Si el número *n* tiene un único dígito ($n \leq 9$), hay exactamente *n* números alternados menores que *n* (por ejemplo, si *n* = 5, los *n* = 5 números serán el 0, el 1, el 2, el 3 y el 4).

Si *n* tiene más de un dígito (*n* > 9), *n* será de la forma (*n/10*)*d*. Puede, entonces, resolverse el problema para (*n*/10). Sea *c'* la solución a dicho problema para (*n/10*) (es decir, sea *c'* la cantidad de números alternados menores que *n/10*):

- Tomemos cualquier número *u* representado por dicha cantidad. Podemos, entonces, formar 5 números alternados menores que *n* a partir de *u*. Cada uno de estos será de la forma *u d'*, donde *d'* es un dígito impar si el último dígito de *u* es par, o un dígito par en otro caso (por ejemplo, si *n* = 536 y el número *u* que hemos elegido es 51, los 5 números a los que hacemos referencia son el 510, el 512, el 514, el 516 y el 518).
- Dado que hay *c'* números alternados entre los que elegir *u*, podemos formar 5*c' números alternados menores que *n* con este método.
- Asímismo, si *u* = 0, con este método formaremos los números *1*,*3*,*5* y *7*. Y, si *u* es distinto de 0, con este método formaremos siempre números de más de un dígito. Por tanto, con el método descrito es imposible generar el 0, el 2, el 4, el 6 y el 8, todos ellos números alternados menores que *n*. Habrá, por tanto, que contabilizar 5 números más.
- Por tanto, tendremos 5*c'* + 5 números alternados menores que *n*, todos ellos de la forma *ud'*, con *u* un número alternado *menor* que *n/10*.

Para completar la cuenta de números alternados menores que *n*, nos falta por considerar los posibles números alternados de la forma (*n*/10)*d'*. Esto tendrá sentido siempre y cuando (*n/10*) sea alternado (si no, no habrá ningún número alternado menor que *n* de la forma (*n*/10)*d'*). En este caso, si el dígito menos significativo de (*n*/10) es impar, *d'* puede ser cualquier dígito par menor que *d* (recuérdese que *d* era el dígito menos significativo de *n*; es decir *d* = n%10). Si el dígito menos significativo de (*n/10*) es par, *d'* podrá ser cualquier dígito impar menor que *d.* Necesitamos determinar, entonces, fórmulas que, dado *n%10*, nos proporcionen, respectivamente, la cantidad de dígitos pares y de dígitos impares menores que *n%10*. Para ello podemos considerar todas las posibilidades:

n%10	Número de impares menores que n%10	Número de pares menores que n%10
0	0	0
1	0	1
2	1	1
3	1	2
4	2	2
5	2	3
6	3	3
7	3	4
8	4	4
9	4	5

Inspeccionando ambas columnas es inmediato darse cuenta de que el número de impares menores que n%10 viene dado por (n%10)/2 (aquí / es la división entera), mientras que el número de pares menores que n%10 viene dado por ((n%10)+1)/2.

Por tanto:

- Si *n*/10 es alternado, y el dígito menos significativo de *n/10* es impar (lo que es equivalente a afirmar que *n/10* es impar), habrá que sumar ((n%10)+1)/2 nuevos números alternados menores que *n* (todos los que se forman como (n/10)*d'*, con *d'* un dígito par menor que n%10). Por ejemplo, para *n* = 5437, habrá que sumar ((n%10)+1)/2 = (7+1)/2 = 4, cantidad que contabiliza los números 5430, 5432, 5434 y 5436.

- Por el contrario, si el dígito menos significativo de *n/10* es par (lo que es equivalente a afirmar que *n/10* es par), habrá que sumar (n%10)/2 (todos los que se formar como (n/10)*d'*, con *d'* un dígito impar menor que n%10). Por ejemplo, para *n* = 5347 habrá que sumar (n%10)/2 = 7/2 = 3, cantidad que cuenta a 5341, 5343 y 5345.

Por último, para permitir decidir en tiempo constante cuando *n/10* es alternado, puede devolverse, en la generalización, una salida *es_alternado* adicional que determine si el numero *n* de entrada es o no alternado:

- En el caso $n \leq 9$, *es_alternado* será directamente *true*.
- En el caso n>9, podemos considerar el resultado *es_alternado'* para *n/10*. Si *es_alternado'* es *false*, *es_alternado* será directamente *false*. En otro caso, *es_alternado* será *true* siempre y cuando la paridad del dígito menos significativo de n/10 (equivalente, la paridad de *n/10*) sea diferente a la de *d*, el dígito menos significativo de *n* (equivalente, la paridad de *n*). Por tanto, en este caso, el valor de *es_alternado* coincidirá con el del predicado (*n/10)%2 != n%2*.

Con todas estas consideraciones, el problema puede resolverse con ayuda de una generalización `cuenta_alternados` que tome como entrada el número *n*, y devuelva como resultados (i) si *n* es o no alternado; y (ii) la cantidad de números alternados menores que *n*. Teniendo en cuenta el análisis realizado anteriormente, dicha generalización puede codificarse como:

```
void cuenta_alternados(t_num n, bool& es_alternado, t_num& c) {
      if (n <= 9) {
            c = n;
            es_alternado = true;
      }
      else {
            cuenta_alternados(n / 10, es_alternado, c);
            c *= 5;
            c += 5;
            if (es_alternado) {
                  if ((n / 10) % 2 != 0) {
                        c += ((n % 10) + 1) / 2;
                  }
                  else {
                        c += (n % 10) / 2;
                  }
                  es_alternado = ((n / 10) % 2 != n % 2);
            }
      }
}
```

La definición de la función `num_alternados` que realiza el algoritmo por inmersión en base a esta generalización es directa:

```
t_num num_alternados(t_num n) {
      t_num c;
      bool es_alternado;
      cuenta_alternados(n, es_alternado, c);
      return c;
}
```

Por último, la complejidad del algoritmo viene dada por la complejidad de `cuenta_alternados`. Si se toma *n* como tamaño del problema, la complejidad puede determinarse a partir de la siguiente recurrencia:

$T'(n) = C_0$, si $n \leq 9$

$T'(n) = C_1 + T'(n/10)$, si $n > 9$

donde C_0 es el coste del caso base, y C_1 es el mayor coste del código que rodea al caso recursivo. Ambos costes son constantes. Esta recurrencia puede resolverse aplicando el patrón de descomposición por división, obteniéndose que $T'(n) \in O(\log n)$. Como la función de tiempo del algoritmo $T(n)$ cumplirá siempre que $T(n) \leq T'(n)$, $T(n)$ estará también en $O(\log n)$.

Si en lugar de *n* se toma como medida de la complejidad del problema el número de dígitos D de *n*, la recurrencia a la que se llega es:

$T'(1) = C_0$

$T'(D) = C_1 + T'(D-1)$, si $D>1$

lo que finalmente permite decidir que $T(n)$ está en $O(D)$.