Lista de Exercícios Sobre Algoritmos Gulosos

1. A via Dutra é uma rodovia que liga Campinas ao Rio de Janeiro. A rodovia possui pontos de atendimento aos transitantes, que são lugares com uma pequena construção com telefone, banheiros e outras facilidades. A administração da via Dutra resolveu colocar também, em alguns destes pontos, ambulâncias para atendimento a emergências. Como ficaria muito caro alocar uma ambulância a cada ponto, a administração resolveu que não alocaria ambulâncias a todos os pontos, mas que nenhum ponto ficaria distante mais que q quilômetros de uma ambulância. A administração quer também minimizar o número de ambulâncias que serão usadas para este serviço.

   Exemplo: x———x——x————————————x——x—x——x

   - Quantas ambulâncias seriam necessárias para resolver o problema do exemplo cima, considerando q=4? 2
   - Elabore um algoritmo que resolve o problema dado que sejam fornecidos q e um vetor d de dimensão n (número de pontos) no qual d_i indica a distnância do ponto i ao ponto $i+1$.

Algoritmo 0.1 (AmbulanciasDutra(d, n))
[Entrada] d, n, e q.

```
count:=0; i:=1;
while i ≤ n do
  dist := 0;
  while d ≤ q and i ≤ n do
    dist = dist + d[i];
    i++;
  end while
  count++;
  locate[count] := i-1;
  dist := 0;
  while dist ≤ q and i ≤ n do
     dist = dist + d[i];
     i++;
  end while
end while
retorne(count, locate);
```

   - Qual a complexidade do seu algoritmo? Linear.
   - Argumente porque o seu algoritmo está correto. O algoritmo sempre posiciona cada ambulância na posição mais longe da primeira posição à esquerda coberta por ela. Desta forma, não seria possível posicionar mais à direita para não extrapolar q. E se posicionar mais à esquerda, cobriria um número menor de locais ao lado direito, e seria necessário um maior número de ambulâncias.

2. (Problema da Mochila fracionária) Considere um conjunto de elementos $S = s_1, s_2, ..., s_n$. Cada elemento s_i possui dois valores associados que se referem ao peso p_i e ao custo do elemento c_i. Suponha que alguns elementos serão inseridos numa mochila que suporta no máximo um peso P. Considere o problema fracionário, ou seja, um elemento pode ser selecionado por inteiro, ou apenas uma fração do mesmo pode ser inserida na mochila.

(a) Desenvolva um algoritmo guloso que seleciona um conjunto de elementos cuja soma dos seus custos seja máxima, respeitando a capacidade de peso da mochila. O algoritmo deve retornar informação sobre que elementos foram selecionados, e qual a fração usada de cada um.

```
MochilaFrac(s, p, c)
inicio
para i = 1 ate |s| faca
  vetAux.beneficio[i] = c[i]/p[i];
  vetAux.indice[i] = i;
  OrdeneDecr(vetAux pelo valor do beneficio);
  peso = p
  para i = 1 ate |s| faca
    se p[vetAux.indice[i]] > peso
      inicio
      imprime(``Selecionado '' + vetAux.indice[i] +
              `` quantidade '' +
              100 * (peso/p[vetAux.indice[i]]));
      break;
      fim;
    senao
      inicio
      imprime(``Seleciona '' + vetAux.indice[i] +
              `` quantidade 100%'');
      peso = peso - p[vetAux.index[i]];
      fim;
  fim para;
fim para;
fim MochilaFrac;
```

(b) Analise o seu algoritmo. $O(n \log n)$

(c) Argumente o porque o algoritmo é correto. O algoritmo seleciona sempre o máximo de cada objeto com maior benefício de custo pelo peso.

Suponha que uma fração de algum objeto não deveria ter sido selecionada. Então esta deveria ser substituída por uma fração de outro(s) objeto(s) com menor benefício. Se tem menor benefício é porque tem um peso maior ou um custo menor, o que geraria uma solução pior. Desta forma o algoritmo proposto gera a solução ótima.

3. (Troco em moedas) Considere o problema de fazer a troca de K centavos usando o menor número de moedas. Suponha que o valor de cada moeda seja um inteiro. Suponha que as moedas disponíveis tenham as denominações que são potências de c, isto é, as denominações são c^0, c^1, ..., c^n para inteiros $c > 1$ e $n \geq 1$.

(a) Para moedas de valores R$ 3, R$ 1, R$ 9, qual o número mínimo de moedas para fornecer troco para R$ 52? 5 x R$ 9 = R$ 45
2 x R$ 3 = R$ 6
1 x R$1 = R$ 1
= R$ 52

(b) Descreva um algoritmo guloso que efetue a troca consistindo em n valores de moedas diferentes. O algoritmo recebe como entrada um conjunto de n moedas e K. Se considerar como entrada o conjunto M de denominações de moedas

```
OrdeneDecr(M); //Ordene M em ordem decrescente
               //do valor das moedas.
r = k;
count = 0;
para i = 1 ate |M| faca
   x = ⌊r / S[i]⌋
   se x > 0 entao
      imprime(x + `` moedas de valor S[i].'');
      count += x;
      r -= x * S[i];
      se r == 0 entao i = |M|;
   fim se;
fim para;
```

Se considerar como entrada apenas c e n

```
r = k;
count = 0;
para i = n ate 1 decremento 1 faca
   x = ⌊r / c[i]⌋
   se x > 0 entao
      imprime(x + ``moedas de valor '' + c[i]);
      count += x;
      r -= x * c[i];
      se r == 0 entao i = 1;
   fim se;
fim para;
```

(c) Analise seu algoritmo. Se considerar a implementação que tem como entrada o conjunto M de denominações de moedas a complexidade é $O(n) + O(n \log n) = O(n \log n)$
Se considerar a implementação que tem como entrada apenas c e n a complexidade é $O(n)$

(d) Argumente porque o seu algoritmo sempre produz uma solução ótima. Como as moedas são potências de c, uma moeda maior é sempre composta por um conjunto inteiro de moedas menores.

Suponha que a resposta do algoritmo não seja a ótima, ou seja, uma moeda selecionada não deveria estar no conjunto. Como o algoritmo proposto sempre seleciona a moeda de maior valor possível, se uma moeda fosse removida do conjunto selecionado, deveria ser trocada por mais de uma moeda, gerando uma solução pior. Desta forma, o algoritmo retorna a solução ótima.

4. (Prim)

Suponha que o grafo G=(V,E) seja representado como uma matriz de adjacências. Forneça uma implementação simples do algoritmo de Prim para esse caso que seja executada no tempo $O(V^2)$.

a) Escreva o pseudocódigo do algoritmo.

Algoritmo 0.2 (Prim)
Entrada Um grafo G, a raiz r da árvore a ser aumentada

```
for (u=1; n; u++)
  Q[u] = 1
  custo[u] = ∞
  Π[u] = nil
end for
custo[r] = 0
indice = r
 for (i=1; n; i++)
  u = indice
  Q[u] = 0
  for (v=1; n; v++)
    if (mat[v][u] == 1&Q[v] == 1&w[u,v] < custo[v])
        then
          custo[v] = w[u,v]
          Π[v] = u
    end if
  end for
  min = ∞
  for (v=1; n; v++)
  if (Q[v] == 1&custo[v] < min)
        then
          min = custo[v]
          indice = v
    end if
  end for
end for
```

b) Argumente brevemente porque a sua estratégia gulosa garante respostas ótimas de acordo com a definição do problema.

Pela propriedade do corte, mesmo argumento do prim original - a árvore é aumentada em cada etapa com uma aresta que contribui com a quantidade mínima possível para o peso da árvore