Name:

Roll No: 23BCS3

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

Department of Computer Science and Engineering

MID SEMESTER EXAMINATION- September, 2024

ICS 211 Design and Analysis of Algorithms

Date & Time: 23-09-2024, 2:30 PM - 4:00 PM
Course Instructors: Nandini J. W, Chakradhar P., Krishnendhu S. P

Max marks: 50
Semester:III
Batch: 1, 2 & 3

Answer all Questions.

Part A (Each Question carry 6 marks, 6*5=30 marks)

1. State whether the following statements are True or false? If it is true, give a short explanation. If it is false, give a counter example. **(3 marks each)**

   (i) Suppose we are given an instance of the Minimum Spanning Tree Problem on a graph **G**, with edge costs that are all positive and distinct. Let T be a minimum spanning tree for this instance. Now, suppose we replace each edge cost c_e by its square c_e^2,thereby creating a new instance of the problem with the same graph but different costs.
   Statement:T must still be a minimum spanning tree for this new instance. True

   •(i) Suppose we are given an instance of the Shortest s-t Path Problem on a directed graph **G**, with edge costs that are all positive and distinct. Let P be a minimum-cost s-t path for this instance. Now, suppose we replace each edge cost c_e by its square c_e^2,thereby creating a new instance of the problem with the same graph but different costs.
   Statement:P must still be a minimum-cost s-t path for this new instance. True

2. Using master's theorem, solve the following. **(3 marks each)**

   ✓(i) Check whether transitivity property exists between the given functions.
   $f(n) = 2n, g(n) = 3n^2, and\ h(n) = 4n^2$. Justify your answer.

   (ii) Solve the recurrence: $T(n) = T(n/2) + n(2 - cos(n))$

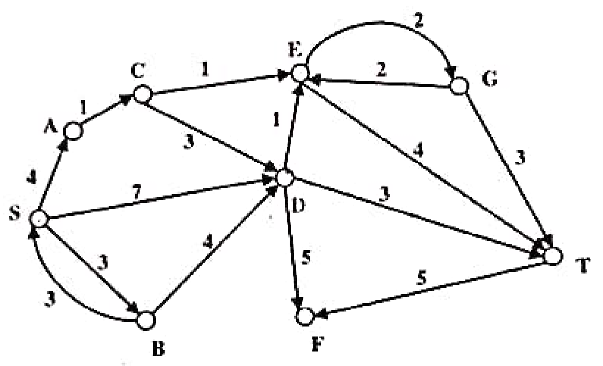
×3. Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are equivalent if they correspond to the same account. It's very difficult to read the account number of a bank card directly, but the bank has a high-tech 'equivalence tester' that takes two bank cards and, after performing some computations, determines whether they are equivalent. Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another. Assume that the only feasible operations you can do with the cards are to pick two of them and plug them into the equivalence tester. It is not difficult to see that this problem can be solved using majority element detection method in $O(n\ log\ n)$. **You are required to prove the correctness of this method.**

✓4. Use a recursion tree to determine a good asymptotic upper bound on the following recurrence.

$$T(n) = T(\tfrac{n}{3}) + T(\tfrac{2n}{3}) + n.$$

P.T.O

5. Consider the directed graph shown in the figure below. There are multiple shortest paths between vertices S and T. Find the shortest path by using Dijkstra's algorithm. Assume that, in any iteration, the shortest path to a vertex v is updated only when a strictly shorter path to v is discovered.



Part B (Each Question carry 10 marks, $2 * 10 = 20$ marks)

• 6. Suppose you are a diamond wholesale dealer and you noticed that there is significant fluctuation in the price of diamond in the recent past. As a part of an analysis, you look at n consecutive days at some point in the past. Let the number of days be $i = 1, 2, 3 \ldots n$; for each day i let $p(i)$ be the price of diamond on that particular day. Now, you are trying to analyse which day would have been the best day to buy diamond and which day would have been the best day to sell diamond so as to make maximum profit.

For example, suppose $n = 3$, $p(1) = \$1000$, $p(2) = \$200$, $p(3) = \$600$. Then you should have purchased diamond on day 2 and sold diamonds on day 3, so as to make the maximum profit of \$400 per diamond. Give an efficient algorithm to find of the day i for buying the diamond and day j for selling the diamond, so as to maximize the profit. It is easy to see that brute force algorithm will run in $O(n^2)$.

✱7. As a part of improvising health care facility in the sparsely populated region of Great Australian Desert, the Australian government propose to establish new emergency health clinics. The houses in this desert area can be imagined as to be scattered along the left and right side a single straight road. The government propose to establish the clinics such that every house is within 5 miles of one of the health clinics.

Give an efficient algorithm that achieve this goal, using minimum number of health clinics. Justify your answer.

*** **Best wishes** ***

Name: Aditya Roll No: 23BCS3

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

Department of Computer Science and Engineering

END SEMESTER EXAMINATION- NOVEMBER, 2024

ICS 211 Design and Analysis of Algorithm

Date & Time: 18-11-2024, 9:30 AM - 12:30 PM
Course Instructors: Nandini J. W, Chakradhar P., Krishnendhu S. P.

Max marks: 100
Semester:III
Batch: 1, 2 & 3

Answer all Questions.

Part A (Each Question carry 10 marks, 10*4=40 marks)

1. You are a bioinformatics researcher studying genetic similarity between two organisms. Each organism's DNA sequence is represented as a string composed of the characters A, C, G, and T. Name the algorithm through which you can perform DNA Sequence Matching. Run the algorithm on the following sequences:
   Organism A: ACGTCGTA
   Organism B: GACTGTA

2. Imagine you work at a healthcare facility that processes high-resolution medical scans (like MRI or CT images) for diagnosis. Each processing stage of these scans involves a series of matrix transformations, where each transformation is represented by a matrix with specific dimensions. You have four transformations for each image each of order: 50×20, 20×30, 30×15 and 15×10. Each transformation step must be applied sequentially. Name an algorithm to optimize processing speed so that its sequential nature will not get affected, but we can reduce the number of scalar operations involved in it. Justify your suggestion w.r.t the given four transformations.

3. Perform the amortized analysis on Augmented stack using Accounting and Potential methods with a short description.

4. Write the recurrence relation for merge sort and solve it using recursion tree method.

Part B (Each Question carry 15 marks, 15*4=60 marks)

5. Consider the following problems given below. Justify which are the complexity classes that the problems fall into, namely P, NP, NP-Hard and NP-Complete. If a problem is identified to be in P, NP, NP-Hard or NP-Complete class, then for each of it you need to provide a proper justification.
   (7.5 marks for each subquestion.)

   i) The **Set Cover Problem** is a classical problem in combinatorial optimization. It is defined as follows:
   Given a universe U of n elements, and a collection of sets $\{S_1, S_2, \ldots, S_m\}$ such that each S_i is a subset of U, the goal is to find a sub-collection of sets that covers the entire universe U using the fewest number of sets. More formally:

   - Let $U = \{u_1, u_2, \ldots, u_n\}$ be the universe of elements.
   - Let $S = \{S_1, S_2, \ldots, S_m\}$ be a collection of sets, where each set $S_i \subseteq U$.
   - The objective is to select a subset of sets $S' \subseteq S$ such that:

   $$\bigcup_{S_i \in S'} S_i = U$$

   and the size of the sub-collection S' is minimized.

The goal is to minimize the number of sets in the sub-collection S' while ensuring that every element of U is included in at least one selected set from S. As an example, consider the universe $U = \{1, 2, 3, 4, 5\}$ and the following collection of sets:

$$S_1 = \{1, 2, 3\}, \quad S_2 = \{1, 4\}, \quad S_3 = \{2, 3, 4\}, \quad S_4 = \{3, 5\}, \quad S_5 = \{4, 5\}$$

To cover all elements of U, the following sub-collection of sets can be selected:

$$S' = \{S_1, S_5\}$$

This sub-collection covers all the elements of U because:

$$S_1 = \{1, 2, 3\}, \quad S_5 = \{4, 5\}$$

and together they cover $\{1, 2, 3, 4, 5\}$.

ii) In graph theory, the **Dominating Set** problem is a classic problem in combinatorial optimization and theoretical computer science. Given an undirected graph $G = (V, E)$, a set $D \subseteq V$ is called a *dominating set* if for every vertex $v \in V$, either $v \in D$ or there exists an edge $(u, v) \in E$ such that $u \in D$. In other words, every vertex in the graph is either in the dominating set or adjacent to a vertex in the dominating set.

The objective is to find a dominating set of minimum size, i.e., a dominating set with the smallest number of vertices.

Formally, the problem is stated as follows:

- **Input:** An undirected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges.
- **Output:** A dominating set $D \subseteq V$ such that:
  - For every vertex $v \in V$, either $v \in D$ or there exists an edge $(u, v) \in E$ such that $u \in D$.
  - The size of D, denoted $|D|$, is minimized.

Consider the following undirected graph G:

$$V = \{1, 2, 3, 4, 5\}, \quad E = \{(1, 2), (1, 3), (2, 4), (3, 5), (4, 5)\}$$

The size of the dominating set is 2, and $D = \{1, 4\}$ is one of minimum dominating set for the graph G. Another dominating set for G is $D = \{1, 5\}$.

6. In the new swap-puzzle game *Candy Swap Saga XIII*, there are n cute animals numbered from 1 to n. Each animal holds one of three types of candy: Circus peanuts, Heath bars, and Cioccolateria Gardini chocolate trues. At the start of the game, you have a Circus peanut candy in your hand.

To earn points, you will visit each of the animals in order from 1 to n. For each animal, you can either keep the candy in your hand or exchange it with the candy the animal is holding. The rules for earning or losing points are as follows:

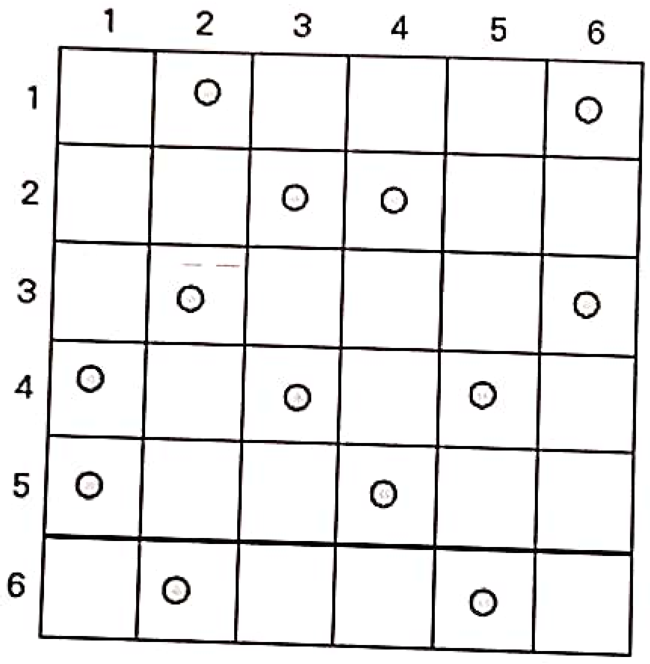
- If you swap your candy for another candy of the *same type*, you earn 1 point.
- If you swap your candy for a candy of a *different type*, you lose 1 point.
- If you decide *not to swap candy*, your score does not change.

You must visit the animals in order, and once you visit an animal, you can never visit it again. The input is a list of size n, where the i-th element of the list, denoted $C[i]$, represents the type of candy the i-th animal holds. The candy types are represented by integers as follows:

- 0 for Circus Peanuts,
- 1 for Heath Bars,
- 2 for Cioccolateria Gardini chocolate trues.

The game starts with a circus peanut in your hand (i.e., your initial candy is of type 0).
Your objective is to determine the maximum score that can be achieved by visiting all n animals, starting with a circus peanut in your hand. Your input is an array $C[1..n]$, where $C[i]$ is the type of candy that the i^{th} animal is holding.
You should output the maximum possible score you can obtain by following the rules of the game. You may use dynamic programming stategy to solve the problem.

7. Several coins are placed in cells of a 6 x 6 board (n x m board) shown below, with no more than one coin per cell. A robot, located in the upper left cell of the board, needs to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it always picks up that coin. Design a Dynamic Programming algorithm to find the maximum number of coins the robot can collect and a path it needs to follow to do this. You need to write the recurrence relation and clearly define all the terms/variables involved in it. Briefly explain how would you trace back your computation and determine the optimal path for the robot. Execute the above on the board.



8. Write the proof of correctness for the following.
(*7.5 marks for each subquestion*)

i) Kruskal's Algorithm

ii) Prim's Algorithm

*** **Best wishes** ***

Roll No: 2022BCS0019 Name: Abhinav Bhagwat

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

Indian Institute of Information Technology Kottayam

Department of Computer Science and Engineering

MID-SEMESTER I EXAMINATION - SEPTEMBER, 2023

ICS 211 DESIGN AND ANALYSIS OF ALGORITHMS

Time and Date: 2.30 PM - 4 PM, 01-09-2023 **Max.Marks:50**
Course Instructor: Nandini J W, Suchithra M S and Ghalib **Batch: 2022**

Answer all Questions

PART-A $[5 \times 7 = 35 \text{ Marks}]$

1. Analyse the following algorithm using the step count method.

```
for (int i = 0; i < n; i++)
    {
          for (int j = i+1; j > =i; j=j/2)
                {
                      for (int k = n; k > j; k=k-1)
                      {
                            printf(''Another-roof,another-proof.'')
                      }
                }
    }
```

2. State Master's Theorem. Further, solve the following recurrence if the recurrence can be solved with the Master's Theorem. Otherwise, indicate that the Master's Theorem does not apply.

   I. $T(n) = 2T(n/2) + n\log n$ [3.5 Marks]
   II. $T(n) = 2^nT(n/2) + n^n$ [3.5 Marks]

3. Find the solution to the following recurrence equation using iterative substitution method:

$$T(2^k) = 3T(2^{k-1}) + 1$$

You can assume $T(1) = c$, where c is a positive constant.

4. Solve the following recurrence relation using recurrence tree method:

$$T(n) = T(n/3) + T(2n/3) + c \cdot n$$

In the above recurrence c is a positive constant, and you can assume any suitable base case.

P.T.O

5. Suppose you are choosing between the following three algorithms:

   I. Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

   II. Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.

   III. Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in quadratic time.

   What are the running times of each of these algorithms (in big-O notation), and which would you choose?

PART-B [$15 \times 1 = 15$ Marks]

6. Suppose you're consulting for a bank that's concerned about fraud detection, and they come to you with the following problem. They have a collection of n bank cards that they've confiscated, suspecting them of being used in fraud. Each bank card is a small plastic object, containing a magnetic stripe with some encrypted data, and it corresponds to a unique account in the bank. Each account can have many bank cards corresponding to it, and we'll say that two bank cards are equivalent if they correspond to the same account.

It's very difficult to read the account number off a bank card directly, but the bank has a high-tech "equivalence tester" that takes two bank cards and, after performing some computations, determines whether they are equivalent.

Their question is the following: among the collection of n cards, is there a set of more than $n/2$ of them that are all equivalent to one another? Assume that the only feasible operations you can do with the cards are to pick two of them and plug them in to the equivalence tester. Show how to decide the answer to their question with only $O(n \log n)$ invocations of the equivalence tester.

******Good Luck******

Roll No:.....2022BCS0019..... Name:.....Abhinav Bhagwat.....

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

Department of Computer Science and Engineering

MID-SEMESTER II EXAMINATION - OCTOBER, 2023

ICS 211 DESIGN AND ANALYSIS OF ALGORITHMS

Time and Date: 9.30 AM - 11 AM, 05-10-2023 Max.Marks:50

Course Instructor: Nandini J W, Suchithra M S and Ghalib Batch: 2022

Answer all Questions

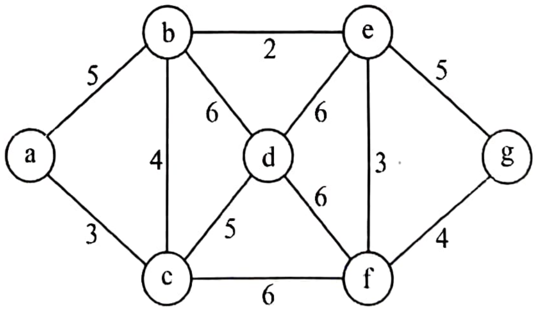
PART-A [5 × 7 = 35 Marks]

1. Find out the time complexity of the following functions using step count method:

```
int f1(int n)
{
        if(n = = 0||n = = 1)
                return n;
    else
                return (2* f1(n - 1) + 3* f1(n - 2));
}

int f2 (int n)
{
        int i;
        int X[N], Y[N], Z[N];
        X[0] =0;  Y[0]=0; Z [0] = 0;
        X[1] = 1; Y[1] = 2; Z[1] = 3;
        for (i = 1; i <= n; i ++)
        {
                X[i] = Y[i - 1] + Z [i - 2];
                Y[i] = 2* X[i];
                Z[i] = 3*X[i];
        }
                return X[n];
}
```

2. List all the Minimum Spanning Trees in the following graph:



P.T.O

3. Let A_1, A_2, A_3 , and A_4 be four matrices of dimensions $10 \times 5, 5 \times 20, 20 \times 10$, and 10×5, respectively. Use matrix chain multiplication algorithm to find the the minimum number of scaler multiplication required to find the product $A_1A_2A_3A_4$?

4. Why Dijkstra's algorithm doesn't work for a graph having negative weight edges? Explain with a small example. The example graph should not have a *negative weight cycle*. We say a graph has *negative weight cycle*, if sum of the weights of all edges in the cycle is negative.

5. The Greedy Interval Scheduling Problem, also known as the Interval Scheduling Problem, is a classic scheduling and optimization problem. Given a set of n tasks or events, each defined by its start time $start[i]$ and end time $end[i]$ for $1 \leq i \leq n$, the goal is to find the maximum number of non-overlapping tasks that can be scheduled without conflicts. We say that two tasks i and j are non-overlaping if $end[i] \leq start[j]$ or $end[j] \leq start[i]$.

   Greedy Algorithm:

   I. Sort the tasks by their end times in ascending order.

   II. Initialize an empty schedule.

   III. Iterate through the sorted tasks:

   - If the current task does not conflict with the last task in the schedule (i.e., its start time is greater than or equal to the end time of the last scheduled task), add it to the schedule.

   IV. Return the schedule as the solution.

   Formally prove that the greedy algorithm given above correctly compute maximum number of non-overlapping tasks that can be scheduled without conflicts.

PART-B [$15 \times 1 = 15$ Marks]

6. Suppose you are managing the construction of billboards on the *Kanyakumari-Srinagar Highway*, a heavily traveled stretch of road that runs south-north for M miles. The possible sites for billboards are given by numbers $x_1, x_2, \ldots, x_n$, each in the interval $[0, M]$ (specifying their position along the highway, measured in miles from its southern end). If you place a billboard at location x_i, you receive a revenue of $r_i > 0$.

   Regulations imposed by the county's Highway Department require that no two of the billboards be within less than or equal to 5 miles of each other. You'd like to place billboards at a subset of the sites so as to maximize your total revenue, subject to this restriction.

   Example: Suppose $M = 20$, $n = 4$,

$$\{x_1, x_2, x_3, x_4\} = \{6, 7, 12, 14\}, \text{ and}$$

$$\{r_1, r_2, r_3, r_4\} = \{5, 6, 5, 1\}.$$

   Then the optimal solution would be to place billboards at x_1 and x_3, for a total revenue of 10.

   Give an algorithm that takes an instance of this problem as input and returns the maximum total revenue that can be obtained from any valid subset of sites. The running time of the algorithm should be polynomial in n.

******Good Luck******

Roll No:...2022BCS0099...... Name:....Abhinav Bhagwat......

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

Indian Institute of Information Technology Kottayam

Department of Computer Science and Engineering

END-SEMESTER EXAMINATION - NOVEMBER, 2023

ICS 211 DESIGN AND ANALYSIS OF ALGORITHMS

Time and Date: 9.30 AM - 12.30 PM, 30-11-2023 **Max.Marks:100**

Course Instructor: Nandini J W, Suchithra M S and Ghalib **Batch: 2022**

Answer all Questions [$10 \times 10 = 100$ **Marks**]

✓ 1. The Fractional Knapsack Problem is a variation of the classic Knapsack Problem where items can be divided into fractions, allowing for a more flexible approach to item selection. In the Fractional Knapsack Problem, a set of items is given, each with a weight w_i and a valu v_i, and there is a knapsack with a maximum weight capacity W. The goal is to determine the combination of items to include in the knapsack to maximize the total value, subject to the constraint that the total weight of the selected items does not exceed the capacity of the knapsack. Formally, we want the following:

Objective Function:

$$\text{Maximize} \quad \sum_{i=1}^{n} v_i \cdot x_i w_i$$

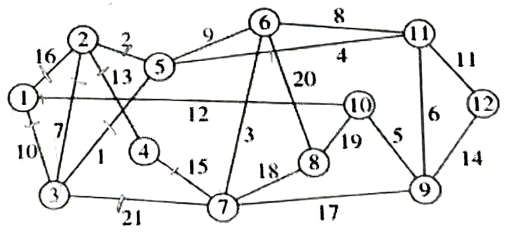
Constraints:

$$\sum_{i=1}^{n} w_i \cdot x_i \leq W \quad \text{(Total weight does not exceed knapsack capacity)}$$

$$0 \leq x_i \leq 1 \quad \text{(Each fraction is between 0 and 1)}$$

Design an algorithm for the problem described above and analyse its time complexity. Furthermore, execute the algorithm on the provided input: $n = 7$, $M = 15$, $(v_1, v_2, \ldots, v_7) = (10, 5, 15, 7, 6, 18, 3)$, and $(w_1, w_2, \ldots, w_7) = (2, 3, 5, 7, 1, 4, 1)$.

✓ 2. Let $G = (V, E, w)$ be a connected, undirected graph with vertices V, edges E, and a weight function w that assigns a non-negative weight to each edge. A Minimum Spanning Tree is a tree $T = (V, E')$ where $E' \subseteq E$ such that T is connected, acyclic, and the sum of weights of edges in E' is minimized.



Consider the graph shown above. If we start with node 10 in V_T as the starting node and use Prim's algorithm to construct the minimum spanning tree, provide the order in which nodes enter V_T. Additionally, provide the minimum total weight.

3. a. Solve using Recursion Tree method

$$T(n) = 3T\left(\frac{n}{4}\right) + n^2$$

where $n = 4^k, k \in \mathbb{N}$, and $T(1) = 1$.

b. Analyse the complexity of the following functions

```
I. function(int n)
   {

        for (int i = 1; i <= n; i++)
        {
              for (int j = 1; j <= n; j++)
              {
               printf(''Obvious is the most dangerous word in mathematics.'');
               break;
               }
        }
   }

II. void function(int n)
      {
       int i = 1, s = 1;
       while (s <= n)
       {
        i++;
        s += i;
        printf(''Music at times is more like perfume than mathematics.");
       }
      }
```

4. *Suppose we perform a sequence of n operations on a data structure where the actual cost of the i-th operation is given by:*

$$c_i = \begin{cases} i & \text{if } i \text{ is an exact power of 2,} \\ 1 & \text{otherwise.} \end{cases}$$

Use aggregate analysis and accounting method to determine the amortized cost per operation.

5. A sequence of stack operations is performed on a stack whose size never exceeds k. After every k operations, a copy of the entire stack is made for backup purposes. Show that the cost of n stack operations, including copying the stack, is $\mathcal{O}(n)$ by assigning suitable amortized costs to the various stack operations.

[Note: Do not worry about how elements are being stored in the backup.]

6. A palindrome is a string over an alphabet that remains unchanged when read in either the forward or backward direction. Palindromes include various instances, such as single-character strings, as well as words like 'civic', 'racecar', 'aibohphobia', etc. Design an efficient algorithm to identify the longest palindrome that serves as a subsequence within a given input string. For instance, when given the input 'character,' the algorithm should yield 'carac.' Additionally, discuss the time complexity of your algorithm.

[Note: A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements.]

7. Professor Saha has cherished a long-held dream of cycling across the scenic state of Rajasthan in India. His planned route spans from Jaipur, situated in the eastern part of the state, to Jaisalmer in the west. Equipped with the capacity to carry two liters of water, Professor Saha can cover a distance denoted as m kilometers before his water supply is exhausted. The generally level topography of Rajasthan eliminates concerns about varying water consumption rates based on the terrain's incline.

In preparation for his journey, the professor examines the official state map of Rajasthan, highlighting locations along the route from Jaipur to Jaisalmer where he can refill his water supply. Distances between these water refill points are also provided.

Professor Saha's primary objective is to minimize the number of water stops during his cycling adventure across the state. Propose an efficient method for him to determine the optimal water stops along his route. Additionally, present evidence supporting the optimality of this strategy and specify its running time.

8. For each of the two questions below, decide whether the answer is (i) "Yes," (ii) "No," or (iii) "Unknown, because it would resolve the question of whether P = NP." Give a brief explanation of your answer.

a. Let's define the decision version of the Interval Scheduling Problem discussed in the class as follows: Given a collection of intervals on a time-line, and a bound k, does the collection contain a subset of nonoverlapping intervals of size at least k?
Question: Is it the case that ***Interval Scheduling*** $\leq_P$ ***Vertex Cover***?

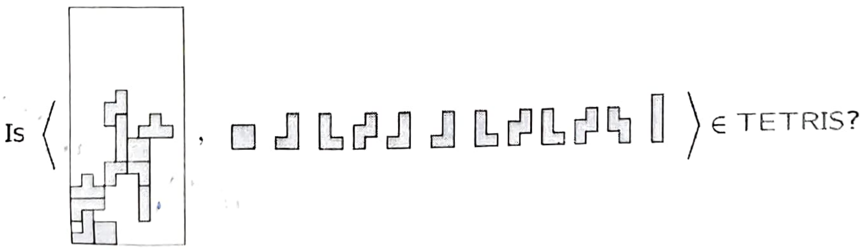
b. Question: Is it the case that ***Independent Set*** $\leq_P$ ***Interval Scheduling***?

9. Show that the following problems are in complexity class NP:

a. Tetris is a popular computer game that was invented by mathematician Alexey Pazhitnov in the mid-1980s. In the Offline Tetris "Advanced Level" game, players are presented with a partially filled gameboard g and a predetermined sequence of Tetris pieces s. The objective is to strategically play these pieces in the given order to clear the entire gameboard. Each Tetris piece can be rotated or moved horizontally, and the player must carefully plan their moves to eliminate filled lines and create space for subsequent pieces. The challenge lies in efficiently utilizing the provided piece sequence to achieve a clear gameboard. The decision problem associated with this game asks whether, given a specific initial gameboard and Tetris piece sequence, it is possible to successfully clear the entire board using the provided pieces.

$$TETRIS = \{\langle g, s\rangle \mid \textit{it is possible to play the piece sequence } s \textit{ to clear the entire gameboard } g\}$$

Decision version of the problem: Given an arbitrary $\langle g, s\rangle$ find out if it belongs to $TETRIS$?
For example following could be an input instance:



b. Consider a generalized version of Candy Crush, a popular match-three puzzle game. In this version, the game is played on an $a \times b$ board filled with six different colored candies. The objective is to achieve a specific score, denoted by s, by strategically swapping neighboring candies on the board. A positive integer k represents the number of allowed swaps.

The scoring mechanism is based on creating chains of three identical candies. Each swap involves interchanging two neighboring candies on the board. When three identical candies align either horizontally or vertically, they form a chain, and these chains are subsequently deleted. Importantly, when multiple chains are formed simultaneously, they are deleted from the bottom of the board to the top. Additionally, the candies above the deleted chains immediately drop down to fill the vacant spaces.

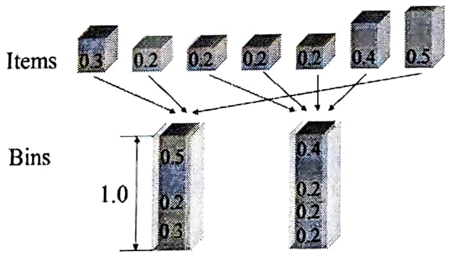
Input:

- An $a \times b$ board filled with six colored candies.
- A positive integer k representing the number of allowed swaps.
- A positive integer s representing the target score to be achieved or surpassed.

Decision version of the problem: Determine if there exists a sequence of k swaps that achieves a score of s or more.

[Note: Do not worry if you have struggled playing Candy Crush, Tetris, Super Mario, Angry Birds, 2048, Snake etc they are proved to be computationally hard to solve as well. Many variants of computer video games falls into the category of NP-Hard, NP-Complete and PSPACE Complete ☹.]

~~10~~ You are given n items with sizes $a_1, a_2, \ldots, a_n$. *Without loss of generality let us assume that* $\forall\ i \in \{1, 2, \ldots, n\}$, $0 < a_i \leq 1$, and an unlimited number of identical bins of integer capacity 1 is *supplied. The Bin Packing Problem (BPP) is to pack all the items into the minimum number of bins so that the total weight packed in any bin does not exceed the capacity.* Design a 2-approximation algorithm for this problem, and prove your performance guarantee. The running time of your input should be polynomial in n.
Just for an example following figure shows that the optimal solution for the given input is 2:



So if your approximation algorithm come up with a packing where you require less than equal to 4 bin for the above instance then you are doing good.
[Note: You can not break the items. So single item can not span through multiple bins.]

******Good Luck******

dafuq bruh!!!