HOMEWORK 6 - NP GRAPH PROBLEMS HW

RAMSEY NUMBERS

A Ramsey number $R(i, j)$ is defined as the smallest number n such that every graph on n vertices has either a clique of size i or an independent set of size j. Ramsey's Theorem states that $R(i, j)$ exists for all i, j in the set of Natural Numbers ($\mathbb{N}$).

PROBLEM 1:

Devise an algorithm that, given i, j, finds $R(i, j)$.

My implementation of the Ramsey numbers algorithm accurately identifies Ramsey numbers for certain pairs (i, j). However, determining exact values for all pairs of i and j remains an open and theoretical question in mathematics (Van Overberghe, 2019). Beyond a few small pairs (i, j), it becomes increasingly challenging to determine exact Ramsey numbers as they grow exponentially. To approximate larger Ramsey values, researchers use various techniques, including linear programming, combinatorics, cyclical graphing, edge counting, and more (McKay & Radziszowski). As finding these exact values for all pairs of i and j remains a significant ongoing mathematical and graph theory challenge, my implementation has made efforts to ensure the accuracy, but it cannot judiciously encompass each advanced technique used by researchers.

```
int RamseyNumber(int i, int j){
    /* * * * * * * * * * * * * * * *
     * Base Case:
     *      R(i,1) = 1
     *      R(1,j) = 1
     * * * * * * * * * * * * * * * */
    if(i==1 || j==1){
        return 1;
    }

    /* * * * * * * * * * * * * * * *
     * Rule 1:
     *      R(i,2)=i
     *      R(2,j)=j
     * * * * * * * * * * * * * * * */
    if(i==2){
        return j;
    }
    if(j==2){
        return i;
    }

    /* Recursive Step */
    /* * * * * * * * * * * * * * * *
     * Rule 2:
     *      R(i,i)<= 4*R(i-2,i)+2
     * * * * * * * * * * * * * * * */
    int result, m, n;
    if (i==j){
        result = 4*RamseyNumber(i-2,j)+2;
    }
    else{
        m = RamseyNumber(i-1, j);
        n = RamseyNumber(i,j-1);


        /* * * * * * * * * * * * * * * * * * * * * * *
         * Rule 3:
         *      if R(i-1, j) and R(i, j-1) are even:
         *          R(i-1, j)+R(i, j-1)-1
         *      else:
         *          R(i-1, j)+R(i, j-1)
         * * * * * * * * * * * * * * * * * * * * * * */
        if (m%2 == 0 && n%2 == 0) {
            result = m + n - 1;
        }
        else{
            result = m + n ;
        }
    }

    /* Return Ramsey Result */
    return result;
}
```

Code 1: Recursive Algorithm for Calculating Ramsey Numbers in C

The Recursive Algorithm shown in Code 1 is designed to determine the Ramsey number $R(i, j)$ using a recursive approach. The algorithm begins by checking for base cases and then implements recursion methods depending on three rules. Please refer to the assignment folder for the complete code implementation of the Recursive Algorithm for calculating Ramsey numbers in C.

Note: The rules presented in the code and table are numbered for clarity and organizational purposes within this document only. They do not have a significance in any official capacity.

Base Case	$R(i,1) = R(1,j) = 1$	*(Barton IV, 2016)*
Rule 1	$R(i,2) = i$ $R(2,j) = j$	*(Barton IV, 2016)* *(Chachamis, 2018)*
Rule 2	$R(i,i) <= 4 \times R(i-2,i) + 2$	*(Weisstein)*
Rule 2 *Alternative, not used in code*	$R(i,i) <= 2 \times R(i-1,i)$	*(Chachamis, 2018)*
Rule 3	$R(i,j) \leq \begin{cases} \text{for } R(i-1,\ j) \text{ and } R(i,\ j-1) \text{ are even} \\ \text{else} \end{cases}$ $R(i,j) \leq \begin{cases} R(i-1,\ j) + R(i,\ j-1) - 1 \\ R(i-1,\ j) + R(i,\ j-1) \end{cases}$	*(Weisstein)*

Table 1: Rules for Calculating Ramsey Numbers

PROBLEM 2:

Analyze your algorithm's time complexity, but you don't need to prove it by induction.

Line	*Code*	*Relation*
1	`int RamseyNumber(int i, int j){`	
2	`    if(i == 1 || j == 1) {`	<u>Base Case 1</u> *this statement executes in constant time whenever $i = 1$ or $j = 1$* c when $i = 1, j = 1$
4	`        return 1;`	
5	`    }`	
6	`    if(i == 2) {`	<u>Base Case 2</u> *these statements execute in constant time whenever $i = 2$ and $j = 2$* e when $i = 2, j = 2$
7	`        return j;`	
8	`    }`	
9	`    if(j == 2) {`	
10	`        return i;`	
11	`    }`	
12	`    int result, m, n;`	
13	`    if(i == j) {`	<u>Recursive Call A</u> *when $i = j$, the function makes one recursive call with the problem size n reduced by 2 for each.* $T(n-2)$
14	`        result = 4*RamseyNumber(i-2,j)+2;`	
15	`    }`	
16	`    else {`	<u>Recursive Call B</u> *although there are 2 recursive calls, there are not two recursive cases because they are within an if-else statement and only one is called during each iteration.* *when $i \neq j$, the function makes two recursive calls with the problem size n reduced by 1 for each.* $2\times T(n-1)$
17	`        m = RamseyNumber(i-1, j);`	
18	`        n = RamseyNumber(i,j-1);`	
19	`        if(m%2 == 0 && n%2 == 0) {`	
20	`            result = m + n - 1;`	
21	`        }`	
22	`        else {`	
23	`            result = m + n;`	
24	`        }`	
25	`    }`	
26	`    return result;`	

All remaining lines (12, 19-24, 26) execute in constant time for each function iteration that is not our base case and will be represented using variable f

$$T(n) = \begin{cases} c & \text{if } i = 1 \text{ or } j = 1 \\ e & \text{if } i = 2 \text{ or } j = 2 \\ T(n-2) + f & \text{if } i = j \\ 2T(n-1) + f & \text{if } i \neq j \end{cases}$$

My recursive implementation for calculating Ramsey Numbers provides a theoretical approach for finding $R(i,j)$ given i, j. However, through analysis of the time complexity, it becomes apparent that the algorithm makes redundant recursive calls for overlapping subproblems. To address this issue and significantly improve runtime performance, dynamic programming techniques may be implemented. By storing the results of previously calculated subproblems in a memoization table, redundant computations could be avoided. Furthermore, if we were to account for the symmetry of Ramsey numbers, $R(i,j) = R(j,i)$, the number of recursive calls could be substantially reduced, resulting in a more efficient algorithm (Weisstein).

References

Barton IV, L. (2016, May 13). Ramsey Theory. Retrieved from https://www.whitman.edu/documents/Academics/Mathematics/2016/Barton.pdf

Chachamis, C. N. (2018, May 13). Ramsey Numbers. Retrieved from https://math.mit.edu/~apost/courses/18.204_2018/ramsey-numbers.pdf

Graham, R. L., & Spencer, J. H. (1990, July). Ramsey Theory. *Scientific America*. Retrieved from https://mathweb.ucsd.edu/~ronspubs/90_06_ramsey_theory.pdf

Graham, R. L., Rothschild, B. L., & Spencer, J. H. (1990). Ramsey Theory. *2*. John Wiley & Sons, Inc. Retrieved from http://people.dm.unipi.it/dinasso/ULTRABIBLIO/Graham_Rothschild_Spencer%20-%20Ramsey%20Theory%20(2nd%20edition).pdf

McKay, B. D., & Radziszowski, S. P. (n.d.). *Subgraph Counting Identities and Ramsey Numbers.* Australian National University, Rochester Institute of Technology, Department of Computer Science. Retrieved from https://users.cecs.anu.edu.au/~bdm/papers/r55.pdf

Van Overberghe, S. (2019). *Algorithms for computing Ramsey numbers.* Ghent University, Department of Applied Mathematics, Computer Science and Statistics. Retrieved from https://libstore.ugent.be/fulltxt/RUG01/002/836/320/RUG01-002836320_2021_0001_AC.pdf

Weisstein, E. W. (n.d.). Ramsey Number. *MathWorld -- A Wolfram Web Resource*. Retrieved from https://mathworld.wolfram.com/RamseyNumber.html