Nome: Beatriz Viana Costa

NUSP: 13673214

Relatório - EP2 – Algoritmos e Estruturas de Dados I (MAC0121):

O código entregue não funciona, contudo, segue lógica utilizada no programa.

No arquivo *biblio.h* foram definidas três *structs,* são elas a pilha, a posição (que é utilizada dentro da palavra) e a palavra.

A pilha contém um vetor onde serão colocados os IDs de cada palavra. A posição contém duas coordenadas em formato *int*. Já as palavras possuem diversas informações referentes a cada palavra colocada no tabuleiro: a posição da primeira letra (pos_ini), a posição da última letra (pos_fim), se já foi colocada no tabuleiro (verificada, em caso afirmativo é 1, caso contrário é 0), sua orientação (orientation, se for na horizontal é igual a 0, se for na vertical é igual a 1), seu tamanho (length, que é conseguido através da função strlen() da biblioteca *string.h*) e o ID da palavra (um número que vai de 1 a quantidade de palavras dada como *input*).

A função principal (*main)* lê a quantidade de linhas e colunas, m e n respectivamente, e em caso de nenhum dos dois serem nulos, chama a função instância, dando esses dois valores como parâmetros. A variável vezes também presente na função principal é inicializada em 1 e é incrementada a cada iteração do único laço presente na função, ela é utilizada na hora da impressão do número da instância.

Já na função instância é criada a matriz *Crossword* de tipo *char, ou seja, a partir da matriz de entrada, caso o valor seja 0 é alocado um espaço vazio, e ser for -1, é guardado um '*', indicando um lugar ocupado. Será nesta matriz que serão colocadas as palavras da palavra cruzada.*

É criada também a matriz matrizID de tipo int, que inicialmente será igual a matriz dada como input, mas à medida que preenchemos a palavra cruzada, o ID da palavra colocada será colocado nessa matriz. Enquanto está matriz é criada, a quantidade de zeros (ou seja, a quantidade de espaços que devem ser preenchidas por letras no Crossword) é calculada e guardada na variável *espacos_vazios*.

Criamos as **palavras* do tipo palavras, e a partir disso a cada palavra que é lida, ela é alocada na string do índice correto, o ID também é colocado e o tamanho idem.

A partir disto chamamos a função *tentaPreencher* e passamos como parâmetros todas as informações que temos até o momento. Nesta temos as variáveis X e Y iniciais (*Xini* e *Yini)* e X e Y finais (*Xfim* e *Yfim)*, que são inicializadas com valor 0.

Tentamos primeiro preencher as palavras na horizontal, denotando o X inicial como o a linha que estamos olhando. Vamos percorrendo esta linha até que encontremos um -1 na *matrizID.* Ao encontrarmos calculamos o tamanho do trecho desta linha, caso seja do mesmo tamanho da palavra que temos, preenchemos as matrizes *Crossword* e *matrizID,* preenchemos as coordenadas iniciais e finais, a orientação, e marcamos a palavra como verificada, empilhamos o ID da palavra colocada e vamos para a proxima palavra.

Caso o tamanho do trecho da linha não seja igual ao tamanho da palavra, atualizamos a coordenada Y inicial e continuamos a olhar até o fim da matriz.

Caso a palavra não consiga ser encaixada horizontalmente, tentamos na vertical, de maneira análoga a tentativa na horizontal.

Quando voltamos para a função instância, verificamos se há espaços vazios na palavra cruzada. Em caso negativo a matriz resposta deveria ser impressa. Já em caso positivo desempilhamos a última palavra colocada e a retiramos da *matrizID* e *Crossword* e então retornamos para a função *tentaPreencher,* isso até que encontremos uma solução ou até que a pilha fique vazia.

Após isso fizemos a última verificação da quantidade de espaços vazios, se for diferente de 0 imprimimos que não há solução para a instância em questão, caso contrário imprimimos a matriz resposta.