CS-2009 Design and Analysis of Algorithms, Spring-2022
Project

Due Date and Time: 17th May 2022 (1130 hrs) **Weight: 10%**

Instructions:

1. *Late submission will not be accepted.*
2. *Project can be done in a group. Max. group size is 3 members. In case of a group size of 3 members, each student is required to solve exactly one problem. In case of group size of 2 members, one member is required to solve any one problem while the other is required to solve the remaining 2 problems. In case of group size of 1 member, the student is required to solve all the problems.*
3. *Groups are required to list their information on the following spreadsheet latest by 9th May, 2022 (you are required to lock the cells related to your group):* https://docs.google.com/spreadsheets/d/1wPX4_R5qKgKPsKb4BAFraUgZFXX_s3PX0-7W69PQgfk/edit?usp=sharing
4. *Changes in the groups are possible till 9th May, 2022, i.e., after 9th May, the groups cannot be changed. The schedule of demonstrations will be made on the basis of groups' information submitted till 9th May, 2022.*
5. *Only the listed student (on the spreadsheet) will give the demo. related to a problem, i.e., other member(s) of the group (if any) cannot give demo. on behalf of some group member.*
6. *Secured project marks of every group member = Secured marks in Problem 1 + Secured marks in Problem 2 + Secured marks in Problem 3, e.g., if there are 3 members in a group; the 1st member secures 4 out of 4 points, the 2nd group member secures 1 out of 3 points, and the 3rd group member secures zero out of the 3 points, then each group member will obtain 5 points out of the 10 points of the project.*
7. *There will be no credit if the given requirements are changed.*
8. *Your solution will be evaluated in comparison with the best solution.*
9. *Plagiarism may result in zero marks in the whole project regardless of the percentage plagiarized.*
10. *Use the additional material provided with this project.*
11. *Additional datasets can be used to test your work so make a generic solution of each problem.*
12. *Provide a "single" integrated report having analysis of your algorithms. Every group member will contribute to the report related to his/her allocation of problem(s).*

Problem 1: (Prepositional Logic Graph) [Weight: 4%]

Consider a propositional formula *F* in conjunctive normal form (CNF). Given a truth assignment *T* to the variables of *F*, a k-flip is another assignment *T'*, which disagrees with *T* on at most *k* variables. K-Flip refers to the problem of deciding, given *F*, *T*, and *k*, whether there exists a k-flip which satisfies strictly more clauses of *F* than *T* does. This problem is intractable.

Your task is to develop, implement and asymptotically analyze an algorithm A that takes input of a formula *F* and an integer *k*. The algorithm first generates a random assignment *T* then it improves the assignment using k-flips as long as possible, or until a specified timeout is reached, aiming to satisfy more and more clauses.

Implement and test algorithm A with a basic prototype. Ideally, the prototype should accept input formula *F* in exactly same format as provided in dataset.txt file.

The problem commences with a formula *F* consisting of *n* number of variables which can be used in any sequence. The formula consists of clauses having any number of OR operations on multiple variables and the clauses are joined through AND operations. Suppose there is an initial random assignment *T* of Boolean values to the variables of formula *F*, it is required to find another assignment *T'* that disagrees with *T* on at most *k* number of bits and strictly satisfies a greater number of clauses than the original *T*. For simplicity, the disagreement is represented as *d*. Therefore, *T* and *T'* are identical for *(n – d)* bits where $1 \leq d \leq k$ exhibits the disagreement constraint.

The details of benchmark dataset are provided in the following table.

Parameter	**Value**
K	{5,10,15,20}
Variables	1040
Clauses	3668

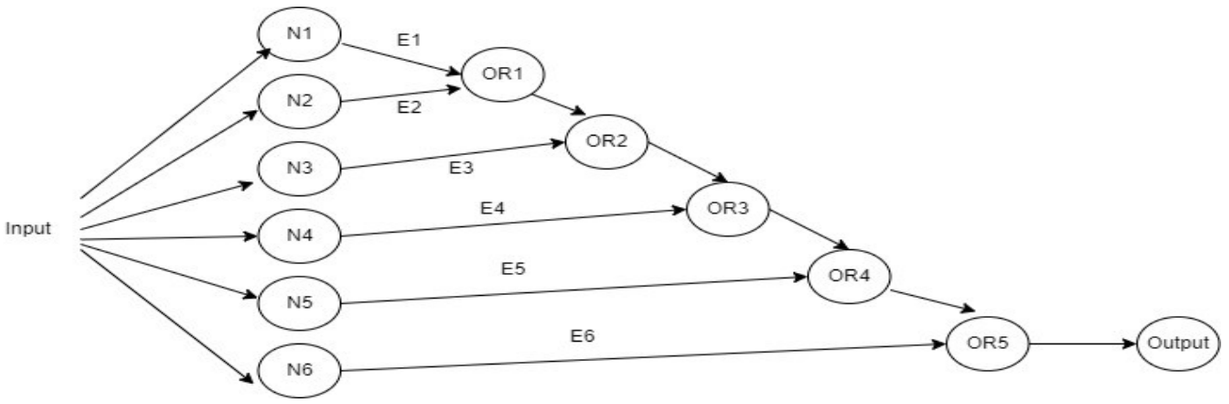

Figure 1 Clause Network

1. Clause Network can have minimum 1 and maximum 6 activated nodes.
2. Each node represents an arbitrary distinct variable.
3. Nodes N2 to N6 are activated only when a clause expression has more than one variable. A sample of Clause Network with edge weights representing a clause {-63, 205, 208, 215, 260} is given below in a table.
4. Weights of E1, E2, E3, E4, E5 and E6 edges represent respective Input Variables.
5. A negative weight edge represents NOT operation on the respective Input Variable.
6. The OR nodes perform OR-operation on two values and also have a read logic. For example, -63 and 205 received in OR1 means access variable 63 with NOT and access variable 205 as it is and later perform OR on both accessed values.

7. Use a suitable data structure to represent this sparse graph.
8. Demonstrating a clause-by-clause traversal of the graph is compulsory.
9. Generate node by node graph traversal stats and show these on the output screen.
10. Clauses are provided in a separate dataset.txt file in which each line represents a clause ending with 0. Remember, for testing a different dataset can be used.
11. The initial truth assignment *T* should be randomly generated for all 1040 variables, whereas *T'* is the output of your algorithm that satisfies the given constraint.
12. Provide analysis of your pseudocode, stat files, output and C++ source code.

	N1	N2	N3	N4	N5	N6	OR1	OR2	OR3	OR4	OR5	Output
N1							-63					
N2							205					
N3								208				
N4									215			
N5										260		
N6											0	
OR1								?				
OR2									?			
OR3										?		
OR4											?	
OR5												?
Output												

Problem 2: (Compartmentalization) [Weight: 3%]

A commercial land of 1000 x 1000 sq. feet needs to be sold in a market area. According to the regulations, a minimum size of saleable plot can be 100 sq. feet or multiples of 100 sq. feet and every dimension is also a multiple of 5 like 5x20. There is an associated standard selling prices for different sizes of plot.

The problem of compartmentalization is as follows: Given a land of size *n* in sq. feet, a list of associated prices in a table and stipulated saleable regulations, determine the maximum profit that can be achieved by making a division of land (further elucidated in an example) for selling. It is to be noted that land can be sold as it is i.e., without making further smaller distributions. Different dimensions summing up to a same size have same associated price.

The land size *n* must fit the multiples of 100 sq. feet; however, actual dimensions can be like 30 x 30, which is also a multiple of 100 sq. feet. You have to provide a Dynamic Programming based solution for this problem, which is explained through an example below.

Example:

Suppose a land of size 40 x 40 sq. feet which is 1.6k sq. feet. The prices of different sizes of plot are as follows:

Sample Sizes	5x20, 10x10, 20x5	5x40, 10x20, 20x10, 40x5	~~5x60~~, 10x30, 15x20, 20x15, 30x10	20x20 etc.	20x25 etc.	20x30 etc.	…	…
Prices	15k	75k	120k	135k	150k	255k	255k	300k

A land of *n* = 20x25 can be split into some of these distributions and there can be other combinations.

1. 20x5 + 10x20 + 10x20; the profit will be 15k+75k+75k=165k
2. 20x10 + 20x15; the profit will be 75k+120k=195k

It is evident considering the above two distributions that for a land of size 20x25, a better value of profit can be achieved which is 195k. The agenda of problem is to find an optimal solution producing maximum profit with any distribution. It is required to consider different split for reaching to a maximum profit but finding the distribution through the proposed algorithm is not required.

Deliverable:

1. Design a Simple Recursive Top-Down without Memoization Algorithm.
2. Design a Recursive Top-Down with Memoization Algorithm.
3. Design an Iterative Bottom-Up with Memoization Algorithm.
4. Optimize Storage in the Iterative Algorithm (if applicable).
5. Provide a C++ file for each of the above 4 and also analyze the time complexity of Iterative Bottom-Up algorithm.
6. Your algorithm should work for any value of *n* and price *P* which is a vector of prices against each plot sizes in multiple of 100 sq. feet. It can be tested against range of test cases so make a generic solution.

Problem 3: (Diagonal Pattern) [Weight: 3%]

Suppose you are given a text T of length *n* x *n* and a pattern P of length *m* x *m,* both in 2D format. Find the diagonal occurrences of the pattern in the given text. The terms and conditions are further elaborated below.

T =

A	**B**	**A**	B	B	A	D	A
A	**C**	**A**	A	A	B	B	B
A	A	D	**B**	**A**	C	D	A
B	C	C	**C**	**A**	A	B	B
C	D	A	D	B	**B**	**A**	C
D	A	D	B	D	**C**	**A**	C

P =

B	**A**
C	**A**

Conditions:

- Pattern can be of any dimension with same number of rows and columns. For example, 1 x 1, 2 x 2, and 3 x 3 etc. up to *n* x *n*.
- There should be at least two blocks of pattern in the diagonal position. Blocks can be more than two. Otherwise, diagonal pattern does not exist.
- The diagonal position begins with the immediate next row of the previous occurrence of pattern. The diagonal pattern can be overlapping some columns of the upper pattern excluding the first column. It is demonstrated in the table.

B	**A**	B	B
C	**A**	A	A
A	**B**	**A**	D
C	**C**	**A**	C

Provide a C++ implementation and analyze your algorithm to find the time complexity. The algorithm will be compared with the best solution. Avoid using more than constant extra space in your logic.

Hint: Make all possible combinations before attempting to implement.