1 Problem 1 (20 points: each subproblem is 4 points)

For each statement below, state if it is true or false.

1. $2^{n \cos(10^{-6}\pi n)} = \Omega(2^{\sqrt{n}})$.

   true false

2. $(\log \log n)^n = \omega(n^3)$

   true false

3. $n(\log n)^{\log n} = o(n^2)$

   true false

4. The running time of the partition step in the Quick Sort algorithm is $O(n)$.

   true false

5. Let $A = \{a_i\}_{i=1}^{n}$ be an array of integers, s.t. $\forall i: \ a_i \in \{1, \ldots, 1000^{10 \log n}\}$. One can sort A using Radix sort algorithm in linear time.

   true false

2 Problem 2 (50 points; each subproblem is 25 points)

Give asymptotic upper bounds for the following recurrences. You can use the master theorem when it is applicable. Assume that $T(0) = T(1) = 1$.

1. $T(n) = T(n/2) + 2\sqrt{n}$

$a = 1;\ b = 2;\ \log_b a = 0;\ f(n) = 2\sqrt{n} = \Omega(n^{0+\varepsilon})$

$\varepsilon = 0.1$

case 3 of MT

$\exists c < 1:\ a f(n/b) \le c f(n)$

$f(\frac{n}{2}) = 2\sqrt{\frac{n}{2}} \le c \cdot 2\sqrt{n}$

$c = \frac{1}{\sqrt{2}}$

$\Rightarrow T(n) = \Theta(f(n)) = \Theta(\sqrt{n})$

2. $T(n) = 2T(n-2) + 2^n$

$n = \log_4 k \Rightarrow T(n) = T(\lg k) = 2T(\lg k - 2) + 2^{\lg k}$

$= 2T(\lg k - \lg 4) + k = 2T(\lg \frac{k}{4}) + k$

$S(k) = T(\lg k) = 2T(\lg \frac{k}{4}) + k$

$S(k) = 2S(\frac{k}{4}) + k$

$a = 2;\ b = 4;\ \log_b 4 = \frac{1}{2};\ f(k) = k = \Omega(k^{\frac{1}{2}+\varepsilon})$

$\varepsilon = \frac{1}{4}$

MT case 3

$\exists c < 1:\ a f(k/b) \le c f(k)$

$2f(\frac{k}{4}) = \frac{k}{2} \le ck$

$c = \frac{1}{2}$

$\Rightarrow S(k) = \Theta(k)$

$\Downarrow$

$T(\lg k) = \Theta(k)$

$\Downarrow$

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

Algo:
(1) Scan through A and B and delete items $> n^3$
(2) Scan through A and replace each item a_i with $(x-a_i)$
call result array $\tilde{A}$
(3) Use radix sort on $\tilde{A}$ and B (all items are integers $< n^3+1$)
(4) merge $\tilde{A}$ and B and call result array C
(5) Scan C (which is sorted), if C has repeated items output Yes
otherwise output No

3 Problem 3 (90 points)

Given two **unsorted** arrays A and B of size n each, and an integer $x < n^3$, find whether there exists an element a_i from A and an element b_j from B whose sum is exactly $a_i + b_j = x$. Prove the correctness and provide the running time analysis for your algorithm. You can assume that all numbers in A are distinct and all numbers in B are distinct, i.e. $\forall i \neq j : a_i \neq a_j$ and $\forall i \neq j : b_i \neq b_j$.

45 points will be given if:
(1) your algorithm works correctly,
(2) your algorithm solves the problem by using the time $\omega(n \log n)$,
(3) your analysis is correct,
(4) your explanations and proofs are clear and with enough details.
If you cannot prove your claims formally, give your best intuition.

75 points will be given if:
(1) your algorithm works correctly,
(2) your algorithm solves the problem by using the time $O(n \log n)$,
(3) your analysis is correct,
(4) your explanations and proofs are clear and with enough details.
If you cannot prove your claims formally, give your best intuition.

Full credit will be given if:
(1) your algorithm works correctly,
(2) your algorithm solves the problem in $O(n)$ time,
(3) your analysis is correct,
(4) your explanations and proofs are clear and with enough details.
If you cannot prove your claims formally, give your best intuition.

RT: each step of the algorithm requires $O(n)$, no cycles
Thus total time is $O(n)$

Correctness: (1) $\exists a_i \in A;\ b_j \in B$ s.t. $a_i + b_j = x \Leftrightarrow \tilde{A} \cap B \neq \emptyset$
i.e. there is a common element in $\tilde{A}$ and B
(this statement can be proved by contradiction)
(2) using the fact that all items in A are unique
we conclude that all items in $\tilde{A}$ are unique
thus (C has repeated items) $\Leftrightarrow \tilde{A} \cap B \neq \emptyset$
due to the fact that all items in B are unique as well

Consider ~~the~~ a linear representation of A:

$\bar{A} = ((A_{11}, (1,1)), (A_{12}, (1,2)), \dots, (A_{nn}, (n,n)))$

value   coordinates

Sort $\bar{A}$ by value and call sorted array $\tilde{A}$ (all items $\le n^2$; $|\bar{A}| = n^2$ | using count sort $\rightarrow RT = O(n^2)$)

$\tilde{A} = (1, (i_1, j_1)), (2, (i_2, j_2)), \dots, (n^2, (i_{n^2}, j_{n^2})))$, where (i_a, j_a) is coordinate of value a.

4 Problem 4 (90 points)

Given an $n \times n$ matrix A, with all the values A_{ij} being **unique** and from the range $\{1, 2, \dots, n^2\}$. Use dynamic programming to find the maximum length path (starting from any cell) such that all cells along the path are in increasing order with the difference of 1. Formally, path is a sequence of pairs $\{(i_k, j_k)\}_{k=1}^{l}$, s.t. (1) $\forall k : |i_{k+1} - i_k| + |j_{k+1} - j_k| = 1$ and (2) $\forall k : A_{i_{k+1}j_{k+1}} - A_{i_k j_k} = 1$, where $i_k, j_k \in \{1, \dots, n\}$.

Valid input: $\begin{bmatrix} 1 & 6 & 7 \\ 8 & 5 & 9 \\ 3 & 4 & 2 \end{bmatrix}$ **Valid output:** $[3, 4, 5, 6, 7]$ $\begin{bmatrix} 1 & \boxed{6} & \boxed{7} \\ 8 & \boxed{5} & 9 \\ \boxed{3} & \boxed{4} & 2 \end{bmatrix}$

Invalid output 1: $[3, 4, 5, 6]$ $\begin{bmatrix} 1 & \boxed{6} & 7 \\ 8 & \boxed{5} & 9 \\ \boxed{3} & \boxed{4} & 2 \end{bmatrix}$ (not the longest)

Invalid output 2: $[3, 4, 5, 6, 7, 9]$ $\begin{bmatrix} 1 & \boxed{6} & \boxed{7} \\ 8 & \boxed{5} & \boxed{9} \\ \boxed{3} & \boxed{4} & 2 \end{bmatrix}$ $(9 - 7 \neq 1$ — violates (2))

Invalid output 3: $[6, 7, 8, 9]$ $\begin{bmatrix} 1 & \boxed{6} & 5 \\ \boxed{9} & 2 & \boxed{7} \\ 4 & \boxed{8} & 3 \end{bmatrix}$ (no diagonal steps allowed — violates (1))

Invalid input 1: $\begin{bmatrix} 1 & 6 & 7 \\ 8 & \boxed{2} & 9 \\ 3 & 4 & \boxed{2} \end{bmatrix}$ (item 2 appears twice)

Invalid input 2: $\begin{bmatrix} 1 & 6 & 7 \\ 8 & \boxed{32} & 9 \\ 3 & 4 & 2 \end{bmatrix}$ (item 32 is not from the range $\{1, \dots, n^2\}$)

45 points will be given if:
(1) your algorithm works correctly,
(2) your algorithm solves the problem by using the time $\omega(n^2)$,
(3) your analysis is correct,
(4) your explanations and proofs are clear and with enough details.
If you cannot prove your claims formally, give your best intuition.

Full credit will be given if:
(1) your algorithm works correctly,
(2) your algorithm solves the problem in $O(n^2)$ time,
(3) your analysis is correct,
(4) your explanations and proofs are clear and with enough details.
If you cannot prove your claims formally, give your best intuition.

$LP(a)$ - length of the longest path starting from (i_a, j_a) | $LP(a) = \begin{cases} LP(a+1) + 1; & \text{if } a \text{ and } a+1 \text{ are neighbors} \\ 1 & \text{otherwise} \end{cases}$

find sequentially ~~LP~~ $LP(n^2), LP(n^2-1), \dots, LP(1)$
Return the longest one.

RT: ① sorting $\bar{A}$ – $O(n^2)$

② computing $LP(\cdot) = O(1) +$ checking if a and $a+1$ are neighbors $= O(1)$

$\times n^2$

$= O(n^2)$

③ finding the longest $O(n^2)$ (linear scan)

④ reconstructing the path itself $O(n^2)$ (or $O(1)$) ~~depends on~~

Total: $O(n^2)$

Correctness:

Statement 1.

[There exist only one qualifying direction for the path starting from cell $(i;j)$

follows from uniqness of all items in the matrix. can be proved by contradiction

Statement 2

[formula for $LP(a)$ is correct.

follows from statement 1 and contradiction argument.