Graf toka kontrole i programske putanje

Šta prikazuje graf toka kontrole?

DEFINICIJA

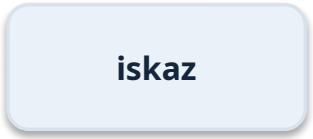
CFG (Control Flow Graph) prikazuje moguće prenose kontrole između delova programa.

Graf je usmeren: grane imaju početak i kraj, a ne moraju obe strane da budu dostižne u konkretnom izvršavanju.

Osnovni elementi



Početak analiziranog metoda



Obrada ili izvršni iskaz



Odluka — obično grana true/false



Kraj analiziranog metoda

Više iskaza u sekvenci može se grupisati u (osnovni) blok.

Primer u Javi: kategorija naknade

Pravilo: nevažeći iznos vraća „invalid“; inače se razlikuju član i ostali korisnici.

```
1  static String feeCategory(int amount, boolean member) {
2      if (amount <= 0) {
3          return "invalid";
4      }
5      if (member) {
6          return "discount";
7      }
8      return "standard";
9  }
```

Šta je odluka?

amount <= 0

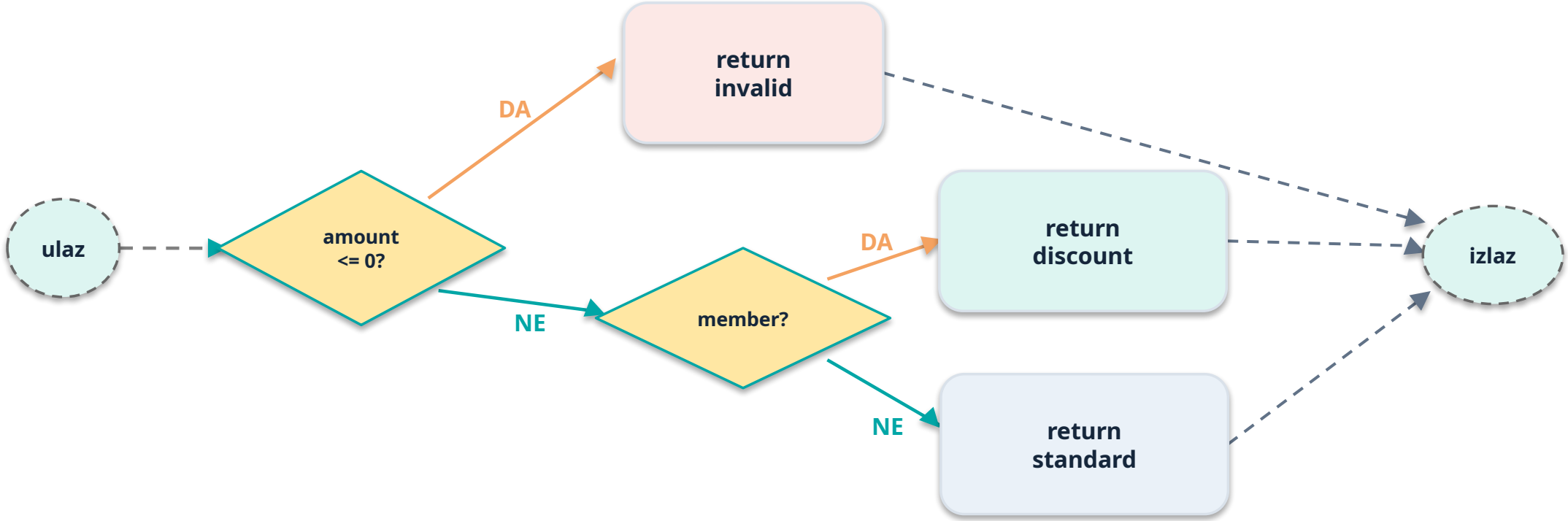
member

Dve tačke grananja

Za graf izdvajamo ulaz, dve odluke, tri povratna iskaza i izlaz.

Kako izgleda CFG za ovaj metod?

Čvorovi predstavljaju kontrolne tačke; grane predstavljaju moguće sledeće korake.



Usmerenje grana je važno: ista dva čvora mogu biti povezana različitim mogućim prenosima kontrole.

Programska putanja je niz čvorova i grana

Putanja: sekvenca čvorova i grana od ulaznog do izlaznog čvora.

Za naš CFG postoje tri putanje od ulaza do izlaza:

P1	ulaz → amount<=0 (DA) → invalid → izlaz
P2	ulaz → amount<=0 (NE) → member (DA) → discount → izlaz
P3	ulaz → amount<=0 (NE) → member (NE) → standard → izlaz

Redosled je značajan: ulaz u isti čvor iz različitih grana ne znači da je prethodna sekvenca ista.

Od putanje do konkretnih ulaza

Svaka odluka duž željene putanje nameće uslov koji ulaz mora da zadovolji.

Putanja	Uslovi	Primer ulaza	Očekivanje
P1	amount <= 0	amount=0; member=false	"invalid"
P2	amount > 0; member=true	amount=100; member=true	"discount"
P3	amount > 0; member=false	amount=100; member=false	"standard"

Test mora da aktivira ciljanu putanju i da proveri očekivano ponašanje (oracle/asercija).

Za jednostavne izraze rešavamo uslove direktno; složeniji primeri mogu zahtevati sistematičnu analizu uslova putanje.

Šta se menja kada program sadrži petlju?

```
static int sumPositive(int[] a) {
    int sum = 0;
    for (int x : a) {
        if (x > 0) sum += x;
    }
    return sum;
}
```

Mogući obrasci izvršavanja

- **0 iteracija**
  Prazan niz.
- **1 iteracija**
  Jedan element.
- **Više iteracija**
  Svaka iteracija ima i uslov x > 0.

Ako dužina niza nije ograničena, putanje mogu da se razlikuju proizvoljnim brojem ponavljanja. Potpuno pokrivanje svih putanja tada nije praktično ostvarivo.

Kasnije ćemo razmatrati kriterijume koji biraju ograničen skup reprezentativnih putanja.

Graf može sadržati strukturno neizvodljivu putanju

```
if (x > 0) {
    if (x < 0) {
        reject();
    }
}
```

Zašto ne postoji test-ulaz?

Unutrašnji if se proverava tek kada je x > 0; isti, neizmenjeni x ne može istovremeno biti < 0.

Prva odluka: x > 0 → DA

Druga odluka: x < 0 → DA

Putanja se može formalno konstruisati u CFG-u, ali nije izvodljiva ni za jednu vrednost x. To treba dokumentovati pri sastavljanju testova.

Drugi primer

```
static int classify(int x) {
    if (x < 0) return -1;
    if (x == 0) return 0;
    return 1;
}
```

Koliko putanja ima u sledećem kodu?

1. Nacrtati CFG.
2. Nabrojati putanje od ulaza do izlaza.
3. Predložiti ulaz za svaku putanju.

Očekivani ulazi: x = −1, x = 0 i x = 5. Oni vode do povratnih vrednosti −1, 0 i 1.

Pitanje: da li izvršiva četvrta putanja koja prvo zadovolji x < 0, a zatim proverava x == 0?

Ključne poruke

- **CFG apstrahuje tok kontrole**

  Čvorovi su kontrolne tačke, a grane predstavljaju moguće prenose kontrole.

- **Putanja je konkretan niz prolazaka kroz graf od ulaznog do izlaznog čvora**

  Ulazi određuju koje odluke i grane će se aktivirati.

- **Potpuno pokrivanje svih putanja često nije praktično**

  Petlje daju proizvoljno mnogo sekvenci, a neke strukturne putanje mogu biti neizvodljive.

Sledi: bazične putanje, linearna nezavisnost i ciklomatska kompleksnost.

Bazične putanje i ciklomatska kompleksnost

Od grafa kontrole toka do bazičnog skupa putanja

1 **Objasniti linearno kombinovanje i nezavisnost putanja**

2 **Izračunati V(G)**
Povezati broj čvorova, grana i binarnih odluka.

3 **Pripremiti testove**
Izabrati ulaze koji izvršavaju svaku putanju bazičnog skupa.

Tok časa

1 Java primer: Euklidov algoritam

2 Linearno slaganje putanja

3 Linearno nezavisne putanje

4 V(G) i McCabe-ov Baseline metod

Cilj nije testirati sve putanje, već odrediti bazični skup koji pokriva nezavisne dimenzije toka kontrole.

Zašto ne testiramo sve programske putanje?

Petlje stvaraju mnogo varijanti

Ista kontrolna struktura može se izvršiti bez ulaska u petlju ili sa 1, 2, 3... iteracija. Broj putanja često nije praktično konačan.

Ideja bazičnih putanja

Izaberemo linearno nezavisan skup putanja. Svaka izabrana putanja doprinosi novoj dimenziji toka kontrole u grafu.

Bazični skup je manji, sistematičan skup reprezentativnih putanja — a ne dokaz da su ispitane sve moguće iteracije.

Koncept uvodi Thomas McCabe tokom 1970-ih, oslanjajući se na ideju linearne nezavisnosti.

Primer u Javi: Euklidov algoritam oduzimanjem

Preuslov: x > 0 i y > 0. Posmatramo samo kontrolni tok algoritma.

```
static int gcdBySubtraction(int x, int y) {
    // Preuslov: x > 0 && y > 0
    while (x != y) {
        if (x > y) {
            x = x - y;
        } else {
            y = y - x;
        }
    }
    return x;
}
```

Dve odluke

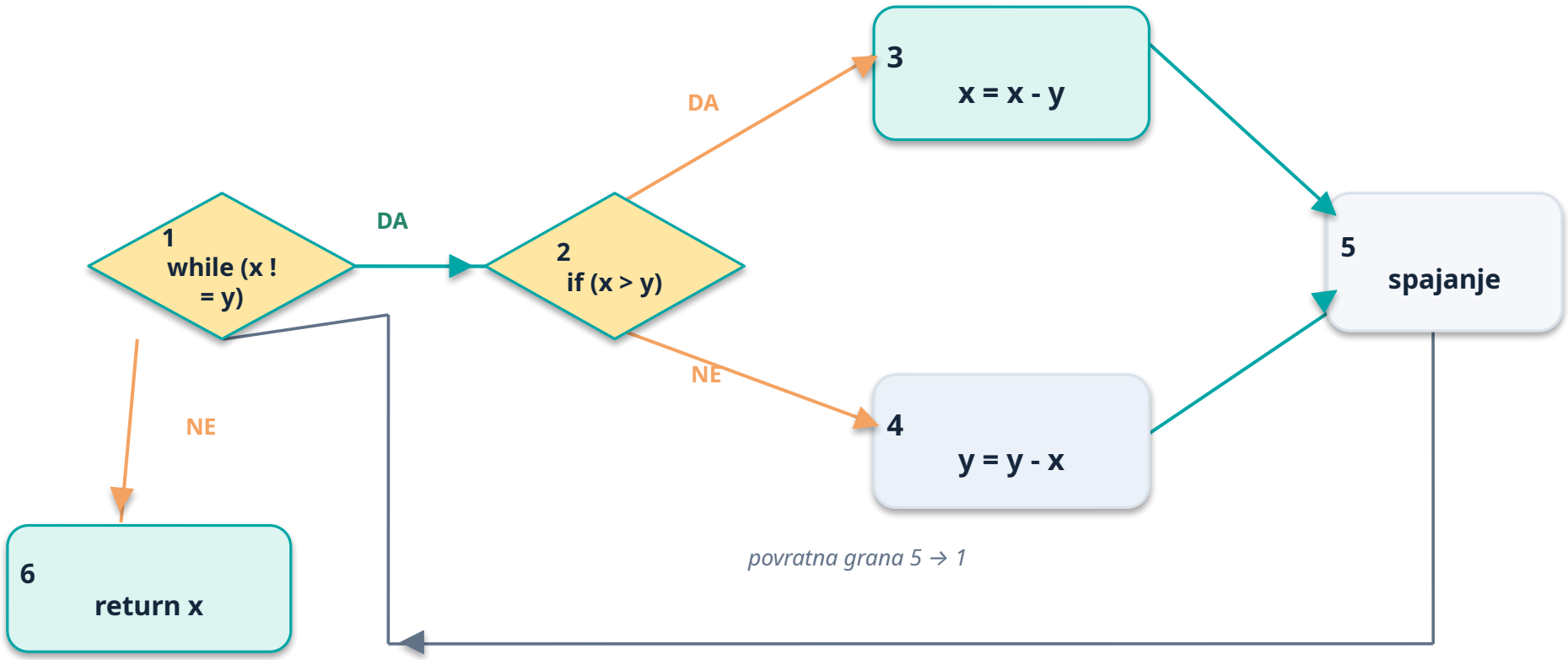
D1 x != y?

D2 x > y?

V(G) očekujemo 3

Algoritam vraća isti rezultat za različite ulaze, ali kontrolni tok zavisi od odnosa x i y.

Graf toka kontrole: 6 čvorova i 7 grana



Grane: 1→2, 1→6, 2→3, 2→4, 3→5, 4→5, 5→1.

N = 6 čvorova; E = 7 grana.

Ciklomatska kompleksnost V(G)

Preko grana i čvorova

V(G) = E − N + 2

E — broj grana; N — broj čvorova; za graf sa jednim ulazom i jednim izlazom.

Preko odluka

V(G) = D + 1

D — broj binarnih odluka u strukturiranom programu.

Za više početnih i završnih čvorova

V(G) = E’ – N + 1

Pre računanja broja grana povezati svaki pojedinačni završni čvor sa svakim od početnih čvorova

E = 7, N = 6 ⇒ V(G) = 7 − 6 + 2 = 3.
D = 2 ⇒ V(G) = 2 + 1 = 3.

Tri putanje koje razdvajaju kontrolne mogućnosti

P1

1 → 2 → 3 → 5 → 1 → 6

while ulazi u telo; if bira x=x−y

Test-ulaz: x=2, y=1

P2

1 → 6

while se ne izvršava

Test-ulaz: x=1, y=1

P3

1 → 2 → 4 → 5 → 1 → 6

while ulazi u telo; if bira y=y−x

Test-ulaz: x=1, y=2

P1 i P3 uključuju jednu iteraciju, dok P2 odmah izlazi iz petlje. To nije pokrivanje svih brojeva iteracija.

Linearno nezavisne putanje: pogled kroz grane

Svaka putanja može se predstaviti **vektorom incidencije**: 1 označava da putanja koristi granu, 0 da je ne koristi.

Grana	P1	P2	P3	Komentar
1 → 2	**1**	**0**	**1**	P2 ne ulazi u petlju
1 → 6	**1**	**1**	**1**	zajednička završna grana
2 → 3	**1**	**0**	**0**	samo P1
2 → 4	**0**	**0**	**1**	samo P3
3 → 5	**1**	**0**	**0**	P1
4 → 5	**0**	**0**	**1**	P3
5 → 1	**1**	**0**	**1**	povratak iz tela

Linearno slaganje putanja je sabiranje/oduzimanje vektora incidencije (uz eventualno množenje skalarnom konstantom).
Putanju P1 nije moguće izraziti kao linearnu kombinaciju ostale dve, jer one nemaju granu 2→3.
Putanju P3 nije moguće izraziti kao linearnu kombinaciju ostale dve, jer jedino ona sadrži granu 2→4.
Putanju P2 nije moguće izraziti kao linearnu kombinaciju ostale dve, jer nemoguće je “poništiti” grane koje sadrže samo P1, odnosno P3.
Prema tome, P1, P2, P3 čini jedan **linearno nezavisni** skup putanja grafa G.

Šta je bazični skup putanja?

Dva uslova

1. Putanje u skupu su međusobno linearno nezavisne.
2. Svaka druga putanja u grafu može se predstaviti kao linearna kombinacija putanja iz skupa.

Za naš CFG

Bazični skup je {P1, P2, P3}.
Njegova kardinalnost je 3.

Jedan CFG može imati više različitih bazičnih skupova, ali svi imaju isti broj putanja.

Broj putanja u bazi jednak je ciklomatskoj kompleksnosti V(G).

McCabe-ov Baseline metod

Postupak sistematski konstruiše nezavisne putanje sve dok ne dobijemo V(G) putanja.

1	**Izabrati baseline putanju**	Predstavlja normalan slučaj; poželjno je da prolazi kroz što više čvorova odlučivanja.
2	**Izmeniti jednu odluku**	Pratiti putanju, naći prvi nerazmatrani čvor odluke i izabrati izlaznu granu koja nije na putanji.
3	**Ponoviti do V(G)**	Svaka nova putanja mora da doda nezavisnu dimenziju; zaustaviti se kada je izabrano V(G) putanja.

Izbor baseline putanje nije jedinstven. McCabe preporučuje putanju koja prolazi kroz što više odluka, kada je to praktično.

Primer izbora putanja i test-ulaza

Putanja	Sekvenca čvorova	Test-ulaz	Efekat
P1	1 → 2 → 3 → 5 → 1 → 6	x=2, y=1	x se smanjuje; izlaz
P2	1 → 6	x=1, y=1	petlja se preskače
P3	1 → 2 → 4 → 5 → 1 → 6	x=1, y=2	y se smanjuje; izlaz

Testovi izvršavaju svaku putanju bazičnog skupa. U praksi treba dodati testove granica, grešaka i više iteracija petlje.

Šta nam govori ciklomatska kompleksnost?

Koristi se za

- procenu broja nezavisnih putanja i planiranje strukturnih testova;
- indikaciju složenosti toka kontrole i napora potrebnog za razumevanje;
- upravljačke smernice za ograničavanje previše složenih metoda.

Ne treba je tumačiti kao

- broj grešaka koje metoda sadrži;
- garanciju da je testiranje dovoljno ili da su rezultati ispravni;
- univerzalni prag kvaliteta: vrednosti poput 10 su smernice, a ne zakon.

Praktična poruka: visoka vrednost V(G) traži pažnju, ali mora se tumačiti zajedno sa veličinom i ulogom jedinice koda.

Kratak zadatak: izračunajte V(G)

```
static int classify(int x) {
    if (x < 0) return -1;
    if (x == 0) return 0;
    return 1;
}
```

1. Koliko je binarnih odluka?
2. Koliko je V(G)?
3. Predložite ulaz za svaku bazičnu putanju.

Rešenje: D = 2, pa je V(G) = 3. Test-ulazi −1, 0 i 5 vode do povratnih vrednosti −1, 0 i 1.

Ključne poruke

1	**Nezavisnost putanja**	Svaka putanja doprinosi novoj dimenziji toka kontrole.
2	**V(G) određuje veličinu baze**	Za tipičan CFG sa jednim ulazom i izlazom: E−N+2; za binarne odluke: D+1.
3	**Baseline metod je sistematičan način konstrukcije skupa nezavisnih putanja**	Kreće se od normalne putanje i dodaju se putanje preko alternativnih grana.