Busca em Largura

1. Para cada $v \in V(G)$ faça: $d[v] = \infty$
2. $d[s], pai[s], S = 0, s, [s]$ ## S é uma fila
3. Enquanto $S \neq \emptyset$ faça:
   - $v = \text{desenfileire}(S)$
   - Para cada $w \in \text{adj}[v]$ faça:
   - se $d[w] = \infty$ então: ## ainda não visitado
   - $d[w], pai[w] = d[v] + 1, v$
   - $\text{enfileire}(w, S)$

Algoritmo de Dijkstra

1. Para cada $v \in V(G)$ faça: $d[v] = \infty$
2. $d[s], pai[s], S = 0, s, V(G)$
3. Enquanto $S \neq \emptyset$ faça:
   - Escolha $v \in S$ tal que $d[v]$ seja mínimo
   - $S = S - v$
   - Para cada $w \in \text{adj}[v]$ faça:
   - se $d[w] > d[v] + c(v, w)$ então:
   - $d[w], pai[w] = d[v] + c(v, w), v$

Algoritmo de Dijkstra

- Qual é o tempo gasto por este algoritmo?
- Depende da implementação do algoritmo de seleção do vértice $v \in S$
- Implementação simples, em que se faz uma passada no vetor, procurando vértices marcados como pertencendo a $S \Rightarrow O(n^2)$
- Implementação utilizando fila de prioridade $\Rightarrow O(m \log n)$
- Nossa escolha da implementação depende
- Para grafos com menos de 3.000 vértices: $O(n^2)$
- Para grafos grandes em que $m = \Theta(n)$, vale a pena usar fila de prioridade

Algoritmo de Floyd-Warshall

- produz tabela completa, para todos os pares de vértices
- $O(n^3)$
- convenção: $c(v, w) = \infty$ para pares (v, w) não adjacentes
- para todo vértice v faça:
-     para todo vértice w faça:
-         se $v = w$ então: $d(v, v) = 0$
-         senão: $d(v, w) = c(v, w)$
- para cada vértice u faça:
-     para cada vértice v faça:
-         para cada vértice w faça:
-             $d(v, w) = \min(d(v, w), d(v, u) + d(u, w))$

Algoritmo de Bellman-Ford

- para cada vértice v faça: $d(v) = \infty$
- $d(s), pai(s) = 0, s$
- para $i = 1$ até $n - 1$ faça:
- para cada aresta (v, w) faça:
- se $d(w) > d(v) + c(v, w)$ então:
- $d(w), pai(w) = d(v) + c(v, w), v$
- para cada aresta (v, w) faça:
- se $d(w) > d(v) + c(v, w)$: retorne Falso
- retorne d