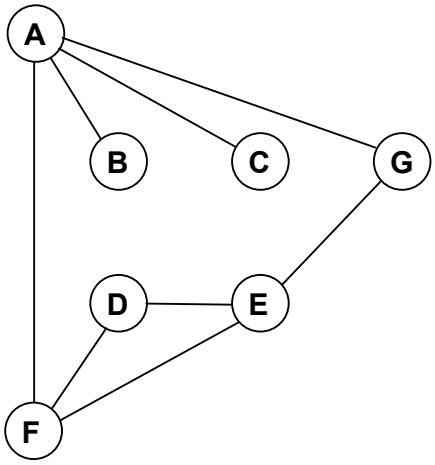
Elementarni algoritmi pretraživanja grafova

Kad jednom imamo graf postavlja nam se problem kako iz zadanog grafa dobiti neke bitne strukturalne informacije o samom grafu, tj. da li je graf povezan i ako nije, od koliko se komponenta povezanosti sastoji te da li graf sadrži ciklus? Postoje dva algoritma koja sustavno obilaze graf i možemo ih iskoristiti pri rješavanju ovih problema, a to su pretraživanje grafa u dubinu i pretraživanje grafa u širinu.

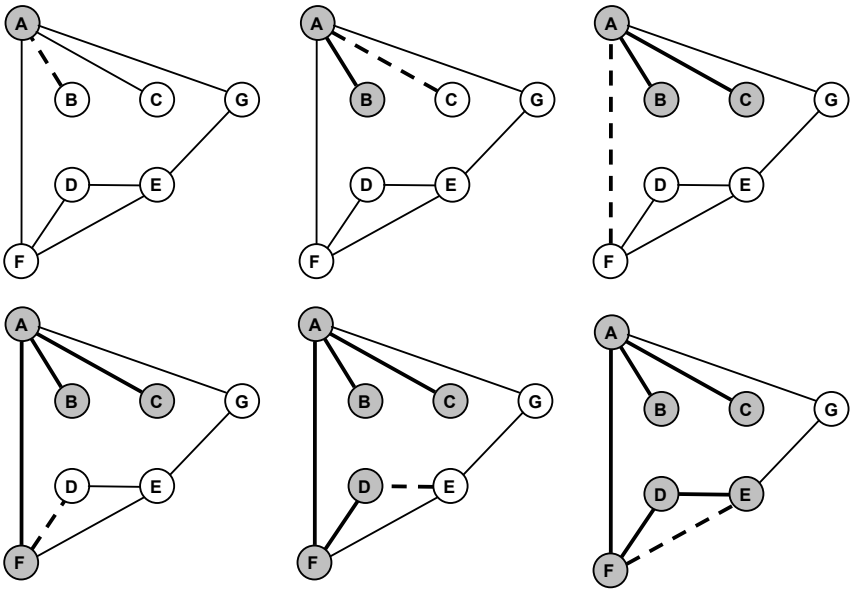
1. Pretraživanje grafa u dubinu.

Sama ideja algoritma je u tome da u svakom koraku pokušamo u grafu otići što je "dublje" moguće. Ta se ideja može veoma lako pokazati na sljedećem primjeru.

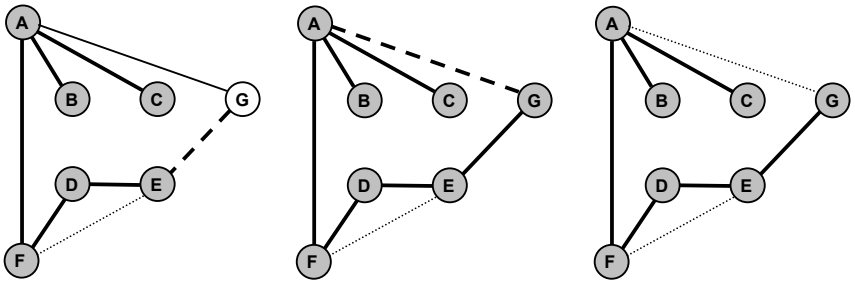


Slika 1. Primjer grafa na kojem ćemo demonstrirati izvođenje pretraživanja u dubinu

Zadani graf obići ćemo u dubinu počevši od vrha A.



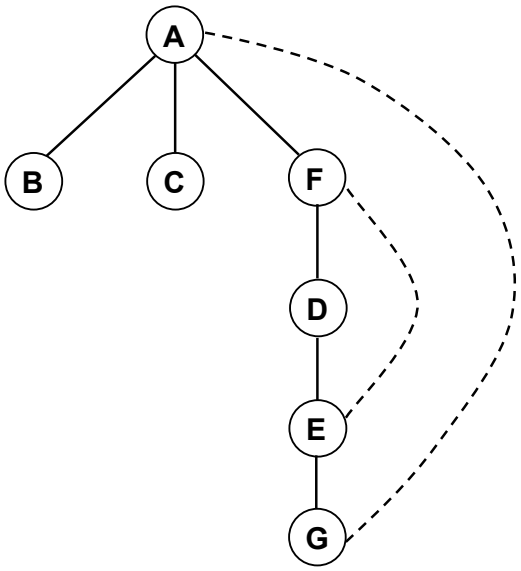
Slika 2. Izvođenje pretraživanja u dubinu



Slika 3. Izvođenje pretraživanja u dubinu (nastavak)

Počevši od vrha A pronalazimo njemu prvi susjedni vrh te nastavimo pretraživati u dubinu od njega na dalje. Ukoliko naiđemo na vrh koji smo već obišli ili vrh od kojeg ne možemo ići dalje, vraćamo se natrag kroz graf dok ne naiđemo na vrh iz kojeg postoji brid kojim nismo prošli. Kao što je iz primjera vidljivo redosljed nailaženja vrhova je A, B, C, F, D, E, G. Nakon što je algoritam naišao na E pokušava otići dalje u dubinu bridom EF, međutim vrh F je već obiđen u nekom prethodnom koraku. Isto tako pokušava se dalje poći bridom GA te je ishod isti kao i s bridom EF.

Uz ovakav način prikazivanja izvođenja algoritma moguć je i prikaz uz pomoć stabla pretraživanja u dubinu. Takvo je stablo u biti samo drugačiji način crtanja grafa kojeg istražujemo. Prikaz takvog stabla za prethodni primjer dan je sljedećom slikom.



Slika 4. Stablo pretraživanja u dubinu

Bridovi označeni punom crtom označavaju da je vrh koji je niže u stablu pronađen u listi susjedstva njemu nadređenog vrha te da taj vrh u prethodnom dijelu algoritma nismo susreli. Crtkani bridovi su oni bridovi koji vode prema već obiđenim vrhovima. Bitno svojstvo neusmjerenih grafova je to da crtkani bridovi uvijek vode od nekog vrha niže u stablu prema nekom vrhu koji mu je predak u stablu.

U toku izvođenja algoritma vrhove grafa možemo u svakom trenutku svrstati u tri klase. U prvu klasu spadaju svi vrhovi koji su potpuno obrađeni, u drugu oni koji su djelomično obrađeni dok u treću oni koji još nisu viđeni. U toku izvođenja aloritma nailaskom na vrh kojeg još nismo susreli u stablo pretraživanja dodajemo brid koji je označen punom crtom, dok kad naiđemo na vrh koji je već obrađen u stablo dodajemo crtkani brid. Kod neusmjerenih grafova nikad nećemo pronaći brid koji vodi od vrha koji je obrađen prema nekom vrhu neke druge klase.

Algoritam:

```
Obilazak_grafa(G)
Za svaki vrh v koji nije označen kao vrh kojeg smo obradili
      Obilazak_u_dubinu(G,v);

Obilazak_u_dubinu(G,v)
      Označi v;
      Za svaki vrh u koji je susjedan v a nije označen
            Obilazak_u_dubinu(G,u);
```

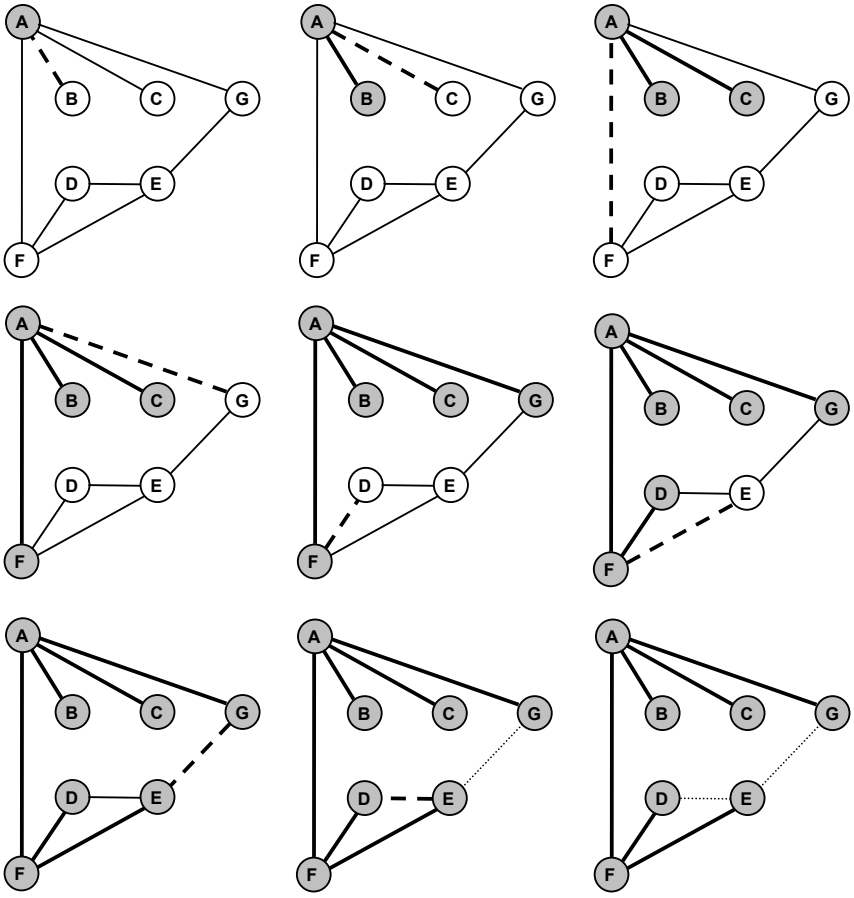
Nerekurzivna implementacija:

Uklanjanje rekurzije vršimo tako da koristimo stog kako bismo pamtili koji vrh treba u sljedećem koraku obraditi.

```
Obilazak_u_dubinu(G,v)
      Stavi v na stog;
      Dok stog nije prazan
            Skini v sa stoga;
            Označi v;
            Za svaki vrh u koji je susjedan v i nije označen
                  Stavi u na stog;
```

2. Pretraživanje grafa u širinu

Slično kao kod pretraživanja grafa u dubinu, ovaj algoritam sistematski obilazi sve vrhove grafa, ali tako da u svakom koraku nastoji ići što je "šire" moguće. Na sljedećoj je slici primjer pretraživanj grafa u širinu za graf na kojem smo prikazivali pretraživanje u dubinu.



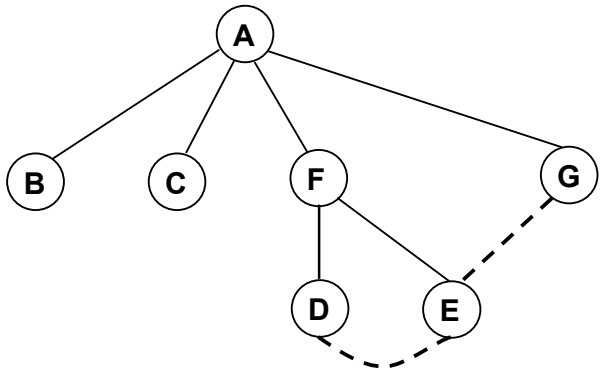
Slika 5. Izvođenje pretraživanja u širinu

Kako bismo pokazali jedno bitno svojstvo pretraživanja grafa u širinu trebamo prvo uvesti pojam udaljenosti u grafu.

Definicija: Za put p u grafu G od v do u, u oznaci $v \rightarrow u$, kažemo da je dužine l ukoliko sadrži l bridova i to označimo s $d(p)=l$. Najkraću udaljenost vrhova u i v definiramo kao dužinu najkraćeg puta između ta dva vrha i označavamo sa $\delta(u,v)$.

Kao što je iz primjera vidljivo, algoritam nikad ne započinje obilazak vrhova koji su udaljeni od početnog vrha n+1 brid prije no što je obradio sve vrhove koji su od početnog vrha udaljeni n bridova.

Slično kao i kod pretraživanja u dubinu, rezultat izvođenja algoritma možemo prikazati stablom pretraživanja u širinu. Takvo stablo za prethodni primjer dano je na sljedećoj slici.



Slika 6. Stablo pretraživanja u širinu

Algoritam:

Kako bi se implementirao ovaj algoritam potrebno je upotrijebiti red koji nam govori koji vrh treba u sljedećem koraku obraditi.

```
Obilazak_grafa(G)
Za svaki vrh v koji nije označen kao vrh kojeg smo obradili
      Obilazak_u_širinu(G,v);

Obilazak_u_širinu(G,v)
      Stavi v u red;
      Dok red nije prazan
            Skini v iz reda;
            Označi v;
            Za svaki vrh u koji je susjedan v i nije označen
                  Stavi u u red;
```

Nakon što smo objasnili kako ova dva algoritma funkcioniraju bilo bi dobro objasniti neke od njihovih primjena. Povezanost grafa je direktno vezana uz broj poziva jedne od procedura za obilazak grafa iz procedure `Obilazak_grafa(G)`, svakim pozivom iz te procedure dobivamo jednu komponentu povezanosti. Ukoliko graf sadrži ciklus tada u stablu kojeg gradimo u toku obilaska postoji brid kojeg označavamo crtkano. Uklanjanjem takvih bridova iz početnog grafa dobivamo razapinjajuće stablo.

Većina algoritama na grafovima sastoji se od većih ili manjih modifikacija ovih osnovnih algoritama pretraživanja grafa. Možemo navesti neke: pronalaženje separirajućih vrhova, pronalaženje mostova, pronalaženje minimalnog razapinjajućeg stabla, pronalaženje minimalnih udaljenosti od jednog vrha do svih ostalih…