Estructuras Ciudad Robotica

- ***robotsXrur SRC dicc(RUR r, datosR)***
  - ***A. SRC sobre vector(datos)***

- ***datosR SRC Tupla(restriccionesXsenda , Estacion e_{actual} , iterator(a) itA, conj(TAG) ts, Boolean habilitado?, Nat cantInf)***
  - ***A. ts SRC sobre vector(TAG)***
  - ***B. habilitado? indica si el RUR se encuentra en circulacion.***
  - ***C. itA es iterador a la posicion del RUR en el heap a.***
  - ***D. cantInf es la cantidad de infracciones del RUR.***
  - ***E. restriccionesXsenda SRC dicc(Estacion e_1 (dicc(Estacion e_2 , Boolean v))***
  - ***F. Donde v indica si comete infraccion al pasar por la senda que conecta e_1 con e_2***
  - ***G. e_{actual} es la estacion en la que se encuentra el robot.***

- ***robotsXestacion SRC dicc(Estacion e_1 , datosE)***
  - ***A. SRC sobre trie(Estacion, datosE))***
  - ***B. datosE SRC Tupla(heap(RUR) a, padreDelUltimo(a))***
  - ***C. Donde a esta ordenado por el mas infractor en la raiz de cada arbol.***

- ***mapa SRC Tupla(dicc(Estacion e_1 , dicc(Estacion e_2 , senda)) sendas, conj(Estacion) estaciones)***
  - ***A. sendas SRC trie(Estacion(trie(Estacion, senda))***
  - ***B. estaciones SRC vector(estaciones)***

- ***senda SRC restriccion***

- ***restriccion SRC ? dificil eh! jaja***

- ***TAG es String ($|TAG| \leq 64$)***

- ***Estacion es String (no acotado)***

- ***RUR es Nat (se asignan secuencialmente, RUR $\in N_0$)***

- ***robots(c) es O(1) y debe devolver un iterador. Siguiente del iterador puede no ser O(1).***
- ***estaciones(c) es O(1) y debe devolver un iterador.***
- ***estación(u, c), #infracciones(u, c) y tags(u, c) son todas O(1).***
- ***entrar(ts, e, c) es O($|e_m| \cdot E + |e_m| \cdot S \cdot R + N_{total}$), donde |em| es la longitud más larga entre todas las estaciones.***

La complejidad real es: O($|e_m| \cdot E + |e_m| \cdot S + |e_m| \cdot E + |e_m| \cdot S \cdot R \cdot 64 + N_{total}$ ***)***

➔ $|e_m| \cdot E + |e_m| \cdot S$ ***resulta de copiar el trie que tiene la estructura de las sendas. Significa recorrer para todas las estaciones (E) suponiendo el peor caso de longitud en todas ellas. Luego para todas las estaciones con las que está conectada esa estacion, que en el peor caso son todas las sendas posibles (S), y todas las estaciones tambien el peor caso en longitud.***

➔ $|e_m| \cdot E + |e_m| \cdot S \cdot R \cdot 64$ ***resulta de recorrer el trie que está en el mapa para calcular las restricciones. 64 es buscar un TAG de la restriccion dentro del trie de caracteristicas del robot. Una caracteristica es acotada luego su complejidad es constante.***

Esto se puede acotar de tal manera que:

$O(|e_m| \cdot E + |e_m| \cdot S + |e_m| \cdot E + |e_m| \cdot S \cdot R \cdot 64 + N_{total}) \in$

$O(2 \cdot |e_m| \cdot E + |e_m| \cdot S \cdot (R \cdot 64 + 1) + N_{total}) \in$

$O(|e_m| \cdot E + |e_m| \cdot S \cdot (R \cdot 64) + N_{total}) \in$

$O(|e_m| \cdot E + |e_m| \cdot S \cdot R + N_{total})$

Que es la complejidad buscada.

- ***mover(u,*** e_2***, c) es O(***$|e_1| + |e_2| + \log N e_1 + \log N e_2$***), donde e1 es la estación donde se encuentra el robot antes de moverse, y e2 es la estación a la que se mueve.***
  Con lo cual eliminarlo es buscarlo en a ($|e_1|$***), eliminarlo (***$\log N e_1$***) y agregarlo en*** e_2 ***(***$|e_2|$***). Hay que tener en cuenta que puedo obtener de a en O(1) dado que tengo itA en los datos del robot que me provee del mismo en esa complejidad. Luego aplicar subir (***$\log N e_2$***). Antes de ingresarlo se le suma si cometio infraccion o no.***
  Buscar si comete o no infraccion requiere usar e_{actual} ***y*** e_2 ***para encontrar en infraccionesXsenda si comete infraccion o no y sumarlo a los datos del robot dado.***
  Esta operacion cuesta O($|e_{actual}| + |e_2|$***).***
  Ademas copiar e_2 ***como la nueva estacion actual en los datos del robot cuesta*** $|e_2|$
  y eliminar e_{actual} ***cuesta*** $|e_{actual}|$ ***(?)***
  Luego la compljidad real es $O(2 \cdot (|e_1| + |e_2|) + (|e_{actual}| + |e_2|) + \log N e_1 + \log N e_2)$
  Luego $e_1 = e_{actual}$ ***con lo cual***
  $O(3 \cdot (|e_1| + |e_2|) + \log N e_1 + \log N e_2) \in$
  $O((|e_1| + |e_2|) + \log N e_1 + \log N e_2)$
  Que es la complejidad buscada.

- ***inspección(e, c) es O(|e| + log Ne).***
  desencolar de a el mas infractor que es O(log Ne) dado que se aplica subir(a) en desencolar
  Para esto primero hay que buscarlo en robotsXestacion (O(|e|)).