Caixeiro Viajante

1. Instalar pacotes se necessario

install.packages(c(“dplyr”, “ggplot2”, “combinat”))

install.packages(“rmarkdown”, “knitr”)

Rodar no bash para gerar o arquivo html

Rscript -e “rmarkdown::render(‘caixeiro_viajante.Rmd’)”

library(dplyr) library(ggplot2) library(combinat)

Criar pasta ‘resultados’ para salvar saídas, se não existir

if (!dir.exists(“resultados”)) { dir.create(“resultados”) }

2. Gerar coordenadas aleatórias para cidades

gerar_cidades <- function(n) { set.seed(42) tibble(cidade = paste0(“C”, 1:n), x = runif(n, 0, 100), y = runif(n, 0, 100)) }

print(gerar_cidades)

3. Calcular distância entre duas cidades

distancia <- function(c1, c2) { sqrt((c1$x - c2$x)^2 + (c1$y - c2$y)^2) }

4. Calcular distância total de um caminho

calcular_distancia_total <- function(caminho, cidades) { total <- 0 for (i in 1:(length(caminho)-1)) { total <- total + distancia(cidades[caminho[i],], cidades[caminho[i+1],]) } total <- total + distancia(cidades[caminho[length(caminho)],], cidades[caminho[1],]) return(total) }

5. Aplica a força bruta

forca_bruta <- function(cidades) { inicio <- Sys.time() perms <- permn(1:nrow(cidades)) melhor_dist <- Inf melhor_caminho <- NULL

for (p in perms) { d <- calcular_distancia_total(p, cidades) if (d < melhor_dist) { melhor_dist <- d melhor_caminho <- p } } tempo <- Sys.time() - inicio return(list(caminho = melhor_caminho, distancia = melhor_dist, tempo = as.numeric(tempo, units = "secs"))) }

6. Aplica heurística do vizinho mais próximo

heuristica_vizinho <- function(cidades) { inicio <- Sys.time() restantes <- 2:nrow(cidades) atual <- 1 caminho <- c(atual)

while (length(restantes) > 0) { proximo <- restantes[which.min(sapply(restantes, function(i) distancia(cidades[atual,], cidades[i,])))] caminho <- c(caminho, proximo) restantes <- setdiff(restantes, proximo) atual <- proximo }

tempo <- Sys.time() - inicio return(list(caminho = caminho, distancia = calcular_distancia_total(caminho, cidades), tempo = as.numeric(tempo, units = "secs"))) }

7. Faz testes para vários tamanhos de cidades

resultados <- tibble()

for (n in 5:8) { cidades <- gerar_cidades(n) res_bruto <- forca_bruta(cidades) res_heuristica <- heuristica_vizinho(cidades)

resultados <- resultados %>% bind_rows(tibble(Cidades = n, Distancia_Bruta = round(res_bruto$distancia, 2), $Tempo_Brut$ $round(res_brutotempo,4)$, Distancia_Heuristica = round(res_heuristica$distancia, 2), $Tempo_Heuristica$ = $round(res_heuristicatempo,4)$)) }

print(resultados)

8. Gera Plot

grafico_resultado <- ggplot(resultados, aes(x = Cidades)) + geom_line(aes(y = Distancia_Bruta, color = "Bruta")) + geom_line(aes(y = Distancia_Heuristica, color = "Heuristica")) + labs(title = "Comparacao de Distancia", y = "Distancia Total", x = "Numero de Cidades") + theme_minimal()

Mostrar o gráfico

print(grafico_resultado)

Salvar o gráfico

ggsave("resultados/grafico-resultado-outras-cidades.png", plot = grafico_resultado, width = 8, height = 6, dpi = 300)

9. Analisando crescimento (Big-O experimental)

Gráfico de tempo de execução

grafico_tempo <- ggplot(resultados, aes(x = Cidades)) + geom_line(aes(y = Tempo_Bruta, color = “Bruta”)) + geom_line(aes(y = Tempo_Heuristica, color = “Heuristica”)) + # scale_y_log10() + labs(title = “Tempo de Execucao - Forca Bruta vs Heuristica”, y = “Tempo (segundos)”, x = “No de Cidades”) + theme_minimal()

print(grafico_tempo)

Salvar o gráfico também

ggsave(“resultados/grafico-tempo.png”, plot = grafico_tempo, width = 8, height = 6, dpi = 300)

write.csv(resultados, “resultados/resultados_cidades.csv”, row.names = FALSE)