Relatório: Resolvedor do Jogo Flood It em C

Gissely de Souza

1 Introdução

Este código é uma implementação em C de um resolvedor para o jogo *Flood It*. Em resumo, ele cria um tabuleiro colorido e, a partir do canto superior esquerdo, expande uma área unida modificando as cores até que todas as células do tabuleiro fiquem com a mesma cor.

2 Funcionamento do Código

O programa realiza as seguintes etapas:

- **Criação do Tabuleiro:** Gera uma matriz $n \times n$ onde cada célula (representada por um nó) recebe uma cor aleatória dentre c cores. Cada nó armazena sua posição (x e y), a cor atribuída e ponteiros para seus vizinhos (cima, baixo, esquerda e direita).
- **Identificação da "União" e da "Borda":** A partir do canto superior esquerdo, o programa utiliza a função `acha_borda` para identificar a região conectada (a "união") que possui a mesma cor. Ao mesmo tempo, os nós que estão na fronteira (com cor diferente) são marcados como parte da "borda".
- **Resolução do Jogo:** São implementadas duas abordagens:
  - **Resolvedor Aleatório:** Escolhe aleatoriamente a próxima cor dentre aquelas presentes na borda.
  - **Resolvedor com Heurística do Maior Bordo:** Seleciona a cor que maximiza a expansão da região unida, ou seja, aquela cuja aplicação resulta na maior área de expansão. Essa estratégia busca minimizar o número total de jogadas.
- **Atualização e Impressão:** A cada jogada, a cor da região unida é modificada, as listas de borda são atualizadas e o estado do tabuleiro é impresso, permitindo acompanhar o progresso até que todas as células tenham a mesma cor.
- **Uso de Listas:** O código utiliza estruturas de lista para gerenciar os nós pertencentes à região unida e aos diferentes conjuntos de borda associados a cada cor, facilitando a expansão da área.

3 Principais Componentes do Código

O código está organizado em partes que incluem:

1. **Definição das Estruturas:** São definidas as estruturas para os nós (células) e para as listas. Exemplo:

Listing 1: Definição das Estruturas

```
typedef struct no *no;
struct no {
    int x, y, cor;
    no vizinhos[4], prox, proxB;
    int visitado, examinado;
};

typedef struct lista *lista;
struct lista {
    no primeiro;
    int tam;
};

```

2. **Criação do Tabuleiro:** A função `cria_tabuleiro_simples` gera a matriz $n \times n$ com cores aleatórias e inicializa os ponteiros dos vizinhos.

3. **Identificação da Região e Bordas:** A função `acha_borda` percorre o tabuleiro, iniciando no canto superior esquerdo, para marcar a área unida (com a mesma cor) e identificar os nós que compõem a borda.

4. **Resolução do Jogo:** A função `resolvedor_max_bordas` utiliza uma heurística que escolhe a cor que maximiza a expansão da região unida. Essa função, juntamente com outras funções auxiliares, atualiza as listas e imprime o estado do tabuleiro a cada jogada.

5. **Função Principal:** A função `main` recebe os argumentos para a dimensão do tabuleiro (n) e o número de cores (c), e inicia a execução do resolvedor.

4 Exemplo de Código Principal

A seguir, apresenta-se um extrato com os pontos principais do código:

Listing 2: Extrato do Código Principal

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define UP vizinhos[0]
#define DOWN vizinhos[1]
#define LEFT vizinhos[2]
#define RIGHT vizinhos[3]
```

```c

/* Estruturas para n e lista */
typedef struct no *no;
struct no {
    int x, y, cor;
    no vizinhos[4], prox, proxB;
    int visitado, examinado;
};

typedef struct lista *lista;
struct lista {
    no primeiro;
    int tam;
};

/*  Funo  que cria o tabuleiro com cores aleatrias */
no **cria_tabuleiro_simples(int n, int c) {
    /*  Implementao  que aloca e inicializa a matriz */
    /* ... */
}

/*  Funo  que identifica a regio unida e a borda */
void acha_borda(no n, int cor, lista *borda, lista uniao) {
    /* ... */
}

/* Resolvedor com a heurstica do maior bordo */
void resolvedor_max_bordas(no **matriz, int n, int n_cor) {
    /* ... */
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("<flood> <nMatriz> <nCores>\n");
        exit(0);
    }
    int n = atoi(argv[1]), c = atoi(argv[2]);
    no **matriz = cria_tabuleiro_simples(n, c);
    resolvedor_max_bordas(matriz, n, c);
    return 0;
}
```

5 Compilação e Execução

Para compilar o programa, utilize um compilador C (como o `gcc`). Supondo que o código esteja salvo em um arquivo chamado `floodit.c`, utilize o seguinte comando:

```
gcc floodit.c -o floodit -std=c99
```

A execução do programa requer dois argumentos:

- `nMatriz`: dimensão do tabuleiro (ex.: 10 para um tabuleiro 10x10);
- `nCores`: número de cores a serem utilizadas.

Exemplo de execução:

```
./floodit 10 6
```

Isso criará um tabuleiro de 10x10 com 6 cores e executará o resolvedor, que, no código, é implementado pela função `resolvedor_max_bordas`.

6 Conclusão

Em resumo, o código cria um tabuleiro colorido, identifica a área conectada a partir do canto superior esquerdo e, utilizando uma heurística (ou abordagem aleatória, se desejado), altera a cor da área unida passo a passo até que o tabuleiro inteiro fique com uma única cor. Essa implementação modular facilita futuras melhorias, como a inclusão de um jogador inteligente que antecipe jogadas.