arvore_enraizada.py

min_height_spanning_tree()

Entrada:

Dois inteiros separados por espaço: V E
V := quantidade de vértices do grafo
E := quantidade de arestas do grafo

E linhas com dois strings separados por espaço: V U
V, U := vértices ligados por uma aresta

Saída:

Árvore geradora enraizada de altura mínima:
{'V': [filhos de V], …, 'root': R}
V := vértices do grafo
R := vértice raiz da árvore geradora enraizada de altura mínima

pontes.py

bridges()

Entrada:

Dois inteiros separados por espaço: V E
V := quantidade de vértices do grafo
E := quantidade de arestas do grafo

E linhas com dois strings separados por espaço: V U
V, U := vértices ligados por uma aresta

Saída:

Pontes do grafo: [('V','U'), …]
V, U := vértices ligados por uma ponte no grafo

djikstra.py

shortest_path()

Entrada:

Dois inteiros separados por espaço: V E
V := quantidade de vértices do grafo
E := quantidade de arestas do grafo

E linhas com dois strings e um inteiro separados por espaço: V U W
V, U := vértices ligados por uma aresta
W   := peso da aresta que liga V, U

Um string: S
S := vértice de origem, sendo S um vértice do grafo

Um string: D
D := vértice de destino, sendo D um vértice do grafo

Saída:

Caminho de custo mínimo de S até D:
{‘path’: [S, … , D], ‘weight’: W(S,D)}
path   := vértices no caminho mínimo de S até D
W(S,D) := custo do caminho mínimo de S até D

egipcio.py

egyptian()

Entrada:

Dois inteiros separados por /: N/D
N := numerador da fração
D := denominador da fração, sendo D >= N

Saída:

Representação egípcia da fração: [‘1/d’, …]
‘1/d’ := fração unitária (parcelas da representação egípcia)

troco.py

coins()

Entrada:

Um inteiro: N
N := valor do troco, sendo N > 0

Saída:

Quantidade de moedas de cada valor:
{Valor da moeda: Quantidade de moedas, … }