Algoritmos y Estructuras de Datos II

Ejercicios voraces

9 de junio de 2014

Problema de cargar combustible 1

- Se quiere hacer un recorrido con un vehículo de autonomía A.
- El vehículo se encuentra en la ciudad 1 con el tanque vacío.
- Se quiere llegar a la ciudad n pasando por las ciudades 2, 3, . . . , $n - 1$.
- Hay exactamente una estación de servicio en cada una de esas ciudades.
- Entre la ciudad i y la ciudad $i + 1$ la distancia d_i es menor o igual a la autonomía A. ¿Por qué este dato?
- Se quiere cargar combustible el menor número de veces posible.
- ¿Hay algún buen criterio para saber dónde y cuánto cargar?

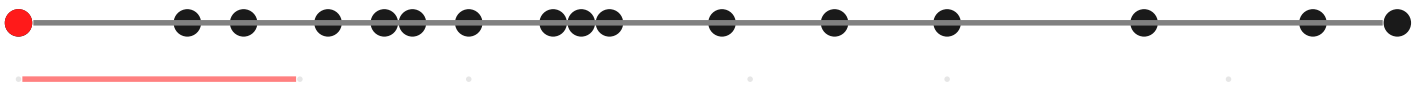
Problema de cargar combustible 2

- Se quiere hacer un recorrido con un vehículo de autonomía A.
- El vehículo se encuentra en la ciudad 1 con el tanque vacío.
- Se quiere llegar a la ciudad n pasando por las ciudades 2, 3, . . . , $n - 1$.
- Hay exactamente una estación de servicio en cada una de esas ciudades.
- Entre la ciudad i y la ciudad $i + 1$ la distancia d_i es menor o igual a la autonomía A.
- El precio de la combustible en la ciudad i es c_i.
- Se quiere llegar a la ciudad n minimizando el gasto en combustible.
- ¿Hay algún buen criterio para saber dónde y cuánto cargar?

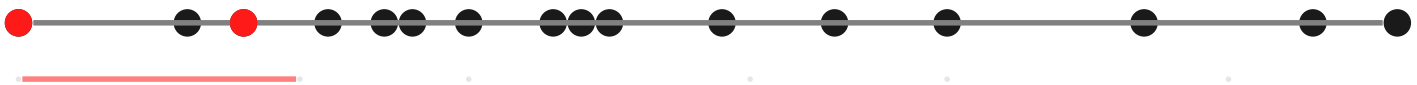
Problema del combustible

Problema del combustible

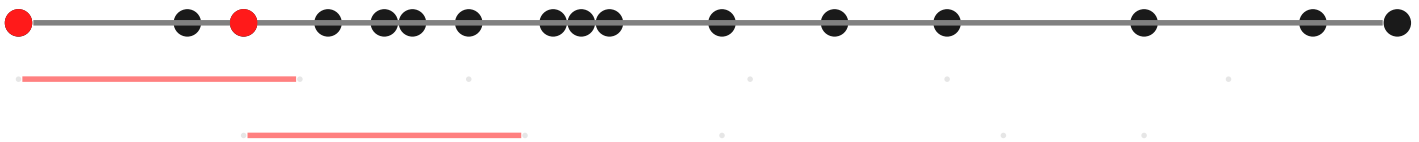
Problema del combustible



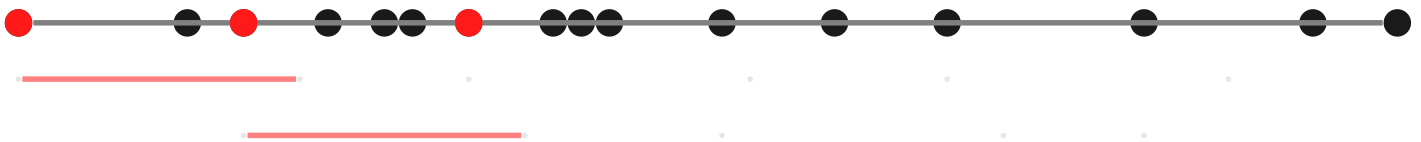
Problema del combustible



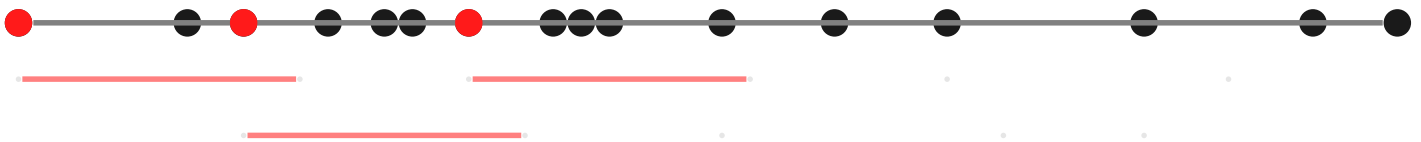
Problema del combustible



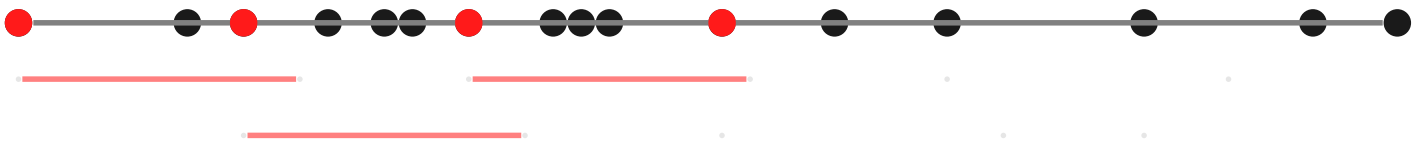
Problema del combustible



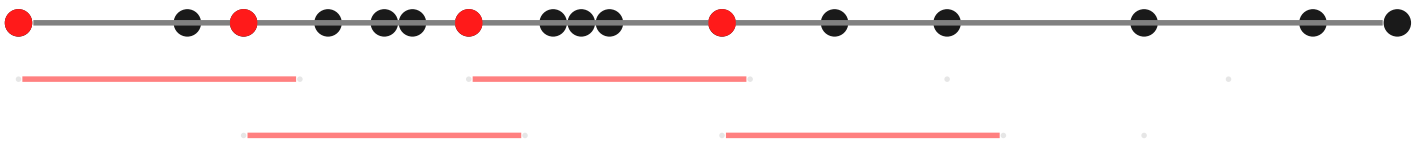
Problema del combustible



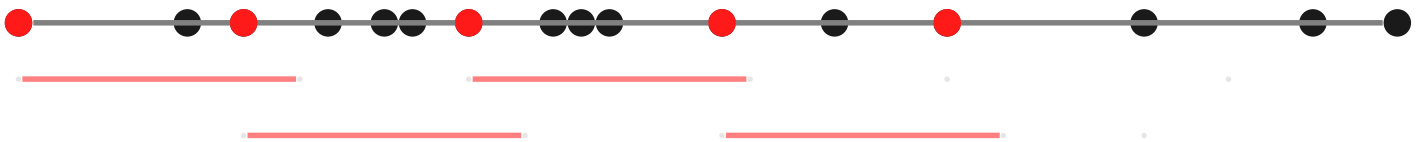
Problema del combustible



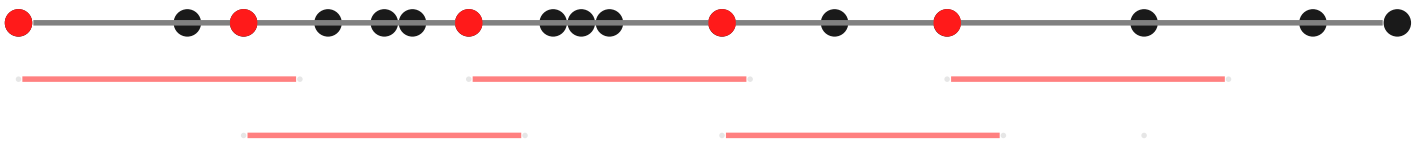
Problema del combustible



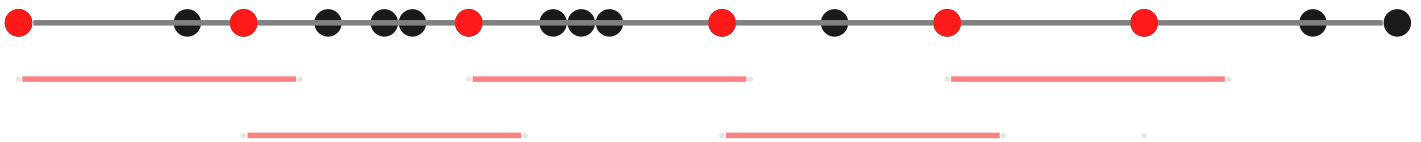
Problema del combustible



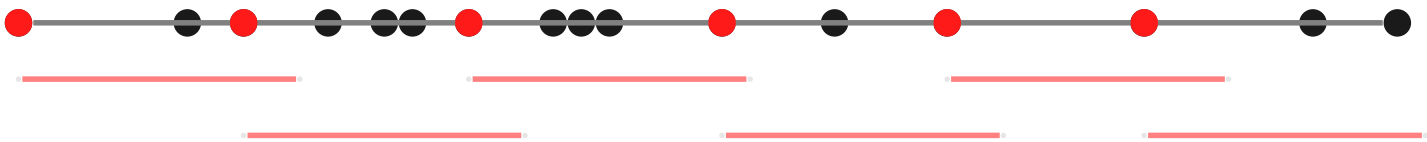
Problema del combustible



Problema del combustible



Problema del combustible



El algoritmo

```
fun combustible(d:array[1..n-1] of nat, A:nat) ret S:array[1..n] of nat
    var j,a : nat
    j:= 0
    a:= 0
    for c:= 1 to n-1 do
        if a < d[c] then
            j:= j+1
            S[j]:= c
            a:= A-d[c]
        else a:= a - d[c]
        fi
    od
end fun
```

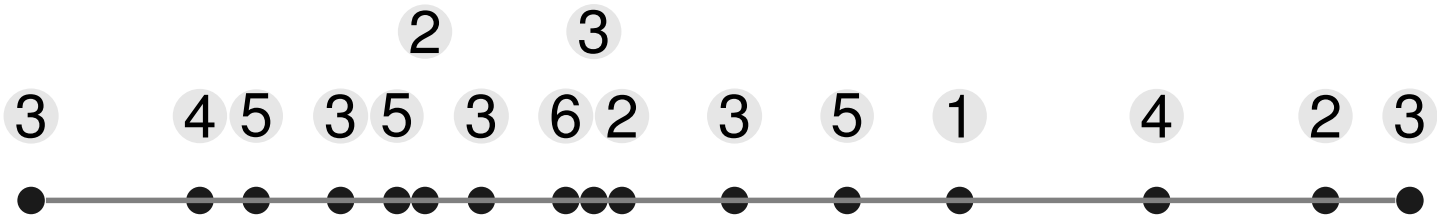
Explicación

- el algoritmo devuelve en S[1] la primer ciudad donde hay que cargar, en S[2] la segunda, etc.,
- el arreglo S se asume que se inicializa en 0,
- no todo el arreglo S se usará (a menos que haya que cargar en todas las ciudades), en el primer j tal que S[j] sea 0, se acaba la información de S,
- se usa el índice j para recorrer S y el índice c para recorrer las ciudades,
- se usa la variable a para llevar la cuenta de la distancia que puede recorrerse con la carga actual de combustible, a=A equivale a afirmar que el tanque esta lleno,
- siempre (si $n > 1$) se carga en la primera ciudad, pero nunca en la última.

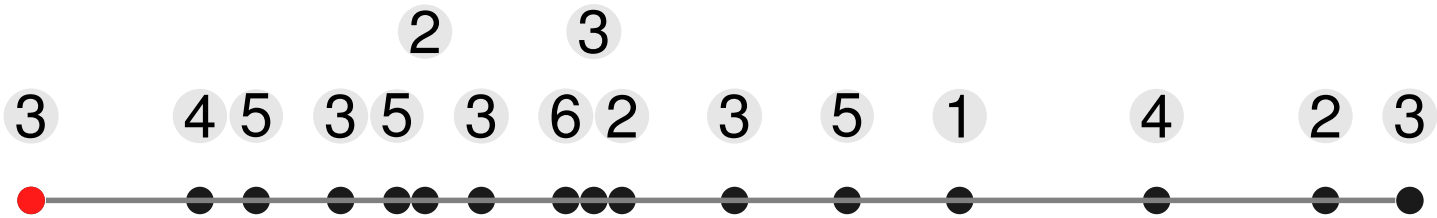
Problema de cargar combustible 2

- Se quiere hacer un recorrido con un vehículo de autonomía A.
- El vehículo se encuentra en la ciudad 1 con el tanque vacío.
- Se quiere llegar a la ciudad n pasando por las ciudades 2, 3, . . . , $n - 1$.
- Hay exactamente una estación de servicio en cada una de esas ciudades.
- Entre la ciudad i y la ciudad $i + 1$ la distancia es menor o igual a la autonomía A.
- El precio de la combustible en la ciudad i es c_i.
- Se quiere llegar a la ciudad n minimizando el gasto en combustible.
- ¿Hay algún buen criterio para saber dónde y cuánto cargar?

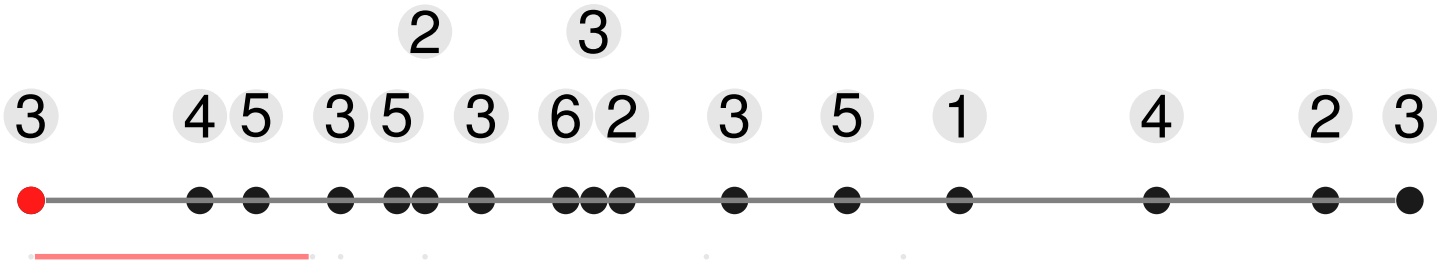
Problema del combustible 2: ejecución incompleta



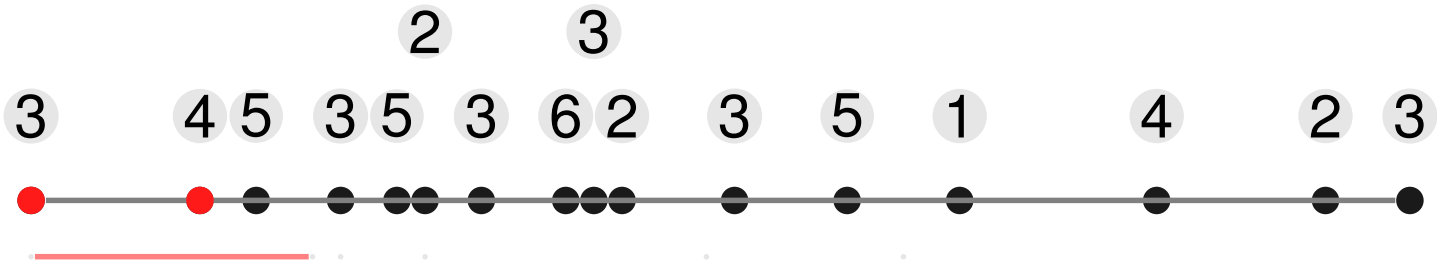
Problema del combustible 2: ejecución incompleta



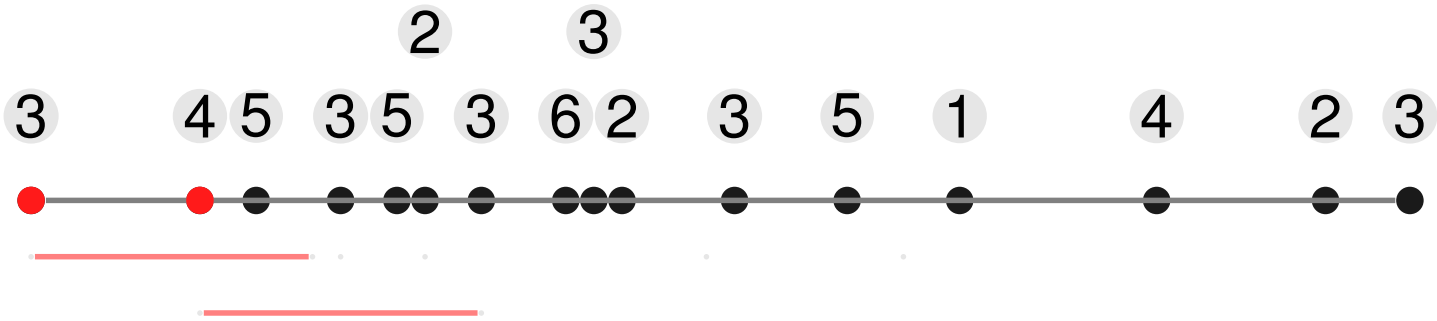
Problema del combustible 2: ejecución incompleta



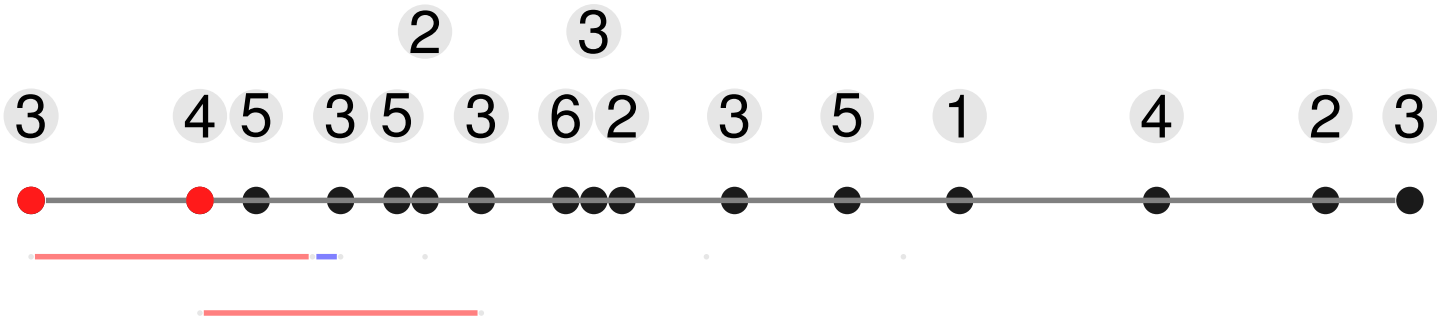
Problema del combustible 2: ejecución incompleta



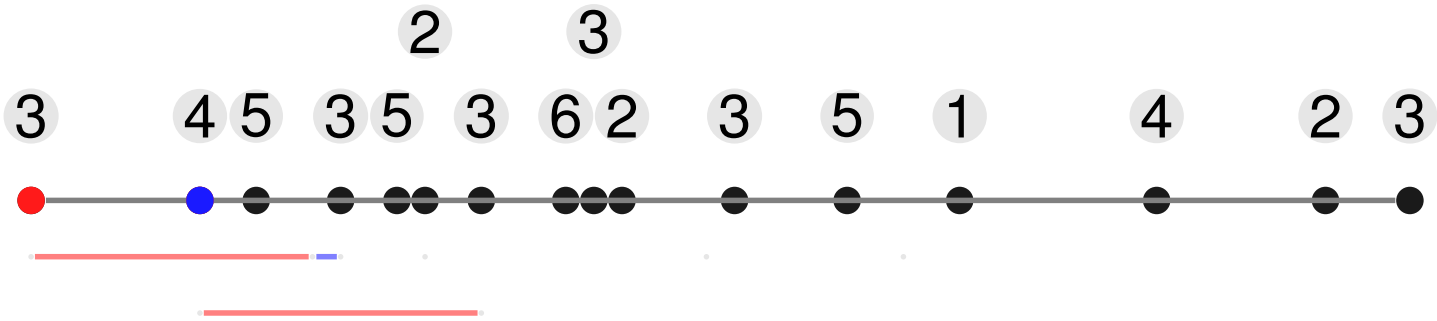
Problema del combustible 2: ejecución incompleta



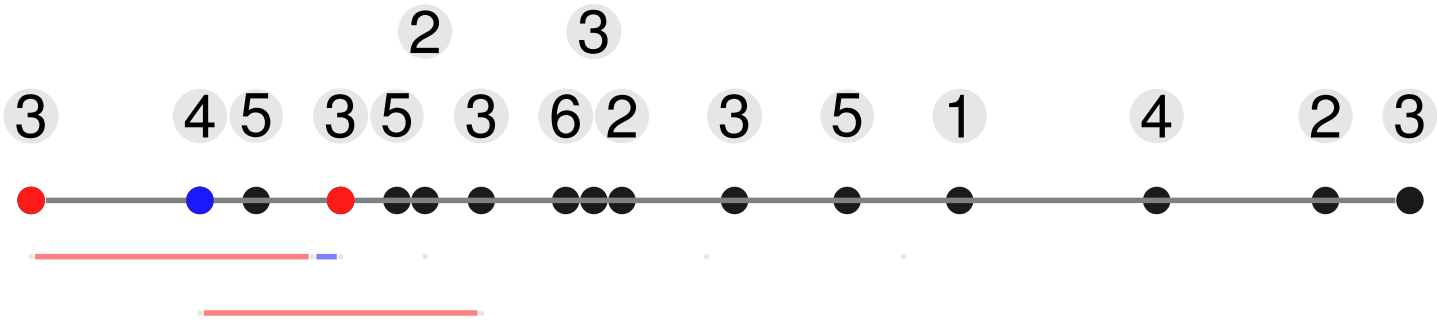
Problema del combustible 2: ejecución incompleta



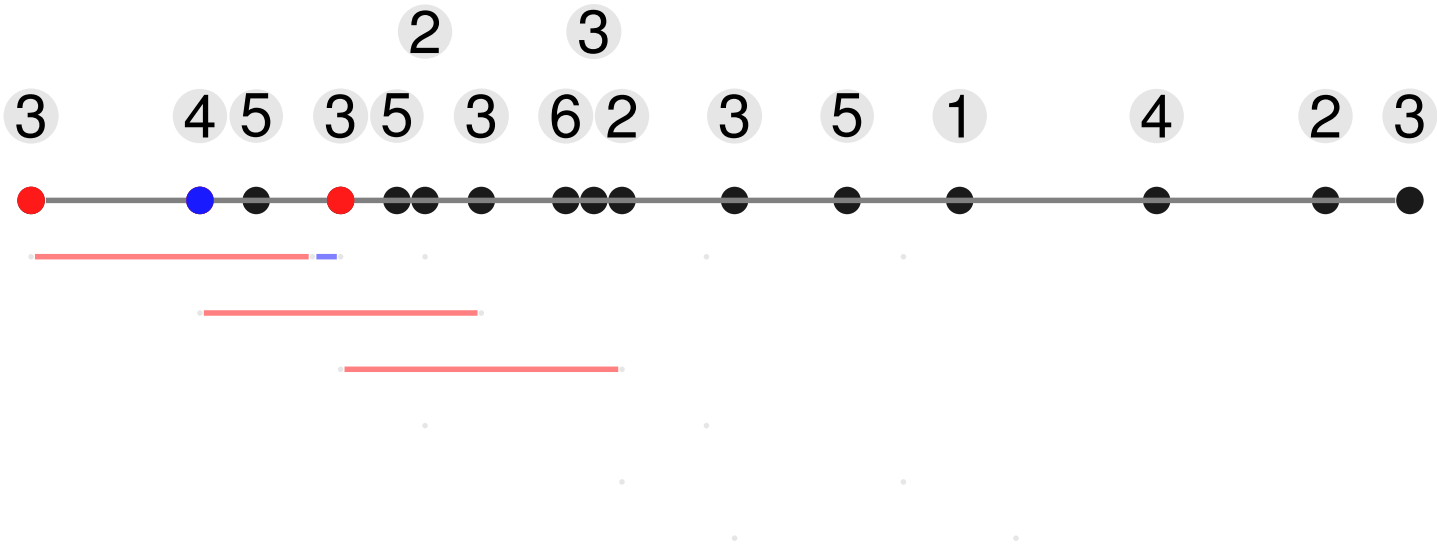
Problema del combustible 2: ejecución incompleta



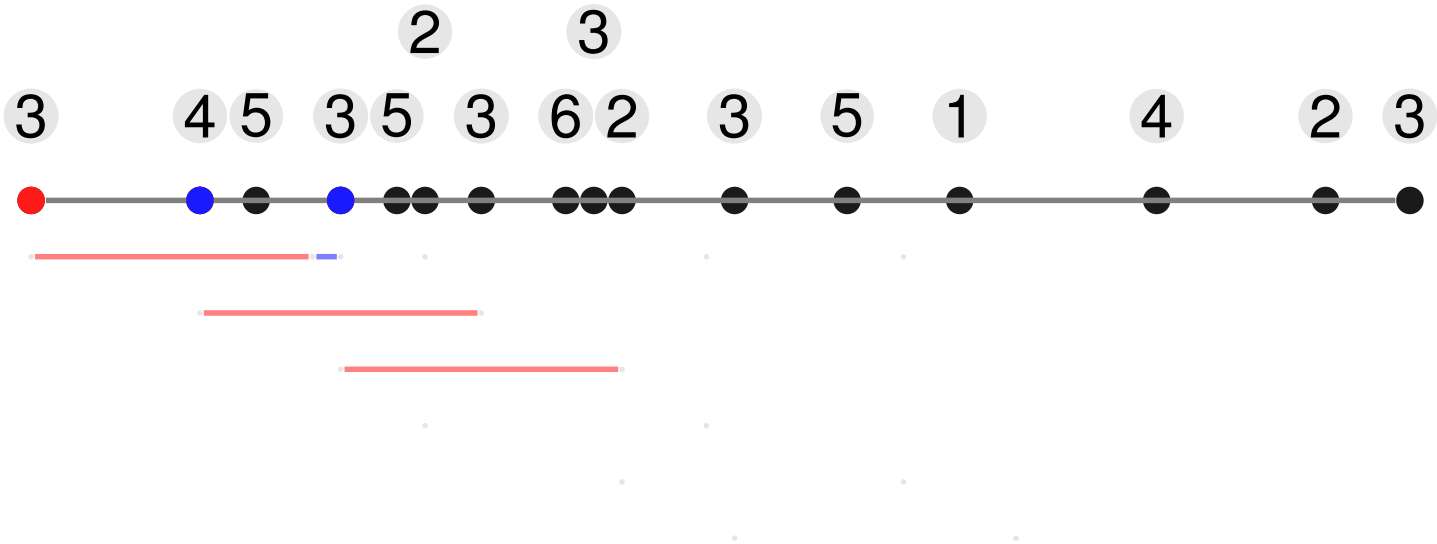
Problema del combustible 2: ejecución incompleta



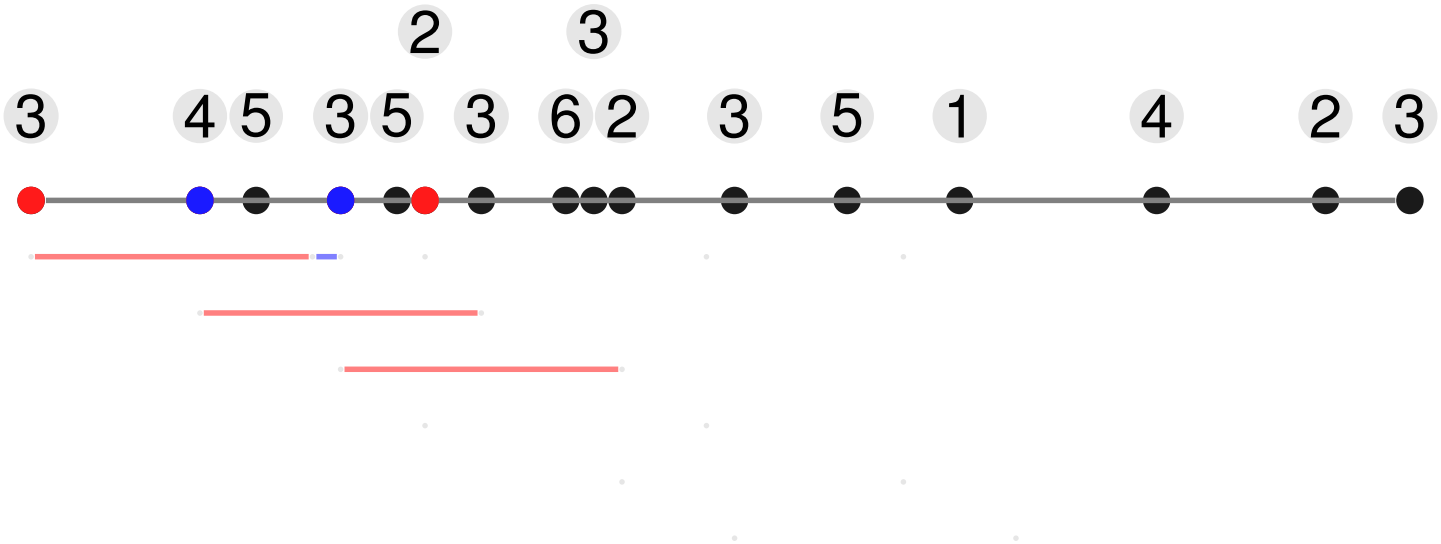
Problema del combustible 2: ejecución incompleta



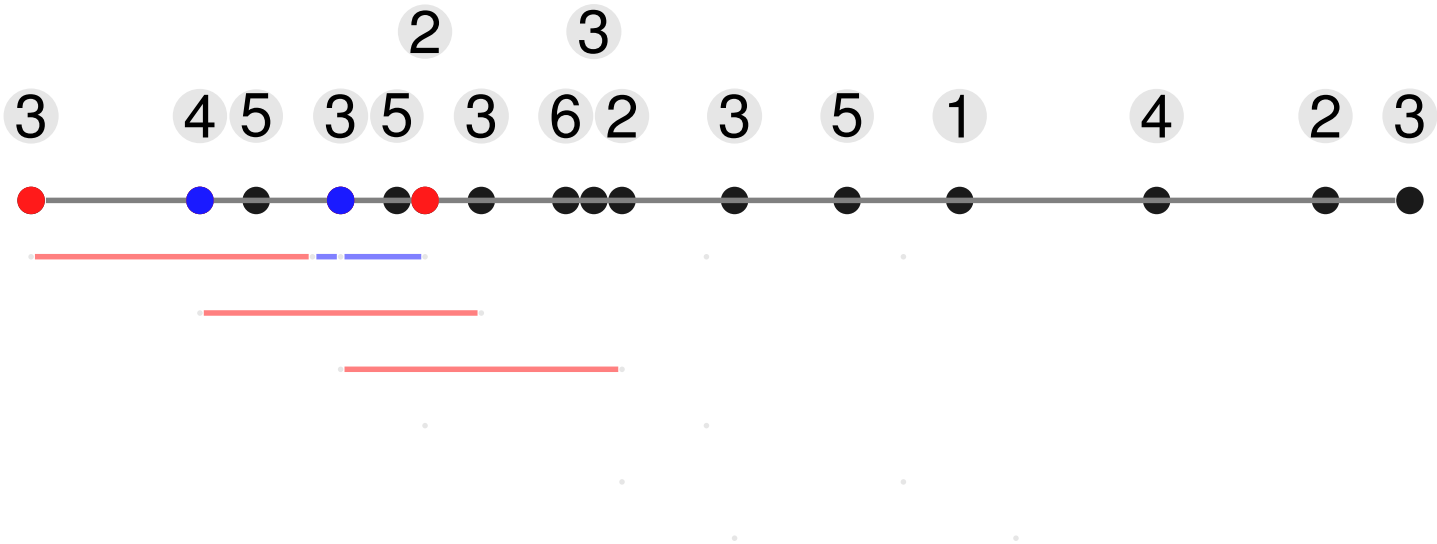
Problema del combustible 2: ejecución incompleta



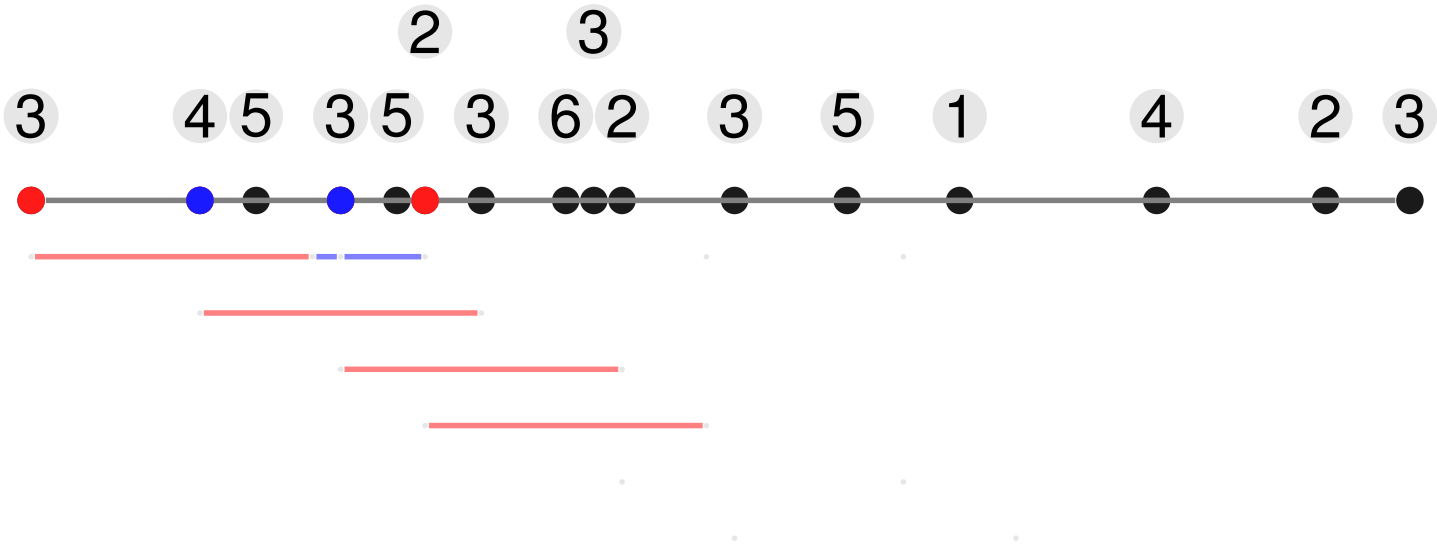
Problema del combustible 2: ejecución incompleta



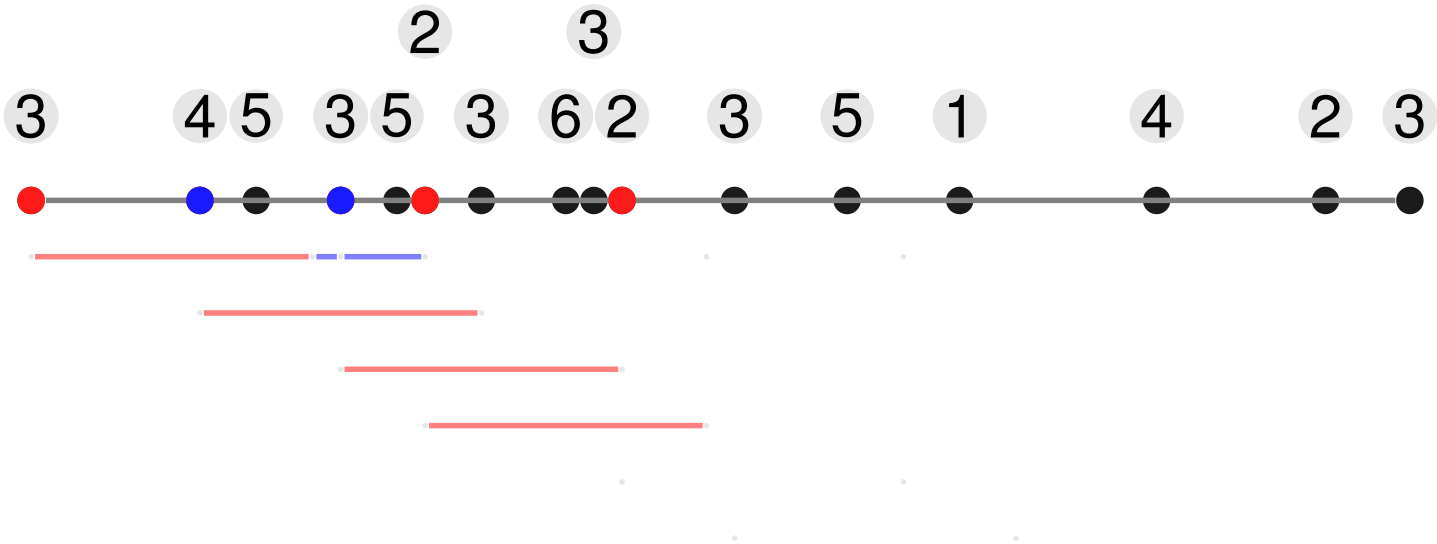
Problema del combustible 2: ejecución incompleta



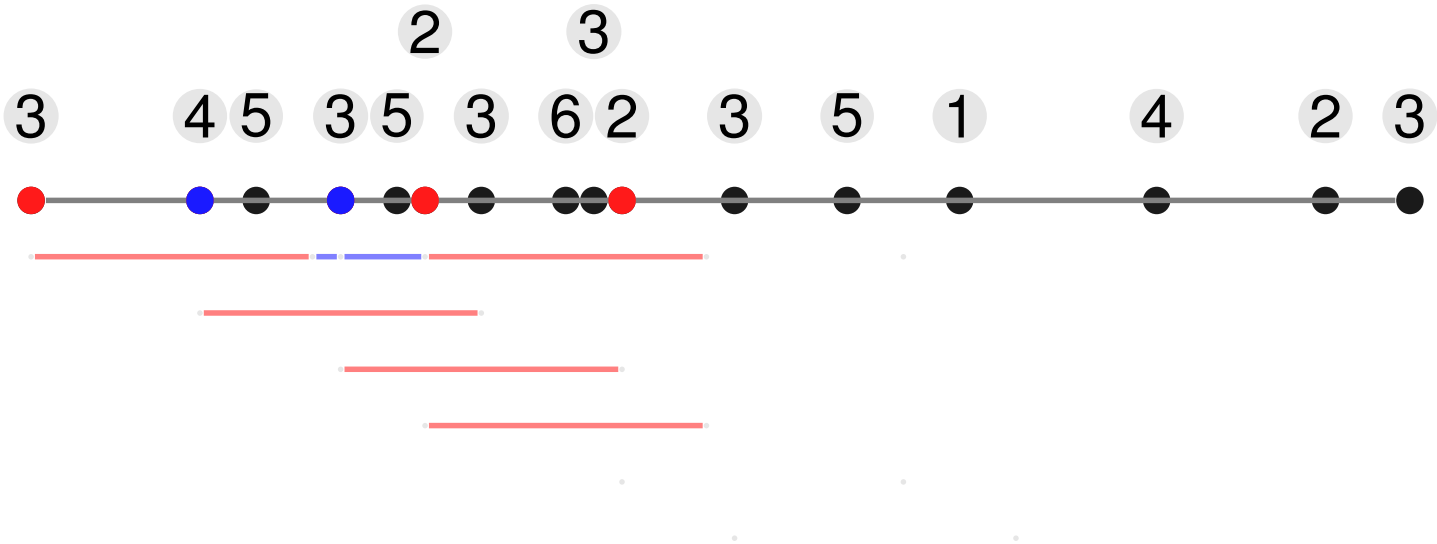
Problema del combustible 2: ejecución incompleta



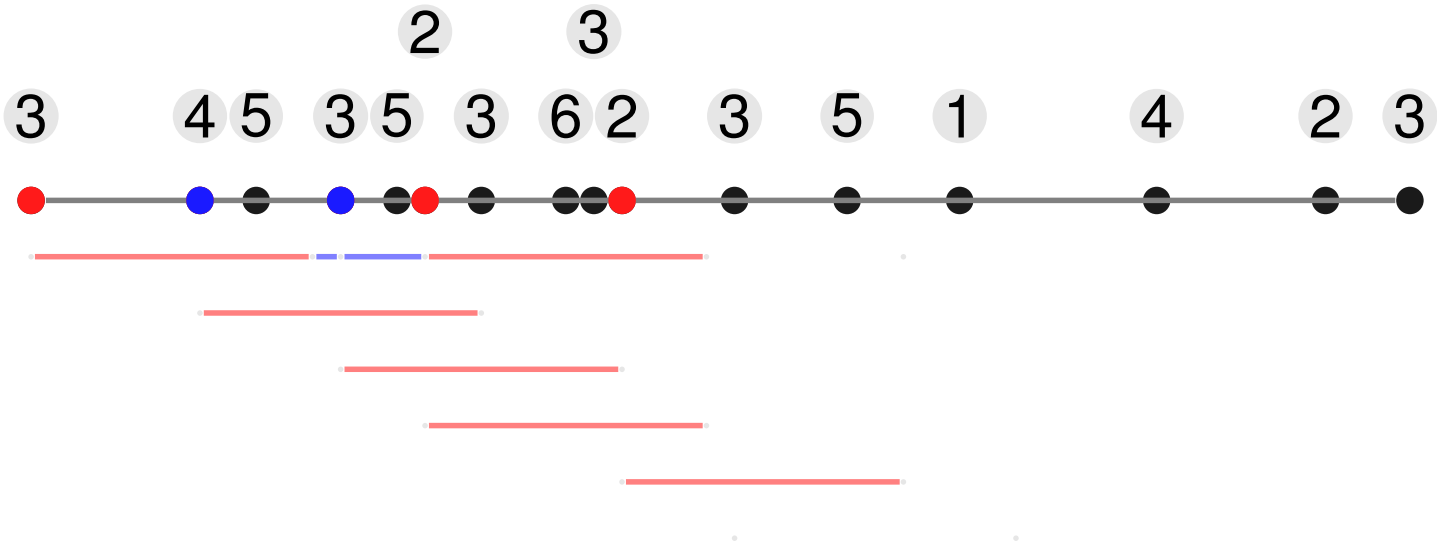
Problema del combustible 2: ejecución incompleta



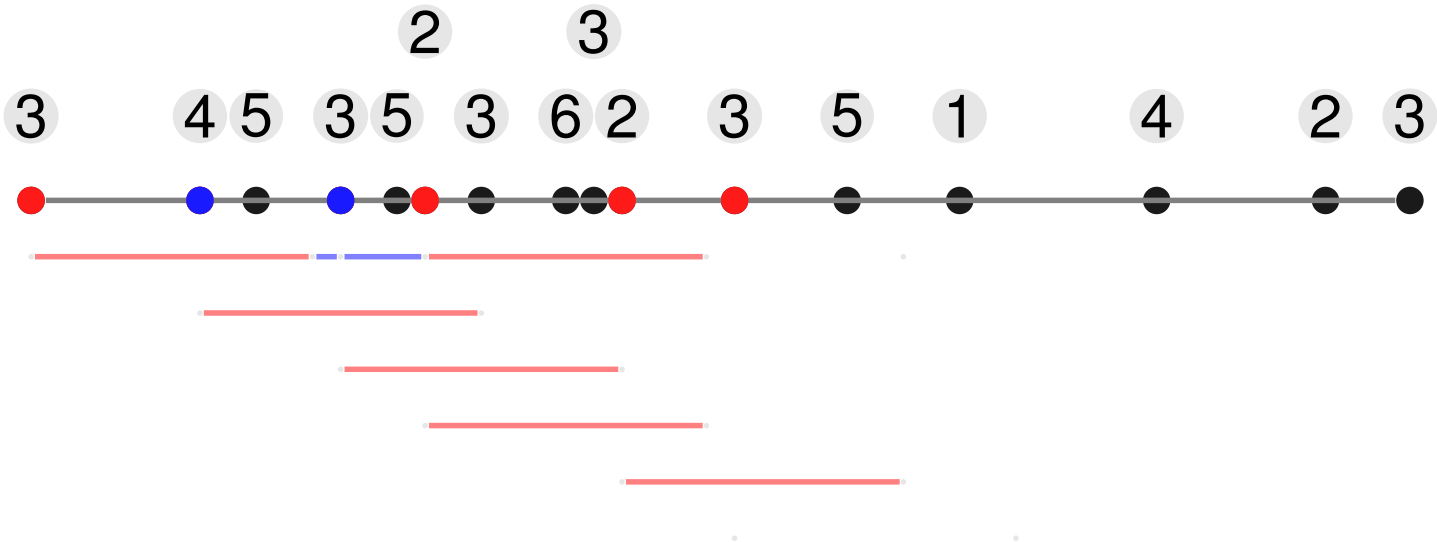
Problema del combustible 2: ejecución incompleta



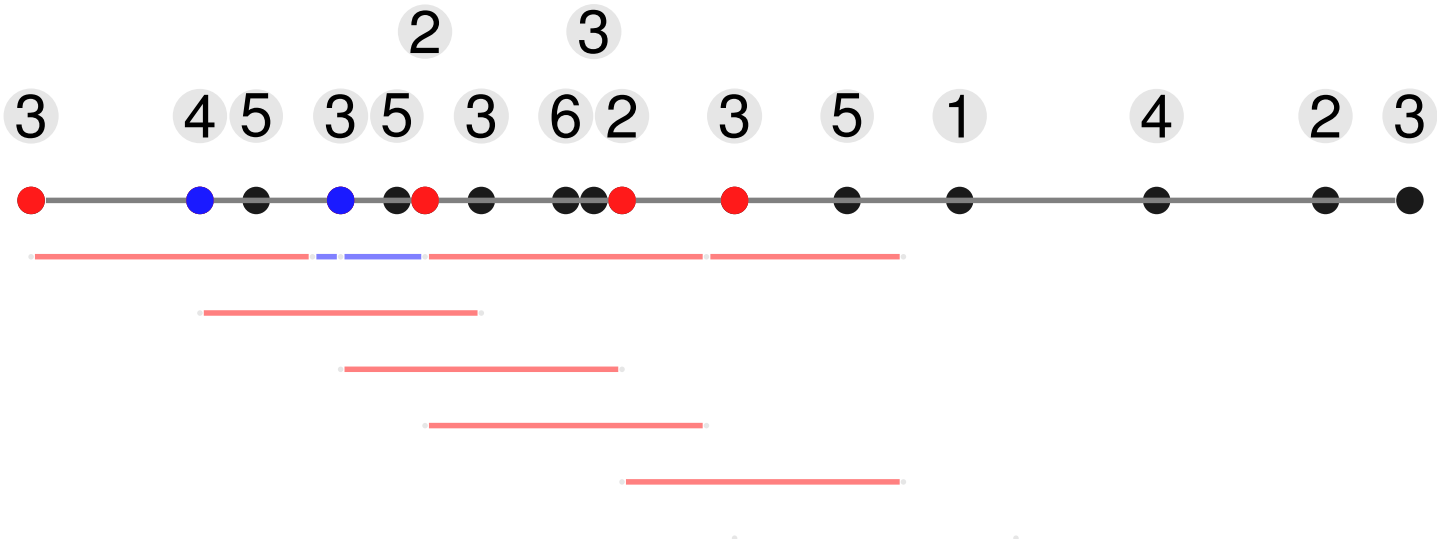
Problema del combustible 2: ejecución incompleta



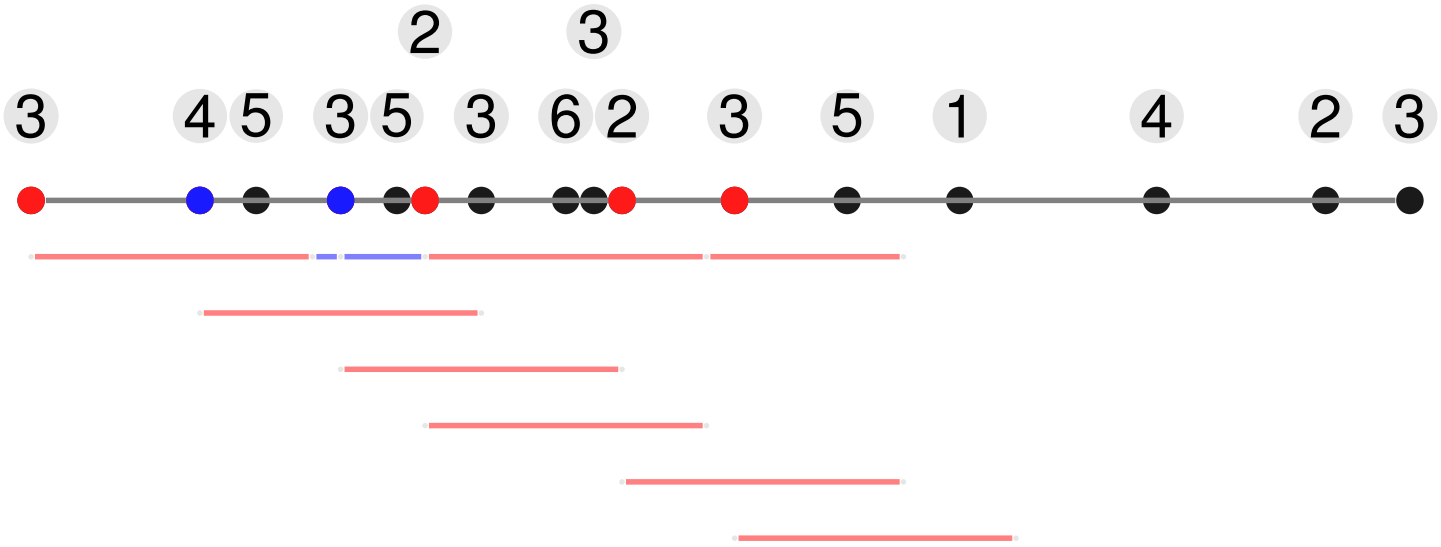
Problema del combustible 2: ejecución incompleta



Problema del combustible 2: ejecución incompleta



Problema del combustible 2: ejecución incompleta



El algoritmo
Ideas

- En la ciudad 1 hay que cargar combustible.
- Cada vez que cargamos combustible debemos decidir cuánto cargar y en cuál ciudad volveremos a cargar.
- Al cargar en la ciudad i, revisamos los precios en las ciudades que se encuentran dentro del alcance A:
  - si todas ellas tienen el combustible al mismo o mayor precio que en la ciudad i, debemos llenar el tanque y elegir para la próxima carga aquélla que tenga **el menor precio** ($c_j \geq c_i$, pero c_j menor que los demás dentro del alcance A a partir de la ciudad i)
    - en caso de empate se puede elegir cualquiera: por ejemplo, la más lejana.
  - si alguna de ellas tiene el combustible más barato que en la ciudad i, debemos cargar solamente el combustible necesario para llegar a la **primera ciudad** con combustible más barato que en i y cargar nuevamente allí.

El algoritmo

```
fun combustible(d:array[1..n-1] of nat, A:nat) ret S:array[1..n] of nat
    var j,a : nat
    j:= 0
    a:= 0
    for c:= 1 to n-1 do
        if a < d[c] then
            j:= j+1
            S[j]:= c
            a:= A-d[c]
        else a:= a - d[c]
        fi
    od
end fun
```