Nombre: Kevin Huertas

Implementación:

```python
class Grafo:
    def __init__(self, vertices):
        self.V = vertices
        self.adj = [[] for i in range(vertices)]

    def agregar_arista(self, u, v, w):
        self.adj[u].append((v, w))
        self.adj[v].append((u, w))

    def prim_arbol_exp_minima(self):
        clave = [float('inf')] * self.V
        padre = [-1] * self.V
        mst_set = [False] * self.V

        clave[0] = 0
        padre[0] = -1

        for i in range(self.V - 1):
            min_clave = float('inf')
            min_idx = -1

            for v in range(self.V):
                if not mst_set[v] and clave[v] < min_clave:
                    min_clave = clave[v]
                    min_idx = v

            mst_set[min_idx] = True

            for adj_v, adj_w in self.adj[min_idx]:
                if not mst_set[adj_v] and adj_w < clave[adj_v]:
                    clave[adj_v] = adj_w
                    padre[adj_v] = min_idx

        return [(padre[i], i) for i in range(1, self.V)]
```

Ejemplo de uso:

```python
from spanningTree import Grafo

grafo = Grafo(4)

# ponemos aristas con sus pesos
grafo.agregar_arista(0, 1, 2)
grafo.agregar_arista(0, 3, 6)
grafo.agregar_arista(1, 2, 3)
grafo.agregar_arista(1, 3, 8)
grafo.agregar_arista(2, 3, 4)

arbol = grafo.prim_arbol_exp_minima()

# Imprimimos
print("Las aristas son:")
for u, v in arbol:
    print(u, v)
```

Resultado:

```
Las aristas son:
(0, 1)
(1, 2)
(2, 3)
```