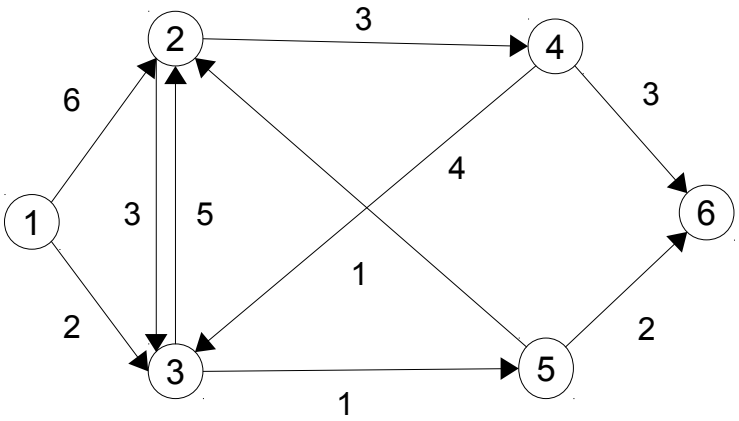
Dijkstra algoritmi kirjeldus

Algoritm leiab lühima tee kaalutud graafis. Kasutatakse sama graafi, mis Standfordi ülikooli materjalis *"The shortest path problem"*, samuti sarnaseid massiviide nimesid lootuses, et niimoodi on kergem mõlema materjali vahel seoseid luua.

Joonisel 1 on kujutatud orienteeritud ja kaalutud graaf. Ülesandeks on leida toodud graafis lühim tee, mis kaalatud graafi puhul on teele jäänud kaarte kaalude summa. Tee leitakse tipust 1 lähtudes.



Joonis 1: Algne graaf

Tee leidmine toimub sarnaselt laiuti otsimisele, kuid selle vahega, et järgmise töödeldava tipu valimisel lähtutakse tipuga seotud teepikkusest (tipu kaugusest lähtepunktist). Originaalalgoritmis kasutatakse järgmise töödeldava tipu valimiseks eelistusjärjekorda. Prioriteetide (eelistuste) aluseks on vastava tipuga seotud teepikkus. Siin kirjelduses leitakse aga asja lihtsustamiseks lihtsalt miinimumi.

Töötlemise käigu ja tulemuste hoidmiseks kasutatakse kolme massiivi. Massiivid peavad olema nii suured, et oleks ruumi kõigi graafi tippude jaoks.

`Prev` – eelmise tipu number (tipp, kust antud tippu satuti).

`Label` – tipu kaugus lähtetipust

`Node` – tipu number koos märkega, kas tipp on "lõpuni" töödeldud

Massiivid algväärtustatakse järgmiselt vt tabel 1 (eeldusel, et algustipuks on 1):

Node	0	0	0	0	0	0
Label	0	999	999	999	999	999
Prev						
Tipud:	1	2	3	4	5	6

Tabel 1: Massiivide algväärtustamine

Algustipu kauguseks määratakse 0, ülejäänud tiippude kauguseks mingi suur arv, mis kindlasti peab olema suurem võimalikest tekkivatest teepikkustest.

Massiiv Node peab arvet töödeldud ja töötlemata tippude üle. Alguses võib sinna kirjutada 0-d ja siis peale töötlemist asendada see 1-ga. Oluline on, et selle massiivi järgi on võimalik arusaada, kas

tipp on juba töödeldud või mitte

Algoritmi käik:

Otsi töötlemata tippude (Node[i] == 0) hulgast väikseima kaugusega (Label[i]) tipp. Tavaline miinimumi leidmine koos lisatingimusega.

Iga selle tipu (u) naabriga (v) tee:

Arvuta tipu v kaugus lähtepunktist: liida tipu u kaugus ja kaare `u->v` pikkus

Kui leitud kaugus < tipu v senine kaugus siis

asenda kaugus massiivfs `Label`

asenda eelmine tipp massiivis Prev (v)

Märgista tipp u töödelduks (massiivi Node kirjuta 1)

Tegevust korratakse, kuni kõik tipud on töödeldud.

Konkreetselt antud graafis:

Samm 1

Minimaalse kaugusega tipp on 1,

Tipu 1naaber on 2, kaare kaal tippude 1 ja 2 vahel on 6, kaugus alguspunktist 0+6=6. 6 on väiksem kui 999, seega kantakse uus teepikkus tipu 2 jaoks tabelisse.

Tipu 1 naaber on 3, kaare kaal tippude 1 ja 3 vahel on 2, kaugus alguspunktist 0+2=2. 2 on väiksem kui 999, seega kantakse uus teepikkus tipu 3 jaoks tabelisse.

Tipp 1 märgitakse töödelduks (vt tabel 2).

Node	1	0	0	0	0	0
Label	0	6	2	999	999	999
Prev		1	1			
Tipud:	1	2	3	4	5	6

Tabel 2: Seis peale 1. sammu

Samm 2

Minimaalse kaugusega tipp on 3,

Tipu 3 naaber on 2, kaare kaal tippude 3 ja 2 vahel on 5, kaugus alguspunktist 2+5=7. 7 ei ole väiksem kui 6, seega tipu 2 infot ei muudeta.

Tipu 3 naaber on 5, kaare kaal tippude 3 ja 5 vahel on 1, kaugus alguspunktist 2+1=3. 3 on väiksem kui 999, seega kantakse uus teepikkus tipu 5 jaoks tabelisse.

Tipp 3 märgitakse töödelduks (vt tabel 3).

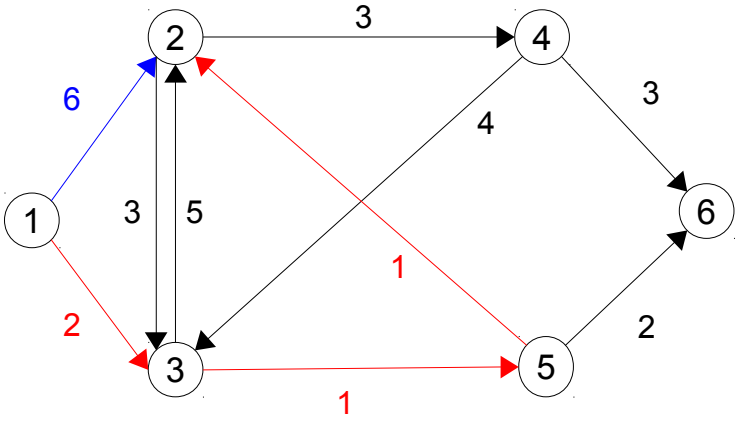
Node	1	0	1	0	0	0
Label	0	6	2	999	3	999
Prev		1	1		3	
Tipud:	1	2	3	4	5	6

Tabel 3: Seis peale 2. sammu

Samm 3

Minimaalse kaugusega tipp on 5,

Tipu 5 naaber on 2, kaare kaal tippude 5 ja 2 vahel on 1, kaugus alguspunktist 3+1=4. 4 on väiksem kui 6, seega kantakse uus teepikkus tipu 2 jaoks tabelisse. Siinkohal toimus tegelik teepikkuse parandus. Ülejäänud parandused olid (ja on) fiktiivsed, mis muudatavad massiivi algväärtuastamisel kasutatud ülisuurt arvu. Joonisel 2 on sinise värviga tähistatud vana tee tippude 1 ja 2 vahel ning punasega uus tee.



Joonis 2: Pikem tee ja lühem tee

Tipu 5 naaber on 6, kaare kaal tippude 5 ja 6 vahel on 2, kaugus alguspunktist 3+2=5. 5 on väiksem kui 999, seega kantakse uus teepikkus tipu 6 jaoks tabelisse.

Tipp 3 märgitakse töödelduks (vt tabel 4).

Node	1	0	1	0	1	0
Label	0	4	2	999	3	5
Prev		5	1		3	5
Tipud:	1	2	3	4	5	6

Tabel 4: Seis peale 3. sammu

Töötlemine lõpuni

Järgmise kolme sammuga valitakse töödeldavateks tippudeks (toodud järjekorras) tipud 2, 6 ja 4. Tabeli seis töötluse lõpuks on tabelis 5.

Node	1	1	1	1	1	1
Label	0	4	2	7	3	5
Prev		5	1	2	3	5
Tipud:	1	2	3	4	5	6

Tabel 5: Seis töötluse lõpus

Tee kahe tipu vahel

Tekkinud tabeli alusel saab leida lühima tee tipust 1 kõigisse teistesse graafi tippudesse. Näiteks tee tippu 4 on leitav järgmiselt, kasutades selleks eellaste massiivi Prev:

Tipu 4 eellaseks on tipp 2. Tipu 2 eellaseks on tipp 5. Tipu 5 eellaseks on tipp 3. Tipu 3 eellaseks on tipp 1. Järelikult tee kulgeb `1 -> 3 -> 5 -> 2 -> 4` ja tee pikkus on 7.

Kui alguspunktiks peab olema mõni teine tipp, tuleb algväärtustada vastava tipu teepikkus (massiiv Label) 0-ga ja tekitada seejärel tabel sarnasel viisil.