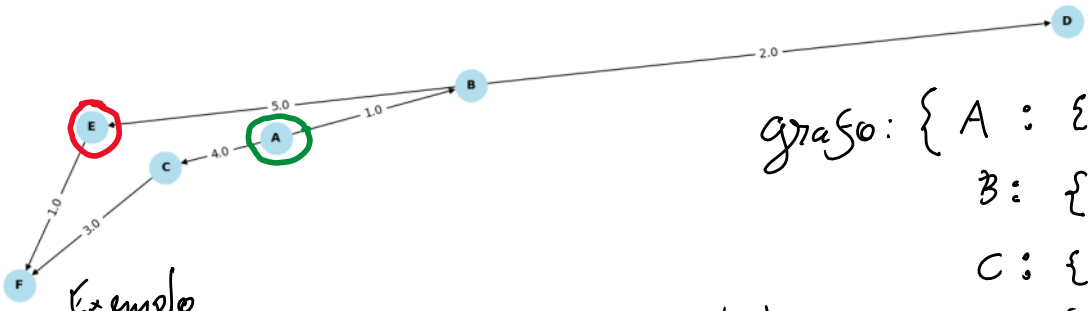
Dijkstra

domingo, 19 de maio de 2024 18:44

Dijkstra Algorithm



grafo: { A : { B: 1, C: 4 }
B: { D: 2, E: 5 }
C: { F: 3 }
D: { }
E: { F: 1 }
F: { }
}

Exemplo

definimos o vértice inicial : 'A'

" " " final : 'E'

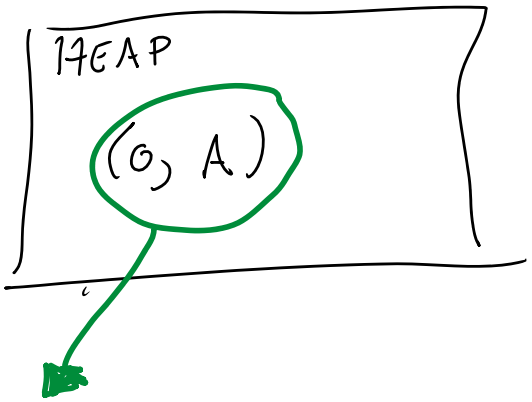
criamos a matriz de alcançabilidade

dijkstra = { A: 0, B: ∞, C: ∞, D: ∞, E: ∞, F: ∞ }

predecessores = { A: NULL, B: NULL, C: NULL, D: NULL, E: NULL, F: NULL }

fila = [(0, vertice_inicial)]

heapq.heapify(fila)



1ª Iteração

While fila:

peso, vértice = heapq.heappop(fila)

peso = 0

vertice = A

1ª iteração dealcance

for vizinho, peso_v in grafo[vertice].items():

distancia = peso + peso_v

vizinho = B

peso_v = 1

distancia = 1

, v: ∞

distancia = 1

1 < B: ∞

if distancia < dijkstra[vizinha]:

dijkstra[vizinho] = distancia {B: 1}

predecessores[vizinho] = vertice {B: A}

heapq.heappush(fila, (distancia, vizinho))

2ª iteração do alcance

vizinho = C e peso_v = 4

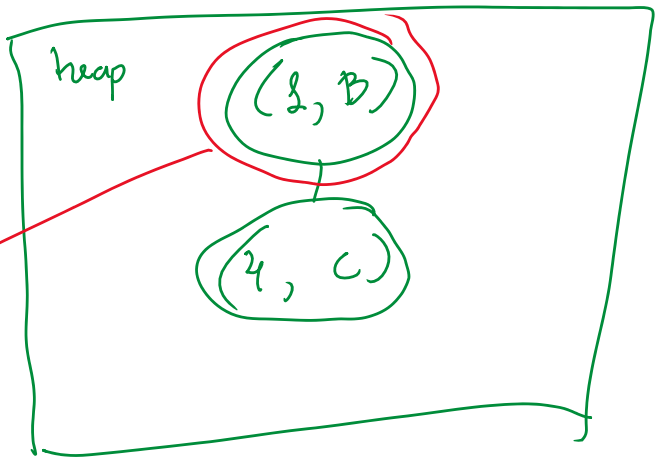
distancia = 4

4 < C: ∞

dijkstra[C] = 4

predecessores[C] = A

heapq.heappush(fila, (4, C))



fila =

2ª Iteração

heapq.heappop(fila)

peso = 1

vertice = B

B alcança D: 2 e E: 5

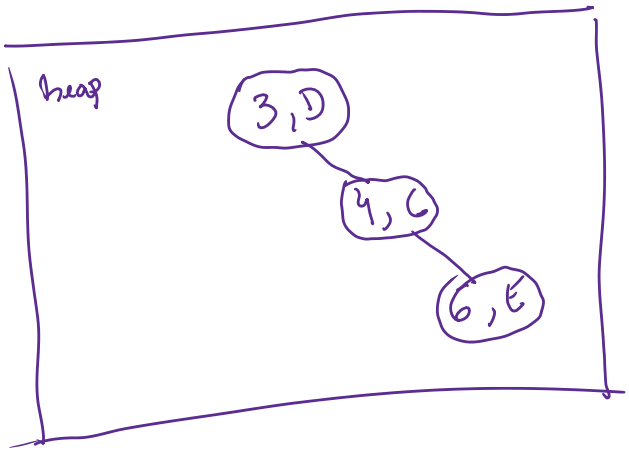
B alcança [D:2] e [E:5]

dijkstra = {A:0, B:3, C:4, D:3, E:6, F:∞}

predecessores = {A:NULL, B:A, C:A, D:B, E:B, F:NULL}

heappush(fila, (3, D)) e heappush(fila, (6, E))

fila:



pop: raiz

raiz sempre é o menor valor

3ª Iteração

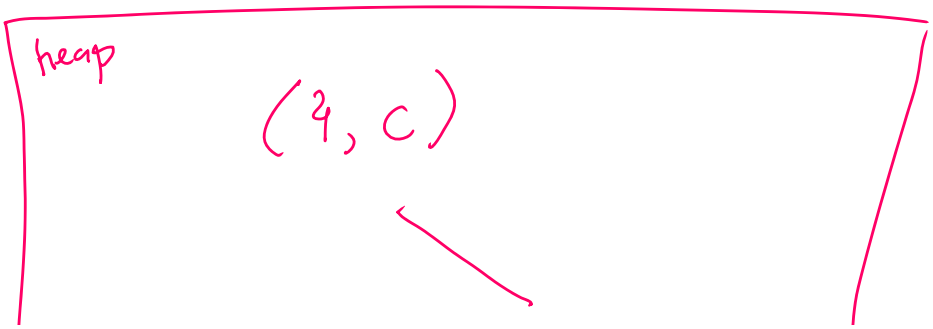
heappop(fila) = (3, D)

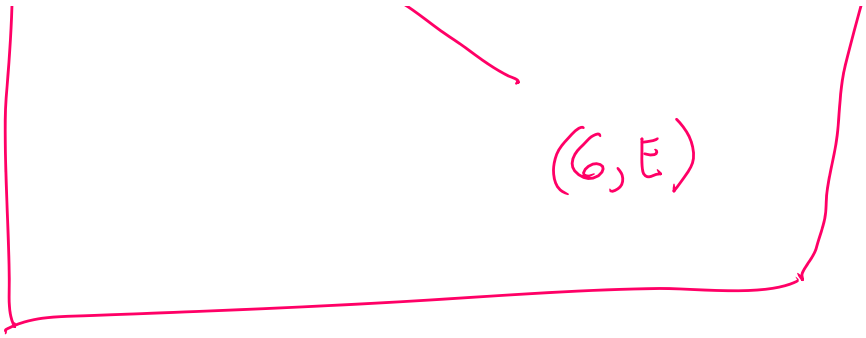
Peso = 3

vertice = D

D não tem vizinhos, então o laço FOR não executa

fila =





Ya Stras

heappop(fila) = (4, C)

peso = 4

vertice = C

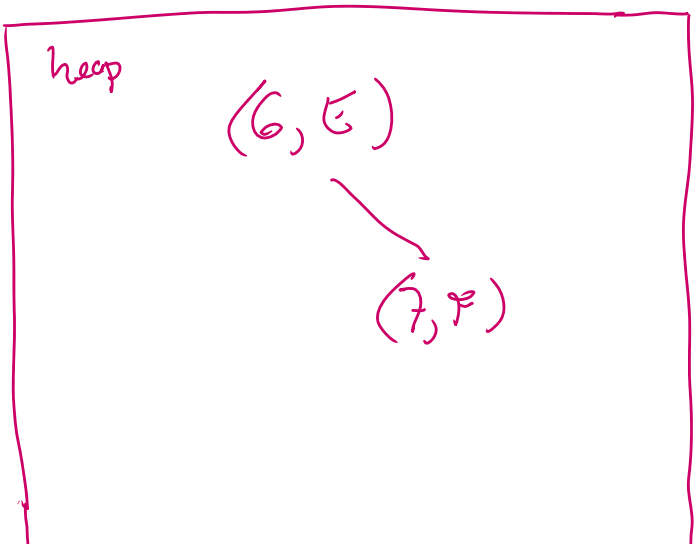
C alcança F: 3

dijkstra = {A: 0, B: 1, C: 4, D: 3, E: 6, F: 7}

predecessores = {A: NULL, B: A, C: A, D: B, E: B, F: C}

heappush (fila, (7, F))

fila:



5ª Iteração

heappush(fila)

peso = 6

vertice = E

E alcança [F: 1] distancia = 6 + 1 = 7

dijkstra = {A: 0, B: 1, C: 4, D: 3, E: 6, F: 7}

predecessores = {A: NULL, B: A, C: A, D: B, E: B, F: E}

heappush(fila, (7, F))

fila = heap

(7, F)

)

(7, F)

As próximas iterações não farão laço FOR

ou seja,

Mas próximas iterações não [illegible]
ou seja)

A Heap se esvazia pop até estar vazia
Encerrando o laço WHILE

Agora iremos construir o caminho

caminho = [ ]

predecessores = { A: NULL, B: A, C: A, D: B, E: B, F: E }

vertice_final = 'E'

while vertice_final :
    caminho . insert(0, vertice_final)
    vertice_final = predecessores [vertice_final]

return caminho