Exercício 4.5 (Tardos)

Alice Duarte Scarpa, Bruno Lucian Costa

2015-06-23

1 Enunciado

Vamos considerar uma rua campestre longa e quieta, com casas espalhadas bem esparsamente ao longo da mesma. (Podemos imaginar a rua como um grande segmento de reta, com um extremo leste e um extremo oeste.) Além disso, vamos assumir que, apesar do ambiente bucólico, os residentes de todas essas casas são ávidos usuários de telefonia celular.

Você quer colocar estações-base de celulares em certos pontos da rodovia, de modo que toda casa esteja a no máximo quatro milhas de uma das estações-base. Dê um algoritmo eficiente para alcançar esta meta, usando o menor número possível de bases.

2 Introdução

Com este exercicio vamos abordar uma técnica chamada de algoritmos gulosos sempre realizando a escolha que parece ser a melhor no momento, fazendo uma escolha ótima local, com intuito de que esta escolha leve até a solução ótima global.

Antes porém, vai ser apresentado duas soluções utilizando um algoritmo "naive" e um força bruta.

3 Soluções para o problema

3.1 Naive algoritmo

Esta primeira solução para o problema é uma das mais simples possiveis de se pensar quando confrontamos o problema.

```
def antena(lista):
   lmax = max(lista) # Valor maximo presente na lista de distancias
   ant = []
   j = 0
   for i in range(lmax):
       if j >= lmax:
           if j - ant[-1] <= 4:
               return ant
       j += 4
       ant.append(j)
   return ant
```

3.2 Força Bruta

```
import math, numpy

def antena(lista):
  lmax = max(lista)
  ant = []

  while lista != []: # Realizar procedimento ate todas as casas cobertas
      torre = numpy.random.randint(1, lmax) #fixando uma torre em um ponto qualquer
      for i in range(len(lista)): #Percorrendo toda a lista
        for i in lista: #
            if i >= torre-4 and i <= torre+4: # Verifica se tem casa esta coberta
                lista.remove(i) # remove a casa coberta
                ant.append(torre) # adciona a torre a lista

  ant = list(set(ant)) # Remove as torres colocadas em duplicatas
  ant.sort() #Ordena as torres

  return ant
```

4 Implementação

4.1 Fluxo válido com demandas não-nulas

5 Complexidade

TODO: calcular a complexidade do algoritmo