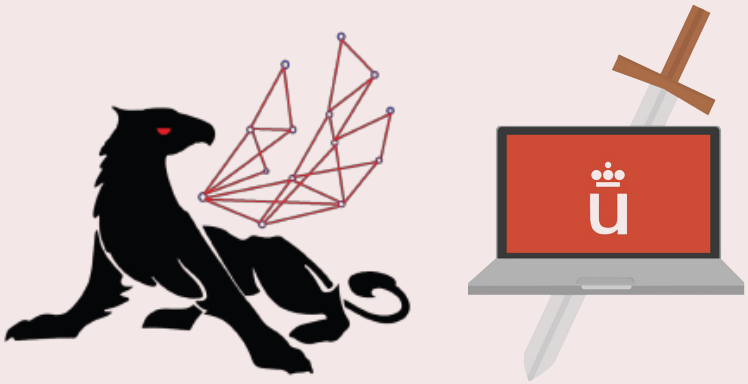


Concurso I 2024: ED

Estadísticas y Soluciones

URJC Escuela Técnica Superior de Ingeniería Informática

Clasificación de los problemas

Problema	Categoría
A - Vuelta a la rutina	Set
B - Victoria Magistral	Dividir
C - Tramitando papeleo	Cola y Set
D - Envíos prioritarios	Colas
E - Cifrado del César	Time Waster
F - El juego de la oca	Simular
G - Teclado Bancario	Mapas. Strings
H - Problema del cambio de monedas	Simular

Estadísticas

Problema	# casos de prueba	Espacio en disco
A - Vuelta a la rutina	7	90MB
B - Victoria Magistral	5	190 KB
C - Tramitando papeleo	11	148MB
D - Envíos prioritarios	8	73MB
E - Cifrado del César	11	80MB
F - El juego de la oca	7	76MB
G - Teclado Bancario	21	148MB
H - Problema del cambio de monedas	20	243KB
- **Total**	90	615MB

Estadísticas*

Problema	Primer chaval en resolverlo	Tiempo
A - Vuelta a la rutina	a.dekeno.2020	16
B - Victoria Magistral	i.penedo.2020	2
C - Tramitando papeleo	e.gomezf.2020	70
D - Envíos prioritarios	de.orna.2020	32
E - Cifrado del César	e.gomezf.2020	19
F - El juego de la oca	-	-
G - Teclado Bancario	a.mayoralg.2020	58
H - Problema del cambio de monedas	i.penedo.2020	47

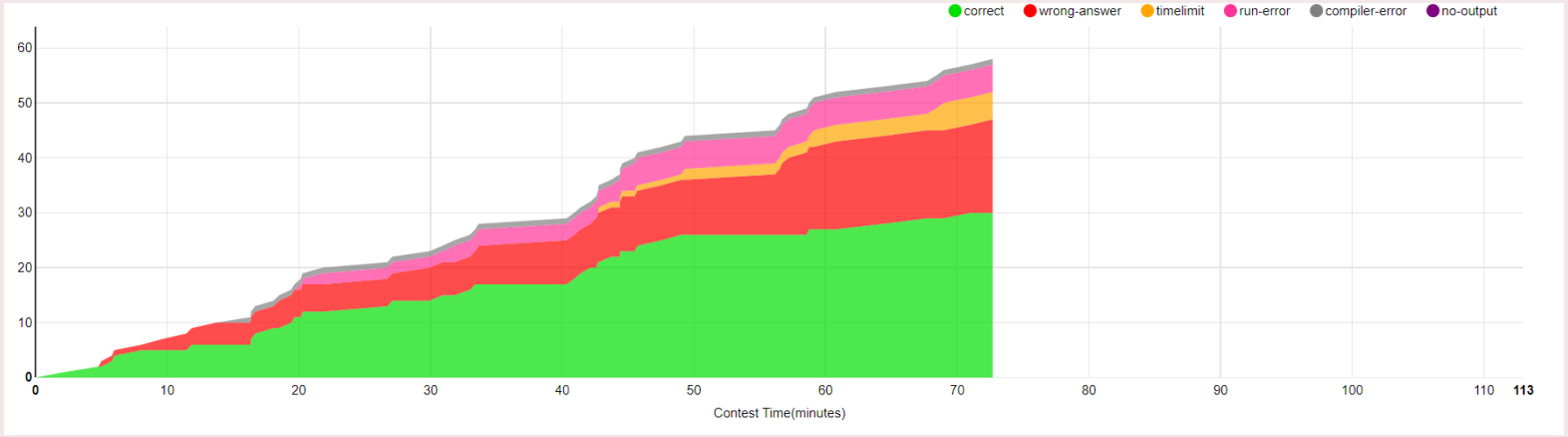
*Antes de congelar el marcador.

Estadísticas*

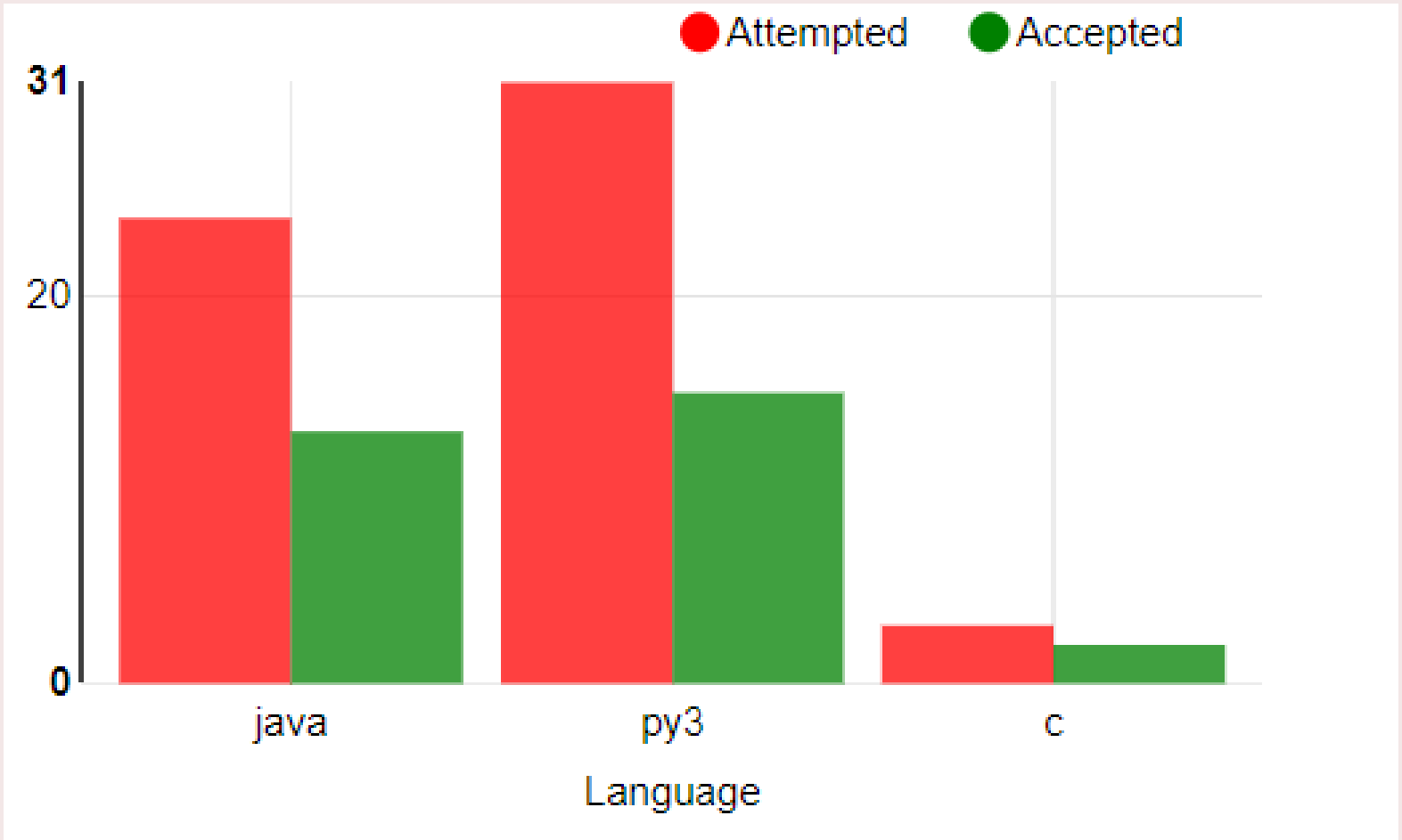
Problema	Envíos	Válidos	% éxito
A - Vuelta a la rutina	11	8	73 %
B - Victoria Magistral	17	14	82 %
C - Tramitando papeleo	3	1	33 %
D - Envíos prioritarios	4	2	50 %
E - Cifrado del César	15	2	13 %
F - El juego de la oca	0	0	0 %
G - Teclado Bancario	2	2	100 %
H - Problema del cambio de monedas	5	1	20 %

*Antes de congelar el marcador.

Estadísticas varias



Estadísticas varias

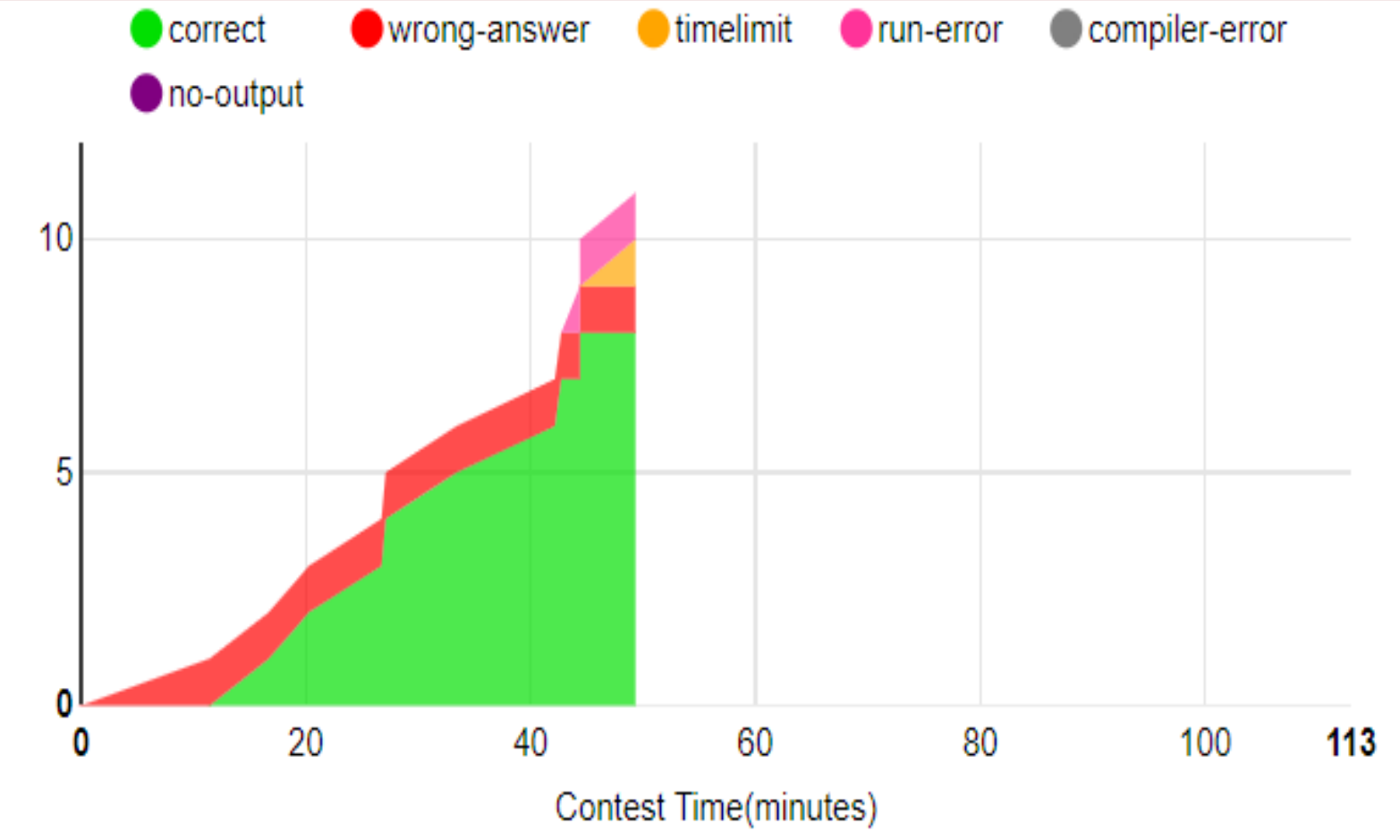


Estadísticas varias

A. Vuelta a la rutina

Envíos	Válidos	% éxito
11	8	73 %

A. Vuelta a la rutina



A. Vuelta a la rutina

Dadas N tareas a realizar y M tareas realizadas...

A. Vuelta a la rutina

Dadas N tareas a realizar y M tareas realizadas...
QUEDAN TAREAS por realizar?

A. Vuelta a la rutina

Dadas N tareas a realizar y M tareas realizadas...
QUEDAN TAREAS por realizar?
Se han realizado todas pero SE HAN REPETIDO tareas?

A. Vuelta a la rutina

Dadas N tareas a realizar y M tareas realizadas...
QUEDAN TAREAS por realizar?
Se han realizado todas pero SE HAN REPETIDO tareas?
Se han realizado EXACTAMENTE las n tareas?

A. Vuelta a la rutina

¿Cómo almacenamos las tareas que se deben realizar?

A. Vuelta a la rutina

¿Cómo almacenamos las tareas que se deben realizar?

↓

Conjunto de elementos que NO se REPITEN y NO tienen un ORDEN específico

A. Vuelta a la rutina

¿Cómo almacenamos las tareas que se deben realizar?

↓

Conjunto de elementos que NO se REPITEN y NO tienen un ORDEN específico

↓

SET

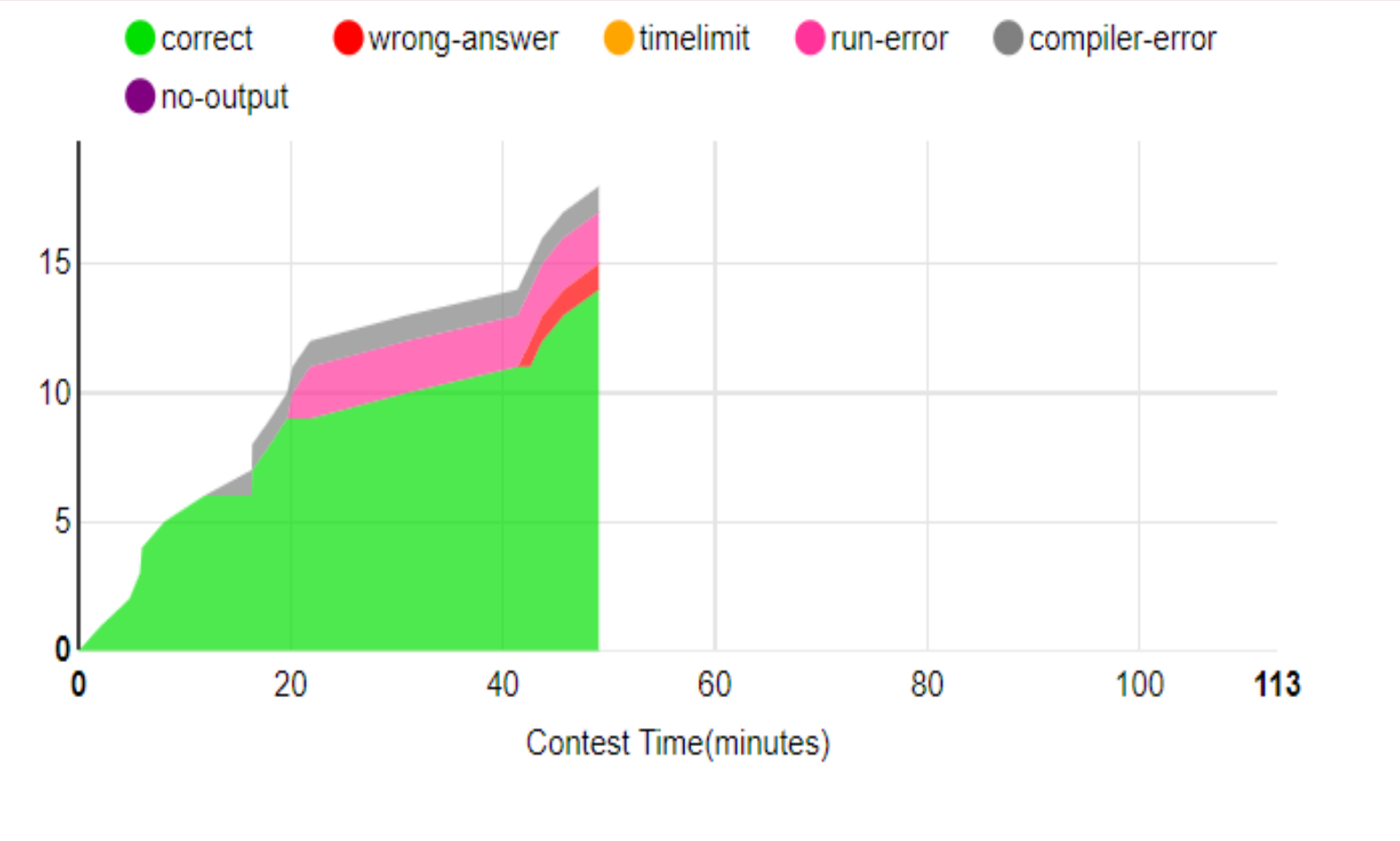
A. Vuelta a la rutina

```
tareasPendientes= new Set;

for i:1...n
      tareasPendientes.add(nuevaTarea);

for i:1...m
      tareasPendientes.remove(tareaRealizada);

//quedan tareas por resolver?
if(!tareasPendientes.isEmpty()) print("A TRABAJAR");
//no hay tareas pendientes pero se han repetido tareas?
else if(m>n) print("TRABAJAS DEMASIADO");
//en cualquier otro caso se han realizado exactamente las n tareas
else print("PUEDES DESCANSAR");
```

B. Victoria Magistral

Envíos	Válidos	% éxito
17	14	82 %

B. Victoria Magistral



B. Victoria Magistral

Dada la vida que tiene un rival y el daño que hace un bala, ¿cuántos disparos hacen falta para derribarle?

B. Victoria Magistral

Si cada disparo quita D de vida, necesitamos

$$\text{disparos}^*D >= V,$$

o lo que es lo mismo,

$$\text{disparos} >= V/D,$$

En este caso (V divisible entre D),

$$\text{disparos} = V/D,$$

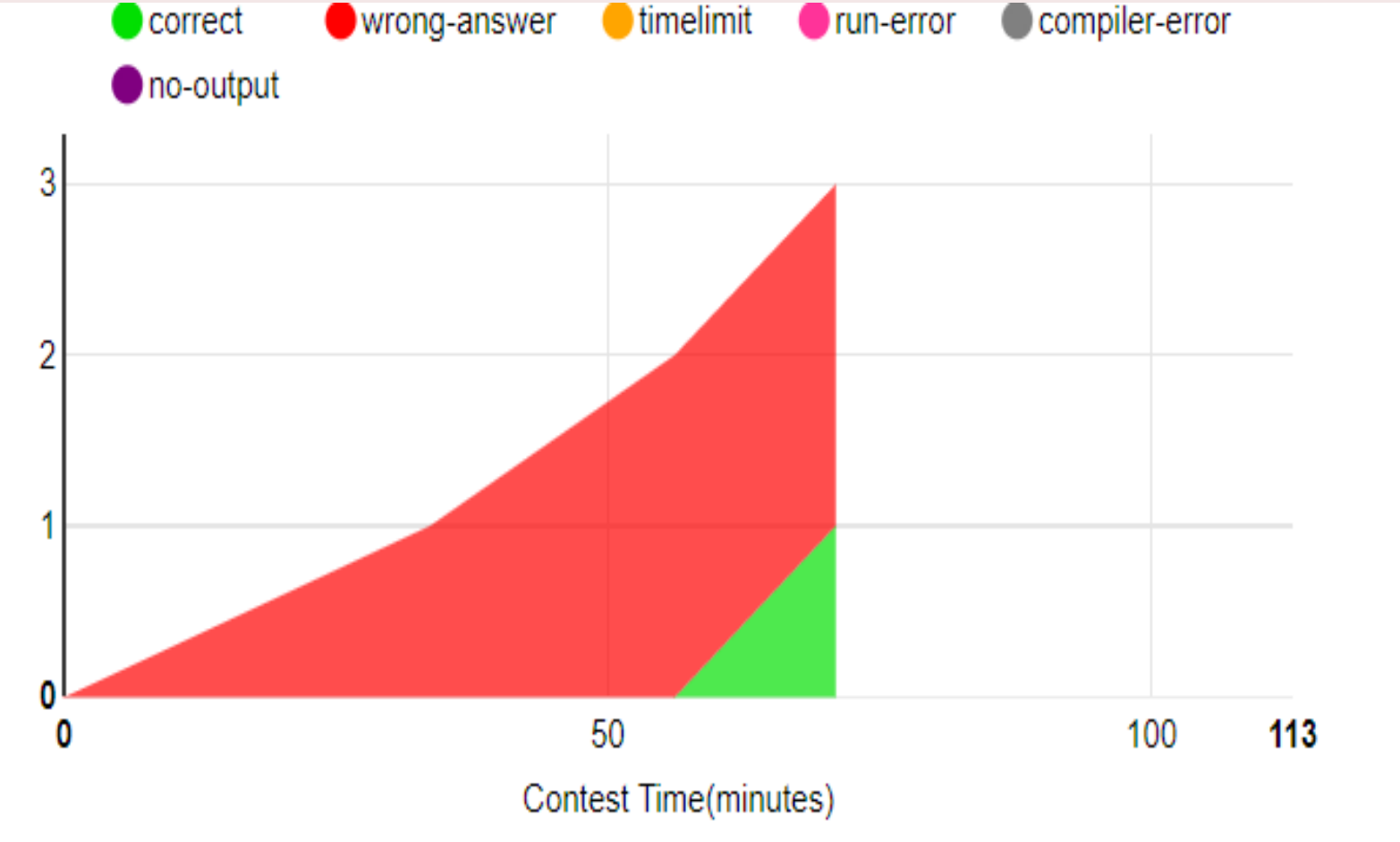
B. Victoria Magistral

```
leer N

for(N)

    leer V
    leer D
    disparos = V/D
    escribir disparos
```

C. Tramitando Papeleo

Envíos	Válidos	% éxito
3	1	33 %

C. Tramitando Papeleo



C. Tramitando Papeleo

Dado una serie de papeles que vienen identificados por el tiempo que se tardan en realizarlos (ya que no hay repeticiones) y un conjunto de papeles que se posponen si o sí, calcular el tiempo que se tarda en tramitarlos. **¿Solución?**

C. Tramitando Papeleo

SIMULAR

C. Tramitando Papeleo

Las estructuras de datos necesarias para la resolución son:

- Cola
- Set

C. Tramitando Papeleo

```
cola = {}
set= {}

leer(N)
for(i=1; i<N ;i++) {
    leer(aux)
    cola.addLast(aux)
}

leer(P)
for(i=1; i<P ;i++) {
    leer(aux)
    set.add(aux)
}
```

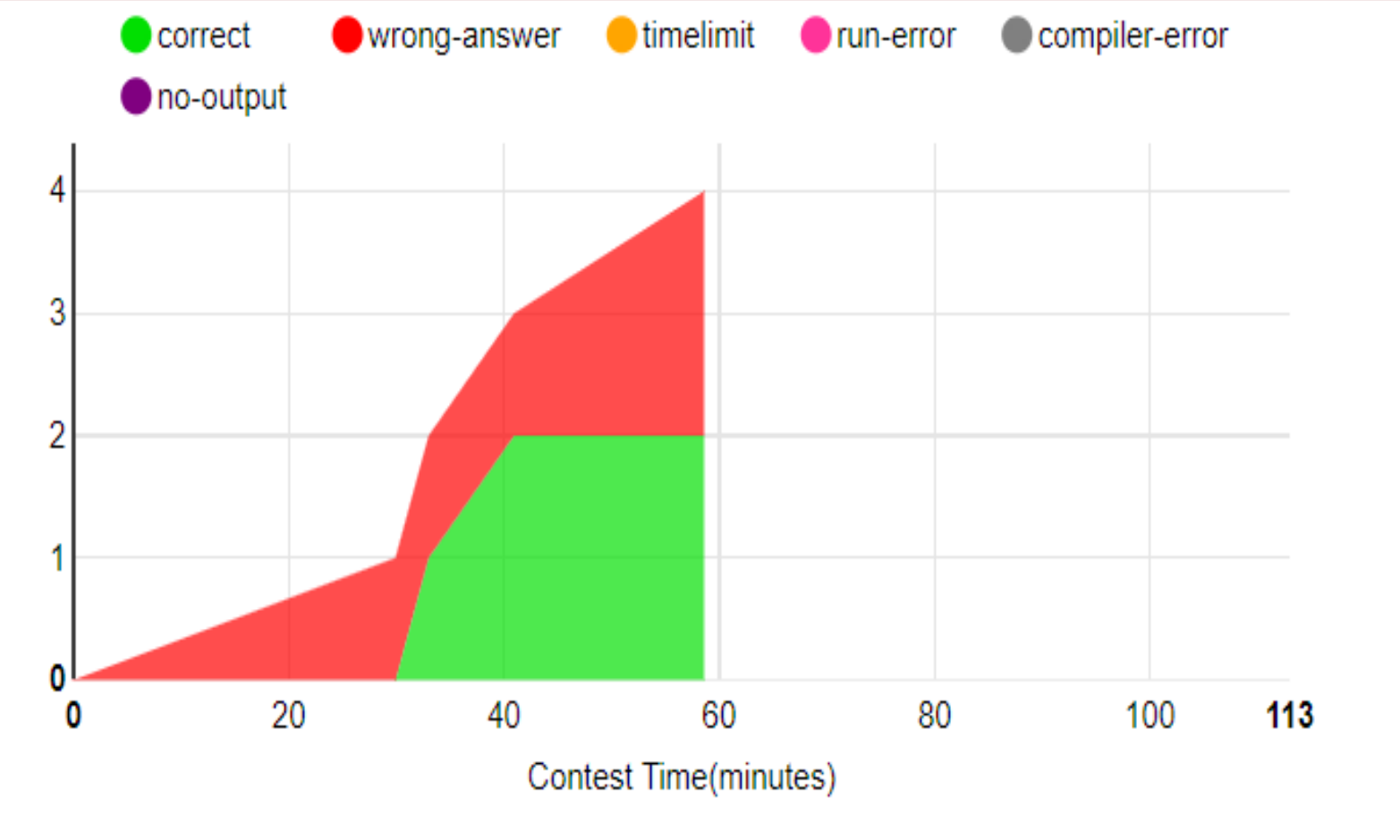
C. Tramitando Papeleo

```
T=0
while(cola.vacia()==false) {
    papel = cola.removeFirst()

    if( set.contains(papel) ) {
        cola.addLast( papel )
        T += 10
        set.remove( papel)
    } else if ( T<= 10 ) {
        T += papel
    } else if ( papel <= T/5 ) {
        T += papel
    } else {
        T += 10
        cola.addLast( papel )
    }
}
```

D. Envíos Prioritarios

Envíos	Válidos	% éxito
4	2	50 %

D. Envíos Prioritarios



D. Envíos Prioritarios

Cada vez que llega un camión de reparto, ¿qué paquetes hay que enviar según su prioridad?

D. Envíos Prioritarios

Primero los P2W, según orden de llegada.
Después los NOR, según orden de llegada.
Así hasta quedarnos sin paquetes o sin capacidad en el camión.

D. Envíos Prioritarios

```
colaP2W = {}
colaNOR = {}
paquetes = 0

leer N

for(n){
    if(PWR) {
        colaP2W.addLast(paquetes)
        paquetes++
    }
    else if(NOR) {
        colaNOR.addLast(paquetes)
        paquetes++
    }
```

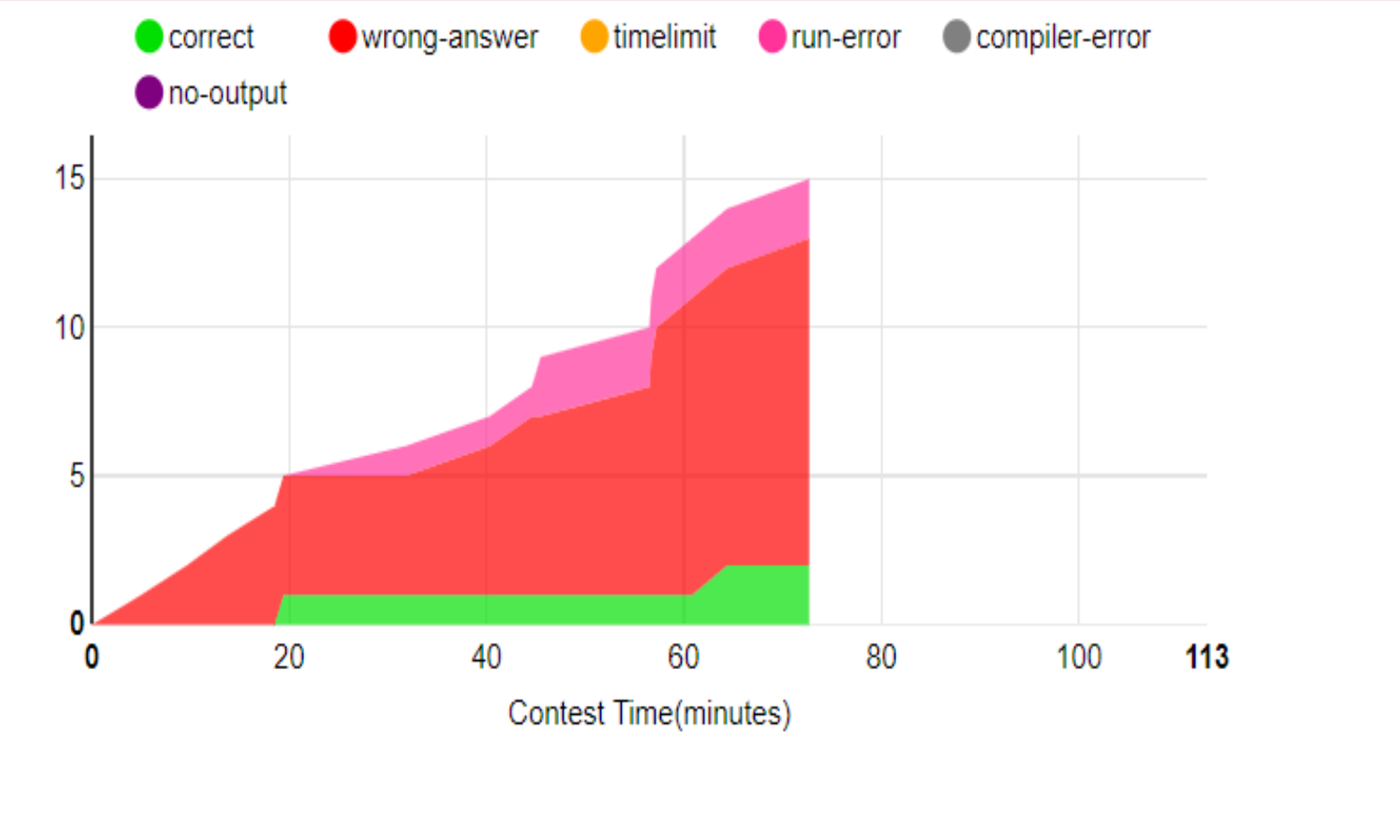
D. Envíos Prioritarios

```
        else {
            leer C
            while(C>0 && !colaP2W.isEmpty()) {
                paquete = colaP2W.removeFirst()
                meterCamión(paquete)
                C--
            }
            while(C>0 && !colaNOR.isEmpty()) {
                paquete = colaNOR.removeFirst()
                meterCamión(paquete)
                C--
            }
        }
    }
```

E. Cifrado del César

Envíos	Válidos	% éxito
15	2	13 %

E. Cifrado del César



E. Cifrado del César

- Este problema se basa en el conocido **Cifrado César**.
- El problema consta de serie de palabras y una clave K
- Para cada palabra tenemos que comprobar si se ha cifrado correctamente
- El cifrado se basa en sumarle K posiciones del abecedario a cada letra

E. Cifrado del César

TIME WASTER

E. Cifrado del César

Hay dos posibles formas de solucionar este problema:

- Con un array de caracteres
- Jugando con la tabla ASCII

E. Cifrado del César

```
leer S
leer W

aux = ""
for(int j=0; j<s.length(); j++) {
            char c = s.[j]
            char t = (char) (c+k)
            if(t>'z') t = (char) (((t-'z')%26)+96)
            aux += t
}
if(aux == w) imprimir(SI)
else imprimir(NO)
```

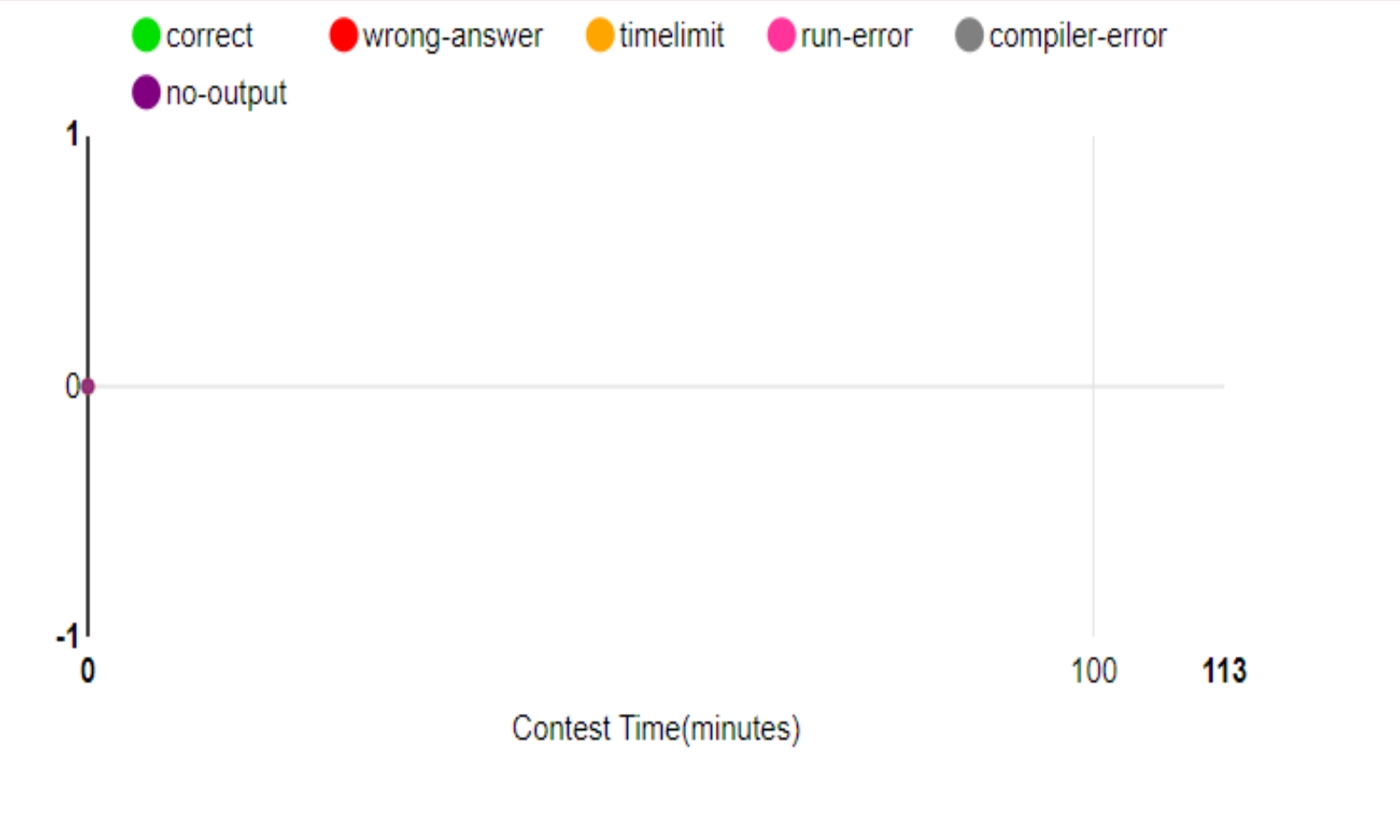
E. Cifrado del César

```
array = {a,b,c,d,e,f,g,h,i,j,k, ..... ,x,y,z}

leer S
leer W

aux = ""
for(int j=0; j<s.length(); j++) {
            int pos = s.[j]-'a'
            pos = (pos + k)%26
            aux += array[pos]
}
if(aux == w) imprimir(SI)
else imprimir(NO)
```

F. El juego de la oca

Envíos	Válidos	% éxito
0	0	0 %

F. El juego de la oca



F. El juego de la oca

Construir el tablero de la oca de tamaño NXN

F. El juego de la oca

Construir el tablero de la oca de tamaño NXN

=

Rellenar una matriz NXN con los números de las casillas

F. El juego de la oca

Simulamos

El recorrido de la oca

F. El juego de la oca

Posible solución:
Descomponer el problema de construir una ESPIRAL en construir CUADRADOS de tamaño decreciente.

F. El juego de la oca

1	2	3	4	5	6
20	21	22	23	24	7
19	32	33	34	25	8
18	31	36	35	26	9
17	30	29	28	27	10
16	15	14	13	12	11

F. El juego de la oca

1	2	3	4	5	6
20	21	22	23	24	7
19	32	33	34	25	8
18	31	36	35	26	9
17	30	29	28	27	10
16	15	14	13	12	11

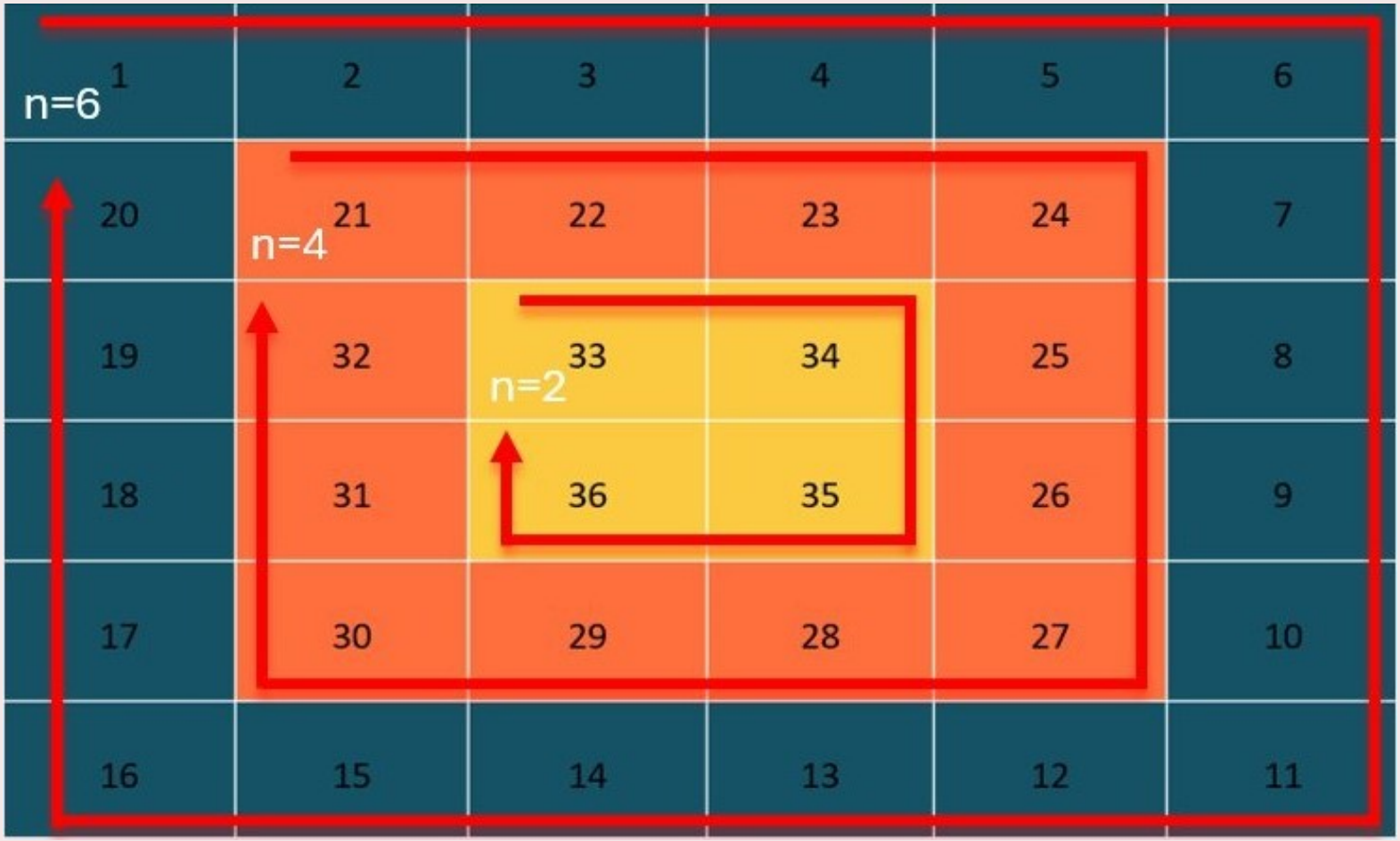
F. El juego de la oca

1	2	3	4	5	6
20	21	22	23	24	7
19	32	33	34	25	8
18	31	36	35	26	9
17	30	29	28	27	10
16	15	14	13	12	11

F. El juego de la oca

1	2	3	4	5	6
20	21	22	23	24	7
19	32	33	34	25	8
18	31	36	35	26	9
17	30	29	28	27	10
16	15	14	13	12	11

F. El juego de la oca



F. El juego de la oca

Podemos iterar desde i=n hasta i>=1, construyendo los cuadrados iterativamente.

F. El juego de la oca

```
static int matriz[n][n];
static int indice=1;

for(int i=n, int comienzo=0; n>=1; n-=2, comienzo++){
    // Construimos el cuadrado de lado n que empieza en la casilla
    // [comienzo][comienzo], por el número indice
    cuadrado(n,comienzo, indice);
}
```

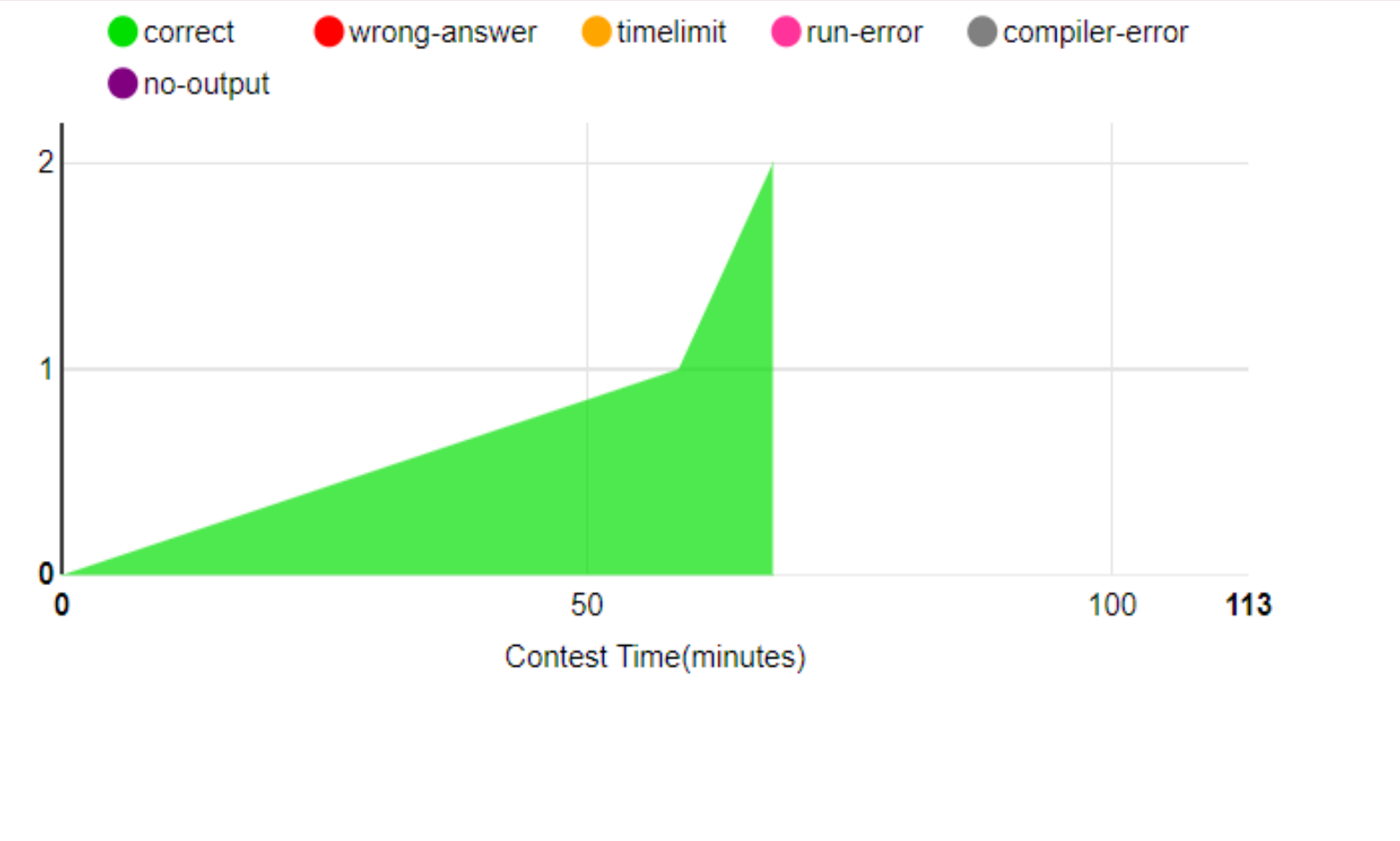
F. El juego de la oca

```
cuadrado(n,comienzo, indice){
    if(n==1) matriz[comienzo][comienzo]=indice ;

    //construimos cada lado del cuadrado
    for(int k=0; k<n-1; k++){
        matriz[comienzo][comienzo+k]=indice++;
    }
    ...para cada lado del cuadrado
}
```

G. Teclado Bancario

Envíos	Válidos	% éxito
2	2	100 %

G. Teclado Bancario

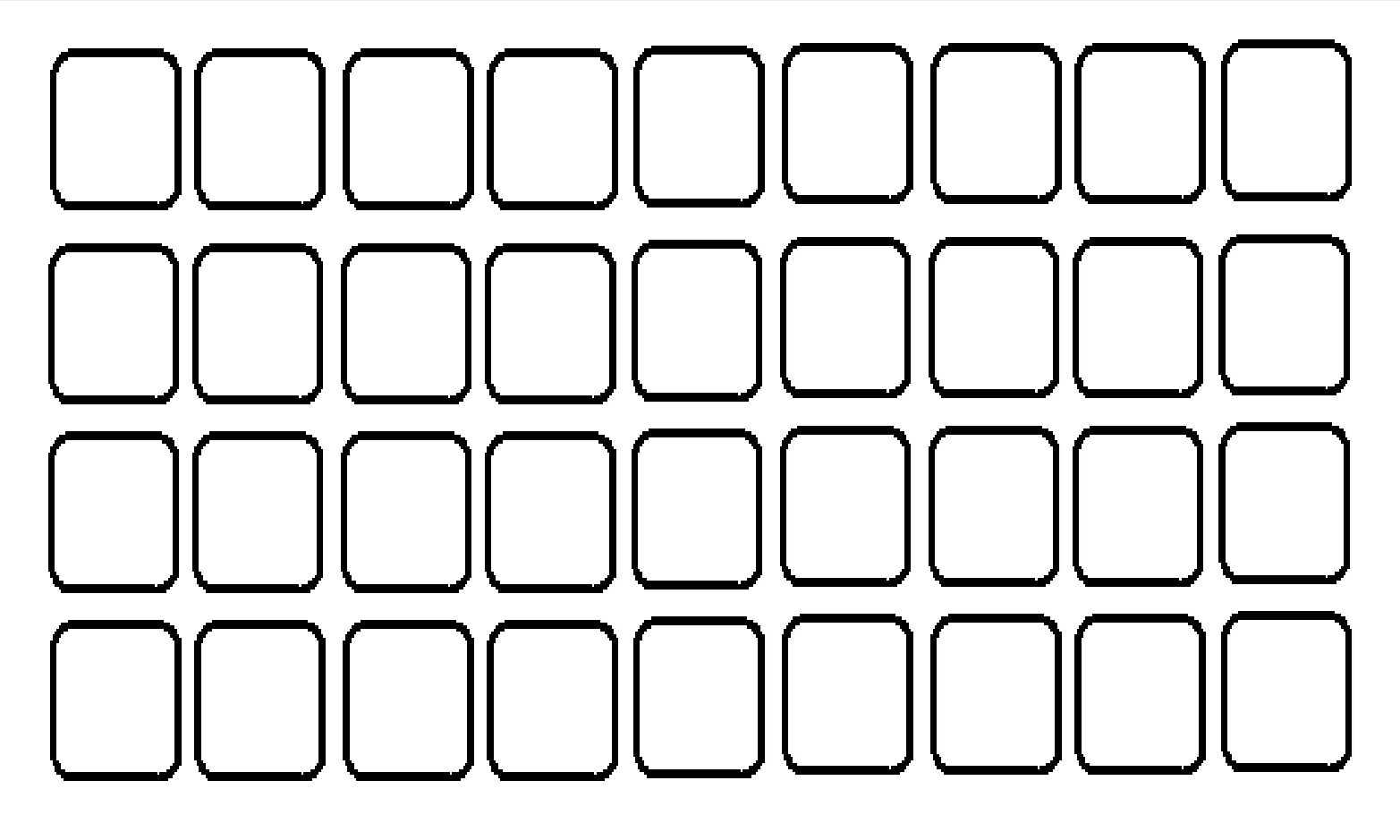


G. Teclado Bancario

Dada una palabra, ¿cuánto tarda un empleado promedio en escribirla?

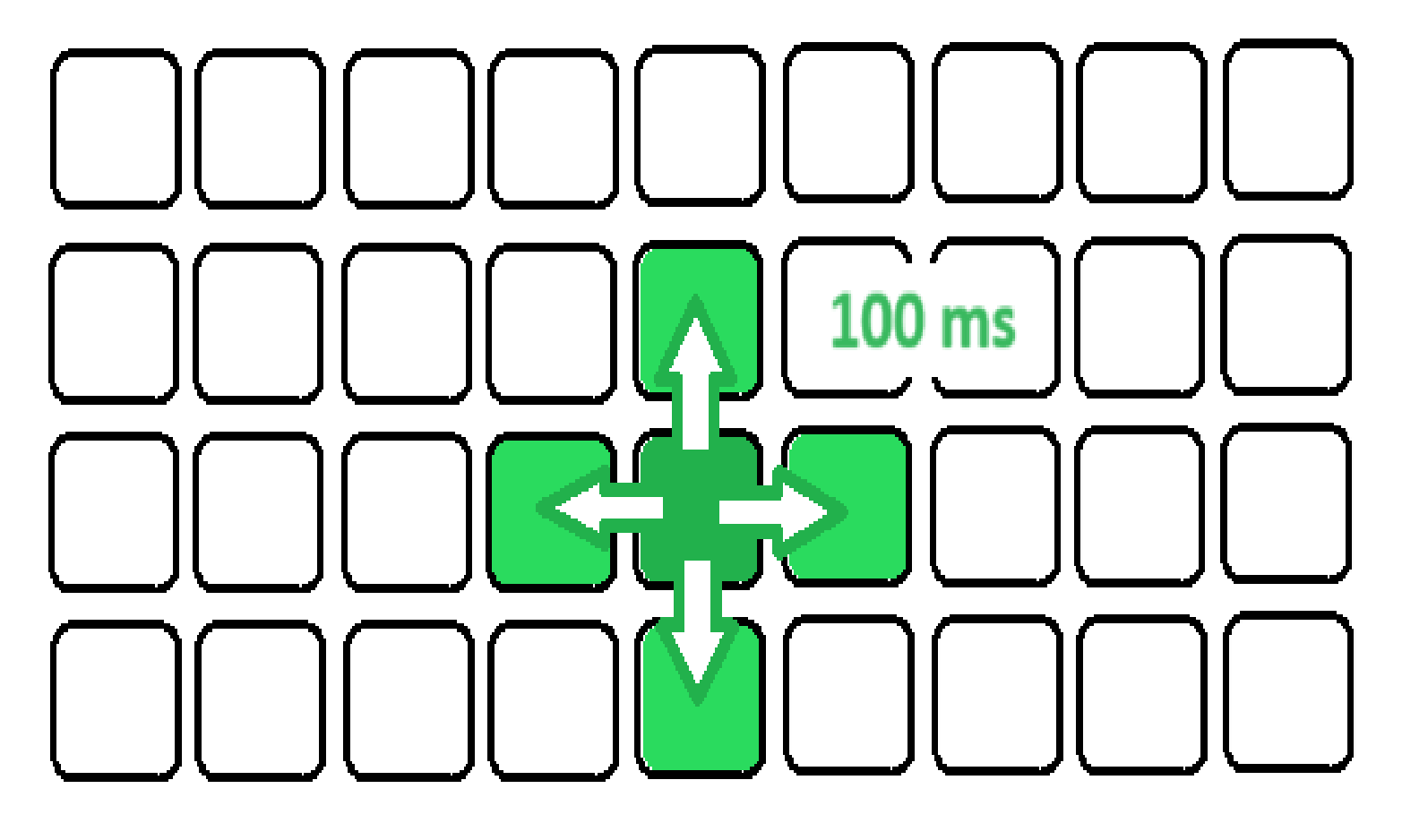
G. Teclado Bancario

El teclado es un matriz de 4 filas y 9 columnas



G. Teclado Bancario

Un empleado promedio tarda 100ms por cada letra que debe recorrer en busca de la siguiente

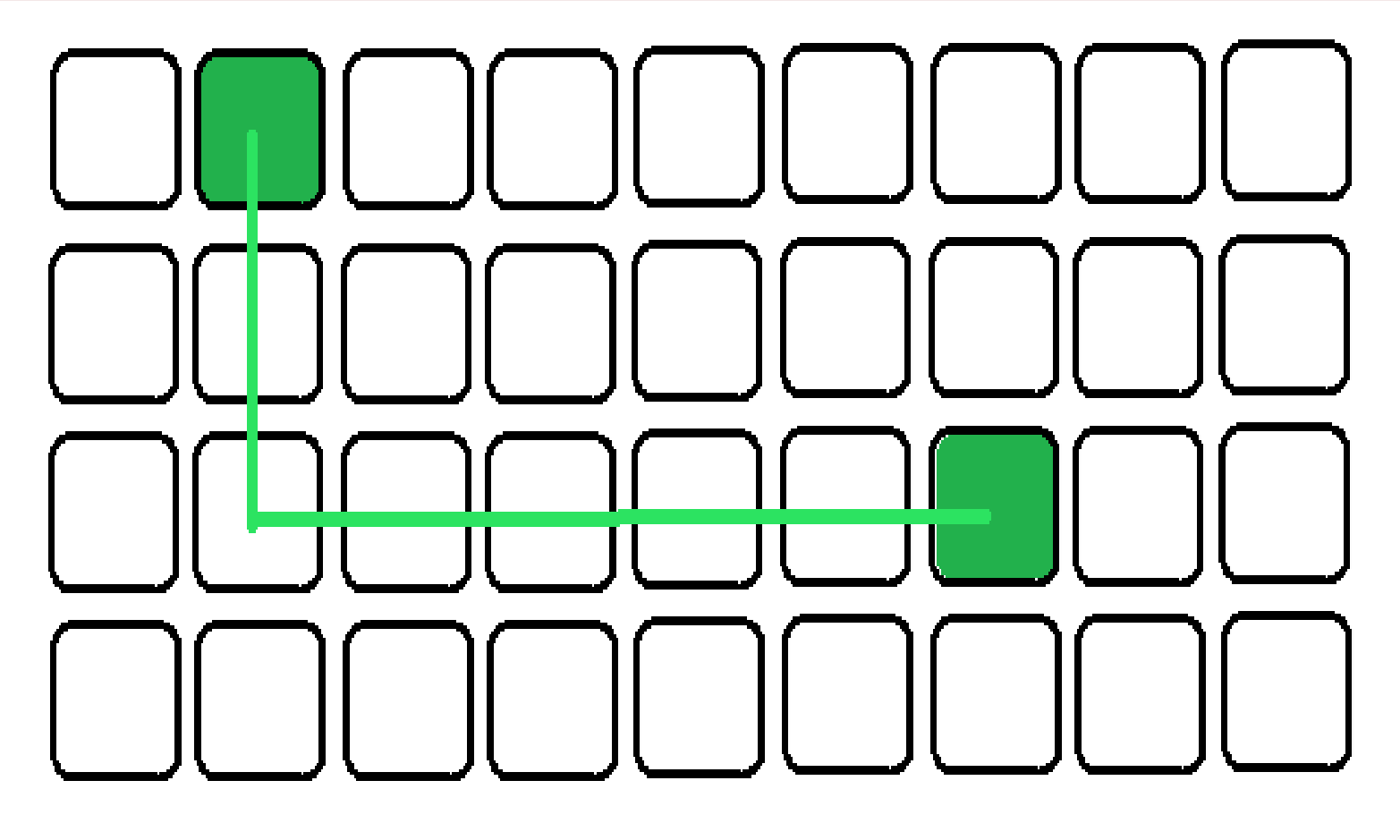


G. Teclado Bancario

Para saber cuánto tarda un empleado promedio en escribir una palabra, por cada letra, multiplicar 100ms por el número de teclas de distancia que haya con la anterior letra

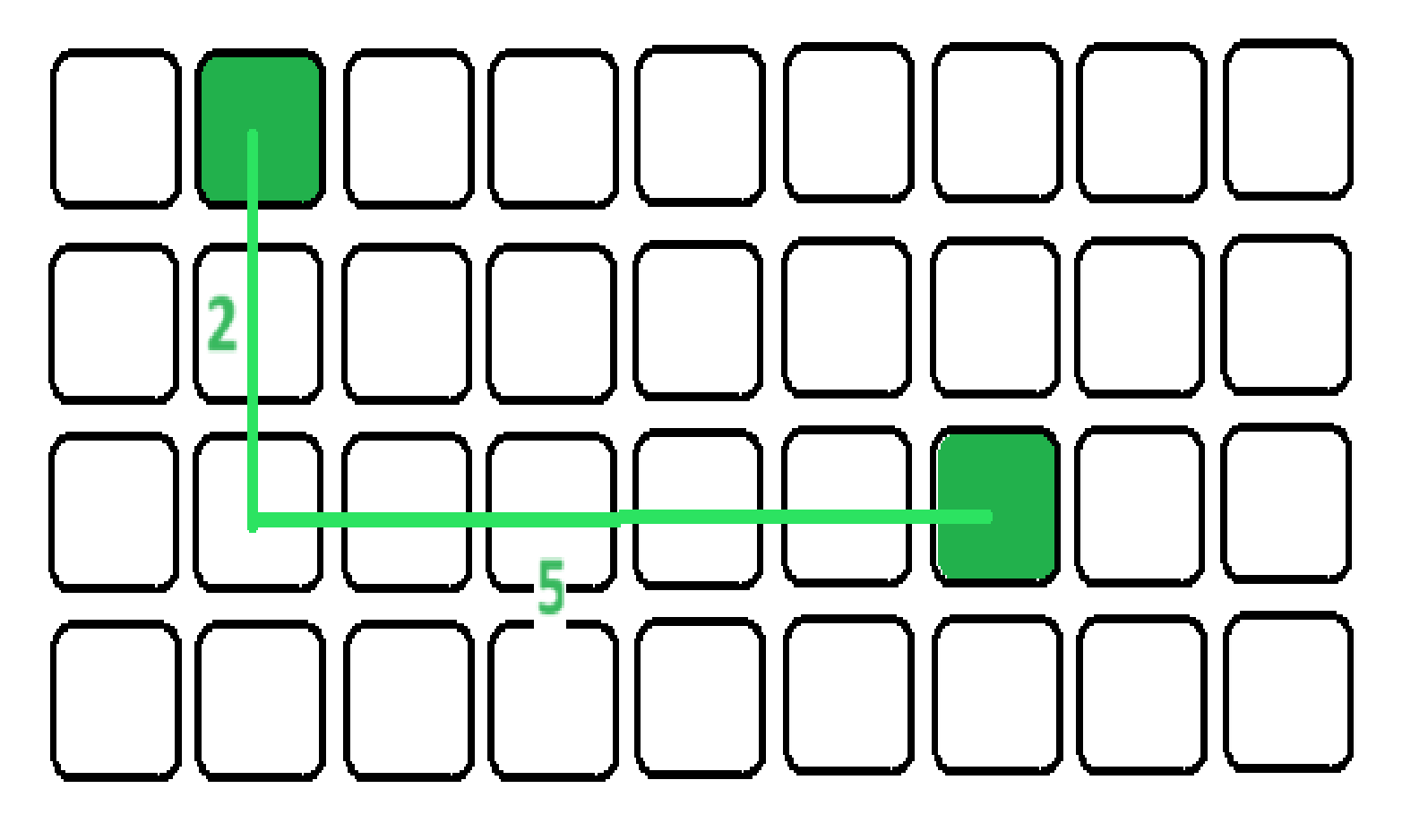
G. Teclado Bancario

¿Cuántas teclas hay que recorrer para ir de una tecla a la siguiente?



G. Teclado Bancario

Filas de distancia + columnas de distancia



G. Teclado Bancario

Necesitamos tener acceso a la fila y la columna en la que está cada letra.

G. Teclado Bancario

Mapa con el valor de [fila, columna] para cada letra.

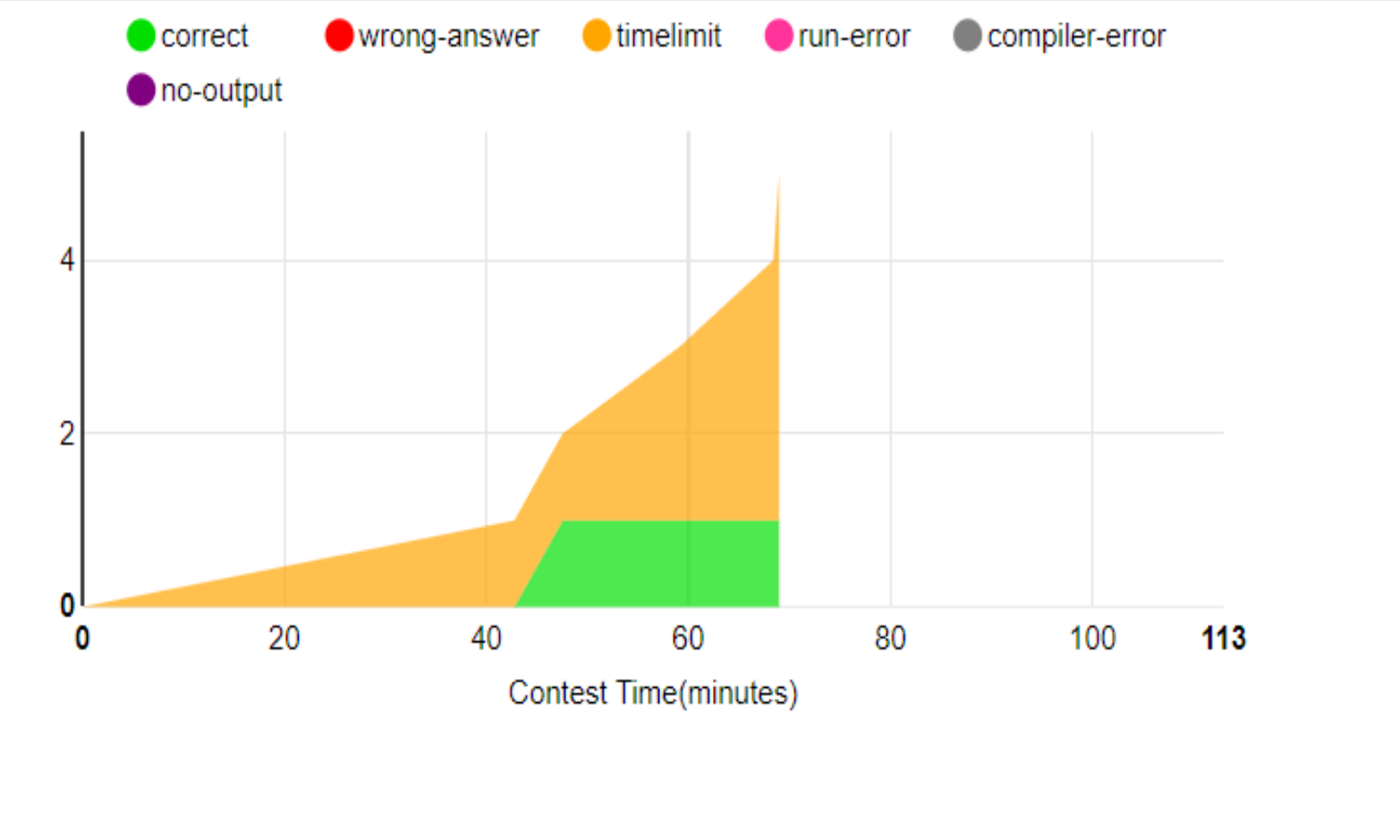
G. Teclado Bancario

```
mapaTeclado = {}
// leer teclado
for(filas)
    for(columnas)
        leer letra
        mapa.put(letra, [fila, columna])

leer N
for(N){
    leer palabra
    tiempo = 0
    for(int l = 1; l < palabra.length(); l++)
        letra = palabra.charAt(l)
        ant = palabra.charAt(l-1)
        teclas = |mapa.get(letra).fila - mapa.get(ant).fila|
        + |mapa.get(letra).columna - mapa.get(ant).columna|
        tiempo += teclas * 100
    escribir tiempo
```

H. Problema del cambio de monedas.

Envíos	Válidos	% éxito
5	1	20 %

H. Problema del cambio de monedas.



H. Problema del cambio de monedas.

Dadas las cantidades de monedas disponibles de valores
500, 200, 100, 50, 20, 10, 5, 2 y 1,
¿mínimo número de monedas necesarias para obtener una cantidad n?

H. Problema del cambio de monedas.

El problema del CAMBIO DE MONEDA es un problema clásico, que puede resolverse:

- VORAZMENTE(=simulando) (los valores de las monedas deben cumplir una serie de condiciones)
- Utilizando técnicas MÁS COMPLEJAS (Como programación dinámica) (funciona para cualquier caso)

De momento, nos centraremos en que, para las monedas dadas en este problema, la solución VORAZ(=simulando) es suficiente.

H. Problema del cambio de monedas.

¿Cómo simulamos la devolución del cambio para devolver el MENOR número de monedas?

H. Problema del cambio de monedas.

Escogiendo siempre la MAYOR moneda posible.

H. Problema del cambio de monedas.

Iteramos sobre la lista de monedas, de mayor a menor valor, escogiendo la mayor cantidad posible de cada moneda.

H. Problema del cambio de monedas.

```
int[] monedas={500,200,100,50,20,10,5,2,1};
int[] disponibles;
//lectura de las cantidades de monedas de cada tipo

//para cada valor num a devolver...
  int total=0;
       for(int i=0; i<9; i++){
           int utilizadas=Math.min(disponibles[i],num/M[i]);
           total+=utilizadas;
           num-=total*M[i];
       }
```