c > 0$$

For the base case, $T(1) = 1 > c1^2$ for $1 > c$.

Let's assume the guess is true for $m < n$. Then $T(m) \geq cm^2$.

$$T(n) = 4T\left(\frac{n}{2}\right) + \sqrt{n} \geq 4c\left(\frac{n}{2}\right)^2 + \sqrt{n} = cn^2 + \sqrt{n} \geq cn^2$$

Therefore, for $0 < c < 1$, $T(n) = \Omega(n^2)$.

Also, guess that $T(n) = O(n^2)$, and prove that there exists c_1, c_2 and n_0 such that,

$$T(n) \leq c_1n^2 - c_2\sqrt{n} \quad \forall n > n_0 \text{ and } c_1, c_2 > 0$$

For the base case, $T(1) = 1 \leq c_1 - c_2$ for $c_1 - c_2 \geq 1$.

Let's assume the guess is true for $m < n$. Then $T(m) \leq c_1m^2 - c_2\sqrt{m}$.

$$\begin{aligned} T(n) &= 4T\left(\frac{n}{2}\right) + \sqrt{n} \leq 4\left(c_1\frac{n^2}{4} - c_2\sqrt{\frac{n}{2}}\right) + \sqrt{n} \\ &\leq c_1n^2 - c_2\sqrt{n} - (2\sqrt{2}c_2\sqrt{n} - c_2\sqrt{n} - \sqrt{n}) \leq c_1n^2 - c_2\sqrt{n} - \sqrt{n}\{(2\sqrt{2} - 1)c_2 - 1\} \\ &\leq c_1n^2 - c_2\sqrt{n} \end{aligned}$$

Therefore, for bit enough c_1 and $c_2 > \frac{1}{2\sqrt{2}-1}$, $T(n) = O(n^2)$.

Grading Criteria:

(+3 points) If your solution is $\Theta(n^2)$.

(+2 points) If you write the correct lower, upper bound.

(-1 points) If you did not mention proper c range.

(-2 points) If you did use substitution method. (If you write correct answer but not using substitution method, then you will get $+3 - 2 = +1$ point.)

(Note) If you use wrong values in process, each mistake deducts 1 point.

- (c) Using the master theorem, find a tight asymptotic bound on the recursion formula.

$$T(n) = 2T\left(\frac{n}{2}\right) + n \lg n$$

Solution:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

In this recursion equation $a = 2, b = 2, f(n) = n \lg n$

$f(n) = n \lg n = \Theta(n^{\log_b a} \lg^k n)$ for constant $k = 1$. So, $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n) = \Theta(n \lg^2 n)$

Grading Criteria:

(+3 points) If your solution is $\Theta(n \lg^2 n)$.

(+1 points) If you use correct master theorem case.

(+1 points) If you calculate correctly.

(Note) If you use wrong values in process, each mistake deducts 1 point.

Problem 3

- (a) (7 points) Suppose we multiply two n -digit octet numbers X and Y using divide-and-conquer algorithms. We divide X into A and B , and Y into C and D , where A , B , C and D are $n/2$ -digit numbers.

$$X = g^{n/2}A + B$$

$$Y = g^{n/2}C + D$$

Consider the first implementation of such an algorithm whose recurrence relation of the time complexity is $T(n) = aT(n/b) + \Theta(n^c)$:

$$XY = g^n AC + g^{n/2}BC + g^{n/2}AD + BD$$

Next, we have the second implementation of an algorithm whose recurrence relation of the time complexity is $T(n) = dT(n/e) + \Theta(n^f)$:

$$XY = (g^n - g^{n/2})AC + g^{n/2}(A + B)(C + D) + (1 - g^{n/2})BD$$

Find the constants a, b, c, d, e, f, g and write in “ a, b, c, d, e, f, g ” format.

Solution:

The first implementation involves 4 multiplication of digits of size $n/2$. Conquering process contains 3 additions of digits which contributes as $\Theta(n)$.

The second implementation involves 3 multiplication of digits of size $n/2$. Conquering process contains 6 additions of digits which contributes as $\Theta(n)$.

g contributes as shifts for octet numbers. Therefore, the final answer is:

$$[4, 2, 1, 3, 2, 1, 8]$$

Grading Criteria:

(+1 points) For each correct number.

- (b) (6 points) Toom-Cook algorithm is a generalized version of the Karatsuba algorithm. Similar to Karatsuba algorithm, it divides the problem into k different subproblems. Toom-Cook 3-way multiplication is a widely known case of Toom-Cook multiplication where $k = 3$. The recurrence relation can be expressed as follows:

$$T(n) = 5T(n/3) + O(n)$$

Write down the running time of the algorithm in (a) and the Toom-Cook algorithm in Θ notation and conclude which one is asymptotically faster. Write down the whole process to get the full score (hint: $2^{1.58} \approx 3, 3^{1.46} \approx 5$).

Solution:

Using the master theorem, we can easily find the running times. For the first and second implementation, we have:

$$a = 2, \quad b = 4, \quad f(n) = n = O(n^{\log_2 4 - \epsilon}) \quad (\because \text{case 1}, 0 < \epsilon \leq 1)$$

$$T_{\text{first}}(n) = \Theta(n^2)$$

$$a = 2, \quad b = 3, \quad f(n) = n = O(n^{\log_2 3 - \epsilon}) \quad (\because \text{case 1}, 0 < \epsilon \leq \log_2 3 - 1)$$

$$T_{\text{second}}(n) = \Theta(n^{\log_2 3})$$

Likewise, the running time of Toom-Cook($k=3$) algorithm is:

$$a = 5, \quad b = 3, \quad f(n) = n = O(n^{\log_3 5 - \epsilon}) \quad (\because \text{case 1}, 0 < \epsilon \leq \log_3 5 - 1)$$

Since $\log_3 5 < \log_2 3 < 2$, Toom-Cook 3-way multiplication is asymptotically fastest algorithm.

Grading Criteria:

(+2 points) For getting correct running time for each algorithm.

(+2 points) Compare each running time and conclude that Toom-Cook 3-way multiplication is asymptotically faster.

(-1.5 points) For showing no process.

Problem 4

- (a) (5 points) Given an array $[9, 7, 5, 11, 12, 2, 14, 3, 10, 6]$, use the *QuickSort Adaptive* to sort the array. Instead of tracing each recursive call, describe the state of the array after the first two levels of recursion (i.e., after the first partitioning and then after the first partitioning of each of those partitions). This approach will show how the algorithm progresses in broader strokes, which is a different perspective from the detailed step-by-step trace.

Grading Criteria:

After the first partitioning with pivot 11 (the middle element): $[9, 7, 5, 3, 6, 2, 10, 11, 12, 14]$
 Pivot 11 is now at its correct position, with elements less than 11 to its left and elements greater than 11 to its right.

After partitioning the left part $[9, 7, 5, 3, 6, 2, 10]$ with pivot 5: $[2, 3, 5, 6, 7, 9, 10]$

After partitioning the right part $[12, 14]$ with pivot 12: $[12, 14]$

- (b) (5 points) Suppose *QuickSort Origin* is used to sort an array where all elements are identical, e.g., $[4, 4, 4, 4, 4]$. How many recursive calls to quicksort will be made before the algorithm terminates? Just write down the number of calls.

Grading Criteria:

8 calls or $2(n - 1)$ calls (5 points)

9 calls or $2n - 1$ calls (including initial call) (5 points)

4 calls or $(n - 1)$ calls (didn't consider that there are 2 QuickSort Origin calls within QuickSort Origin - 4 points)

5 calls or n calls (didn't consider that there are 2 QuickSort Origin calls within QuickSort Origin but including initial call - 4 points)

- (c) (5 points) If *QuickSort Origin* is used to sort an array in reverse order (e.g., $[5, 4, 3, 2, 1]$), what will be the total number of recursive quicksort function calls made to sort the entire array?

Grading Criteria:

8 calls or $2(n - 1)$ calls (5 points)

9 calls or $2n - 1$ calls (including initial call) (5 points)

4 calls or $(n - 1)$ calls (didn't consider that there are 2 QuickSort Origin calls within QuickSort Origin - 4 points)

5 calls or n calls (didn't consider that there are 2 QuickSort Origin calls within QuickSort Origin but including initial call - 4 points)

Problem 5

- (a) (5 points) Describe **the worst-case** scenario for the randomized quicksort algorithm. Explain the condition that leads to the worst-case scenario, and find the worst-case time complexity of quicksort in terms of n , the number of elements in the array.

Grading Criteria:

(+2 points) The worst-case is that only the maximum (or minimum) value is selected as a pivot.

(+3 points) Time complexity = $O(n^2)$

- (b) (5 points) Describe **the best-case** scenario for the randomized quicksort algorithm. Explain the condition that leads to the best-case scenario, and find the best-case time complexity of quicksort in terms of n , the number of elements in the array.

Grading Criteria:

(+2 points) The base-case is that the pivot divides the array into two equal-sized sub-arrays

(+3 points) Time complexity = $O(n \log n)$

- (c) (5 points) Find the recurrence relation $T(n)$ of the worst-case linear-time selection algorithm, when the input array is divided into groups of 7 instead of 5.

$$T(n) \leq T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + O(n)$$

Grading Criteria:

(+2 points) $\frac{n}{7}$

(+2 points) $\frac{5n}{7}$

(+1 points) n

Problem 6

- (a) When drawing the DFS tree, write the number of vertices for each depth. The depth of the root node is 0. (e.g. If there are 1 vertex for depth 0 and 2 vertices for depth 1, write in the form of $\{0 : 1, 1 : 2\}$)

Solution:

$\{0 : 1, 1 : 2, 2 : 1, 3 : 2, 4 : 2\}$

Grading Criteria:

(+5 points) Solution is correct.

(+3 points) Solution is wrong, but the correct DFS tree is drawn.

- (b) Fill in the table below showing the number of outgoing edge by edge type of each vertex.

Solution:

	tree edge	forward edge	back edge	cross edge
<i>A</i>	2	0	0	0
<i>B</i>	0	0	2	0
<i>C</i>	1	0	0	0
<i>D</i>	0	0	1	0
<i>E</i>	0	0	0	1
<i>F</i>	2	0	0	0
<i>G</i>	2	0	0	0
<i>H</i>	0	0	0	1

Grading Criteria:

(+5 points) Solution is correct.

(-1 points) per incorrect one vertex(one row). But the minimum score is 0.

If you write connecting vertex instead of the number, 50% of the total score is deducted.

- (c) Fill in the table below showing the discoveryTime and finishTime for each vertex when you finish the above algorithm.

Solution:

	discoveryTime	finishTime
A	1	28
B	7	10
C	3	23
D	14	16
E	18	20
F	12	21
G	5	22
H	25	27

Grading Criteria:

(+5 points) Solution is correct.

(-1 points) per incorrect one vertex(one row). But the minimum score is 0.

Problem 7

- (a) (6 points) Find all the strongly connected components(SCCs) of G (if the node u and node v are a SCC, write in form of $\{u, v\}$).

Solution:

$\{a, b, d, e\}, \{c, f\}, \{g, j, k\}, \{h, i, m\}, \{l\}, \{n, o\}$

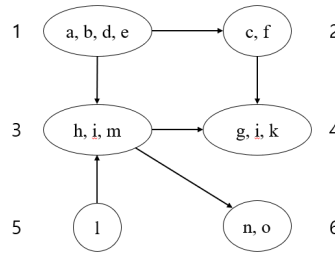
Grading Criteria:

(+1 points) Each correct answer.

(-1 points) If 6 correct answers + additional wrong answers.

- (b) (2 points) Find the source vertices and sink vertices (The vertices should be in the form of SCC).

Solution:



Source: $\{a, b, d, e\}, \{l\}$

Sink: $\{g, j, k\}, \{n, o\}$

Grading Criteria:

(+0.5 points) Each correct answer.

(-0.5 points) Each additional wrong answer.

(-0.5 points) Source is not SCC form.

(-0.5 points) Sink is not SCC form.

- (c) (5 points) How many possible topological orderings does the meta-graph of G have?

Solution:

For convenience, rename each node of the meta-graph of G as 1 through 6 as shown above. Then, let's focus on node 3. If arranged in order, nodes 1 and 5 should be on the left side of node 3, nodes 4 and 6 should be on the right side of node 3, and the order of the same side doesn't matter. Then we only need to count the positions where node 2 can lie between nodes 1 and 4.

$1, 5, 3, 4, 6 \rightarrow 3$

$1, 5, 3, 6, 4 \rightarrow 4$

$5, 1, 3, 4, 6 \rightarrow 2$

$5, 1, 3, 6, 4 \rightarrow 3$

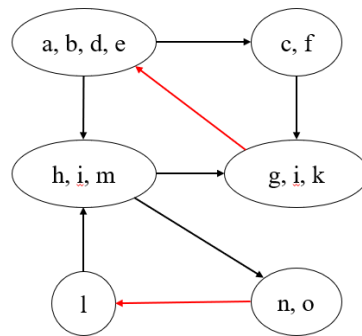
Thus, the answer is $3 + 4 + 2 + 3 = 12$

Grading Criteria:**(+5 points)** Correct answer.

- (d) (2 points) What is the minimum number of edges you must add to G to make it strongly connected?

Solution:

Consider edges that make the meta-graph strongly connected. No matter which edge is selected between the vertices in the SCC connected by those edges, it makes G strong connected.

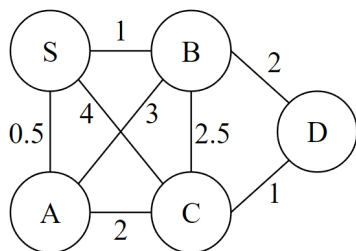


Thus, the answer is 2.

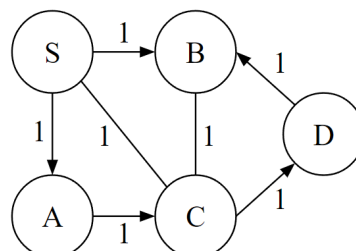
Grading Criteria:**(+2 points)** Correct answer.

Problem 8

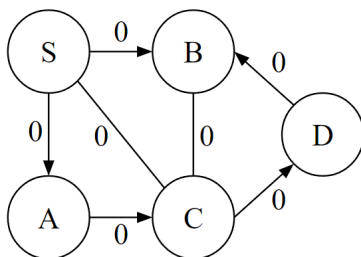
- (a) (3 points) Decide whether Dijkstra's algorithm can get the shortest distance between vertices on the following graphs (circle *yes* or *no*). If the algorithm **cannot** be applied, provide a reason. Notice that there may exist a case where same vertex is visited more than twice.



(a) : **yes/no**

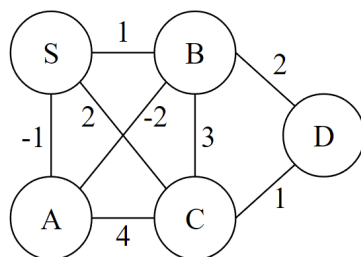


(b) : **yes/no**



(c) : **yes/no**

The graph contains non-positive
weight on edges.



(d) : **yes/no**

The graph contains negative weight
on edges.

Grading Criteria:

(+0.25 points) If your choice is correct.

(+1.0 points) If the provided reason is correct.

Below answers are also considered as **correct**.

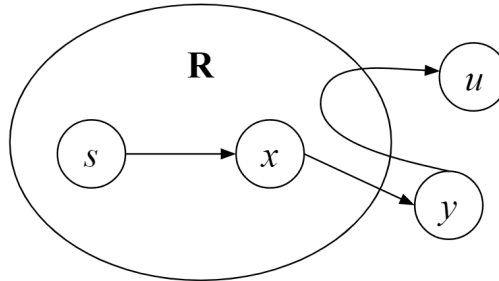
For 8(a) – (c), the line 11 in code does not run (since 0 is not bigger than 0).

For 8(a) – (d), there is an infinite cycle on S-A-B. When running algorithm, $\text{dist}(\text{node})$ is (some value) but actually it is (value smaller than obtained value)

Below answers are considered as **wrong**.

For 8(a) – (c), queue cannot differentiate min to pop out / minimum edge cannot be differentiated / priority queue cannot be build. There are multiple shortest path / there is no shortest path because all path is 0 / all path have same length.

- (b) (7 points) Using the property (i) and (ii), below proves the following theorem – Dijkstra’s algorithm terminates with $dist(v) = \text{distance to } v$ for all $v \in V$ – by contradiction.



- It suffices to show that $dist(v) = \text{distance to } v$ for every $v \in V$ when v is added to R .
- Suppose u is the first vertex added to R for which $dist(u) > \text{distance to } u$.
- Let y be the first vertex in $V - R$ along a shortest path from s to u , and let x be its predecessor.
- Since u is the first vertex violating the theorem, we have $dist(x) = \text{distance to } x$.
- ...

Write down the missing processes and show the contradiction.

When x was added to R , update the distance between $(x, y) - dist(x) + l(x, y) = dist(y)$. It implies that $dist(y) = \text{distance to } y \leq \text{distance to } u < dist(u)$.

It contradicts the initial assumption, $dist(u) \leq dist(y)$ (because u is added prior to y).

Grading Criteria:

(+2 points) get relationship between x and y (or y and u)

(+2 points) get the inequality of $dist(y) \leq \text{distance to } u \leq dist(u)$

(+3 points) show contradiction - u is in R , $dist(u) < dist(y)$. Contradiction.

Below answers are also considered as **correct**.

(+3 points) show contradiction - y is added to R prior to u , Contradiction.

Below answers are considered as **wrong**.

- (a) Applying property (i) from x until reaching the vertex u .

Since $dist(u)$ is corrected to distance to u , it is contradicting to the assumption.

→ this just ran the update operation, and saying it can find correct $dist(u)$.

This is insufficient for proving by contradiction (-3 points).

If there is no relationship between x and y , or y and u (-2 points).

- (b) y is the first vertex in $V - R$, $dist(y) > \text{distance to } y$

u should be the first vertex violating the theorem, but y violates first...

→ you should not assume that being out of R always not having shortest distance found.

(0 points).

- (c) when comparing x to u with $l(x, y) + l(y, u)$, you cannot use triangle inequality because both value are literally same.

- (c) (5 points) Run Dijkstra's algorithm, and get the shortest distance from S to D ($dist(D)$). In the table, write down the $dist(v)$ value (distance from S to particular vertex). First line is written down for you, and you must fill in all the blanks.

S	A	B	C	D
0	1	3	4	∞
0	1	2	3	∞
0	1	2	3	4
0	1	2	3	4

$dist(D) : 4$

Grading Criteria:

(+0.3 points) Each blank.

(+0.5 points) Correct $dist(D)$.

(Note) Writing $dist(D)$ without filling the table is 0 points.