Instituto Nacional de matemática Pura e Aplicada

Curso de Algoritmos

Aluno: Hallison da Paz

Rio de Janeiro, outubro de 2015

Assignment sobre Grafos

Este trabalho objetiva:

```
1 Encontrar uma árvore geradora mínima para um grafo.
2 Encontrar o menor caminho entre uma fonte e um destino.
3 Encontrar a árvore de menores caminhos a partir de uma fonte.
```

Para isso, serão utilizados os grafos representados nos arquivos *maps.txt* e *BR.txt*. O presente documento apresenta as soluções passo a passo. O código (anexo) foi implementado na linguagem de programação *python 3.4*, não sendo compatível com as versões 2.x desta linguagem.

In [1]:
```
from graphs import *
from PIL import Image
```

In [2]:
```
#Carrega o grafo da América do Norte
graph = graph_initialization("map.txt", 128)
print("Tamanho total do grafo da América do Norte: ", graph_size(graph))
```

```
Tamanho total do grafo da América do Norte:  125071.0
```

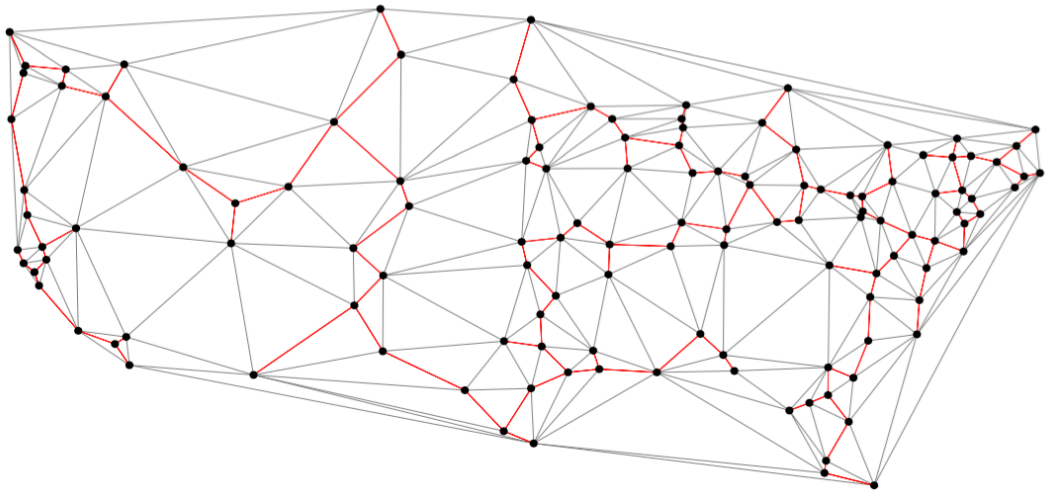
1. Árvore geradora mínima para o grafo das 128 cidades da América do Norte (map.txt).

Escolhemos uma representação do grafo por lista de adjacência. O grafo é representado por uma lista de objetos do tipo nó (Node) e cada instância de nó possui um identificador e uma lista de adjacência de seus vizinhos (representados por identificadores na lista). Para determinação de árvore geradora mínima, implementou-se o algoritmo de Prim.

In [3]:
```
#Calcula a árvore geradora mínima do grafo da América do Norte
root, mst = minimum_spanning_tree(graph)
print("Comprimento da árvore geradora mínima: ", graph_size(mst))
img = Image.open(images_folder + 'mstMap.png')
img
```

```
Comprimento da árvore geradora mínima:  22081.0
```

Out[3]:



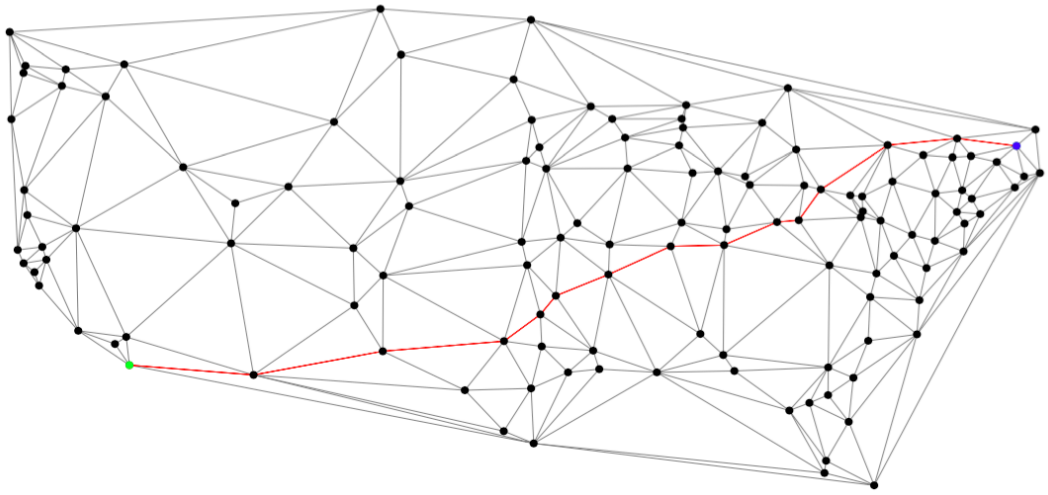
2. Caminho mínimo entre a cidade 93 e a cidade 112.

Para cálculo do caminho mínimo, uma escolha natural é o algoritmo de Dijkstra, uma vez que o nosso grafo não possui arestas de peso negativo e o algoritmo de Dijkstra adotam uma abordagem bastante semelhante ao de Prim (que já foi implementado). Ao encontrarmos o nó de destino, podemos parar a execução do algoritmo e reconstruir o caminho. Na figura *abaixo*, o vértice 93 está representado na cor verde, enquanto o vértice 112, na cor azul. As arestas do caminho mínimo encontram-se destacadas em vermelho.

In [4]:
```
root, min_path_src_dst = minimum_path(graph, 93, 112)
print("Comprimento total do caminho mínimo entre a cidade 93 e a cidade 112 é:
", graph_size(min_path_src_dst, root))
img = Image.open(images_folder + 'path93_112.png')
img
```

```
Comprimento total do caminho mínimo entre a cidade 93 e a cidade 112 é:  4737.
0
```

Out[4]:



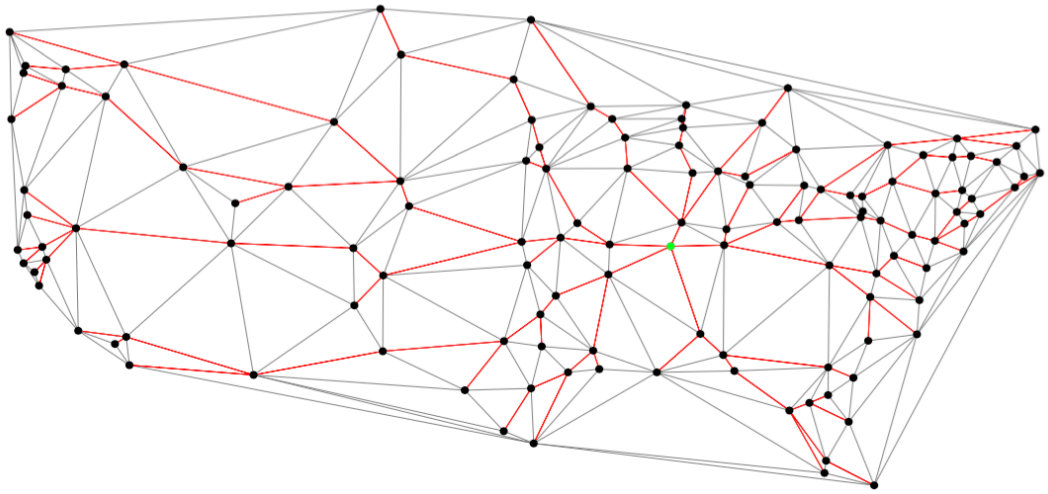
3. Árvore de menores caminhos a partir da cidade 104.

Para cálculo da árvore de caminhos mínimos, resolvemos o problema de caminhos mínimos a partir de uma única fonte apenas executando o algoritmo de Dijkstra a partir do nó 104 e deixando-o executar até o final, isto é, até que todos os nós alcançáveis a partir do nó 104 (em verde, na figura abaixo) tenham sido visitados.

In [10]:
```
root, min_path_single_src = minimum_path(graph, 104)
print("Comprimento da árvore de caminhos mínimos a partir da cidade 104 é: ", g
raph_size(min_path_single_src, root))
img = Image.open(images_folder + 'total_path104.png')
img
```

```
Comprimento da árvore de caminhos mínimos a partir da cidade 104 é:  33314.0
```

Out[10]:



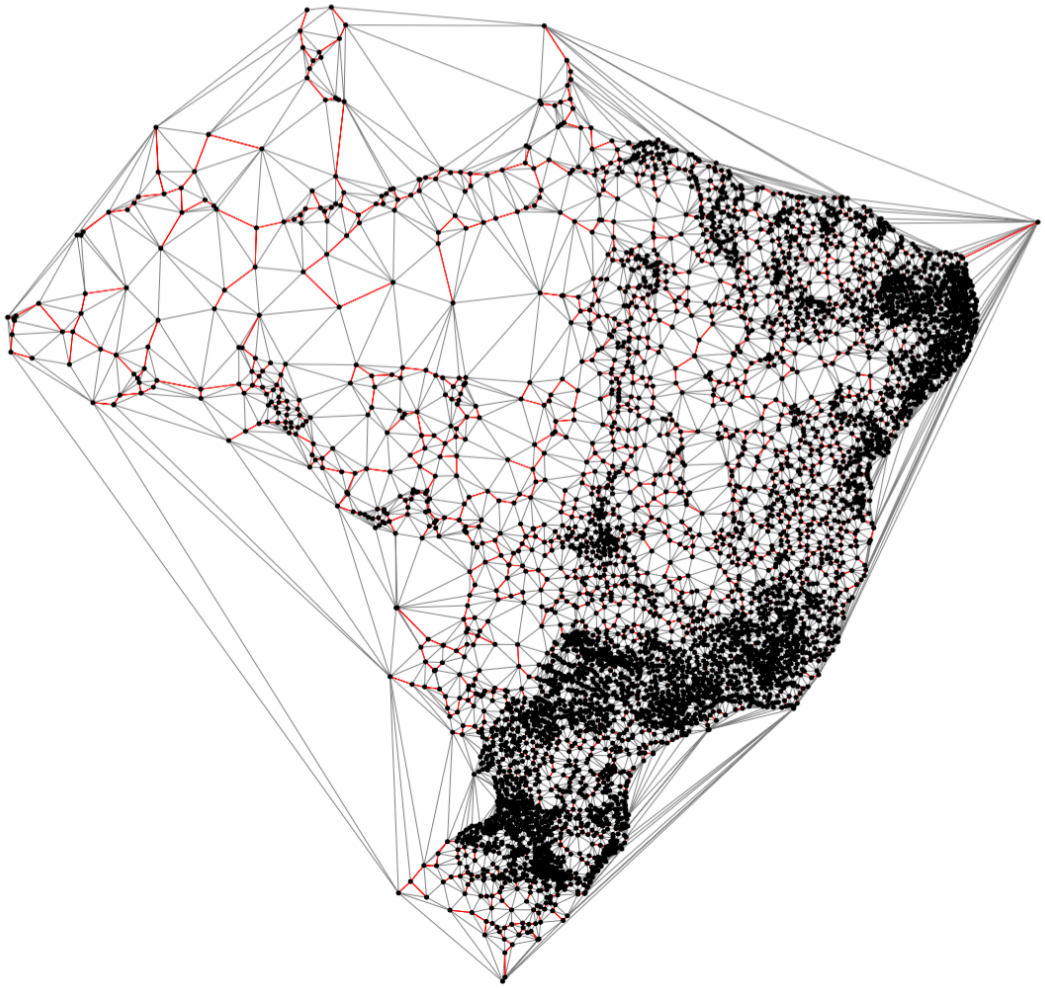
Faça o mesmo para o grafo das 5565 sedes dos municípios do Brasil, encontrando o menor caminho entre a cidade 1 e cidade 2646, e a árvore de menores caminhos a partir da cidade 2646.

In [6]:
```
#Carrega o grafo do Brasil
br_graph = graph_initialization("BR.txt", 5565)
```

In [7]:
```
#Calcula a árvore geradora mínima do grafo do Brasil
root, br_mst = minimum_spanning_tree(br_graph)
print("Comprimento da árvore geradora mínima: ", graph_size(br_mst, root))
img = Image.open(images_folder + 'mstBR.png')
img
```

```
Comprimento da árvore geradora mínima:  1040.5415191100028
```

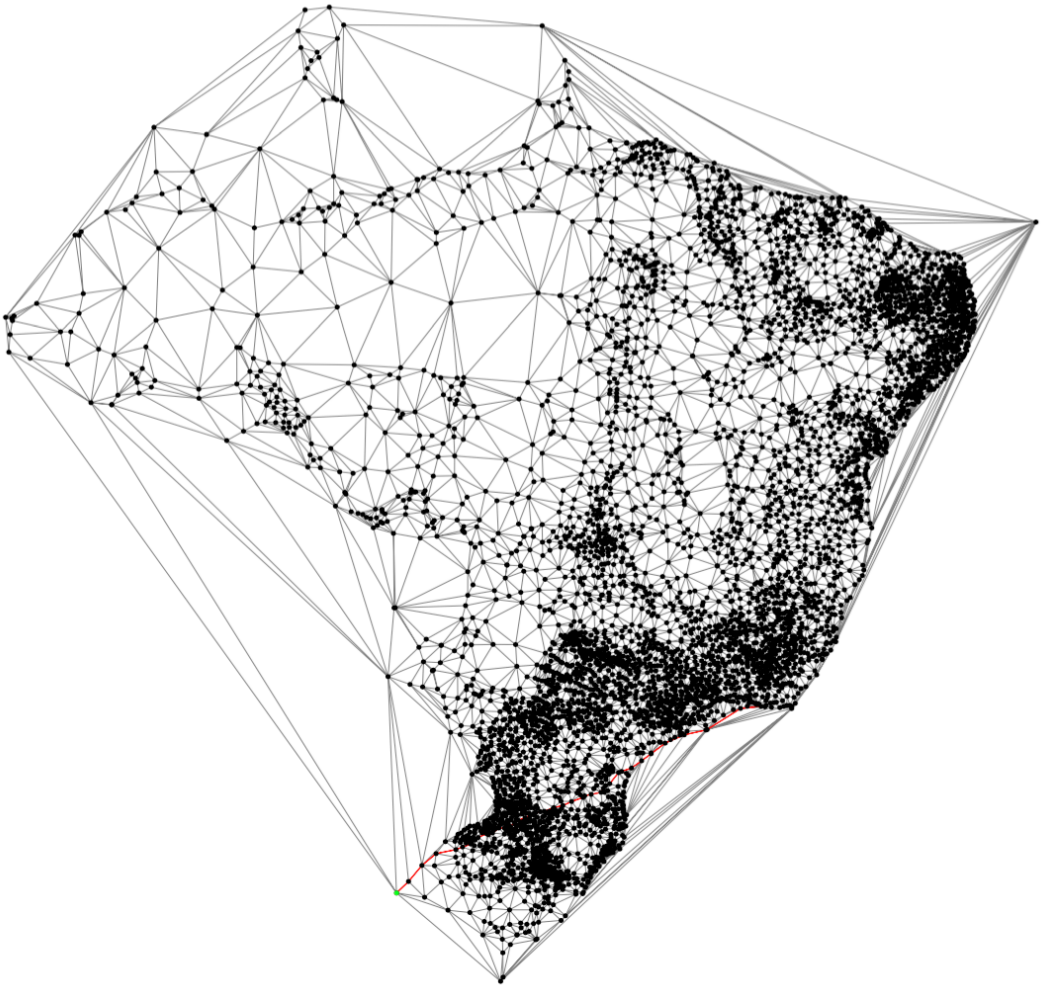
Out[7]:



In [8]:
```
root, br_min_path_src_dst = minimum_path(br_graph, 1, 2646)
print("Comprimento total do caminho mínimo entre a cidade 1 e a cidade 2646 é:
", graph_size(br_min_path_src_dst, root))
img = Image.open(images_folder + 'path1_2646.png')
img
```

```
Comprimento total do caminho mínimo entre a cidade 1 e a cidade 2646 é:  16.65
58691
```

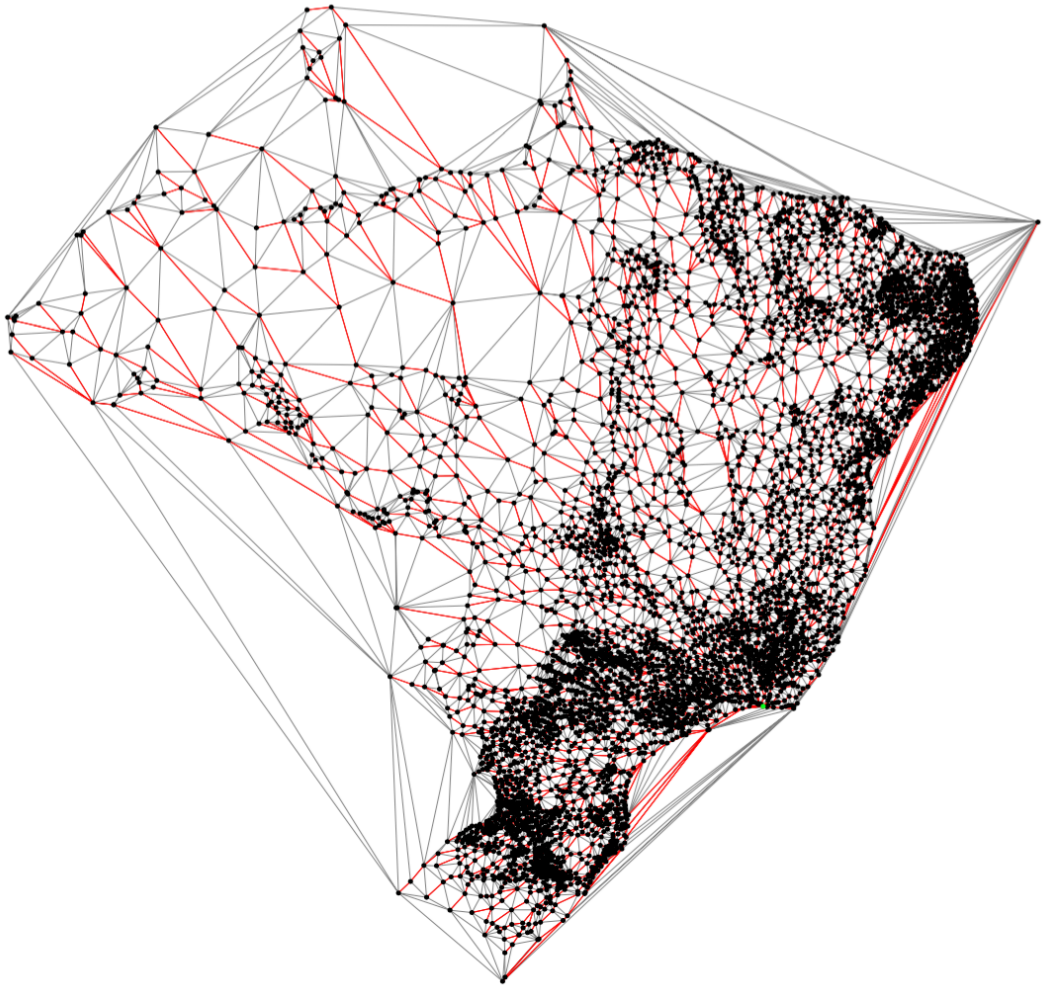
Out[8]:



In [9]:
```
root, br_min_path_single_src = minimum_path(br_graph, 2646)
print("Comprimento total da árvore de caminhos mínimos a partir da cidade 2646
é: ", graph_size(br_min_path_single_src, root))
img = Image.open(images_folder + 'total_path2646.png')
img
```

```
Comprimento total da árvore de caminhos mínimos a partir da cidade 2646 é:  17
15.811753659993
```

Out[9]:



Referências

[1] Trabalhos do curso de algoritmos: http://lhf.impa.br/cursos/algo/trabalhos.html (http://lhf.impa.br/cursos/algo/trabalhos.html)

[2] hallpaz's github: https://github.com/hallpaz/Algorithms-Course-IMPA-2015/tree/master/assignments/graphs (https://github.com/hallpaz/Algorithms-Course-IMPA-2015/tree/master/assignments/graphs)

In [ ]: