Prirodno-matematički fakultet
Odsjek za matematiku
Teorijska kompjuterska nauka
Napredni algoritmi i strukture podataka

Dokumentacija za projekat 1

Implementacija Greedy algoritma najveće uštede

KASUMOVIĆ MUAZ

Sarajevo, maj 2019.

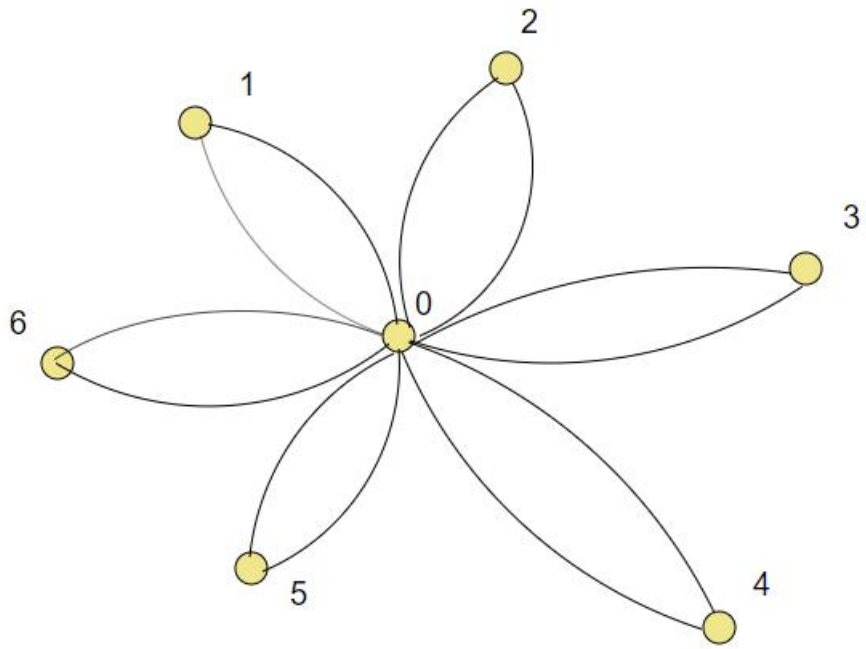
Sadržaj

Uvod

U ovom radu će biti predstavljeni detalji implementacije sljedećeg greedy algoritma: Izabere se grad 0, i napravi se tura na način $0-1-0-2-0-3-0-...$ (ovo nije validna tura u smislu TSP rješenja jer se ne posjećuje svaki grad jednom). Dakle, početna tura ide do svakog grada i odmah se vraća do početnog. Tada se pravi greedy izbor: izabere se par gradova (i, j), $i, j \neq 0$ za koji se dobija najveća ušteda ako se ide od grada i do grada j (umjesto da se vraća do početnog grada). Ovaj postupak se ponavlja dok se ne sastavi validna tura za TSP problem.
Na kraju rada je dat prikaz rezultata kao i poređenja sa druga dva greedy algoritma koji su obrađeni na vježbama (koji traže najmanju granu i najbližeg susjeda).

Opis algoritma

Na početku se formira $n-1$ tura koje imaju dvije grane, od $(0, i)$ i $(i, 0)$ za $i = 1...n$,kao što je prikazano na slici 1. Nakon što je formirano navedenih $n-1$ tura izračuna se ušteda za sve parove



Slika 1: Početnih n-1 tura

gradova (i, j), $i, j \neq 0$ na sljedeći način

$$ušteda(i, j) = d(0, i) + d(j, 0) - d(i, j) \tag{1}$$

pri čemu je d matrica udaljenosti između gradova($d(i, j)$ predstavlja udaljenost od grada i do grada j). Parovi gradova se sortiraju (grane grafa) prema uštedi u opadajućem poretku. Sada se

uzimaju grane iz sortirane liste redom tako da nema ciklusa u grafu osim onih ciklusa koji su inicijalno napravljeni. Dakle, dodaje se grana (i, j), a brišu se grane $(0, i)$ i $(j, 0)$. Kada se svaki čvor posjeti algoritam terminira (kada dobijemo validnu turu) i vraća turu. Pseudokod algoritma je dat listingom ispod.

Algorithm 1: Greedy algoritam najveće uštede

```
Input : Matrica udaljenosti dimenzija n * n: d
Output : Dužina ture
for i ← 1 to n do
    for j ← i + 1 to n do
        ustede.dodaj(i,j,d[0][i] + d[j][0] - d[i][j])
    end
end
sortiraj listu uštede u opadajućem poretku
for k ← 1 to size(ustede) do
    if broj grana u ruti = n-2 then
    break
    end
    if nema ciklusa u ruti dodavanjem k-te grane then
    ruta.dodaj(ustede[k][0],ustede[k][1])
    end
end
dodaj preostale dvije grane u rutu
izračunaj dužinu ture
```

Implementacija algoritma

Za potrebe implementacije ovog algoritma su korištene klase $Ruta$ i $Graf$ koje su korištene na vježbama. U ovim klasama napravljene su određene izmjene te su dodane neke funkcije članice.
U klasu $Ruta$ su dodane sljedeće funkcije članice:

- *vector<int> vratiTuru0()*,
- *bool imaLiCiklusaNova()* i
- *bool pomocnaZaDetekcijuCiklusa(int v, vector<bool> &posjecen, int roditelj)*.

Prve dvije metode se nalaze u javnom dijelu funkcije dok se treća metoda nalazi u privatnom dijelu klase. Prva funkcija pravi turu od liste susjedstva počevši od grada 0. Druga metoda provjera da li u grafu postoji ciklus i ona je bazirana na DFS-u i ima kompleksnost $O(|V| + |E|)$.
Pomoću konstruktora klase $Graf$ se učitaju koordinate gradova te formira matrica udaljenosti. Implementirane su dvije nove funckije u klasi $Graf$:

- *vector<int> vratiTuruNajvecaUsteda(vector<vector<int>> & udaljenosti)*,

- *int najvecaUsteda(vector<vector<int>> & udaljenosti).*

Pozivom prve funckije dobijamo turu koju pravi ovaj algoritam, dok pozivom druge funkcije dobijamo dužinu ture.

Vremenska kompleksnost

Za izračunavanje svih mogućih ušteda potrebno je vrijeme $O(n^2)$. Sortiranje ovih ušteda košta $O(n^2 logn)$. Provjera da li postoji ciklus u ruti ima kompleksnost $O(|E| + |V|)$, pri čemu su E i V broj grana i vrhova, respektivno. U najgorem slučaju imamo n vrhova i n grana u ruti tako da provjera postojanja ciklusa u ruti ima kompleksnost $O(n)$. Pošto se u najgorem slučaju prolazi kroz sve elemente niza uštede to imamo da je kompleksnost ovog algoritma $O(n^3)$.

Rezultati

Nakon implementiranja algoritma provedeno je testiranje algoritma te usporedba sa druga dva greedy algoritma obrađenim na vježbama (algoritam koji uzima najmanju granu i algoritam koji ide do najbližeg susjeda). Rezultati su prikazani u tabeli 1. Testiranje je izvršeno na računaru sa karakteristikama:

- Procesor: Intel(R) Core(TM) i5-3320M CPU @2.60Ghz 2.60Ghz,
- RAM: 4.00 GB,
- Operativni sistem: Windows 10.

Država	**Broj gradova**	**Opt. rješenje**	**Najmanja grana**		**Najbliži susjed**		**Najveća ušteda**	
			Dužina	Vrijeme [s]	Dužina	Vrijeme [s]	Dužina	Vrijeme [s]
Djibouti	38	6656	7019	0.004	9745	0.009	6664	0.04
Sahara	29	27603	39691	0.003	36388	0.003	28953	0.023
BIH	179	-	30030	0.023	30591	0.002	26476	0.61
Qatar	194	9352	11499	0.015	11640	0.002	10487	0.59
Uruguay	734	79114	91588	0.1	99247	0.018	87154	9.73
Luksemburg	980	11340	13464	0.28	14370	0.042	12381	26.15
Oman	1979	86891	111191	1.49	120908	0.16	99517	99.2

Tablica 1: Rezultati testiranja tri greedy algoritma na različitim državama

Iz priloženih rezultat se može zaključiti da je na svih sedam instanci podataka algoritam dao značajno bolju rutu (kraću) u odnosu na preostala dva greedy algoritma. Pošto su i algoritam koji bira najbližeg susjeda i algoritam koji uzima najmanju granu kvadratni algoritmi odnos vremena izvršavanja ova dva algoritam i implementiranog algoritma u sklopu projekta je očekivan.

Zaključak

U ovom projektu je predstavljen jedan način implementacije algoritma najveće uštede te su uspoređeni rezultati sa druga dva greedy algoritma. Na kraju možmo zaključit da ovaj algoritam daje bolje ture od druga dva algoritma, ali da za $n > 750$ mu je potrebno puno više vremena u odnosu na druga dva.