1) Calcule a complexidade de tempo do algoritmo de *Dijkstra*, que retorna o custo do caminho mínimo entre um vértice inicial *s* e qualquer outro vértice de um grafo. Sendo *G.Adj* a lista de adjacência que representa as arestas do grafo, *w* o custo associado as arestas e *Q* uma lista de vértices. Considere dois cenários:

   a) *Q* é uma lista encadeada.

   b) *Q* é uma lista de prioridade, implementada como um *heap* binário.

   Explique a complexidade das linhas em vermelho e dos laços **for** e **while**.

```
RELAX (u, v, w)
  if v.d > u.d + w (u, v)
    v.d ← u.d + w (u, v)
    u.pi ← u
```

```
DIJKSTRA (G, w, s)
  for each vertex v ∈ G.V
    v.d = ∞
    v.pi = NIL
  s.d = 0
  S ← ∅
  Q ← G.V
  while Q ≠ ∅
    u ← EXTRACT_MIN(Q)
    S ← S ∪ {u}
    for each vertex v ∈ G.Adj[u]
      RELAX(u, v, w)
```

2) Mostre cada iteração do algoritmo da questão anterior considerando o grafo abaixo, considere o vértice ***a*** como o vértice inicial: