TAREFA TEG 1#A

Aluno: Alex Halatiki Vicente

Primeiramente, o arquivo IrisDataset foi modificado eliminando a primeira linha e a última coluna das demais linhas para facilitar a leitura do mesmo. Após isso, a função exibida na imagem abaixo faz a leitura do arquivo recebendo um ponteiro de int como parâmetro, que será modificado conforme a leitura do arquivo, para assim, se ter o controle de quantos vértices existirão. Com isso, lê-se linha por linha do arquivo separando cada valor por virgula utilizando a função strtok(), realocando a matriz (ponteiro de ponteiro de double) que será retornada e convertendo os valores presentes na linha para double com a função atof() para atribuí-los na matriz.

```c
double **leEntrada(int *linhas)
{
    FILE *arquivo;
    double ** matrizVertices = NULL;
    char linha[50];

    arquivo = fopen("irisDataset.csv", "r");

    if (arquivo == NULL)
        return NULL;

    while (fgets(linha, 50, arquivo) != NULL)
    {
        matrizVertices = realloc(matrizVertices, sizeof(double*) * (linhas[0]+1));
        matrizVertices[linhas[0]] = malloc(sizeof(double) * N);

        char *token = strtok(linha, ",");
        int aux = 0;

        while (token != NULL)
        {
            matrizVertices[linhas[0]][aux] = atof(token);
            token = strtok(NULL, ",");
            aux++;
        }
        linhas[0]++;
    }

    fclose(arquivo);

    return matrizVertices;
}
```

Após isso, uma matriz quadrada m x m, sendo m o número de vértices, é alocada e passada como parâmetro junto da matriz de vértices (retornada anteriormente pela função leEntrada) para a função exibida na imagem abaixo, que percorre a matriz de vértices calculando a distância euclidiana dos valores de cada vértice com os demais vértices, e atribuindo esse valor na nova matriz m x m passada como parâmetro.

```
void distanciaEuclidiana(double **matrizVertices, double **matriz, int linhas, int colunas)
{

    for(int i=0;i<linhas;i++)
        for(int j=0;j<colunas;j++)
        {
            double aux = 0;
            for(int c=0;c<N;c++)
                aux += (pow(matrizVertices[i][c] - matrizVertices[j][c], 2));

            matriz[i][j] = sqrt(aux);
        }
}
```

Com a matriz de distância euclidiana formada, utiliza-se a função exibida na imagem abaixo para percorrê-la e encontrar o maior valor presente na matriz, valor esse que será usado posteriormente para normalizar a matriz.

```
double maiorValor(double **matriz, int linhas, int colunas)
{
    double maior = matriz[0][0];

    for(int i=0;i<linhas;i++)
        for(int j=0;j<colunas;j++)
            if(matriz[i][j] > maior)
                maior = matriz[i][j];

    return maior;
}
```

E então, para a função exibida na imagem abaixo, passamos a matriz e o maior valor encontrado pela função anterior para percorrer a matriz dividindo os valores dela pelo valor, assim tem-se a matriz normalizada.

```
void normalizaMatriz(double **matriz, int denominador, int linhas, int colunas)
{
    for(int i=0;i<linhas;i++)
        for(int j=0;j<colunas;j++)
            matriz[i][j] = matriz[i][j] / denominador;
}
```

Por fim, basta percorrer a matriz novamente, agora verificando os elementos que são menores ou iguais ao limiar (0.3), colocando o valor 1 para os mesmos e 0 para os que não atendem o requisito, como mostra a imagem abaixo. Assim temos a matriz de adjacência formada.

```
for(int i=0;i<linhas;i++)
    for(int j=0;j<colunas;j++)
    {
        if(matrizNormalizada[i][j] <= valor)
        {
            matrizNormalizada[i][j] = 1;
            quant++;
        }
        else
            matrizNormalizada[i][j] = 0;
    }
```

Com a matriz de adjacências formada, podemos gerar o arquivo texto de arestas no formato que condiz com o script python disponibilizado para gerar o grafo como mostra a imagem abaixo.

```
FILE *arquivo;

arquivo = fopen("arestas.txt", "w");

//fprintf(arquivo, "%d", quantArestas);

for(int i=0;i<m;i++)
{
    int aux = 0;
    for(int j=0;j<m;j++)
        if(matrizDistancia[i][j] == 1)
        {
            fprintf(arquivo, "%d %d\n", i+1, j+1);
            aux = 1;
        }
    if(!aux)
        fprintf(arquivo, "%d\n", i+1);
}

fclose(arquivo);
```

O resultado 3D do grafo gerado a partir do script fornecido se encontra abaixo.