DeliveryPlusPlus

Opšte o samom projektu

Sama srž projekta jeste jedan multigraf gde gradovi predstavljaju čvorove, a putevi veze između tih gradova, tj. čvorova. Putu se može dodeliti naziv, dužina, kao i tip. Trenutno je moguće da put bude kopnenog, vazdušnog, vodenog tipa ili da bude šina. U zavisnosti od njegovog tipa, tim putem mogu da idu samo ona vozila koja imaju odgovorajuć tip vožnje. Pošto je u pitanju multigraf, između svakih gradova moguć je neograničen broj veza, nebitno da li već postoji put tog tipa. Što se tiče gradova, svaki grad pripada nekoj državi i to može pripadati samo jednoj državi. Pri kreiranju grada, navodi se kojoj državi taj grad pripada. Svaka država zna koji su njeni gradovi, kom kontinentu pripada, kao i da izlista informacije o njima.

Kretanje kroz ovaj graf je obezbeđeno pomoću Dajkstrinog algoritma za nalaženje najkraćeg puta u grafu. Znači, možemo naprimer, za grad Kragujevac da nađemo najoptimalniji put ka recimo gradu Berlinu. Uz to, možemo izfiltrirati puteve kroz koje smemo da se krećemo, tako što ćemo eksplicitno navesti kog tipa su ti putevi. Na primer, ako kažemo da želimo da nađemo najoptimalniji put od Kragujevca do Berlina kretući se isključivo putevima kopnenog tipa, algoritam će da nam vrati najoptimalniji put, što je autoput (u ovom slučaju preko autoputa – što i nije baš najbrži put). Ako ne navedemo ni jedan tip puta, onda će se gledati kao da može svaka vrsta puta da se koristi, i sada bi nam za isti primer algoritam rekao da treba da odemo do Beograda, i onda od Beograda avionom do Berlina (jeste najoptimalniji put – najkraće je rastojanje). Dok na primer, ako pokušamo da nađemo isključivo kopneni put od Kragujevca do Njujorka, nećemo biti uspešni.

Što se tiče paketa, svaki paket ima svoje naziv, svoju težinu, grad odakle se šalje i grad koji mu predstavlja destinaciju. Što se tiče vozila, vozila su lako proširiva i može se dodati bilo koji tip vozila, samo da njegov tip vožnje odgovara nekom od već navedenih tipova puta. Pri kreiranju vozila, navodi se koliki je maksimalni kapacitet koji može da prenese, cena po kilometru i u kom gradu se na početku nalazi.

Sam prevoz paketa je jako problematičan problem koji sam rešio tako što se ide paket po paket, stvara se optimalna putanja od izvorišta paketa do njegovog odredišta, i onda se proverava da li uopšte mi imamo odgovarajuća vozila koja mogu da pristupe tom optimalnom putu. Primera radi, zamislimo da sva vozila koja mogu voziti po autoputu se nalaze u Evropi, a ni jedno takvo vozilo se nalazi u Americi, ali zato Amerika ima dosta vozeva, i dosta razvijeni železnički sistem, i neka je Njujork jedini način da se stigne iz Evrope u Ameriku. Najoptimalniji put nam kaže da kada paket stigne u Njujork najbolje bi bilo da autoputem odemo do nekog tamo grada koji nam je odredište, međutim problem nastaje pošto mi nemamo ni jedno vozilo koje je povezano sa tim putem, pa zato je ovaj najoptimalniji put zapravo nemoguć za nas. Zato, isključićemo taj deo puta koji je nedostupan, i probaćemo da generišemo najoptimalniji put još jednom. Sada, algoritam nam kaže da kada stignemo u Njujork, najoptimalnije bi bilo da vozom pošaljemo taj paket u našu destinaciju, što nam i odgovara, jer imamo voz koji se nalazi u Americi koji može da dođe do Njujorka. E sada, da je naprimer Njujork jedino bio povezan sa drugim gradovima preko autoputa, ovaj paket bi bio nemoguć za isporuku i preskočili bi ga.

Ako je sve sa optimalnim putem ok, onda će se ići grad po grad, i za svaki put će se naći jedno vozilo koje bi pružilo najbolje usluge transporta paketa (bira se prvo vozilo čiji je maksimalni kapacitet veći ili jednak težini paketa). Ako nema ni jedno tako vozilo, izabraće se dostupno vozilo sa najvećim kapacitetom, i paket će se prevesti u sledeći grad iz nekoliko tura. Cena prevoza se uvećava

za pređene kilometra, ali samo kada se ceo paket ili neki njegov deo nalazi unutar vozila. Kada je vozila prazno (npr. Vraća se po ostatak paketa ako ga prevozi iz nekoliko tura), ti pređeni kilometri neće se naplatiti. Tako se ide put po put, dok paket ne stigne do svoje destinacije.

Tehnička strana projekta

class Entity

Abstraktna klasa koju će mnoge druge klase u ovom projektu nasleđivati. Poenta ove klase jeste da se obezbedi da sve bitne klase imaju auto generisani id koji će uvek biti različit, da imaju ime i da moraju obezbediti opis samih sebe preko virtualne metode ***getInfo***.

class Country : public Entity

Ovo je klasa koja predstavlja države, njen konstruktor je privatni jer je njeno instaciranje obezbeđeno preko statičke metode ***createCountry*** koja čuva pokazivač na tu državu u privatnom, statičkom rečniku ***countries*** gde je ključ njen id koji je obezbeđen preko Entity klase koju nasleđuje. Taj id se vraća korisniku i on posle može da ga pristupi bi pristupio ovoj državi preko statičke metode ***getCountryById***. Sve države se mogu izbrisati iz memorije preko statičke metode ***clearCountries***.

class City : public Entity

Ova klasa predstavlja gradove, i može se reći da je jedna od ključnih u ovom projektu. Kao i kod klase Country, konstruktor ove klase je privatan i njeno instanciranje jeste obezbeđeno preko metode ***createCity***, čuva se u rečniku i pristupa se preko statičke metode gde se prosledi id tog grada. Preko statičke metode ***connectTwoCities*** dva grada se mogu povezati tako što se metodi proslede id-ovi dva grada koja želite da povežete, naziv puta, rastojanje puta kao i tip tog puta. Nakon toga, ta metoda ima da pozove privatnu metodu za oba grada koja se naziva ***addConnection***, koja će smestiti taj put u vektor puteva, koji će biti smešten u rečniku nazvan ***connections*** gde će ključ biti id grada sa kojim je povezan. Svi gradovi se mogu izbrisati iz memorije preko statičke metode ***clearCities***.

class Path : public Entity

Ova klasa predstavlja put, tj. vezu između neka dva grada tj. čvora. Ona ne sadrži nikakve informacije o gradovima koja ona povezuje, i kao sa ostalim klasama do sad, njen konstruktor je privatan i kreiraju se preko statičke metode, takođe se čuvaju u rečniku. Svi putevi se mogu izbrisati iz memorije preko statičke metode ***clearPaths***.

class PathSolver

Ova klasa sadrži veliki deo logike ovog programa, tj u njoj se nalazi Dajkstrin algoritam. Konstruktor ove klase zahteva da se prosledi id početnog grada, kao i nesortirani skup koji će sadržiti sve tipove puta kroz koje ćemo moći da se krećemo. Ako je taj skup prazan, onda ćemo moći da se krećemo kroz sve tipove puta. Sam algoritam kreće da radi u konstruktoru, i ako postoji, naćiće najkraću putanju do svih gradova od početnog grada. Metoda ***getPathTo*** će u tekstualnoj formi da opiše putovanje od početnog grada do prosledjenog id-a grada, ispisaće kroz koje puteve i gradove smo sve prošli, kao i koliko kilometra smo prošli. Metoda ***isCityReachable*** za prosleđeni grad proverava da li je moguće da se uopšte stigne do prosleđenog grada.

class UnexpectedBehavior : public exception

Klasa koja predstavlja generičku grešku kojoj se može proslediti tekst i dati detaljno objašnjenje o grešci koja se desila.

class Package : public Entity

Klasa predstavlja paket koji se mora prevesti iz grada X u grada Y. Pri instanciranju navode se ti id-ovi tih gradova, naziv paketa, težina paketa.

class Vehicle : public Entity

U ovoj klasi se nalazi drugi deo logike ovog programa. Njen konstruktor je protected, iz razloga zato što želimo da se instancira preko izvedenih vozila, jer preko izvedenih vozila mi možemo navesti kog tipa je to vozilo u konstruktoru i za koji tip puta je to vozilo namenjemo (npr klasa avion će konstruktoru vozilo proslediti da je namenjen za vazdušni tip puta i da je tipa Letelica). Navodi se cena po kilometru, maksimalni kapacitet, i u kom gradu se nalazi to vozilo. Ako npr kažemo da se neki brod nalazi u Kragujevcu, desiće se greška zato što Kragujevac nije povezan ni sa jednim putem koji je vodenog tipa (što je i logično). Preko statičke metode ***deliverPackages*** se započinje proces transporta paketa, tako što se metodi prosledi vektor svih paketa koji treba da se prenese kao i vektor naših raspoloživih vozila. Ide se paket po paket, i koristeći klasu PathSolver se rešava problem najoptimalnijeg puta u skladu sa raspoloživim vozilima, ako se nađe put koji je ok i preko kojeg je moguć prevoz koristeći dostupna vozila, taj paket i vektor svih vozila se prosleđuje privatnoj metodi ***startShipping*** koja će se baviti transportom iz grada u grad dok ne stigne u svoje odredište, kao i računanje cene transporta.