Pseudocódigo Problema Trabajadores

Función para explorar nodos vecinos

```
void explorarVecinos(vector<vector<int>> tabla, vector<bool> trabajoAsignado,
                     vector<int> stack, int costoActual, vector<int> orden,
                     int nivel, int mejorCosto, vector<int> mejorAsignacion){

  //Si ya asignamos todos los trabajadores, ver si es la mejor solución
  if(nivel == tabla.size()){
    if (costoActual < mejorCosto){
      mejorCosto = costoActual
      mejorAsignacion = stack
    }
    return
  }

  //Seleccionar trabajador actual
  int trabajadorActual = orden[nivel]

  //Explorar todos los trabajos posibles
  for (int i=0; i<tabla.size(); i++){

    //Si el trabajo no ha sido asignado aún
    if (!trabajoAsignado[i]){

      //Costo de asignar este trabajo
      int nuevoCosto = costoActual + tabla[trabajadorActual][i];

      //Si el costo aún puede mejorar, explorar
      if (nuevoCosto < mejorCosto){
        trabajoAsignado[i] = true    //Asignar trabajo
        stack[trabajadorActual] = i  //Guardar asignación

        //Llamar recursivamente al siguiente nivel
        explorarVecinos(tabla, trabajoAsignado, stack, nuevoCosto,
                        orden, nivel+1, mejorCosto, mejorAsignacion)

        //Backtracking
        trabajoAsignado[i] = false
        stack[trabajadorActual] = -1
      }
    }
  }
}
```

Función ProblemaTrabajadores

```
void problemaTrabajadores(vector<vector<int>>tabla, vector<int>orden){
  int n= tabla.size()
  vector <bool>trabajoAsignado(n, false) //Trabajo ya asignado
  vector <int> mejorAsignacion(n,-1)   // Guardar mejor solución
  vector<int> stack(n,-1)  // Guarda el trabajo para cada trabajador
  int mejorCosto = ∞          // Costo mínimo

  // Llamada inicial
  explorarVecinos(tabla, trabajoAsignado, stack, 0, orden, 0,
                  mejorCosto, mejorAsignacion)

  //Imprimir resultados para cada uno de los trabajadores
   for (int i =0 ; i <n; i++){
      print i , mejorAsignacion[i], tabla[i][mejorAsignacion[i]]
   }

  print mejorCosto
}
```