Podatkovne strukture in algoritmi 1
Fakulteta za Matematiko in Fiziko
5.12.2023

Domača naloga 2 - Skladišče

1 Opis

V večjem distribucijskem podjetju si želijo moderniziratisvoja skladišča. Njihova skladišča so velika, ter zaposleni potrebujejo veliko časa, da v skladišču najdejo primeren izdelek. V podjetju so se odločili, da bodo iskanja izdelkov znotraj skladišča modernizirali z uporabo avtonomnih robotov. Naloga robota bo, da v skladišču najde določen izdelek, ter ga dostavi na izhodna vrata skladišča. Vsako skladišče je kvadratne oblike s stranicami velikosti 100 (dolžinska enota tukaj ni pomembna). V skladišču se nahajajo police pravokotne oblike z izdelki. Vsaka polica P_i je predstavljena kot 4-terica števil $(x_1^i, y_1^i), (x_2^i, y_2^i)$, ki predstavljajo levo spodnje ter desno zgornje ogljišče pravokotnika. Skladišče vsebuje L polic, ki se med seboj ne prekrivajo, med njimi pa je vedno dovolj prostora, da robot lahko gre mimo (robota si predstavljamo kot eno točko, ki torej ne zaseda nobene ploščine). Robot se torej lahko premika po stranicah polic. Poleg tega so roboti izjemno napredni in se v izrednih primerih lahko »tunelirajo« čez neko polico. Pri tem porabijo sicer ogromno energije zato je število tuneliranj omejeno. Vsako skladišče ima vhodna vrata na koordinatah $(0, 0)$ ter izhodna vrata na koordinatah $(100, 100)$. Naloga robota je, da svojo pot začne v začetnih vratih, prevzame nek izdelek na določeni polici (izdelek lahko prevzame le na določenem ogljišču police) ter ga dostavi do izhodnih vrat. Vaša naloga bo, da izračunate najkrajšo možno pot, ki jo mora robot opravit od začetnih vrat, do določenega izdelka ter nato do izhodnih vrat.

2 Vhodni podatki

V prvi vrstici se nahajajo tri števila n, p, k. Nato sledi n vrstic vsaka pa je oblike $x_1^i, y_1^i, x_2^i, y_2^i$ ter predstavlja opis police P_i. Za tem sledi prazna vrstica in nato še p vrstic z poizvedbami oblike x_i, y_i. To predstavlja izdelke, ki jih mora robot prinesti do izhodnih vrat.

Omejitve vhodnih podatkov:

- $n \leq N$
- $p \leq 4N$
- $N \leq 40$
- $k \leq 5$
- vse koordinate so decimalna števila, zaokrožena na dve decimalni mesti.
- vsaka izmed p vrstic x_i, y_i predstavlja eno izmed ogljišč neke police.

3 Izhodni podatki

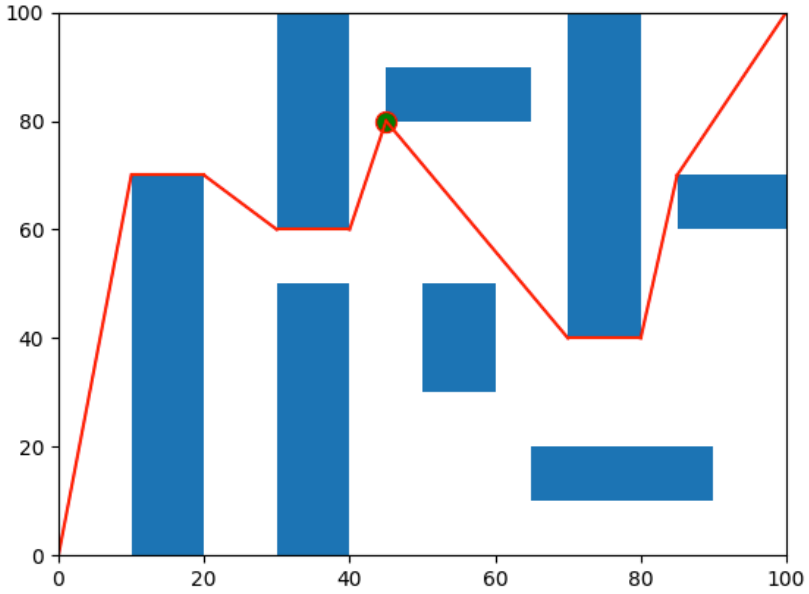
Izpišite p vrstic, eno za vsako poizvedbo, ki predstavlja najkrajšo možno pot, ki jo mora robot prepotovati iz $(0, 0)$ do (x_i, y_i) ter nato do $(100, 100)$, če lahko opravi največ k tuneliranj. Dolžina poti je evklidska razdalja.

Časovna zahtevnost: Vaš algoritem mora delovati v $O((k + 1)N^3)$.

4 Primer 1

Vhod	Izhod
`8,4,0`	`246.59307984474498`
`10,0,20,70`	`227.67114123010037`
`30,0,40,50`	`255.31046145510248`
`30,60,40,100`	`214.5725582817884`
`45,80,65,90`	
`50,30,60,50`	
`70,40,80,100`	
`65,10,90,20`	
`85,60,100,70`	
`45,80`	
`50,30`	
`65,10`	
`80,40`	

V tem primeru, se robot ne more tunelirati čez police. Za prvo poizvedbo torej iščemo najkrajšo pot $(0, 0) \to (45, 80) \to (100, 100)$, ki ne seka nobene police.



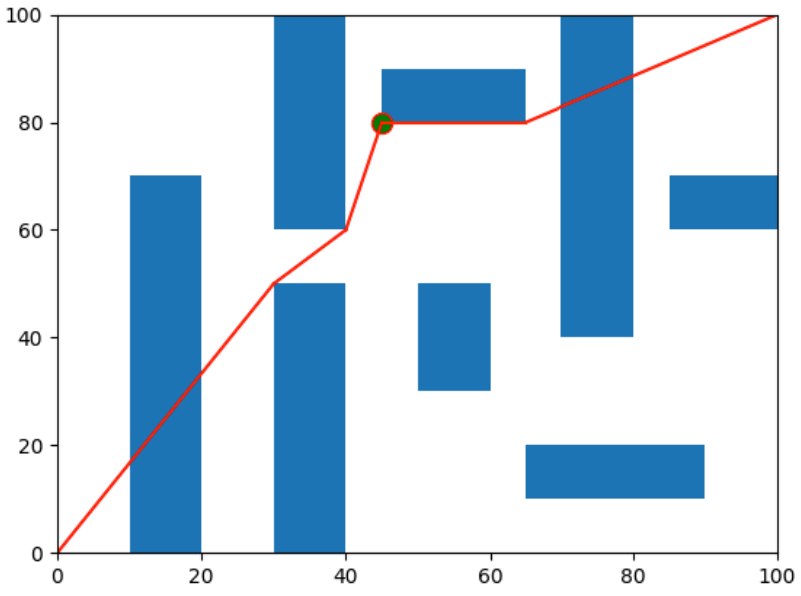
Shematični prikaz primera.

Na zgornji sliki je prikazano skladišče z policami ter pot za prvo poizvedbo.

5 Primer 2

Vhod	Izhod
8,4,2	153.378471441765
10,0,20,70	154.62503103743884
30,0,40,50	164.71956450381748
30,60,40,100	153.39755141397953
45,80,65,90	
50,30,60,50	
70,40,80,100	
65,10,90,20	
85,60,100,70	
45,80	
50,30	
65,10	
80,40	

V tem primeru, se robot na svoji poti lahko tunelira čez največ dve polici. To mu omogoča, da isto poizvedbo opravi po krajši pot. Za isto poizvedbo kot prej opazimo, da robot



Shematični prikaz primera z tuneliranje.

lahko opravi precej krajšo pot. V tem primeru je eno tuneliranje porabil preden je pobral izdelek, nato pa še eno ko je izdelek prenesel do izhodnih vrat. Ni pa nujno, da bo vedno tako. Lahko bi recimo porabil vse tuneliranje preden pobere izdelek (ali pa potem). Kot rezultat je potrebno najti najbol ugodno opcijo.

6 Točkovanje

Testni primeri so sestavljeni, tako da ima prvih 10 testnih primerov paramter k vedno enak 0. Torej tuneliranja v tem primeru ni. Če bodo te testi uspešno opravljeni (ter ustrezali časovni zahtevnosti) boste dobili največ 7 točk. Za preostale 3 točke morate rešiti še preostale testne primere, ki pa imajo parameter $k > 0$. Testni primer vam bo rešitev štel kot pravilno, če se od pravilne rešitve ne razlikuje za več kot 10^{-5}.

7 Dodatni napotki

Med reševanjem naloge se boste verjetno srečali z kakšnim preprostim geometrijskim problemom, morda pa tudi z kakšnim težjim. Časovna zahtevnost bi morala biti dovolj "splošna", da ne boste rabili razviti kakšnih kompleksnih podatkovnih struktur. Vseeno pa pri lažjih problemih (recimo ali se dve daljici sekata) probajte kodo spisati čim bolj razumljivo. Če boste imeli kakšna vprašanja oziroma težave pa me seveda lahko vprašate.

8 Testni primeri

Na voljo imate več testnih primerov ter pripadajočih izhodov. Vsak testni primer ima vhodne podatke v datoteki `testXY.in` ter izhodne podatke v `testXY.out`. Testni primeri se nahajajo v mapi `Tests`.
S pomočjo priložene skripte `test_runner.py` lahko zaženete enega, ali pa vse testne primere. Datoteko `resitev.py`, kjer ste rešili nalogo prestavite v isto mapo, kjer se nahaja `test_runner.py` ter mapa `Tests`. Skripta `test_runner.py` ima 3 možne argumente:

- `-script` (obvezen): ime oziroma pot do vaše datoteke z rešitvijo
- `-test_nb` (neobvezen): številka testa, ki naj se izvede. Če ni podano se izvedejo vsi testi.
- `-pypath` (neobvezen): Pot do Python interpreterja oz. okrajšava. Če ni podano, se vzame `python3`.

Opozorilo: Na različnih sistemih ima Python lahko različne okrajšave. Na linux in MacOS operacijskih sistemih Python ponavadi zaganjamo z `python3`, medtem ko na Windows operacijskih sistemih ponavadi z `python` (v kolikor je python dodan v okoljsko spremenljivko). V kolikor boste imeli kakšne težave me kontaktirajte.

Odprite terminal v mapi, kjer imate zgornje datoteke. Nato izvedite ukaz:

```
> python3 test_runner.py -script resitev.py -test_nb 0
```

Ob uspešno izvedenem testu se izpiše:

```
> Test test00.in uspešen.  Čas:  0.024 sekunde.
```