Nonogramma

Il Nonogramma è un rompicapo molto conosciuto. Consideriamo una versione uno-dimensionale di questo rompicapo: in questo caso, il giocatore ha una singola riga di n celle.

Le celle sono numerate da 0 a $n-1$ da sinistra a destra. Il giocatore deve colorare ogni cella di nero o di bianco: con 'X' indicheremo una cella nera e con '_' una cella bianca.

Il giocatore riceve quindi una sequenza $c = [c_0, \ldots, c_{k-1}]$ di k interi positivi: gli *indizi*. Deve quindi colorare le celle di modo che le celle nere nella riga formino esattamente k blocchi di celle consecutive. Inoltre, il numero di celle nere nel blocco i-esimo (0-based) a partire da sinistra deve essere uguale a c_i.

Per esempio, se gli indizi sono $c = [3, 4]$, il risultato finale dovrà avere esattamente due blocchi consecutivi di celle nere: uno di lunghezza 3 e l'altro di lunghezza 4. Quindi, se $n = 10$ e $c = [3, 4]$, una soluzione agli indizi forniti è "_XXX__XXXX".

Nota che "XXXX_XXX__" non è invece una soluzione agli indizi: i blocchi di celle nere non sono nell'ordine corretto. Inoltre, anche "__XXXXXXX_" non è una soluzione agli indizi: c'è un singolo blocco di celle nere anziché due separati.

Ti viene dato un Nonogramma parzialmente risolto, cioè il numero n, la sequenza c e l'informazione che alcune delle celle sono colorate di nero o di bianco. Il tuo compito è di dedurre informazioni addizionali.

Più precisamente, diciamo *soluzione valida* una colorazione che soddisfa gli indizi dati e rispetta le informazioni sui colori inizialmente note. Devi scoprire quali celle sono nere in ogni soluzione valida, e quali sono bianche in ogni soluzione valida.

È garantito che l'input è tale per cui esiste sempre almeno una soluzione valida.

Dettagli di implementazione

Devi implementare la seguente funzione (metodo):

- `string solve_puzzle(string s, int[] c)`.
  - `s`: stringa di lunghezza n. Per ogni i compreso tra 0 e $n-1$ il carattere i-esimo è:
    - 'X', se la cella i deve essere nera,
    - '_', se la cella i deve essere bianca,
    - '.', se non ci sono informazioni riguardo la cella i.
  - `c`: array di lunghezza k contenente gli indizi (definiti nel testo),

- la funzione deve restituire una stringa di lunghezza n. Per ogni i compreso tra 0 e $n-1$ il carattere i-esimo della stringa restituita deve essere:
  - 'X', se la cella i è nera in ogni soluzione valida,
  - '_', se la cella i è bianca in ogni soluzione valida,
  - '?', altrimenti (cioè se esistono due soluzioni valide tali per cui la cella i è nera nella prima soluzione e bianca nella seconda).

Nel linguaggio C la signature della funzione è leggermente diversa:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
  - `n`: lunghezza della stringa `s` (numero di celle),
  - `k`: lunghezza dell'array `c` (numero di indizi),
  - gli altri parametri sono come sopra,
  - invece di restituire una stringa, la funzione deve scrivere la risposta nella stringa `result`.

I codici ASCII dei caratteri usati in questo problema sono:

- 'X': 88,
- '_': 95,
- '.': 46,
- '?': 63.

Vedi i template forniti per ulteriori dettagli di implementazione nel tuo linguaggio di programmazione.

Esempi

Esempio 1

```
solve_puzzle(".........", [3, 4])
```

Queste sono tutte le possibili soluzioni valide del rompicapo:

- "XXX_XXXX__",
- "XXX__XXXX_",
- "XXX___XXXX",
- "_XXX_XXXX_",
- "_XXX__XXXX",
- "__XXX_XXXX".

Nota che le celle con indici 2, 6 e 7 sono nere in ogni soluzione valida, mentre ogni altra cella può essere nera oppure no.

La risposta corretta è quindi "??X???XX??".

Esempio 2

```
solve_puzzle(".......", [3, 4])
```

In questo esempio la soluzione è univocamente determinata.

La risposta corretta è quindi "XXX_XXXX".

Esempio 3

```
solve_puzzle("..._._....", [3])
```

In questo caso possiamo dedurre che la cella 4 deve anche essere bianca — non c'è modo di di far stare tre celle nere consecutive tra le celle bianche 3 e 5.

La risposta corretta è quindi "`???___????`".

Esempio 4

```
solve_puzzle(".X........", [3])
```

Ci sono soltanto due soluzioni valide che corrispondono con la descrizione sopra:

- "`XXX_______`",
- "`_XXX______`".

La risposta corretta è quindi "`?XX?______`".

Subtask

In tutti i subtask $1 \leq k \leq n$, e $1 \leq c_i \leq n$ per ogni $0 \leq i \leq k-1$.

1. (7 punti) $n \leq 20$, $k = 1$, s contiene solo '`.`' (schema vuoto),
2. (3 punti) $n \leq 20$, s contiene solo '`.`',
3. (22 punti) $n \leq 100$, s contiene solo '`.`',
4. (27 punti) $n \leq 100$, s contiene solo '`.`' e '`_`' (in altre parole, le informazioni riguardano solo le celle bianche),
5. (21 punti) $n \leq 100$,
6. (10 punti) $n \leq 5\,000$, $k \leq 100$,
7. (10 punti) $n \leq 200\,000$, $k \leq 100$.

Grader di esempio

Il grader di esempio legge l'input nel formato seguente:

- riga 1: la stringa s,
- riga 2: l'intero k seguito da k interi $c_0, \ldots, c_{k-1}$.