

Mal etter tall

Mal etter tall er et velkjent hjernetrim-spill. Vi ser på en enkel versjon av spillet, med bare en dimensjon. I dette spillet er spilleren gitt en rad med n celler. Cellene er nummerert fra 0 til $n - 1$, fra venstre til høyre. Spilleren skal male hver celle enten svart eller hvit. Vi bruker 'X' for å merke svarte celler og '_' for å merke hvite celler.

Spilleren er gitt en sekvens $c = [c_0, \dots, c_{k-1}]$ med k positive heltall: *hintene*. Han må male cellene på en slik måte at de svarte cellene i raden utgjør nøyaktig k blokker med påfølgende celler. Videre skal antallet i den i -ende blokken (0-basert) fra venstre skal være lik c_i . For eksempel, hvis hintene er $c[3, 4]$, skal det løste spillet ha nøyaktig to blokker med påfølgende svarte celler: en med lengde 3 og den andre med lengde 4. Derav, hvis $n = 10$ og $c = [3, 4]$, så er løsning som tilfredsstill hintene "XXX XXXX". Merk at "XXXX XXX" ikke tilfredsstill hintene: blokkene med svarte celler er ikke den riktige rekkefølgen. "XXXXXXXX" tilfredsstill heller ikke hintene, fordi det ikke er to separate blokker.

Du er gitt et delvist løst spill. Det vil si, du er gitt n og c , du vet også at noen celler må være svarte og noen celler må være hvite. Din oppgave er å finne mer informasjon om cellene.

Nærmere bestemt, en *gyldig løsning* er en som tilfredsstill alle hintene, og som også stemmer overens med med de kjente cellene. Programmet ditt skal finne de cellene som er malt svart i alle gyldige løsninger, og de cellene som er malt hvitt i alle gyldige løsninger. Du kan anta at inputten er slik at det finnes minst én gyldig løsning.

Implementasjonsdetaljer

Du skal implementere følgende funksjon (metode):

- `string solve_puzzle(string s, int[] c)`.
 - s : string av lengde n . For hver i ($0 \leq i \leq n - 1$) så er tegn i :
 - 'X', hvis celle i må være svart,
 - '_', hvis celle i må være hvit,
 - '.', hvis det er ingen informasjon om celle i .
 - c : array med lengde k som inneholder hint, som definert over,
 - funksjon skal returnere en streng av lengde n . For hver i ($0 \leq i \leq n - 1$) tegn i av outputt-strengen skal være:
 - 'X', hvis celle i er svart i alle gyldige løsninger,
 - '_', hvis celle i er hvit i alle gyldige løsninger,
 - '?', ellers (dvs., hvis det finnes to gyldige løsninger slik at celle i er svart i en av dem og hvit den andre).

I C er funksjonen litt annerledes:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
 - `n`: lengden av strengen `s` (antall celler),
 - `k`: lengden av arrayet `c` (antall celler),
 - de andre parametrene er som over,
 - istedenfor å returnere en streng med `n` tegn, skal funksjonen skrive svaret til strengen `result`.

ASCII-koden på følgende tegn brukt i oppgaven er:

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

Bruk de vedlagte templatfilene for detaljer om implementasjonen i ditt språk.

Eksempler

Eksempel 1

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

Dette er alle mulige løsninger på spillet:

- `"XXX_XXXX_"`,
- `"XXX__XXXX_"`,
- `"XXX___XXXX"`,
- `"_XXX_XXXX_"`,
- `"_XXX__XXXX"`,
- `"__XXX_XXXX"`.

En kan se at cellene med (0-basert) indekser 2, 6 og 7 er svarte i alle gyldige løsninger. Hver av de andre cellene kan, men behøver ikke, være svarte. Derfor er riktig svar `"??X???XX??"`.

Eksempel 2

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

I dette eksempelet er løsningen unik, og det rette svaret er `"XXX_XXXX"`.

Eksempel 3

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

I dette eksempelet kan vi utlede at også celle 4 må være hvit, det er ingen måte man kan plassere tre påfølgende svarte celler mellom de hvite cellene på indeks 3 og 5. Derav er korrekt svar `"???__????"`.

Eksempel 4

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

Det er bare to gyldige løsninger som passer med beskrivelsen over:

- "XXX _____",
- " _XXX _____".

Derav er rett svar "?XX? _____".

Subtasks

I alle subtasks $1 \leq k \leq n$, og $1 \leq c_i \leq n$ for hver $0 \leq i \leq k - 1$.

1. (7 poeng) $n \leq 20$, $k = 1$, s inneholder bare '.' (tomt spill),
2. (3 poeng) $n \leq 20$, s inneholder bare '.',
3. (22 poeng) $n \leq 100$, s inneholder bare '.',
4. (27 poeng) $n \leq 100$, s inneholder bare '.' og '_' (informasjon bare om hvite celler),
5. (21 poeng) $n \leq 100$,
6. (10 poeng) $n \leq 5\,000$, $k \leq 100$,
7. (10 poeng) $n \leq 200\,000$, $k \leq 100$.

Sample grader

Sample graderen leser input i følgende format:

- linje 1: string s ,
- linje 2: heltall k fulgt av k heltall c_0, \dots, c_{k-1} .