Universidade Federal de Santa Catarina Centro Tecnológico - CTC
Departamento de Informática e Estatística - INE

Relatório

Estrutura de Dados

Profs.
Aldo Von Wangenheim e
Alexandre Goncalves Silva

Alunos
Eduardo Gwoszdz De Lazari (22100612) e
Micael Angelo Sabadin Presotto (22104063)

Parte 1:

Para o desenvolvimento da verificação das tags XML, utilizamos os documentos disponibilizados pelo professor para criar nossa lógica. Analisamos a documentação sobre como abrir um arquivo XML e também sobre algumas funções que poderiam ser utilizadas com strings.

Em nosso código, utilizamos a estrutura de dados chamada Pilha (LIFO). Resumidamente, abrimos o arquivo XML e o lemos linha por linha. Para verificar se uma tag estava sendo aberta e fechada, utilizamos o método find(). Guardamos a posição inicial que indica a abertura da tag e a posição final que indica o fechamento da tag. Com essas informações, usamos a função substr() para extrair a tag entre a abertura e o fechamento.

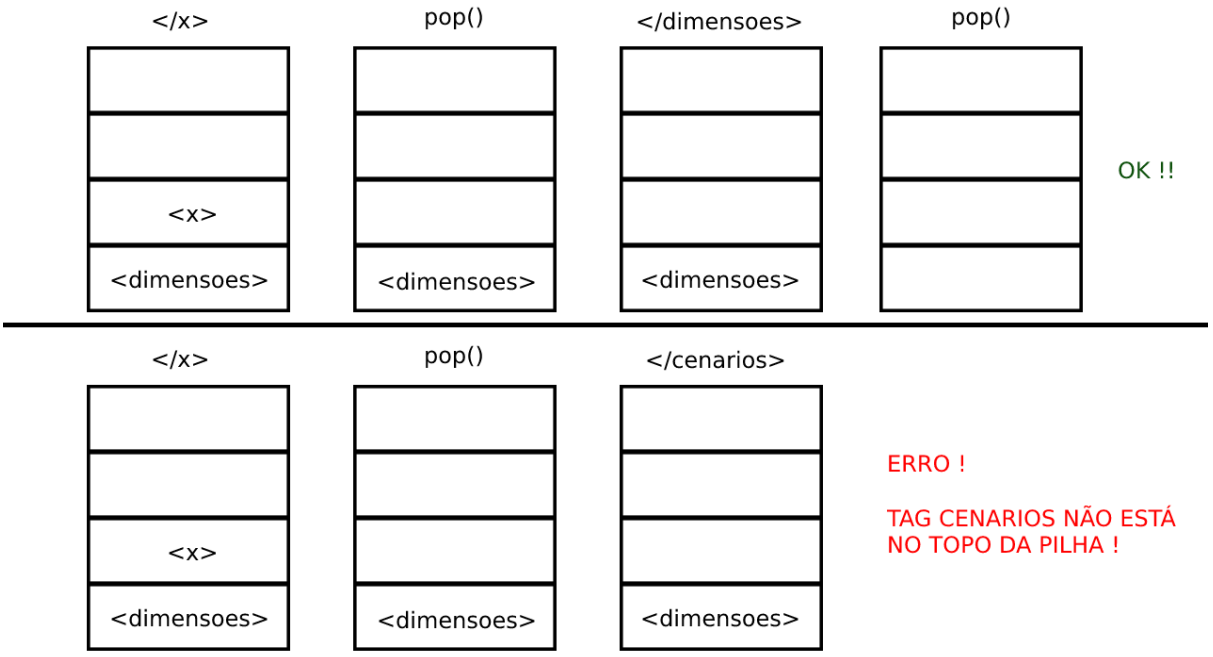
Em seguida, analisamos se essa tag era uma tag de fechamento, ou seja, se iniciava com '/'. Caso fosse uma tag de fechamento, verificamos se ela era igual à última tag inserida na pilha, ou seja, a tag que estava no topo. Se fosse verdadeiro, utilizamos o método pop() para removê-la e continuávamos nossa verificação. Nessa situação, os erros que poderiam ocorrer eram quando a última tag inserida na pilha não correspondia à que estava sendo fechada, o que indicaria um fechamento de tag incorreto. Outra situação seria quando a pilha estivesse vazia, o que significaria que estava sendo fechada uma tag que nem sequer foi aberta. Nessas situações, a função retorna falso, pois não faria sentido continuar as iterações se houvesse algum erro.

Finalmente, após todas as iterações que verificavam se as tags estavam sendo abertas e fechadas corretamente, verificamos se a pilha estava vazia. Se estivesse vazia, isso indicaria que tudo ocorreu corretamente e a função retornará true. Caso contrário, se a pilha não estivesse vazia, isso significaria que uma tag não foi fechada corretamente e, portanto, a função retorna falso.

Segue uma figura representando dois casos da implementação da verificação das tags via Pilha.

1º caso = <dimensoes> <x> </x> </dimensoes> => ok

2º caso = <dimensoes> <x> </x> </cenarios> => errado



Parte 2:

Na segunda parte do projeto, utilizamos filas para processar as tags do arquivo XML. Cada fila representava o valor de uma tag específica. Por exemplo, criamos uma função chamada getValorTag() que obtinha o conteúdo dentro de uma determinada tag. Por exemplo, se quisermos obter o valor inteiro da tag <altura>, podemos chamar essa função, passando como parâmetros a linha que estamos analisando e a tag da qual queremos obter o valor. Para passar a linha como parâmetro, precisamos abrir o arquivo XML novamente. Dessa forma, armazenamos os valores de cada tag em suas respectivas filas.

Para obter o conteúdo da matriz, nossa abordagem foi localizar a abertura da tag da matriz e concatenar os valores de suas linhas em uma string chamada "valorMatriz". Para fazer isso, criamos uma variável booleana que, quando encontrávamos uma tag de abertura de matriz, era definida como verdadeira. Enquanto essa variável fosse verdadeira, realizamos a concatenação. Quando encontrávamos uma tag de fechamento, a variável era definida como falsa e a concatenação não ocorria mais.

Com essas informações em mãos, pudemos implementar a função de limpeza do robô. Para isso, utilizamos duas matrizes: a matriz de entrada e a matriz resultado. A matriz de entrada é aquela obtida na função anterior, enquanto a matriz resultado é inicializada com todos os valores em 0. Para isso, utilizamos os valores de altura e largura, que representam o número de linhas e colunas, respectivamente.

Para determinar as células a serem limpas pelo robô, utilizamos um método iterativo que percorre uma fila. A execução continua até que a fila esteja vazia, ou seja, todas as

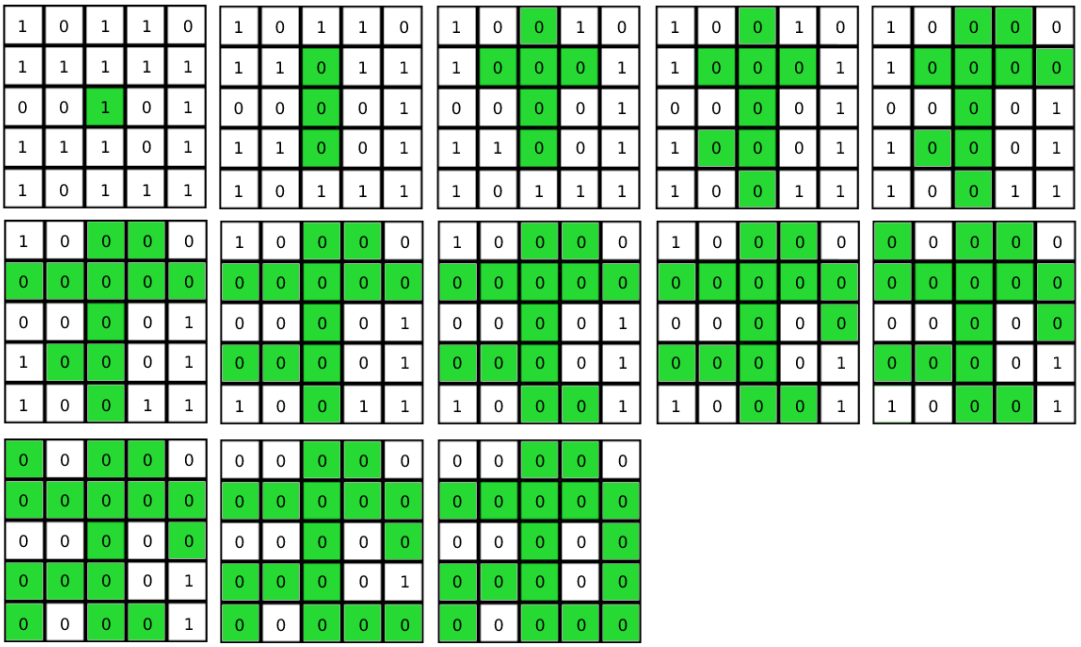
células que podem ser limpas foram processadas. Para armazenar as posições na fila, utilizamos uma lógica de decompor uma matriz em um vetor. Assim, as coordenadas Matriz[x][y] foram transformadas em Vetor[x * numero_colunas + y]. Por exemplo, em uma matriz de tamanho 3x3, a célula matriz[1][0] corresponde à célula vetor[1 * 3 + 0], ou seja, vetor[3].

Após explicarmos a lógica de decomposição e busca dos valores, podemos prosseguir com o código. Começamos verificando se a posição em que o robô foi colocado na matriz de entrada é igual a 1. Se for o caso, esse valor 1 também será colocado na matriz resultado e as coordenadas correspondentes serão adicionadas à fila. O processo iterativo consiste em:

1. Remover o primeiro item da fila (dequeue()).
2. Achar seus vizinhos adjacentes.
3. Se os vizinhos forem iguais a 1 na matriz de entrada e 0 na matriz resultado, suas posições são adicionadas à fila e seus valores na matriz resultado são definidos como 1.

Ao final do processo, contamos a quantidade de 1s existentes na matriz resultado, que será a resposta da função "limpar".

Segue exemplo do robô limpando a matriz =>



Fila correspondente:
[2,2]
[1,2], [3,2]
[3,2], [1,3], [1,1], [0,2]
[1,3], [1,1], [0,2], [3,1], [4,2]
[1,1], [0,2], [3,1], [4,2], [1,4], [0,3]
[0,2], [3,1], [4,2], [1,4], [0,3], [1,0]

[4,2], [1,4], [0,3], [1,0], [3,0]
[1,4], [0,3], [1,0], [3,0], [4,3]
[1,0], [3,0], [4,3], [2,4]
[3,0], [4,3], [2,4], [0,0]
[4,3], [2,4], [0,0], [4,0]
[2,4], [0,0], [4,0], [4,4]
Encerrado

Vale ressaltar que nesse caso específico, todas as casas com valor 1 foram limpadas, porém isso não é possível em todos casos. Um exemplo seria quando existe uma célula de valor 1 cercada por 0s. Ela nunca será acessada, portanto, não será limpada pelo robô.

Conclusão

O grupo enfrentou algumas dificuldades na execução do projeto, incluindo a forma de extrair o conteúdo das tags. Inicialmente, foi desafiador pensar em uma maneira de obter os conteúdos das tags. Após um tempo de análise e reflexão, conseguimos desenvolver a ideia de usar uma função que recebe como parâmetro a linha e a tag desejada. Essa abordagem foi possível ao examinarmos o arquivo XML e percebermos que as tags principais que procurávamos estavam em uma única linha.

No entanto, a obtenção do conteúdo da matriz foi um desafio adicional, pois a função criada não funcionaria para a tag da matriz. Contudo, surgiu rapidamente a ideia de extrair a matriz através da concatenação do seu conteúdo em uma única string. Além disso, optamos por utilizar um vetor em vez de uma matriz, visando simplificar o processo.

Referências

http://www.cplusplus.com/doc/tutorial/files/
http://www.cplusplus.com/reference/string/string/