

MAZE RUNNER

PROGRAMMERINGSKONKURRENCE

Forfattere: Mads Okholm Bjørn, Andreas Halkjær From, Marcus Skov Hansen, Martin Hemmingsen, Thomas Søren Henney, Johan Bloch Madsen, Marcus Pagh, Tord Joakim Stordalen

1 Introduktion

Formålet med denne konkurrence er at designe og implementere et program, der kan føre robotter gennem en dynamisk labyrint. De indsendte programmer bliver sat til at dyste mod hinanden, og gruppen, der har lavet det bedste program vinder selvfølgelig en præmie! TODO: præmie?

Man kan deltage i konkurrencen i grupper bestående af op til fire studerende, og deltagelse tæller som en bestået skriftlig obligatorisk afleveringsopgave for alle gruppemedlemmer. Der skal ikke skrives nogen rapport. Flere praktiske informationer findes sidst i dette dokument. Bemærk dog følgende vigtige datoer:

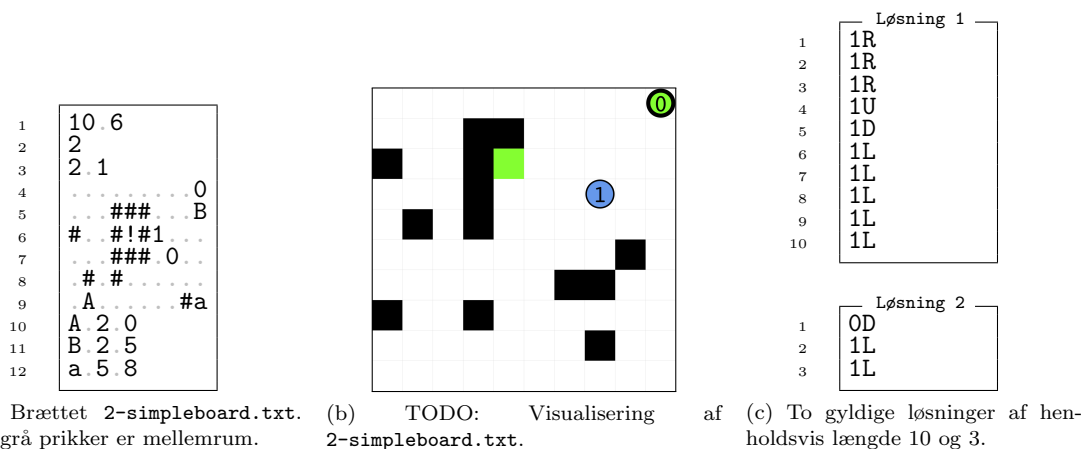
TODO Deadline for at uploade sit program til CodeJudge.

TODO Præmieoverrækkelse. Resultatet af konkurrencen præsenteres og vinderne annonceres. Sted: Bygning 116/81.

1.1 Kort om labyrinten

I Maze Runner gælder det om at give instruktioner til robotter, så de bevæger sig rundt på en forudbestemt bane, så en hvilken som helst robot når en bestemt position i labyrinten. En robot kan på et træk bevæge sig enten op, ned, til venstre eller til højre, dog ikke ind i labyrintens vægge, andre robotter eller ud af banen. Det gælder om at få en robot til at ende på målfeltet med færrest mulige samlede træk, altså inklusive træk med eventuelle andre robotter. For at opnå en løsning med få træk, er det meget ofte en fordel (og nogle gange også nødvendigt) at flytte på mere end en robot.

I labyrinten findes også to slags kontakter der tænder og slukker for stykker af labyrintens mure. Begge slags kontakter aktiveres ved at en robot stiller sig på dem. Den første slags, kaldet skift-kontakter, har fortsat effekt efter robotten flytter sig til et andet felt. Den anden slags, kaldet stå-kontakter, slukker kun for et stykke mur mens en robot står på den.



Figur 1: Eksempel på en bane af størrelse $n = 10$ med $r = 2$ robotter. De sorte felter er vægge, og det grønne felt er målfeltet for robotterne. Den optimale løsning til denne bane har længde 3.

2 Konkurrencen

I konkurrencen vil vi arbejde med en tekst-baseret repræsentation af labyrinten. Labyrinten ligger på et rektangulært gitter (kaldet banen) bestående af $n \times m$ felter, hvor $2 \leq n, m \leq 1000$. I banen findes r robotter, hvor $1 \leq r \leq 10$. Robotterne er nummeret $0, \dots, r-1$, og det gælder blot om at få en af robotterne hen på det unikke målfelt.

Figur 1 viser et eksempel på en bane med $n = 10, m = 6$ og $r = 2$ samt to mulige løsninger. I de næste to sektioner beskrives det formelle input- og outputformat samt kravene til jeres program.

2.1 Inputformat

Inputtet til jeres program er en bane som vist i Figur 1a.

Den første linje angiver henholdsvis bredden, n , og højden, m , af banen som to mellemrumsadskilte heltal mellem 2 og 1000. Den anden linje angiver antallet af robotter, r , som et heltal mellem 1 og 10. Den tredje linje angiver antallet af skift-kontakter, Λ , og stå-kontakter, λ , som to heltal mellem 0 og 26 adskilt af mellemrum.

De efterfølgende m linjer angiver banen. Hver af disse linjer består af n symboler, der repræsenterer indholdet på denne position i banen.

De lovlige symboler i en linje er:

: Angiver at positionen er en væg.

(mellemrum) : Angiver at positionen er tom.

! : Angiver målpositionen.

0, 1, ..., $(r-1)$: Angiver, at robotten med dette tal er på denne position.

A, B, ..., Z : Angiver skift-kontakter.

a, b, ..., z : Angiver stå-kontakter.

Det garanteres, at symbolet !, robotterne $0, 1, \dots, (r-1)$ og kontakterne $a-z$ og $A-Z$ optræder på præcis én position i banen. Det garanteres desuden at banen kan løses.

De resterende $\Lambda + \lambda$ linjer består af navnet på en kontakt og dernæst koordinaterne for det stykke mur, som kontakten påvirker, se figur 1a. Alle skift-kontakter angives først, dernæst alle stå-kontakter. Koordinaterne er givet som 0-indekserede række-søjle par, så eksempelvis har robot 1 på figur 1a koordinaterne 2, 6.

2.2 Outputformat

En gyldig løsning til banen er en sekvens af træk, der bringer en robot hen på målpositionen. Hvert træk udskrives som en linje bestående af præcis to symboler: Et heltal i der angiver robotten, hvor $0 \leq i \leq r-1$, efterfulgt (uden mellemrum) af et af symbolerne R (højre), L (venstre), U (op) eller D (ned) der angiver retningen. F.eks. angiver 0R at robot 0 flyttes mod højre. Et eksempel på to gyldige løsninger er vist i Figur 1c.

2.3 Krav til jeres program

Jeres program skal indlæse en bane fra konsollen (`stdin`), beregne en gyldig løsning og udskrive denne løsning til konsollen (`stdout`). Det er ikke et krav at jeres program finder den korteste løsning, men jo kortere løsningen er, desto flere points får jeres program.

3 Sådan kommer du igang – og anden praktisk information

Start med at hente filen `mr.zip`, der ligger på DTU Inside under fildeling. Filen indeholder en skabelon i hhv. Java og Python med forskellige brugbare metoder til indlæsning af banefiler m.m. Prøv at uploade filen `MazeRunner.java` til CodeJudge – så burde den klare den første test.

Det er ikke et krav at tage udgangspunkt i disse skabeloner – I må også meget gerne skrive jeres løsning i andre sprog, der er understøttede på CodeJudge.

Filen `mr.zip` indeholder desuden en række forskellige baner, som I kan teste jeres løsning på. Alle banefilnavne starter med et tal, der angiver, hvor mange robotter, der er i banen.

Herunder følger yderligere relevant information.

Piazza Brug Piazza til alle spørgsmål, der ikke er besvaret her.

Gruppedannelse Husk at danne gruppe på CodeJudge. Maksimum er fire studerende per gruppe. Aflevering foregår ved at uploade sin løsning til CodeJudge.

CodeJudge Konkurrencen foregår på CodeJudge og er opdelt i to delopgaver: *Obligatorisk* og *Konkurrence*. Den obligatoriske del består af forholdsvis nemme baner med en enkelt robot. Start med at lave en løsning til denne del. Konkurrencedelen indeholder større og mere udfordrende baner med op til 10 robotter.

Den obligatoriske del Uploades en løsning, der består alle tests (5 synlige og 5 skjulte) i den obligatoriske del, tæller dette som en bestået skriftlig obligatorisk afleveringsopgave for alle medlemmer af gruppen.

Konkurrencedelen Efter deadline tager vi den nyeste version af jeres program i konkurrencedelen og lader det dyste mod de andre programmer på en ukendt samling af baner.

Tids- og hukommelsesgrænser På CodeJudge bliver jeres program tildelt T sekunders CPU-tid og S MB hukommelse. I kan læse værdierne T og S som de første to kommandolinjeargumenter til jeres program. Som udgangspunkt kan I regne med at $T = 5$ sekunder og $S = 500\text{MB}$, men når den endelige konkurrence afvikles efter deadline, bliver jeres program tildelt en del flere ressourcer.

Pointtildeling For hver bane i konkurrencedelen som jeres program løser får det $100 \frac{E}{U}$ points, hvor U er længden af jeres løsning, og E er et estimat af den bedst mulige løsning for banen.

Scoreboard På Scoreboardet på CodeJudge kan man løbende følge med i, hvordan ens løsning klarer sig i forhold til andre gruppers løsninger. Grupperne er sorteret efter det totale antal points deres løsning har opnået på konkurrencebanerne. I tilfælde af point-lighed er den bedste gruppe den med det korteste tidsforbrug. Bemærk, at vi til den endelige konkurrence også tester jeres program på nye og ukendte baner, så derfor er det ikke sikkert, at den endelige vinder bliver den gruppe, der ligger øverst på Scoreboardet.

Webapp På hjemmesiden <https://319.dk/MR/> findes en app, hvor man kan importere/eksportere baner og løsninger – og selv prøve at spille banen vha. tastaturet. Man kan også komme direkte til denne app ved at klikke på visualiseringen af en bane i CodeJudge. Appen er kun testet i Chrome og Safari, men burde også virke i andre browsere og på tablets og touchenheder. Bugs kan mailes til Mads Bjørn.

Nogle gode råd Start med at lave en løsning, der virker for helt små baner, hvor der kun er en robot. Overvej, hvordan du kan bruge algoritmiske redskaber fra kurset som fx bredde-først-søgning og hashing til at søge efter en løsning.