ROTAS DE ÔNIBUS ENTRE CIDADES PAULISTAS

A Rede Estadual Rodoviária de São Paulo - deseja fornecer aos seus usuários um aplicativo que permita verificar os caminhos entre cidades, através de viagens de ônibus.

Para tanto, o arquivo **texto** GrafoOnibusSaoPaulo.txt é fornecido. Cada registro contém nomes de cidades de origem e de destino e a distância entre elas em quilômetros. As ligações são bidirecionais, ou seja, valem tanto para a ida quanto para a volta de uma cidade para outra.

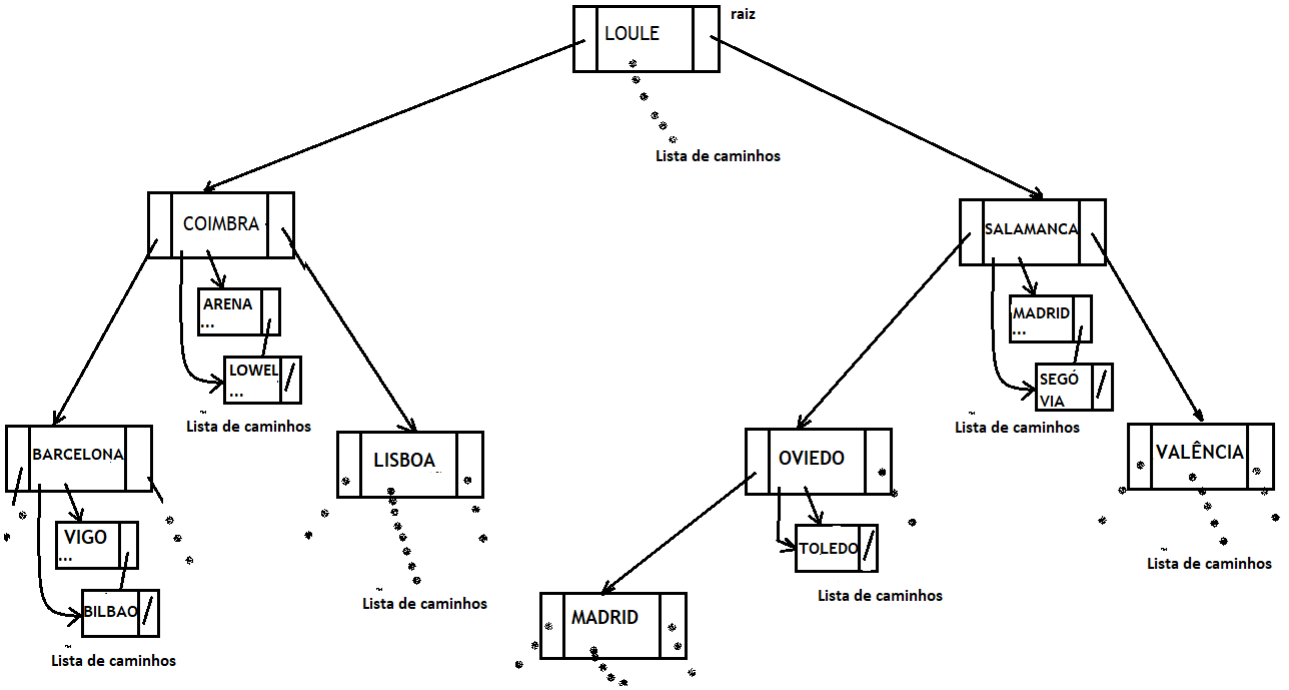
Há também um arquivo **binário** chamado CidadesSaoPaulo.dat, com o nome da cidade e a coordenada cartesiana (x, y) onde a cidade deve ser localizada no mapa que será exibido na tela. X e y são valores reais que representam proporções da largura e da altura do mapa, respectivamente, no momento e nas dimensões (em pixels) em que o mapa está sendo exibido. Os dados desse arquivo estão dispostos em ordem alfabética de nome de cidade.

Deseja-se que o aplicativo efetue a manutenção do arquivo de cidades e do arquivo de caminhos entre as cidades, bem como permita a busca de caminhos entre cidades através de ônibus. Codifique as classes Cidade e Ligacao de forma que sejam compatíveis com as interfaces e restrições definidas nas classes NoArvore, Arvore, ListaSimples e NoLista. Use, para essas classes, os códigos desenvolvidos durante as aulas e apresentados na apostila. Não use outras classes para este programa.

O arquivo binário de cidades deve ser lido recursivamente e seus registros armazenados em uma árvore de busca binária balanceada AVL, para fazer a manutenção dos dados de cidades. Em cada nó dessa árvore de cidades, o registro de cidade armazenado deverá, também, possuir um atributo da classe ListaSimples, inicialmente vazio. Depois da leitura do arquivo de cidades e da montagem da árvore balanceada AVL, o arquivo de caminhos será lido e os caminhos que tem origem em cada cidade serão armazenados na lista ligada encapsulada dentro do objeto Info de cada nó de cidade, representando a lista de cidades acessíveis a partir da cidade de origem armazenada no nó da árvore.

Após o arquivo de cidades ter sido lido e sua árvore montada, deve-se passar a ler o arquivo de caminhos, sequencialmente. Cada registro de caminho deverá ser armazenado ao final da lista ligada armazenada no nó da cidade de origem na árvore de cidades.

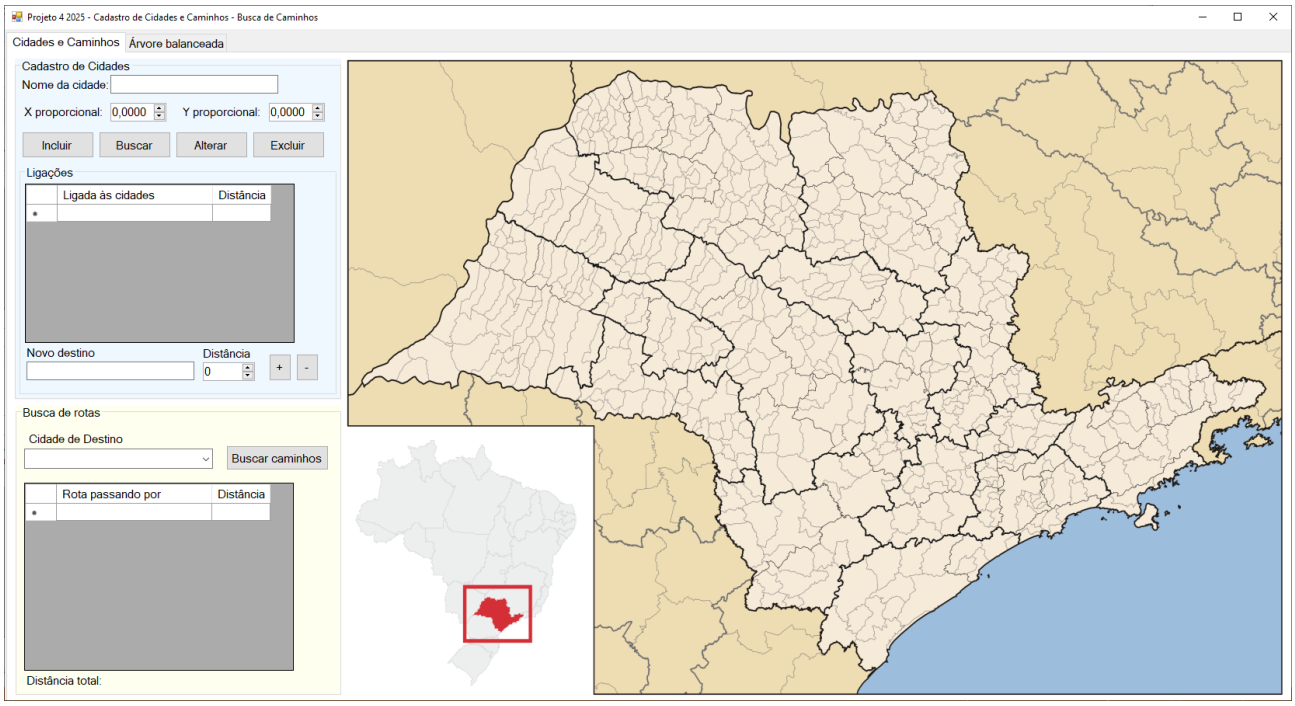
Assim, teremos uma estrutura semelhante à abaixo:



Como se observa na figura acima, em cada nó de árvore se origina uma lista de caminhos, que representa todas as saídas da cidade representada pelo nó. Observe que a árvore é mantida

balanceada logo depois da leitura recursiva dos registros do arquivo de cidades e a cada inclusão ou exclusão, pois é uma árvore AVL.

O formulário de manutenção de cidades terá um TabControl com duas guias (tabPages)., conforme modelo abaixo:



A primeira guia deverá ter campos para exibição e digitação de dados (nome de cidade, coordenadas proporcionais), bem como botões para incluir, alterar, exibir e excluir a cidade cujo nome for digitado no textBox de nome de cidade.

Na inclusão de cidade, pressiona-se o botão [Incluir] e, a partir daí, o usuário deverá digitar o nome da cidade que deseja incluir, verificar (evento Leave do textBox) se a cidade não existe. Não existindo, deve-se clicar no local do mapa onde fica essa cidade para que suas coordenadas X e Y proporcionais sejam preenchidas nos numericUpDowns udX e udY. Após isso, a cidade deve ser incluída na árvore de busca balanceada AVL

No botão [Buscar], deve-se mostrar os dados da cidade cujo nome foi digitado anteriormente no txtNomeCIdade. Caso a cidade exista na árvore de busca, deve-se exibir suas coordenadas e as cidades para as quais tem ligação.

Quando o botão [Alterar] for pressionado, deve-se alterar as coordenadas proporcionais X e Y da cidade cujo nome foi buscado anteriormente e está exibida na tela, nesse momento. Não altere nomes de cidades.

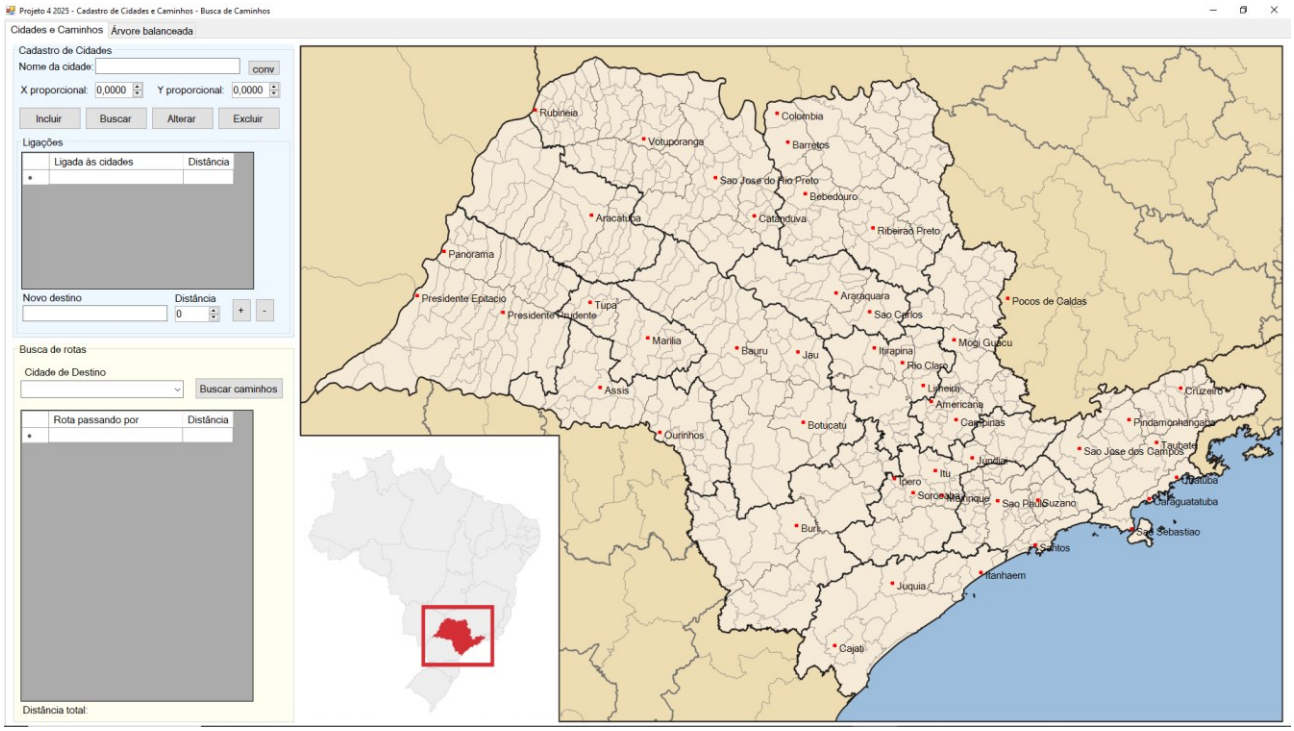
Quando o botão [Excluir] for alterado, deve-se marcar como excluida da árvore de busca a cidade cujo nome foi buscado anteriormente e está exibida na tela, nesse momento. No entanto, caso essa cidade possua ligação com alguma outra cidade, não poderá ser excluída.

Abaixo desses controles visuais, a guia terá um dataGridView onde serão exibidos os dados dos caminhos originários (ligações) na cidade que está sendo apresentada na tela. Em cada linha desse dataGridView teremos colunas para exibir o nome da cidade de destino e a distância até ela.

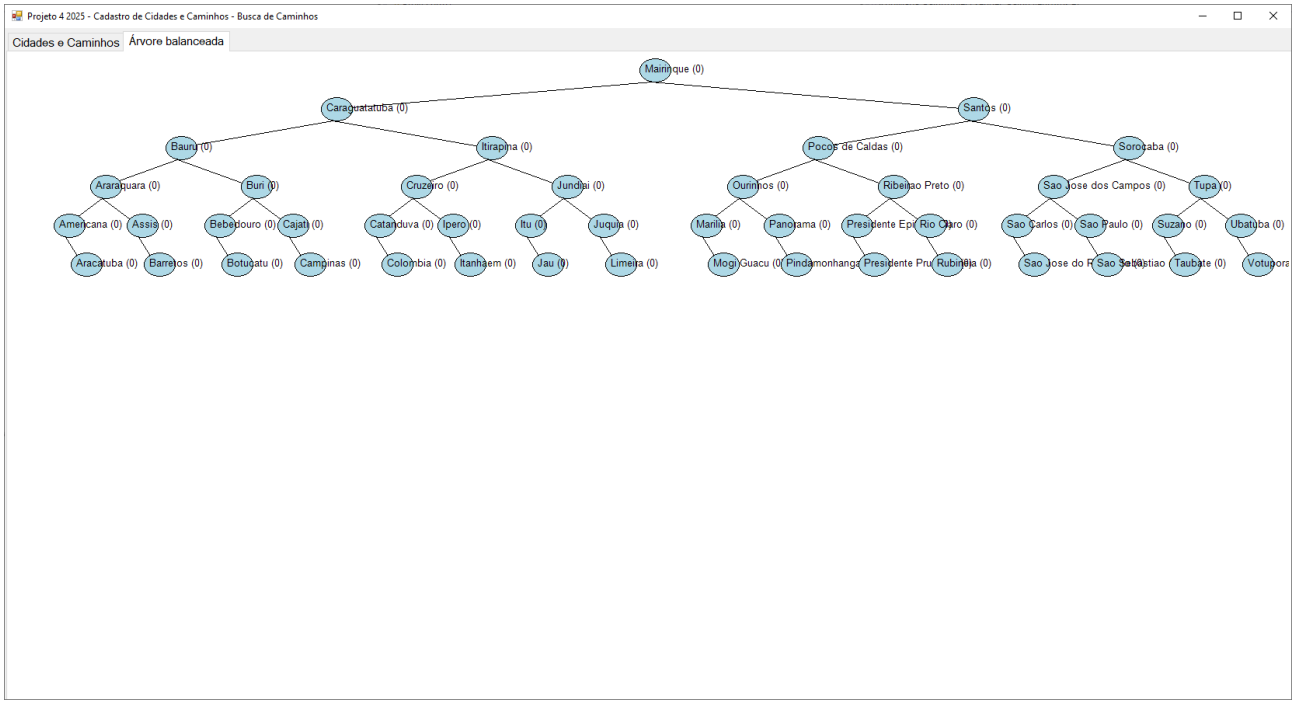
Quando o botão [+] for pressionado, deve-se incluir, na lista ligada de cidades originárias da cidade atual, um nó com o nome da cidade Novo destino e sua distância. Deve-se garantir que o nome da cidade de destino exista também na árvore, e uma ligação desta cidade com a cidade exibida deve também ser incluída na lista de cidades ligadas da cidade de destino, pois as ligações são bidirecionais.

Quando o botão [-] for pressionado, deve-se excluir a ligação que corresponde à linha selecionada no dataGridView dgvLigacoes. Lembre-se de excluir essa ligação na lista ligada de ligações da cidade de origem e na cidade de destino, em seus respectivo nós da árvore de busca.

Do lado direito do formulário, há um PictureBox onde se desenhará o mapa do Estado de São Paulo, no qual ainda deverão ser apresentados um pequeno círculo preenchido na localização geográfica de cada cidade presente na árvore de busca, bem como escrever o nome da cidade próximo a esse círculo. Abaixo temos um exemplo, não exatamente igual ao dos arquivos. É interessante, também, exibir todas as ligações entre as cidades, sempre que se atualizar o mapa.



Numa outra guia do TabControl (tpArvore), há um Panel interno onde a árvore AVL será desenhada, mostrando o nome de cada cidade nos nós exibidos e a quantidade de caminhos que ela possui, não sendo esperado que se desenhe a lista ligada originária de cada nó da árvore.



O arquivo "SaoPaulo_MesoMicroSemMunicip.png" apresenta o mapa acima sem os nomes das cidades e sem as ligações do grafo. Ele possui 2560 x 1600 pixels e as coordenadas proporcionais apresentadas no arquivo Cidades.dat foram determinadas de acordo com essas dimensões.

Obviamente, o mapa no seu tamanho original não caberá na tela. Portanto, permita que o mapa seja armazenado num componente PictureBox que se ajuste ao tamanho da tela e lembre-se que isso mudará as coordenadas de exibição de cada cidade no mapa **proporcionalmente** à mudança da altura y e largura x do mapa apresentado na tela, numa proporção entre a largura e a altura da tela com a coordenada (X, Y) original da cidade.

No evento Paint do PictureBox - exibir os nomes e locais das cidades no mapa, de acordo com a proporção entre coordenadas das cidades referentes ao tamanho original (2560 x 1600) e as dimensões **atuais** do picturebox.

Quando o usuário desejar buscar uma rota de ônibus entre a cidade atualmente exibida e uma cidade de destino qualquer, deverá selecionar a cidade de destino no combobox cbxCidadeDestino e pressionar o botão [Buscar Caminho]. Nesse momente, deverá ser usado o algoritmo de Dijkstra para determinar um caminho entre as cidades, avisando caso não exista e exibindo cada vértice da rota determinada pelo algoritmo caso um caminho seja encontrado. Também deve-se desenhar as linhas retas ligando as cidades que fazem parte dessa rota.

Quando terminar a execução do programa, a árvore de busca deverá ser percorrida em ordem, e seus registros gravados no arquivo binário de cidades em ordem crescente. Como cada nó possui uma lista ligada de caminhos, os respectivos registros armazenados em cada uma dessas listas deverão ser gravados no arquivo texto de caminhos.

Dica:Não use acentuação nos nomes de cidades a menos que você modifique o conjunto de caracteres de seu programa, para que preveja caracteres acentuados. Nos arquivos originais, estão sem acentuação.

Descrição dos arquivos

Cidades.dat

NomeCidade – string, 25 posições
CoordenadaX – float
CoordenadaY – float

Campos em binário

GrafoOnibusSaoPaulo.txt

CidadeOrigem – string
CidadeDestino – string
distancia – inteiro

campos separados por ;

```
Americana;Campinas;40
Americana;Limeira;30
Araçatuba;Bauru;195
Araçatuba;Panorama;195
```

IMPORTANTE

- Trabalho feito **em dupla;**
- Desenvolver em Windows Forms/C# no Visual Studio;
- Usar as classes ListaSimples, NoLista, NoArvoreAVL e ArvoreAVL desenvolvidas nas aulas
- Comentar adequadamente o programa e o código programado;
- Nomear os identificadores de forma adequada;
- No início dos arquivos fonte, digitar comentário com os RAs e nomes dos alunos;
- Entrega: **21/11/2025** pelo Google Classroom
- Material a ser entregue: pasta completa **do projeto contendo arquivos de dados compactados em um único arquivo, cujo nome será** raMenor_raMaior_Proj4ED.zip (24187_24202_Proj4ED.zip, por exemplo).