Problema BinSearch

Archivo de entrada	`stdin`
Archivo de salida	`stdout`

```
bool binary_search(int n, int p[], int target){
    int left = 1, right = n;
    while(left < right){
        int mid = (left + right) / 2;
        if(p[mid] == target)
            return true;
        else if(p[mid] < target)
            left = mid + 1;
        else
            right = mid - 1;
    }
    if(p[left] == target) return true;
    else return false;
}
```

Es bien sabido que si `p` está ordenado, entonces este código devuelve `true` si y solo si `target` aparece dentro de `p`. Por otro lado, este puede no ser el caso si `p` no está ordenado.

Se le da un entero positivo n y una secuencia $b_1, \ldots, b_n \in \{\texttt{true}, \texttt{false}\}$. Se garantiza que $n = 2^k - 1$ para algún k entero positivo. Debe generar una permutación p de $\{1, \ldots, n\}$ que siga ciertas condiciones. Sea $S(p)$ el número de índices $i \in \{1, \ldots, n\}$ para los cuales `binary_search(n, p, i)` devuelve **not** b_i. Debe establecer p para que $S(p)$ sea pequeño (como se detalla en la sección "Restricciones").

(Nota: una permutación de $\{1, \ldots, n\}$ es una secuencia de n enteros que contiene cada entero desde 1 a n *exactamente* una vez).

Entrada

La entrada contiene varios casos de prueba. La primera línea de entrada contiene T, el número de casos de prueba. Siguen los casos de prueba.

La primera línea de un caso de prueba contiene el número entero n. La segunda línea de un caso de prueba contiene una cadena de longitud n que contiene solo los caracteres '0' y '1'. Estos caracteres no están separados por espacios. Si el carácter i-ésimo es '1', entonces $b_i = \texttt{true}$, y si es '0', entonces $b_i = \texttt{false}$.

Salida

Los datos de salida constan de las respuestas para cada uno de los casos de prueba de T. La respuesta para caso de prueba particular consiste en la permutación p generada para ese caso de prueba.

Restricciones

- $\sum n$ Es la suma de todos los valores de n en una sola entrada.
- $1 \leq \sum n \leq 100\,000$.
- $1 \leq T \leq 7\,000$.
- $n = 2^k - 1$ para algunos $k \in \mathbb{N}$, $k > 0$.
- Si $S(p) \leq 1$ para todos los casos de prueba dentro de una subtarea, se otorga el 100% de los puntos para esa subtarea.
- En caso contrario, si $0 \leq S(p) \leq \lceil \log_2 n \rceil$ (por ejemplo, $1 \leq 2^{S(p)} \leq n + 1$) para todos los casos de prueba dentro de una subtarea, se otorga el 50% de los puntos para esa subtarea.

#	Puntos	Restricciones
1	3	$b_i =$ `true`.
2	4	$b_i =$ `false`.
3	16	$1 \leq n \leq 7$.
4	25	$1 \leq n \leq 15$.
5	22	$n = 2^{16} - 1$ y cada b_i se selecciona de manera uniforme e independiente al azar de {`true`, `false`}.
6	30	Sin restricciones adicionales.

Ejemplos

Archivo de entrada	Archivo de salida
4 3 111 7 1111111 3 000 7 000000000	1 2 3 1 2 3 4 5 6 7 3 2 1 7 6 5 4 3 2 1
2 3 010 7 0010110	3 2 1 7 3 1 5 2 4 6

Explicaciones

Ejemplo 1. En los dos primeros casos de prueba del primer ejemplo, tenemos $S(p) = 0$.

En el tercer caso de prueba, tenemos $S(p) = 1$. Esto es porque `binary_search(n, p, 2)` devuelve `true`, aunque $b_2 =$ `false`.

En el cuarto caso de prueba, tenemos $S(p) = 1$. Esto es porque `binary_search(n, p, 4)` devuelve `true`, aunque $b_4 =$ `false`.

Ejemplo 2. Tenemos $S(p) = 0$ para ambos casos.