

## Mazes (mazes)


Hampton Court Palace (in Richmond, net ten zuidwesten van London) is beroemd voor zijn doolhof, rond 1690 aangeplant door koning William III. Omdat het paleis nu geopend is voor publiek als toeristische attractie, hebben de koninklijke autoriteiten besloten om het doolhof te vervangen door een nieuwe versie. Ze schatten in dat er  $K$  toeristen het doolhof zullen bezoeken tijdens het seizoen, en willen dat het nieuwe doolhof precies  $K$  mogelijke routes erdoorheen bevat zodat elke bezoeker een unieke ervaring kan beleven.

Het doolhof is een grid van  $N$  rijen en  $M$  kolommen, waarbij elk vakje leeg is of een heg bevat. Het doolhof is omringd door een hek, en er is een enkele ingang en een enkele uitgang. De ingang is het vakje linksboven, de uitgang het vakje rechtsonder. Een bezoeker mag in het doolhof alleen naar rechts of beneden bewegen. Vanwege de beschikbare ruimte mag het doolhof **niet meer dan 200 rijen** en **niet meer dan 200 kolommen** bevatten.

Je opdracht is om een doolhof te ontwerpen met exact  $K$  routes van de ingang naar de uitgang waarbij alleen naar rechts en omlaag bewogen mag worden.

## Implementatie

Je moet een enkel `.cpp` bestand inleveren.

 In de bijlagen van deze opgave vind je een template `mazes.cpp` met een voorbeeld-implementatie.

Je moet de volgende functie implementeren:

```
C++ | vector<vector<char>> solve(long long K);
```

- Integer  $K$  staat voor het gewenste aantal routes door het doolhof.
- De functie moet een twee-dimensionale vector van karakters teruggeven die het doolhof beschrijven.
- Het teruggegeven doolhof moet  $N$  rijen en  $M$  kolommen bevatten voor een  $N, M \leq 200$ .
- Elk vakje van het doolhof is of een `.` (punt) voor een leeg vakje of een `#` (hash) voor een vakje met een heg.
- De ingang is in het vakje  $(0, 0)$  en de uitgang in het vakje  $(N - 1, M - 1)$
- Het doolhof moet precies  $K$  verschillende paden van de ingang naar de uitgang bevatten waarbij alleen naar rechts en beneden wordt bewogen.

De grader zal de functie `solve` aanroepen en de teruggegeven waarde printen naar het uitvoerbestand.

## Voorbeeld Grader

De map van de opgave bevat een versimpelde versie van de jury's grader, die je kan gebruiken om je oplossing lokaal te testen. De versimpelde grader leest de invoer van `stdin`, roept de functies aan die je moet implementeren en schrijft uiteindelijk de uitvoer naar `stdout`.

De invoer bestaat uit 1 regel, die een enkel getal  $K$  bevat.

De uitvoer bestaat uit meerdere regels die bevatten:

- De eerste regel bevat twee integers  $N$  en  $M$  ( $N, M \leq 200$ ), het aantal rijen en het kolommen van het doolhof.
- De volgende  $N$  regels bevatten elk  $M$  karakters, die het doolhof beschrijven

- Elk karakter is of een . (dot) voor een leeg vakje of een # (hash) voor een vakje met een heg.

## Randvoorwaarden

$$1 \leq K \leq 10^{18}.$$

## Scoring

Je programma wordt getest op een set testgevallen, gegroepeerd per deelopgave. Om de score te krijgen die hoort bij een deelopgave, moet je alle testgevallen oplossen die daarbij horen.

- **Subtask 1** [ 0 punten]: Voorbeeld testgevallen.
- **Subtask 2** [ 9 punten]:  $K \leq 10$ .
- **Subtask 3** [26 punten]:  $K \leq 99$ .
- **Subtask 4** [26 punten]:  $K$  is een macht van twee.
- **Subtask 5** [39 punten]: Geen extra randvoorwaarden.

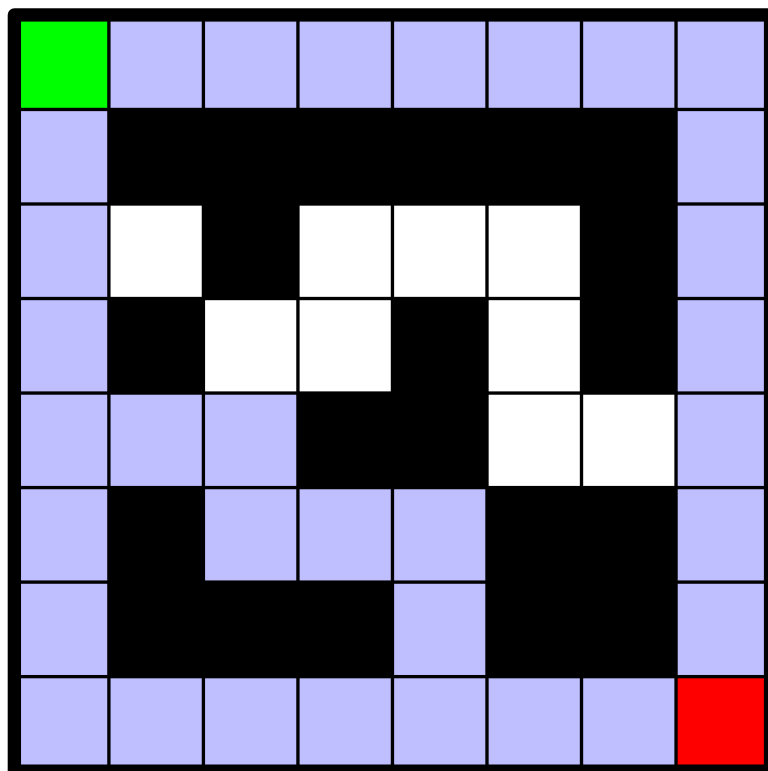
## Voorbeelden

stdin	stdout
1	2 2 .# ..
3	8 8  ..... .#####. ..#...#. .#..#.#. ...##... .#...##. .###.##. .....

## Uitleg

In het **eerste voorbeeld** is er duidelijk maar een pad door het doolhof.

In het **tweede voorbeeld** zijn er drie mogelijke paden door het doolhof. Het startvakje is groen gekleurd, het eindvakje rood en de vakjes die een pad vormen zijn blauw gekleurd.



Figuur 1: Second sample case