<pre>In [2]: from IPython.display import Image, display x = Image('pictures/1_2.png', height=300) y = Image('pictures/1_2_b.png', height=300) display(x) display(y)</pre> 1.2 Grafy
Dany jest graf nieskierowany V = (V, E), gdzie V = {v ₁ , v ₂ ,, v _V } i E = {e ₁ , e ₂ ,, e _E }, gdzie e _i = {j, k} : j, k ∈ V. Napisz program który działa jak następuje. 1. Dla każdego wierzchołka losuje pozycję (v _i ^x , v _i ^y) na płaszczyźnie [0, 100] → [0, 100]. 2. Dla zadanego V generuje drzewo rozpinające wierzchołki V w następujący sposób: (a) dzieli graf na dwa podgrafy A, B po połowie (±1) wierzchołków (b) wywołuje się rekurencyjnie na utworzonych podgrafach A, B
 (c) w wyniku rekurencji zwracane są dwa grafy C, D (d) z C i z D algorytm wybiera po jednym wierzchołku, o najmniejszej liczbie sąsiadów w C i w D (jeśli jest wiele takich wierzchołków – losowy wybór) (e) łączy wybrane wierzchołki i zwraca tak utworzony graf
 (f) jeśli w podgrafie, podczas rekurencyjnego wywoływania jest tylko jeden wierzchołek, to algorytm zwraca ten właśnie wierzchołek 3. Wyświetla generację drzewa krok po kroku (np. co sekundę), zaczynając od (kolejno ponumerowanych) wierzchołków bez połączeń, i dodając połączenia jedno za drugim. Na połączeniach wyświetlana jest odległość zaokrąglona do 2 miejsc po przecinku. Uwaga: Upewnij się, że przy każdym uruchomieniu programu dla takich samych V program zawsze generuje to samo drzewo – wykorzystaj random.seed().
<pre>In [1]: import networkx as nx import numpy as np from math import sqrt from time import sleep import matplotlib.pyplot as plt</pre> In [2]: # ilość wierzchołków n = 20
<pre>In [3]: # losowanie pozycji z przedziałów [0, 100], [0, 100] V_pos = list() for i in range(n):</pre>
<pre>In [5]: G = nx.Graph() V = np.arange(n) G.add_nodes_from(V) nx.draw(G, pos=V_pos, with_labels=True)</pre> 1 6 15 19
8 11 ₁₂ 13 18 18
Treść zadanie można interpretować na rózne sposoby: 1. Program kończy działanie i pozostawia dwa grafy które są drzewami rozpinającymi. W treści nie jest wyraźnie napisane czy algorytm ma dodać jeszcze jedno połączenie ostatecznych grafów A i B. Połączenia między krawędziami mają być dodawane po podziale na podgrafy C, D.
2. Ma zostać utworzone drzewo rozpinające - więc musimy dodać krawędź między dwoma końcowymi grafami A i B. Aby spełnić wymaganie tej interpretacji wystarczy dodać tylko jeden warunek w kodzie aby na koniec dodać jeszcze jedną krawędź. In [6]: # interpretacja 1 def euclides(a, b): """ Funkcja oblicza odległość euklidesową między dwoma wierzchołkami :param list a: wzpórzędne wierzchołka a [x, y]
<pre>:param list b: wzpórzędne wierzchołka b [x, y] :return float: odległość euklidesowa między dwoma wierzchołkami zaokrąglona do 2 miejsc po przecinku """ return round(sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2), 2) def least_neighbor(g, nodes): """ Funkcja wyznacza wierzchołek o najmniejszej liczbie sąsiadów</pre>
<pre>:param g: graf dla którego działa algorytm :param list nodes: lista wierzchołków podgrafu :return int: wierzchołek o najmniejszej liczbie sąsiadów """ node = None min_nei = float('+inf') for n in nodes: if len(list(g.neighbors(n))) < min_nei: min_nei = len(list(g.neighbors(n))) node = n</pre>
<pre>return node def plot_graph(g): """ Funkcja wyświetlająca graf :param g: graf :return: None """ nx.draw(g, pos=V_pos, with_labels=True, node_color='purple')</pre>
<pre>edge_labels = nx.get_edge_attributes(g, 'length') nx.draw_networkx_edge_labels(g, pos=V_pos, edge_labels=edge_labels) plt.show() def split_graph(g, nodes): """ Funkcja przeporwadzająca algorytm podziału na 2 grafy i utworzenia drzewa rozpinającego :param nx.Graph g: graf na którym stosujemy algorytm :param list nodes: lista wierzchołków grafu</pre>
<pre>:return a_nodes, b_nodes: dwie listy z wierzchołkami podgrafu """ # sprawdzenie możliwości podziału na 2 podgrafy A, B if len(nodes) > 2: # podzial na wierzchołki A i B a_nodes = nodes[0:len(nodes)//2] b_nodes = nodes[len(nodes)//2:] # sprawdzenie możliwości podziału na 2 podgrafy C, D if len(a_nodes) >= 2: # rekurencyjne wywołanie funkcji</pre>
<pre># vierzchołki o najmniejszej liczbie sąsiadów v1 = least_neighbor(g, ac) v2 = least_neighbor(g, ad) dist = euclides(V_pos[v1], V_pos[v2]) # dodanie krawędzi do grafu g.add_edge(v1, v2, length=dist) # wyświetlenie kolejnego etapu na grafie</pre>
<pre>sleep(.5) plot_graph(g) # sprawdzenie możliwości podziału na 2 podgrafy C, D if len(b_nodes) >= 2: # rekurencyjne wywołanie funkcji bc, bd = split_graph(g, b_nodes) # wierzchołki o najmniejszej liczbie sąsiadów v1 = least_neighbor(g, bc) v2 = least_neighbor(g, bd)</pre>
<pre>dist = euclides(V_pos[v1], V_pos[v2]) # dodanie krawędzi do grafu g.add_edge(v1, v2, length=dist) # wyświetlenie kolejnego etapu na grafie sleep(.5) plot_graph(g) return a_nodes, b_nodes</pre>
<pre>else: return nodes[:1], nodes[1:] split_graph(G, list(G.nodes)) 6</pre>
90.67
90.67
90.67
6 9 15 17 9 0.67 19 19 19 19 19 19 19 19 19 19 19 19 19
9
90.67 10 13 13 13 13 13 13 13 13 13 13
12 18 19 19 19 19 19 19 19 19 19 19 19 19 19
10 10 10 10 10 10 10 10 10 10
0
13 5 24.5 12 18 18
13.07 - 30.07 - 8 - 44.1 - 13.07 - 13.05 - 13.
1
10 10 10 10 10 10 10 10 10 10
5
90.67 10 10 10 10 10 10 10 10 10 10
93.65 44.1 93.65 44.1 93.65 144.1 15.28.0 16.09 17.09 18.00 19
10
99 17 18 19 19 19 19 19 19 19 19 19 19 19 19 19
1
19.06 19
19.06 19.07 19.06 19.07 19.06 19
19 06 19 06 10
100 (100 (100 (100 (100 (100 (100 (100
19 86 5 4 4 1 1 10 20
15 W 10 1 10 2
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
$\begin{array}{cccccccccccccccccccccccccccccccccccc$
Circ(A): (ED, 11, 2, 3, 4, 5, 5, 5, 7, 8, 7), 115, 11, 12, 13, 13, 14, 15, 16, 17, 18, 131; Treef, Zadanie mozina interpretować na różnie sposoby: 1. Tak jak powyzaj program konczy działanie i pososławia dwa grafy które są drzewami rozpinającymi. W treści nie jest wyzatnie napisane czy spopymin meddecij jestze jelkon polszenie o statecznych grafów A I B. Polsczenia miectry krawadzami mają być dodoswane po podziale na podziale i pososławia dwa grafy które są drzewami rozpinającymi. W treści nie jest wyzatnie napisane czy spopymin w opisane o potzenie o statecznych grafów A I B. Polsczenia miectry krawadzami mają być dodoswane po podziale na podziale na podziale krawadzami czy spopymin w opisane w
Out (6): 11°C, 1, 2, 3, 4, 5, 5, 7, 8, 9), (10, 11), 12, 13, 14, 15, 15, 14, 17, 18, 19): Trend radiation marizon interpretational via ridure sproadly 1. Tak jak powyzej prugnem kończy działamie i pozostawa dwa grafy które są dziewami ruzprusącymi. W treści mie jest wyrażnie ruspisanie cy adopróm na dociał jeszczej edno połączenie osiałacznych grafik Ari B. Połączenia między krawędziami ruspi być dodawame po podacie mogodnych C. D. 2. Ma zodski utwozone dziewo rozpinające - wjec musimy dodak iczwędź między dwoma końcowymi grafiami A i B. Aby spełnić wymanie laj interpreta wystarczy dodać tykio jeden warune w kodze. 13. (8): 4 Teorogony pozoszanie w znać śreby facesi intra kować z połączenia interpreta podacie z pożecnia w kodze. 13. (8): 4 Teorogony pozoszanie w znać śreby facesi intra kować z połączenia interpreta podacie z poda
Durt(e): (Ch. ** 2-2, 3, 4, 5, 5, 6, 7, 8, Mt. ** 10, 11, 13, 13, 14, 10, 18, 17, 18, 111) Field Zaddnie moma interpretovać na różne sposoby: 1. Tak jak powyżej pregram koncy zdriażnia je prowtawia dwa groły które sy dzewanii nzpirogcymi. W treści nie jest wyrażnia majskimi czy utjęcyti m na dodać jest-ze jedniu połążenia o-satz-czych gradów A i B. Połążenia mędzy kowydzianii mają był dodawana po podada i na podajady C. D. 2. Mie zostać zdrożnenia dzewa z doży "szedzaszawac" dodaczenia. 1. (8): - ** ** ** ** ** ** ** ** ** ** ** ** **
Total continue to the state of the processor of the state of the processor of the state of the s
Co. (C) 1 1 2 2 2 2 2 2 2 2
to (2) If the travel of the tr
20.00. The State A. C.
The country of the co
The control of the co
Figure 1 and
The content of the
The control of the co
The Control of the Co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co
The control of the co

