Univerzitet u Banjoj Luci

Prirodno-matematički fakultet

Informatika

Napredni koncepti baza podataka

Neo4j u rješavanju problema ograničenja

Profesor

Dr Marko Đukanović

Student

Teodora Milanović 22/19

Sadržaj

Opis problema

Problemi ograničenja su klasa problema čija rješenja zadovoljavaju sva data ograničenja. Tačnije, potrebno je svakoj promjenljivoj dodijeliti vrijednost iz njenog domena, tako da su sva ograničenja zadovoljena.

Formalno, problem ograničenja se sastoji iz:

- skupa promjenljivih $X = \{X_1, X_2, \ldots, X_n\}$
- skupa domena datih promjenljivih $D = \{D_1, D_2, \ldots, D_n\}$
- skupa ograničenja $C = \{C_1, C_2, \ldots, C_n\}$

Problem koji ćemo riješiti u nastavku jeste problem mosta i baklje. Cilj ovog problema jeste pronalazak najkraćeg vremena koje je potrebno da n ljudi pređe most uzimajući u obzir da najviše dvoje ljudi može da pređe most zajedno. Svaka osoba se kreće određenom brzinom i potreban joj je određen vremenski period da pređe most, te se dvoje ljudi prilikom prelaska kreću brzinom one osobe koja je sporija. S obzirom da je noć, prilikom prelaženja je potrebno nositi baklju. Međutim, na raspolaganju je samo jedna baklja koja se ne može baciti. Dodatno ograničenje jeste maksimalno vrijeme za koje svih n osoba mora preći most prije nego što baklja ne izgori.

Dakle, postoje naredna ograničenja:

- Najviše dvoje ljudi može da pređe most zajedno.
- Dvoje ljudi se prilikom prelaska kreću brzinom one osobe koja je sporija.
- Na raspolaganju je samo jedna baklja koja se mora nositi prilikom svakog prelaska.
- Postoji vrijeme za koje most mora da se pređe prije nego što baklja izgori.

Na osnovu prethodnih ograničenja slijedi da postoje naredna stanja:

- Početno stanje gdje se sve osobe nalaze na jednoj strani mosta.
- Niz prelaznih stanja koja nam daju informacije o položaju svake osobe.
- Krajnje stanje gdje su sve osobe prešle na drugu stranu.

Opis algoritma

Da bismo ovaj problem mogli riješiti pomoću Neo4j-a potrebno je da prvobitno napravimo grafovsku bazu. Čvorovi će predstavljati sva moguća stanja, a prelaske u stanja ćemo predstaviti vezama. Početni čvor predstavlja stanje u kom se sve osobe nalaze na jednoj strani mosta. Osobe pripadaju skupu $P = \{P_1, P_2, ..., P_n\}$, te se pomoću donje crte označava most. U svakom čvoru se nalazi informacija o položaju svake osobe u odnosu na most.
Promjene stanja se modeluju pomoću veza i novih stanja. Tačnije, svaka veza sadrži informacije o prelasku kao što su osobe koje prelaze most i vrijeme koje im je potrebno da pređu most, koje u ovom slučaju odgovara vremenu koje je potrebno sporijoj osobi da pređe most. Slijedi da novo stanje sadrži informaciju o položaju ljudi u odnosu na most nakon novog prelaska.

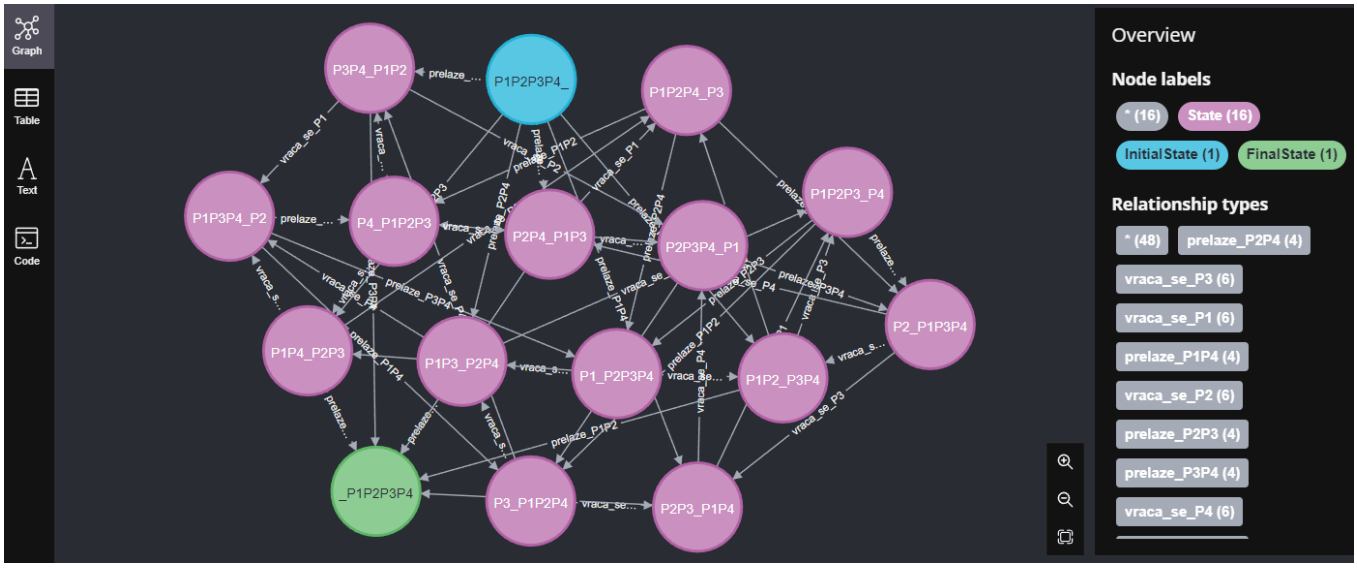
graf - Notepad

File Edit View

```
MERGE (:InitialState:State{attribute:"P1P2P3P4_"})
MERGE (:State{attribute:"P3P4_P1P2"});
MATCH (s1:State{attribute:"P1P2P3P4_"}), (s2:State{attribute:"P3P4_P1P2"})
MERGE (s1)-[:prelaze_P1P2{attribute:2}]->(s2);
MERGE (:State{attribute:"P1P3P4_P2"});
MATCH (s1:State{attribute:"P3P4_P1P2"}), (s2:State{attribute:"P1P3P4_P2"})
MERGE (s1)-[:vraca_se_P1{attribute:1}]->(s2);
MERGE (:State{attribute:"P4_P1P2P3"});
MATCH (s1:State{attribute:"P1P3P4_P2"}), (s2:State{attribute:"P4_P1P2P3"})
MERGE (s1)-[:prelaze_P1P3{attribute:5}]->(s2);
MERGE (:State{attribute:"P1P4_P2P3"});
MATCH (s1:State{attribute:"P4_P1P2P3"}), (s2:State{attribute:"P1P4_P2P3"})
MERGE (s1)-[:vraca_se_P1{attribute:1}]->(s2);
MERGE (:FinalState:State{attribute:"_P1P2P3P4"});
MATCH (s1:State{attribute:"P1P4_P2P3"}), (s2:State{attribute:"_P1P2P3P4"})
MERGE (s1)-[:prelaze_P1P4{attribute:10}]->(s2);
MERGE (:State{attribute:"P2P4_P1P3"});
MATCH (s1:State{attribute:"P4_P1P2P3"}), (s2:State{attribute:"P2P4_P1P3"})
MERGE (s1)-[:vraca_se_P2{attribute:2}]->(s2);
MATCH (s1:State{attribute:"P2P4_P1P3"}), (s2:State{attribute:"_P1P2P3P4"})
MERGE (s1)-[:prelaze_P2P4{attribute:10}]->(s2);
MATCH (s1:State{attribute:"P4_P1P2P3"}), (s2:State{attribute:"P3P4_P1P2"})
MERGE (s1)-[:vraca_se_P3{attribute:5}]->(s2);
MATCH (s1:State{attribute:"P3P4_P1P2"}), (s2:State{attribute:"_P1P2P3P4"})
MERGE (s1)-[:prelaze_P3P4{attribute:10}]->(s2);
MERGE (:State{attribute:"P3_P1P2P4"});
```

Ln 92, Col 39

Slika 1. Primjer skripte za kreiranje grafovske baze



Slika 2. Primjer grafovske baze u Neo4j-u

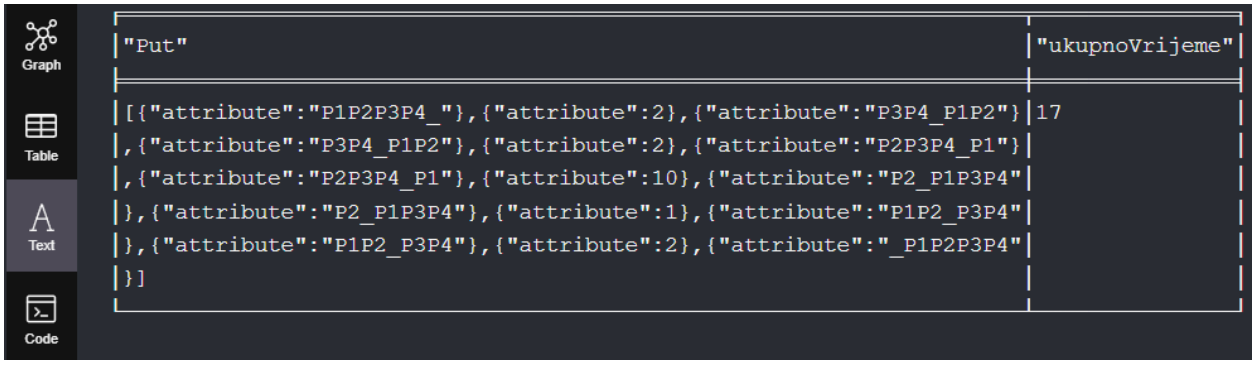
Kako bismo pronašli puteve u grafu koji zadovoljavaju sva ograničenja potrebno je da implementiramo uskladištenu proceduru koristeći Javin Neo4j API. Algoritam koji pronalazi puteve koji zadovoljavaju sva ograničenja funkcioniše na sljedeći način:

1. Početni čvor je ujedno i onaj od kog kreće obilazak grafa.
2. Vrši se posjećivanje svih susjednih čvorova, te se putevi proširuju u dubinu.
3. Provjerava se da li se uključivanjem datog susjednog čvora u put krše ograničenja tj. da li je nošena baklja prilikom prelaska i da li se nakon prelaska prekoračilo vrijeme koje je dostupno prije nego što baklja izgori.
4. U slučaju da čvor zadovoljava ograničenja, dati čvor se uključuje u put, te se nastavlja obilazak grafa.
5. U slučaju da čvor ne zadovoljava ograničenja, dati čvor se isključuje i obilazak grafa se ne nastavlja.
6. Konačno, iterira se kroz kolekciju puteva koji zadovoljavaju sva ograničenja, te ako je krajnji čvor puta završni, dati put se vraća kao jedan od rezultata.

Posljednji korak jeste poziv procedure koju smo implementirali tako što joj kao parametre proslijedimo broj osoba i vremensko ograničenje koje predstavlja trajanje baklje. Takođe, vrši se manipulacija rezultatom kako bi se prikazao najbolji put iliti put kojeg karakteriše najkraće vrijeme potrebno da n osoba pređe most.

```
1 CALL constraint.findValidPaths(4, 17) YIELD path
2 WITH path, relationships(path) AS rels
3 WITH path, rels, reduce(acc = 0, r IN rels | acc + r.attribute) AS ukupnoVrijeme
4 RETURN path as Put, ukupnoVrijeme
5 ORDER BY ukupnoVrijeme
6 LIMIT 1
```

Slika 3. Primjer upita



Slika 4. Primjer rezultata izvršavanja upita

Opis instanci i specifikacije računara

Za potrebe upoređivanja korištene su tri instance. Instance sadrže informacije o broju osoba koje treba da pređu most, trajanju baklje i vremenu koje je potrebno svakoj od tih osoba da pređu most.

	Broj osoba	Trajanje baklje (min)	Lista vremena (min)
1.	4	17	[1, 2, 5, 10]
2.	5	35	[1, 2, 5, 10, 20]
3.	6	54	[1, 2, 5, 10, 17, 30]

Specifikacije:

- Procesor: AMD Ryzen 7 5825U,
- RAM: 16 GB,
- Operativni sistem: Windows 11 Pro.

Upoređivanje sa drugim algoritmima

Za upoređivanje rezultata su izabrane tehnike pohlepnog pristupa i dinamičkog programiranja, a njihovi algoritmi koji rješavaju problem mosta i baklje su implementirani u Javi. U nastavku se nalaze rezultati izvršavanja.

Broj instance	Vrijeme izvršavanja	Rezultat
Pohlepni pristup		
1	<1ms	17
2	<1ms	33
3	<1ms	52
Dinamičko programiranje		
1	5ms	17
2	5ms	33
3	6ms	52
Neo4j		
1	14ms	17
2	86ms	33
3	525ms	52

Iz prethodne tabele se može zaključiti da svi algoritmi vraćaju ista, optimalna, rješenja. Međutim, rješavanje problema ograničenja pomoću Neo4j-a je najmanje efikasan pristup, pogotovo na većim instancama. I pohlepni algoritam i algoritam dinamičkog programiranja su brže došli do rješenja u 100% slučajeva. Pohlepni algoritam se na datim instancama izvršava 14 do 525 puta brže, a algoritam dinamičkog programiranja 5 do 6 puta brže. Neka n označava broj ljudi. Vremenska složenost pohlepnog algoritma je O(n). Što se tiče, algoritma dinamičkog programiranja vremenska složenost je $O(n^2)$. Algoritam za obilazak stabla ima eksponencijalnu vremensku složenost $O(2^n)$, koja je ujedno i najgora.

Zaključak

Neo4j u rješavanju problema ograničenja ima izrazito loše performanse. Posmatrajući problem mosta i baklje, broj čvorova koji je potreban za kreiranje baze eksponencijalno raste i iznosi 2^n, pri čemu n označava broj ljudi. Slijedi da se sa svakim povećanjem veličine instance drastično mijenja veličina baze i količina resursa koja je potrebna za njeno skladištenje. Iz tog razloga i minimalno povećanje veličine instance dovodi do velikog gubitka performansi. Samim tim, drugi pristupi su se pokazali kao dosta bolja opcija pri rješavanju problema ograničenja, pogotovo na većim instancama.

Literatura

[1] Anon., *Solving Constraint Problems using Neo4j.*
Available at: https://medium.com/@dhananjay.ghanwat/solving-constraint-problems-using-neo4j-88bc3e456bac

[2] Rote, G., *Crossing the Bridge at Night.*
Available at: http://page.mi.fu-berlin.de/Rote/Papers/pdf/Crossing+the+bridge+at+night.pdf

[3] Wikipedia, *Bridge and torch problem.*
Available at: https://en.wikipedia.org/wiki/Bridge_and_torch_problem