Trabalho 3 - Balanceamento de Recursos

Introdução à Ciência de Computação

Prof. Moacir Ponti / João Batista
PAE: Leonardo Tórtoro Pereira, Matheus Takata
Monitores: George Yoshio Tamanaka, Henry Shinji Suzukawa

Implemente suas atividades sem compartilhar ou olhar código de seus colegas. Procure usar todos os conceitos vistos nas aulas. Documente a sua aplicação por meio de comentários no programa

Descrição do trabalho

Esse trabalho tem como objetivo treinar os conceitos de funções, recursividade, ponteiros e alocação dinâmica, fazendo o aluno pensar em:

1. Solução de problemas através de recursão;
2. Passagem de parâmetros por referência;
3. Manipulação de matrizes na memória *stack*;
4. Manipulação de vetores na memória *heap*;

Tarefa

A distribuição de recursos e posicionamento das bases iniciais de jogadores em jogos do gênero estratégia em tempo real -do inglês, real-time strategy (RTS)- é um problema muito relevante para os desenvolvedores. É preciso garantir uma distribuição justa de recursos para ambos os jogadores para que o jogo seja equilibrado e mais desafiador para ambos.

Neste trabalho, você deverá implementar a verificação da distribuição de recursos em termos de distância às bases de dois jogadores diferentes. Para isso, primeiro você deverá ler o conteúdo de uma matriz 9x9, representando o mapa no qual ocorre o jogo, denominada de *map*. Nela, o valor 0 representa um espaço transponível e 1 um espaço **in**transponível. Todos os valores de coordenadas serão passados seguindo o sistema (L, C), em que L é o índice da linha da matriz, e C o índice da coluna da matriz.

Os valores serão passados linha a linha, da esquerda para a direita, ou seja: (0,0) é o primeiro valor, (0,1) o segundo, (0,2) o terceiro, etc.

Considere as coordenadas Após a leitura e atribuição do valor de cada célula de, uma dupla de coordenadas no formato (L,C) será passada, correspondendo, respectivamente, à posição dos jogadores 1 e 2 no mapa. Ambas serão referidas como $pos1$ e $pos2$, respectivamente.

Por fim, um número correspondente ao total de recursos ($nRec$) será passado, seguido das coordenadas X e Y, respectivamente, para cada recurso, formando o conjunto de

coordenadas que chamaremos de *posRec*. **Lembrem-se de alocar na memória heap a variável para salvar as coordenadas dos recursos!**.

Lidas as entradas, seu programa deverá calcular o valor de exploração[1] entre a base de cada jogador e cada elemento de recurso do mapa. O artigo mencionado contém a fórmula para o valor de exploração na Equação 2, apresentada abaixo:

$$E_i(S_N) = \frac{1}{N-1} \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{E_{i \to j}}{P}$$

Na qual N é o número de recursos somado de 1 (a base do jogador), também conhecido como elementos de referência. P é o número de células transponíveis em *map*, $E_{i \to j}$ é a cobertura do mapa quando um algoritmo de *flood-fill*[2] de quatro direções é aplicado iniciando-se do elemento i e parando quando o elemento j foi encontrado, ou seja, o tanto de células visitadas pelo algoritmo durante a trajetória (**contar a célula da base e do recurso como parte do resultado**).

Portanto, você deverá calcular o valor E_1 e E_2, respectivamente, separadamente partindo de ambas as bases. Lembre-se de **não colocar** a base oposta no conjunto de elementos a serem explorados! O algoritmo de *flood-fill* mencionado é explicado no link do rodapé, que também contém um pseudo-código do mesmo.**Lembrem-se de utilizar a ordem de visita do pseudo-código do link: Sul, Norte, Oeste e Leste**

Note que quanto **menor** o valor de E, menos o jogador terá de explorar para encontrar recursos e, portanto, possui **vantagem**.

Ao final, você deverá comparar ambos os valores e verificar se um dos jogadores possui vantagem em relação ao outro. Para evitar erros de cálculo com ponto flutuante, você deverá considerar que a diferença entre os valores E_1 e E_2 só será significativa se o valor **absoluto** da diferença for **maior ou igual que 0.001**. Caso contrário, você deverá considerar os valores como iguais.

[1] `https://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/viewFile/7374/7585`
[2] `https://en.wikipedia.org/wiki/Flood_fill`

Entrada

Portanto, seu programa deverá ler 100 inteiros, com o conteúdo de cada célula da matriz, que pode receber os valores 0 ou 1:

- `map(0,0)`
- `map(0,1)`
- `map(0,2)`
- ...
- `map(8,7)`
- `map(8,8)`

Em seguida, ler 4 inteiros, sendo cada par as coordenadas L e C, respectivamente, de cada jogador

- `pos1L`
- `pos1C`
- `pos2L`
- `pos2C`

O número de recursos dever ser lido, seguido pelas coordenadas L e C de cada recurso.

- `nRec`
- `posRec[1]L`
- `posRec[1]C`
- `posRec[2]L`
- ...
- `posRec[nRec-1]C`
- `posRec[nRec]L`
- `posRec[nRec]C`

No exemplo a seguir, a matriz foi reduzida para 5x5, para facilitar a visualização:

```
0 0 0 0 0
0 1 1 1 1
0 0 0 0 0
0 0 0 0 0
0 1 1 1 1
0 3
3 3
2
0 0
0 4
```

Saída

Seu programa deverá imprimir como saída, separado por linhas, os valores de E_1, E_2 (ponto flutuante) e uma mensagem identificando se o mapa está balanceado ou algum jogador tem vantagem. Caso seja balanceado:

- O mapa eh balanceado \n

Caso contrário, dizer qual dos dois jogadores ($p = 1$ ou $p = 2$) tem a vantagem:

- O jogador p possui vantagem\n

Exemplo:

```
2.3\n
1.25\n
O jogador 2 possui vantagem\n
```

Note os símbolos `\n` denotando uma quebra de linha na função `printf()`

Exemplo

Para a entrada:

```
0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1 0
0 0
8 8
1
4 4
```

A saída será:

```
0.152542
0.152542
O mapa eh balanceado
```

Instruções

O trabalho será avaliado levando em consideração:

1. Realização dos objetivos
2. Representação correta da entrada e saída dos dados
3. Uso de comentários e estrutura no código (e.g. indentação, legibilidade)
4. Número de acertos no sistema Run.Codes
5. Uso da memória

ATENÇÃO:

- O projeto deverá ser entregue apenas pelo (`http://run.codes`) no formato de **código fonte**, ou seja apenas o código C.
- O prazo está no sistema run.codes
- Em caso de projetos **copiados** de colegas ou da Internet, todos os envolvidos recebem nota zero. Inclui no plágio a cópia com pequenas modificações, cópia de apenas uma parte ou função. Portanto programe seu próprio trabalho.