UNIVERSIDADE FEDERAL DE VIÇOSA
DEPARTAMENTO DE INFORMÁTICA
INF332 – Projeto e Análise de Algoritmos

Trabalho Computacional

As implementações devem ser feitas nas linguagens C++ ou Python. Apresentar os resultados e os códigos dos programas em um documento PDF.
Este trabalho pode ser feito em DUPLA.

1. Dado um conjunto de n pontos $p_1, p_2, ..., p_n$ no plano euclidiano. Implemente um algoritmo de **Força Bruta** para determinar a solução ótima do Caixeiro Viajante (ciclo hamiltoniano com menor distância total). O algoritmo deve gerar todas as permutações dos números 2 a n. Para cada permutação gerada, adicione o ponto 1 no início e no final da permutação e calcule a distância total do ciclo hamiltoniano.

Exemplo (para $n = 6$): para a permutação 4-6-2-5-3, obtém-se o ciclo hamiltoniano **1**-4-6-2-5-3-**1**. A distância total do ciclo será: distT = $d(p_1, p_4) + d(p_4, p_6) + d(p_6, p_2) + d(p_2, p_5) + d(p_5, p_3) + d(p_3, p_1)$, onde $d(p_i, p_j)$ é a distância euclidiana entre os pontos p_i e p_j.

Seu algoritmo deve executar no máximo 3600 segundos (1 hora). Se nesse tempo o algoritmo não consegue gerar todas as $(n\text{-}1)!$ permutações, ele deve retornar o melhor ciclo hamiltoniano encontrado (em 3600 segundos).

O algoritmo deve mostra/graficar/plotar o **melhor ciclo hamiltoniano encontrado** com seu respectivo **valor de distância total** e o **tempo de execução** (em segundos) gasto.

2. **Divisão e Conquista** para determinar uma solução aproximada do Problema do Caixeiro Viajante.

Implemente o seguinte algoritmo baseado em Divisão e Conquista (OBS. use a mesma estrutura, os mesmo nomes das variáveis e funções):

- Ordene o conjunto dos n pontos p = {p[1], p[2], ..., p[n]} em ordem crescente da coordenada x.

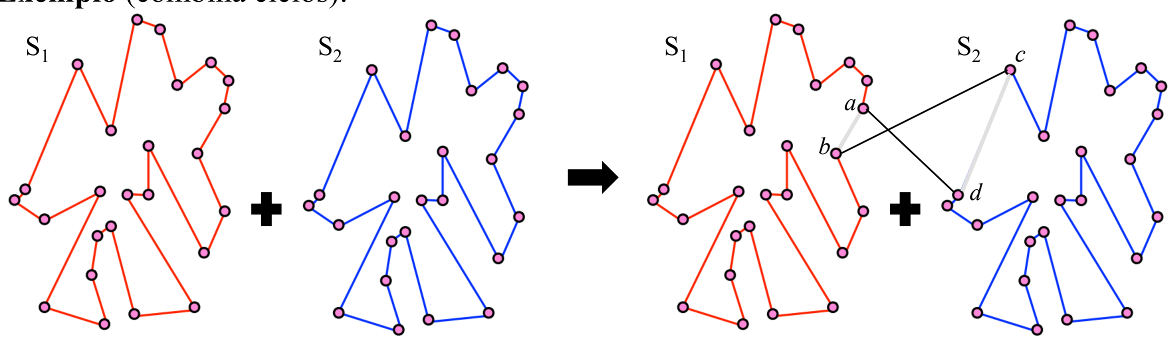
DivConqPCV(p, l, r):
//no início $l = 1$, $r = n$
Se $r - l <= 2$:
    Se $r - l == 1$: **return** {(p[l], p[r]), (p[r], p[l])}, *dist*1 *//se existem 2 pontos*
    Se $r - l == 2$: **return** {(p[l], p[l+1]), (p[l+1], p[r]), (p[r], p[l])}, *dist*2 *//se existem 3 pontos*
Senão:
    $m = (l + r)/2$
    (S_1, $distS_1$) = **DivConqPCV**(p, l, m)
    (S_2, $distS_2$) = **DivConqPCV**(p, m+1, r)
    (S, $distS$) = **CombinaCiclos**(S_1, S_2, $distS_1$, $distS_2$)
    return (S, $distS$)

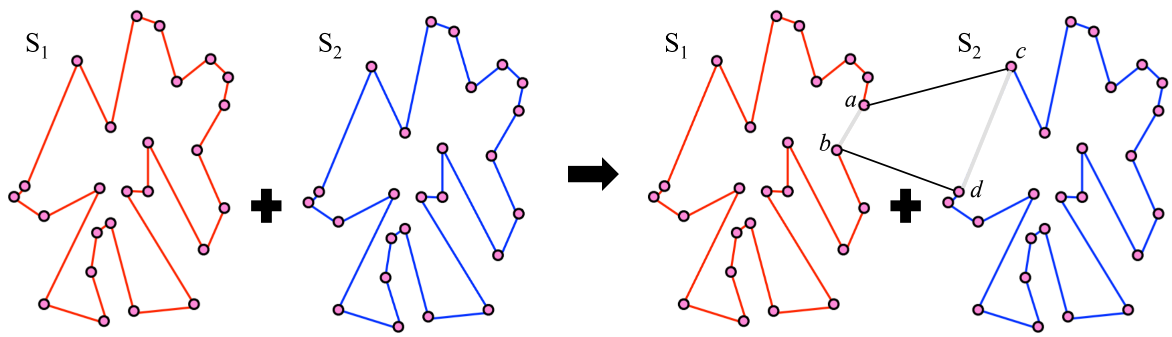
Combina Ciclos:
Para combinar os ciclos S_1 e S_2 faça o seguinte:
- Procurar as arestas (a, b) em S_1 e (c, d) em S_2 tal que um dos seguintes valores seja o mínimo possível:

$Valor1 = dist(a, d) + dist(b, c) - dist(a, b) - dist(c, d)$
$Valor2 = dist(a, c) + dist(b, d) - dist(a, b) - dist(c, d)$
Onde $dist(p, q)$ denota a distância euclidiana entre os pontos p e q.

- Se $Valor1$ é o menor:
  $S = S_1 + S_2 - \{(a, b)\} - \{(c, d)\} + \{(a, d)\} + \{(b, c)\}$, $distS = distS_1 + distS_2 + Valor1$
- Se $Valor2$ é o menor:
  $S = S_1 + S_2 - \{(a, b)\} - \{(c, d)\} + \{(a, c)\} + \{(b, d)\}$, $distS = distS_1 + distS_2 + Valor2$

Exemplo (combina ciclos):





Mostrar/graficar/plotar o **ciclo** S encontrado com seu respectivo **valor de distância total** ($distS$) e o **tempo de execução** (em segundos) gasto pelo algoritmo de Divisão e Conquista.

3. Além das soluções obtidas (gráficos), apresente os resultados para as instâncias fornecidas, da seguinte maneira:

Instâncias	Força Bruta		Divisão e Conquista	
	Distância total	Tempo execução (seg)	Distância total	Tempo execução (seg)
TSP_10				
TSP_12				
TSP_15				
TSP_20				
TSP_30				
TSP_50				
TSP_80				
TSP_100				

Cada instância contém o número de pontos (n) e as coordenadas (x, y) dos pontos $p_1, p_2, ..., p_n$.