Submission

ID	DATE	PROBLEM	STATUS	CPU	LANG
	TEST CASES				
8031930	06:10:37	10 Kinds of People	✖ **Time Limit Exceeded**	> 1.00 s	C++
	✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✖☐☐				

Submission contains 1 file: download zip archive

FILENAME	FILESIZE	SHA-1 SUM	
10kindsofpeople.cpp	2338 bytes	259fc7f9d9721ff7fa75b5f5435cdc6db35794b5	download

Edit and resubmit this submission.

10kindsofpeople.cpp

```
#include <bits/stdc++.h>
class Solution {
  public:

  void DFS(int r0, int c0, int** board, int rows, int cols, char filler, char filled){
    if (board[r0][c0] == filled){
      board[r0][c0] = filler;
      if (r0 + 1 < rows){
        DFS(r0 + 1, c0, board, rows, cols, filler, filled);
      }
      if (r0 - 1 > -1){
        DFS(r0 - 1, c0, board, rows, cols, filler, filled);
      }
      if (c0 + 1 < cols){
        DFS(r0, c0 + 1, board, rows, cols, filler, filled);
      }
      if (c0 - 1 > -1){
        DFS(r0, c0 - 1, board, rows, cols, filler, filled);
      }
    }
    else{
      return;
    }
  }

  void fill(int r0, int c0, int** board, int rows, int cols, char filler){
    char filled = board[r0][c0];
    DFS(r0, c0, board, rows, cols, filler, filled);
  }

  void PBoard(int** board, int rows, int cols){
    for (int i = 0; i < rows; i++){
        for(int j = 0; j < cols; j++){
            std::cout << board[i][j];
        }
        std::cout << "\n";
    }
  }

  std::string queryB(int r1, int c1, int r2, int c2, int** board, int rows, int cols){
    char start = board[r1][c1];
  std::string return_value = "neither";
  fill(r1, c1, board, rows, cols, 4);
  if (board[r1][c1] == board[r2][c2]){
   if (start == 0){
    return_value = "binary";
   }
   else{
    return_value = "decimal";
   }
  }
  fill(r1, c1, board, rows, cols, start); // Return board to original state.
  return return_value;
 }
};
int main(){
 std::ios_base::sync_with_stdio(false);
 std::cin.tie(NULL);
 Solution S1= Solution();
 int r,c,n;
 std::cin >> r;
 std::cin >> c;
 int** board = new int*[r];
 for(int i = 0; i < r; i++){
  board[i] = new int[c];
 }
 std::cin.ignore();
 for(int i = 0; i < r; i++){
  std::string this_line;
  getline(std::cin, this_line);
  for (std::string::size_type j = 0; j < this_line.size(); j++)
  {
   int this_entry = (int)(this_line[j]) - 48;
   board[i][j] = this_entry;
  }
 }

 std::cin >> n;
 for (int i = 0; i < n; i++){
  int r1,c1,r2,c2;
  std::cin >> r1;
  std::cin >> c1;
  std::cin >> r2;
  std::cin >> c2;
  std::cout << S1.queryB(r1 - 1, c1 - 1, r2 - 1, c2 - 1, board, r, c) << "\n";
 }
 return 0;
}
```