Deeksha Sharma

10/30/2017

1. Design: Describe your approach to solve this problem.  Include your algorithm (pseudocode) and explain what is your base case and what the recursive step are.  Why did you decide to approach the problem this way?

My solution is to write a recursive function that takes in a char array. As per the directions of the assignment this method is going to try every possible move and try to get all the explorers and cannibals to that other bank while making sure that the number of cannibals never outnumbers the number of explorers on either bank.

**Pseudocode:**

**Header file:**

class Cannibals

{

public:

//last char is L if boat is on left bank and is R if boat is on right bank

char leftBank[7];

char rightBank[6];

//possibleMoves is a vector of vectors of chars that holds all possible combination of moves

vector<vector<char>> possibleMoves = {{'E', NULL},{ 'C',NULL },{ 'E','E' },{ 'C','C' },{ 'E','C' }};

//this will hold all the moves that lead to the solution

char solutionArray[40];

bool ValidState(char[]); //checks if banks are at a valid state

bool MoveBoat(char[]); //moves people between banks

bool TryAMove(char[]); //the recursisve method

int SizeOfArray(char[]); //gets size of array

char PrintArray(char[]); //prints the array

bool Remove(int, char[]); //removes target char from targetArray

bool Add(int, char[]); //adds target to targetArray

Cannibals(); //default constructor

};

**.cpp file:**

Cannibals::Cannibals()

{

for (int i = 0; i < 6; i++)

{

Initialize all of rightBank elements to 0

}

for (int i = 0; i < 40; i++)

{

Initialize all of solutionArray elements to 0

}

for (int i = 0 through size of leftBank array; i++)

{

if (i < 3) leftBank[i] = 'E';

else if (i < 6) leftBank[i] = 'C';

else leftBank[6] = 'L';

}

}

bool Cannibals::ValidState(char target[])

{

for (int i = 0 through size of target; i++)

{

Count the number of Es and number of Cs.

}

if (number of Cs > number of Es) return false;

return true;

}

int Cannibals::SizeOfArray(char target[])

{

for (int i = 0 through size of target; i++)

{

Count the number of elements.

}

return number of elements;

}

char Cannibals::PrintArray(char[] target)

{

for (int i = 0 through size of target; i++)

{

Print the element at index i

}

cout << endl;

}

bool Cannibals::Remove(vector<char> target, char[] targetArray)

{

bool sucRemove = false; //tracks if the remove was successful

if (index 1 of target char vector isn’t null)

{

for (int i = 0 through size of targetArray; i++)

{

if (the index of targetArray equals the char in index 1 of target)

{

Set that index of targetArray to null

sucRemove = true;

}

else sucRemove = false;

}

}

for (int i = 0 through size of targetArray; i++)

{

if (the index of targetArray equals the char in index 0 of target)

{

Set that index of targetArray to null

sucRemove = true;

}

else sucRemove = false;

}

return sucRemove;

}

bool Cannibals::Add(vector<char> target, char[] targetArray)

{

bool sucAdd = false; //bool to keep track of whether the addition was successful

if (index 1 of target char vector isn’t null)

{

for (int i = 0 through size of targetArray; i++)

{

if (index of targetArray equals null)

{

Set that index of targetArray to char in index 1 of target

sucAdd = true;

}

else sucAdd = false;

}

}

for (int i = 0 through size of targetArray; i++)

{

if (index of targetArray equals null)

{

Set that index of targetArray to char in index 1 of target sucAdd = true;

}

else sucAdd = false;

}

return sucAdd;

}

bool Cannibals::MoveBoat(int targetIndex)

{

if (boat is on left bank)

{

if (removing the vector at targetIndex in possibleMoves vector from leftBank is successful)

{

Move boat to right bank

return add the vector at targetIndex of possibleMoves to rightBank

}

}

else

{

if (removing the vector at targetIndex in possibleMoves vector from rightBank is successful)

{

Move boat to left bank

return add the vector at targetIndex of possibleMoves to leftBank

}

}

return false;

}

bool Cannibals::TryAMove(int targetIndex)

{

if (move of targetIndex is successful)

{

//base case

if (size of rightBank is 6 which means its full)

{

return true;

}

if (both left and right bank are at a valid state)

{

for (int i = 0 through the size of possibleMoves; ++i)

{

if (move to try equals the move just tried) continue;

if (TryAMove of the next possible move is successful)

{

return add the move to the solutionArray

}

}

}

MoveBoat(targetIndex) //if at an invalid state this reset arrays to previous valid state

}

//if either bank is not at a valid state

return false;

}

**Driver file:**

int main()

{

Create an instance of Cannibals class

//this loop calls the recursive function and tries all possible moves

for (int i = 0 through the size of possibleMoves vector; ++i)

{

Call TryAMove for every possible move stored in the possibleMoves vector

}

Print out solution

}

1. What is the complexity of your code?  First, analyze it "by hand", and then formalize it by writing down the recurrence analysis.   Explain your findings.

1. Clearly state any assumptions you make

I am assuming that there is only 3 explorers and 3 cannibals on the left bank. I am assuming that the boat can only carry 2 people. I am assuming that there needs to be one person in the boat in order for it to move. I am assuming that if there are more cannibals than explorers on any side at a time then that bank is at an invalid state. I assume that there is only one boat being used to move the explorers and cannibals.

1. Show and explain your output

With the code that I have when I try to run it in Linux I get the following output:

deeksha@uw1-320-06:~$ g++ -std=c++11 \*.cpp

deeksha@uw1-320-06:~$ ./a.out

very beginning of TryAMove

Segmentation fault (core dumped)

When I looked up this error in Google it said that it is a result of me trying to access memory that I do not have access to. I have test cout lines in the code to see the progress of my code and to understand what is being executed and what isn’t. Because of the output “very beginning of TryAMove” I know that the faulty access of memory is happening after the first TryAMove function call. I reviewed my code over and over however I was not able to identify where this error was occurring.

1. If you use any external references (i.e., not the textbook) point out what the reference is -- website, another book, a classmate.

Ryan Peters: A UWB student who is also in 342 but with a different professor.

1. Compiling and running instructions (see deliverable 3 below).