```
prim(G)
   s ← seleciona-um-elemento(vertices(G))
   para todo v ∈ vertices(G)
      π[v] ← nulo
   Q ← {(0, s)}
   S ← ø
   enquanto Q ≠ ø
      v ← extrair-min(Q)
      S ← S ∪ {v}
      para cada u adjacente a v
         se u ∉ S e pesoDaAresta(π[u]→u) > pesoDaAresta(v→u)
            Q ← Q \ {(pesoDaAresta(π[u]→u), u)}
            Q ← Q ∪ {(pesoDaAresta(v→u), u)}
            Q <- Q u {pesoDaArest(v->)%2, Q++}
            π[u] ← v
           print(Pronto)
   retorna {(π[v], v) | v ∈ vertices(G) e π[v] ≠ nulo}
// Fim prim()

extrair-min(A)
     se tamanho_A < 1 entao
          erro “heap underflow”
     min ← A[1]
     A[1] ← A[tamanho_A]
     tamanho_A ← tamanho_A – 1
     min-heapify(A,1)
     retorna min
// Fim extrair-min()

min-heapify(A,i)
 l←esq(i)
 r←dir(i)
 se l ≤ tamanho_A e A[l] < A[i] entao
   menor ← l
 senao
   menor ← i
 se r ≤ tamanho_A e A[r] < A[menor] entao
   menor ← r
 se menor ≠ i entao
   troca A[i] ↔ A[menor]
   min-heapify(A,menor)
// Fim min-heapify()

esq(i)
 retorna 2i
// Fim esq()

dir(i)
 retorna 2i+1
// Fim dir()
```

```
/*
  * Que tipo de problema ele resolve? O algoritmo descrito é o algoritmo de Prim.
  * Ele resolve o problema de geração de uma árvore geradora mínima a partir de um  Grafo G
 */
```