**Homework 6 - NP Graph Problems HW**

Ramsey Numbers

A Ramsey number is defined as the smallest number such that every graph on vertices has either a clique of size or an independent set of size . Ramsey's Theorem states that exists for all in the set of Natural Numbers ().

# Problem 1:

Devise an algorithm that, given , finds .

My implementation of the Ramsey numbers algorithm accurately identifies Ramsey numbers for certain pairs . However, determining exact values for all pairs of and remains an open and theoretical question in mathematics (Overberghe). Beyond a few small pairs , it becomes increasingly challenging to determine exact Ramsey numbers as they grow exponentially. To approximate larger Ramsey values, researchers use various techniques, such as linear programming, combinatorics, cyclical graphing, edge counting, and more [mkkay]. As finding these exact values for all pairs of and remains a significant ongoing mathematical and graph theory challenge, my implementation has made efforts to ensure the accuracy, but it cannot reasonably 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 : 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. It checks for base cases and then implements recursion methods depending on three rules. The entire code is available in the assignment folder.

*Note: Rules are numbered for clarity, the numbers are not related in any official capacity.*

|  |  |  |
| --- | --- | --- |
| Base Case |  | [Barton] |
| Rule 1 |  | [Barton]  [Chachamis] |
| Rule 2 |  | [Mathworld] |
| Rule 2  Alternative, not used in code |  | [Chachamis] |
| Rule 3 |  | [Mathworld] |

Table : 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) { | | | Base Case 1  *this statement executes in constant time whenever or*  when , |
| 4 |  |  | return 1; | |
| 5 |  | } | | |
| 6 |  | if(i == 2) { | | | Base Case 2  *these statements execute in constant time whenever and*  when , |
| 7 |  |  | return j; | |
| 8 |  | } | | |
| 9 |  | if(j == 2) { | | |
| 10 |  |  | return i; | |
| 11 |  | } | | |
| 12 |  | int result, m, n; | | |  |
| 13 |  | if(i == j) { | | | Recursive Call A  *When ,* *the function makes one recursive call with the problem size n reduced by 2 for each.* |
| 14 |  |  | result = 4\*RamseyNumber(i-2,j)+2; | |
| 15 |  | } | | |
| 16 |  | else { | | | Recursive Call B  *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 ,* *the function makes two recursive calls with the problem size n reduced by 1 for each.*  *2* |
| 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 execute in constant time for each function iteration that is not our base case and will be represented using variable .* |

This could be improved using a ?? to keep track of already discovered values. If this is implemented, keep in mind the following rule:

|  |  |
| --- | --- |
|  | [mathworld](https://mathworld.wolfram.com/RamseyNumber.html) |