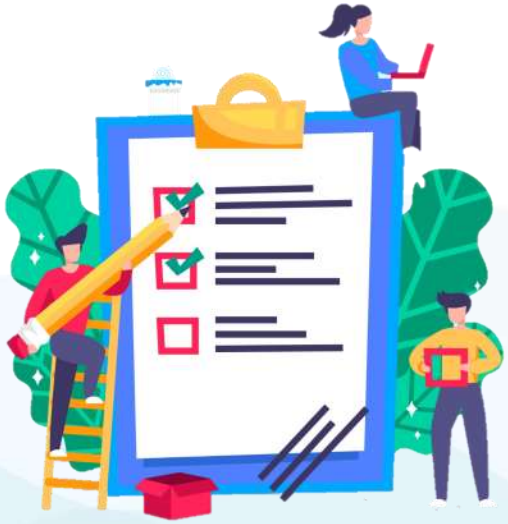


White Box Testing : Basis Path

Tantangan Pengujian

Tantangan utama dalam pengujian adalah menentukan set awal yang baik dari kasus uj:

- Hilangkan pengujian yang berlebihan
- Berikan cakupan tes yang memadai
- Izinkan pengujian yang lebih efektif
- Manfaatkan pengujian terbatas sumber daya

Apa itu Basis Path Testing?

Basic Path Testing adalah gabungan antara pengujian jalur (path testing) dan cabang pengujian (branch testing):

- Pengujian Jalur (path testing) : Pengujian dirancang untuk menjalankan semua atau jalur yang dipilih melalui program komputer [IEEE610]
- Branch Testing (branch testing): Pengujian dirancang untuk mengeksekusi setiap hasil setiap titik keputusan dalam program komputer [IEEE610]
- Basis Path Testing (basis path testing): Pengujian yang memenuhi persyaratan pengujian cabang & juga menguji semua jalur independen itu dapat digunakan untuk membangun jalur arbitrer apa pun melalui program komputer [berdasarkan NIST (National Institute of Standards and Technology – Amerika Serikat)]

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI
TEKNIK INFORMATIKA
MATA KULIAH
SOFTWARE QUALITY & TESTING

Apa itu Basis Path Testing?

- Basis Path adalah jalur unik melalui perangkat lunak di mana iterasi tidak diperbolehkan - semua jalur yang memungkinkan melalui sistem adalah kombinasi linier dari mereka.
- Basis Path Testing adalah teknik White Box Testing yang mengidentifikasi pengujian kasus berdasarkan alur atau jalur logis yang dapat diambil melalui perangkat lunak.

McCabe's Basis Path Testing

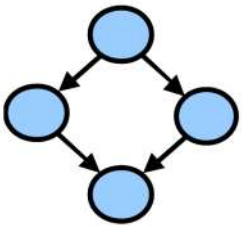
Langkah:
1: Gambarlah grafik aliran kontrol (control flow graph)
2: Hitung kompleksitas siklus
3: Pilih kumpulan jalur dasar (basis set of path)
4: Buat kasus uji (test case) untuk melatih setiap jalur

Step 1: Gambar Grafik Aliran Kontrol (Control Flow Graph) #1

Grafik Aliran Kontrol (Control Flow Graph)

Setiap desain prosedural dapat diterjemahkan ke dalam aliran kontrol grafik:

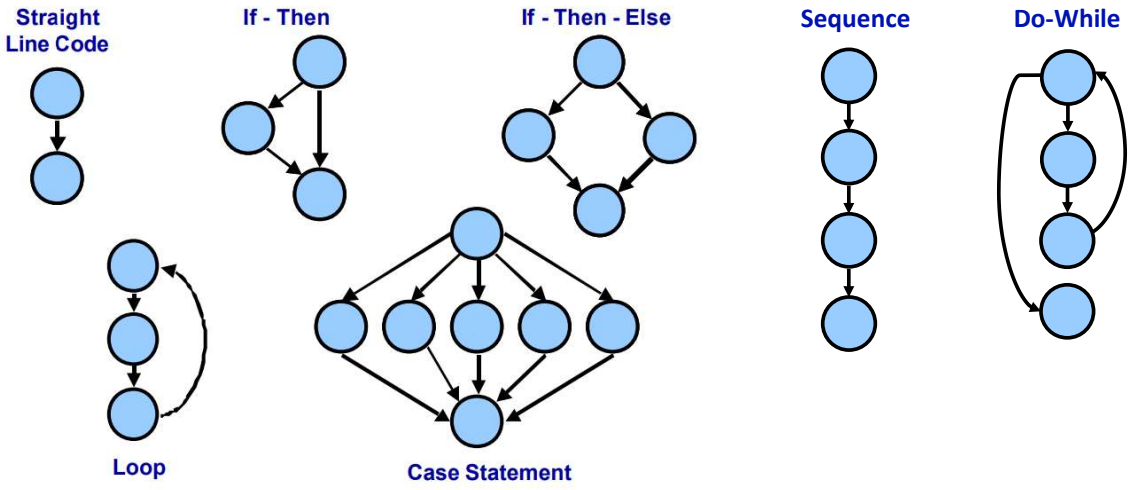
- Garis (atau panah) yang disebut **tepi (edges)** mewakili aliran kontrol
- Lingkaran yang disebut **node** mewakili satu atau lebih tindakan
- Area yang dibatasi oleh edge dan node disebut **region**
- **Node predikat/ predicate node** adalah node yang berisi suatu kondisi



FAKULTAS ILMU KOMPUTER
PROGRAM STUDI
TEKNIK INFORMATIKA
MATA KULIAH
SOFTWARE QUALITY & TESTING

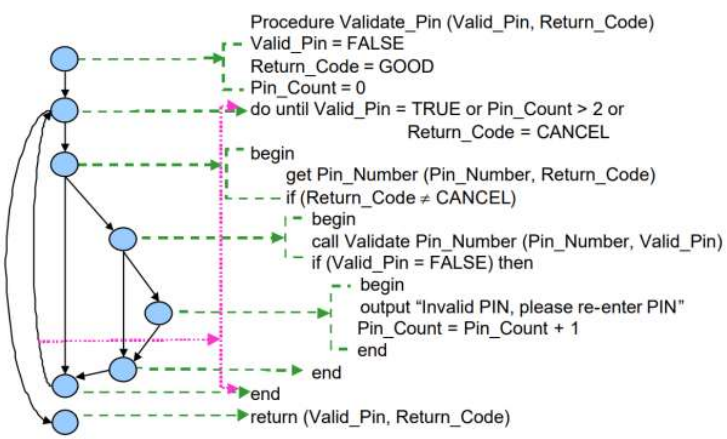
Step 1: Gambar Grafik Aliran Kontrol (Control Flow Graph) #2

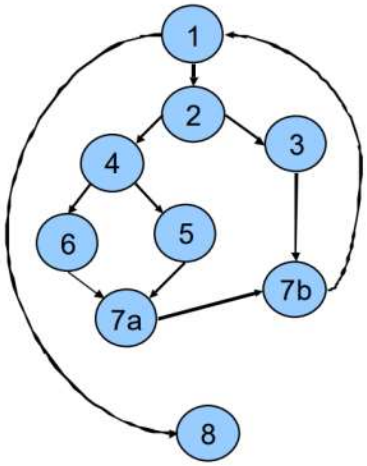
Struktur Dasar Grafik Aliran Kontrol (Control Flow Graph)



Step 1: Gambar Grafik Aliran Kontrol (Control Flow Graph) #3

Contoh



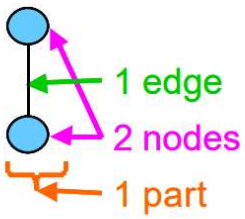



```
1.    do while records remain
          read record;
2.        if record field 1 = 0
3.            then process record;
                  store in buffer;
                  increment counter;
4.        elsif record field 2 = 0
5.            then reset record;
6.            else process record;
                  store in file;
7a.       endif;
      endif;
7b.   enddo;
8.  end;
```

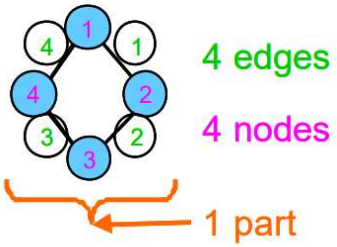
Langkah pertama dalam basis path testing adalah menggambar grafik aliran kontrol. Seperti yang diilustrasikan pada contoh di atas, dapat dilakukan langsung dari kode sumber.

Step 2: Hitung Kompleksitas Siklus #1

Model: V(G) = edges – nodes + 2p
Dimana p = jumlah bagian grafik yang tidak terhubung



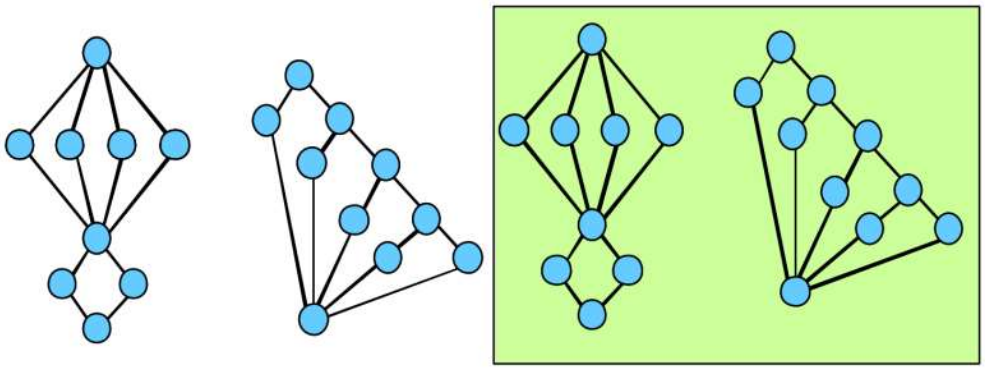
V(G) = 1 - 2 + 2x1 = 1
Straight line code always has a complexity of 1



V(G) = 4 - 4 + 2x1 = 2

Step 2: Hitung Kompleksitas Siklus #2

Model: V(G) = edges – nodes + 2p
Dimana p = jumlah bagian grafik yang tidak terhubung



12 edges
9 nodes
12-9+2(1)=5

13 edges
10 nodes
13-10+2(1)=5

25 edges
19 nodes
25-19+2(2)=10

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI
TEKNIK INFORMATIKA
MATA KULIAH
SOFTWARE QUALITY & TESTING

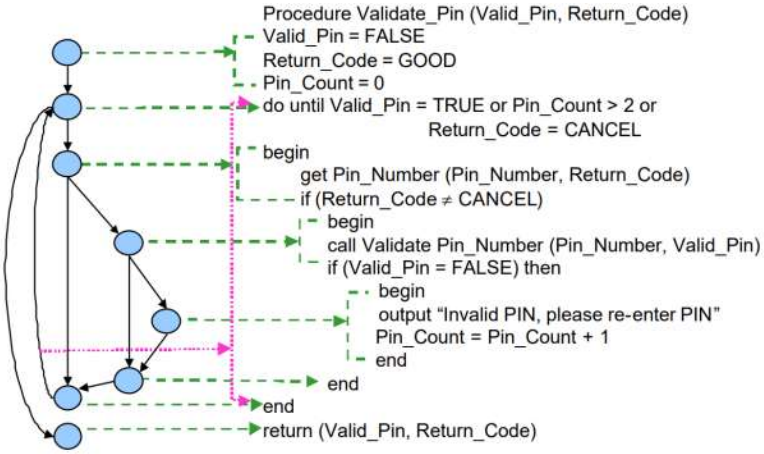
Kegunaan Kompleksitas Siklus

Kompleksitas siklus dapat digunakan untuk:

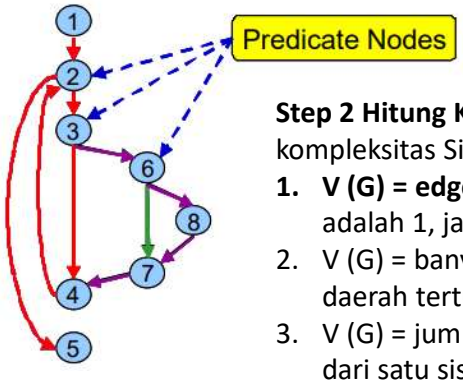
- mengetahui berapa banyak jalur yang harus dicari dalam basis path testing
- Membantu menentukan area potensi ketidakstabilan
- Menunjukkan unit / komponen yang dapat diuji & dimengerti (pemeliharaan)
- Memberikan indikasi kuantitatif kompleksitas aliran kendali unit / komponen
- Menunjukkan upaya yang diperlukan untuk menguji unit / komponen

Uji Kasus - Step 1

- **Step 1 Gambarlah grafik control flow-** untuk contoh ini kita akan menggunakan grafik control flow yang kita gambar pada contoh kode pertama.



Uji Kasus – Step 2



Step 2 Hitung Kompleksitas Siklomatik - ada tiga cara untuk menghitung kompleksitas Siklomatik dari grafik control flow

1. **V (G) = edges - nodes + 2p**. Untuk contoh ini ada 10 edge, 8 node dan p adalah 1, jadi V (G) = 10 - 8 + 2 = 4
2. V (G) = banyaknya daerah pada grafik control flow. Untuk contoh ini ada 3 daerah tertutup ditambah daerah luar, jadi V (G) = 4.
3. V (G) = jumlah node predikat + 1. Node predikat adalah node dengan lebih dari satu sisi yang memancar darinya (node yang berisi kondisi). Untuk contoh ini, node 2, 3 dan 6 adalah node predikat, jadi V (G) = 3 + 1 = 4.

Uji Kasus – Step 2

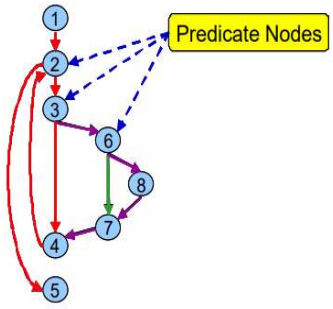
Matrix Grafik Koneksi

- Untuk informasi tambahan, publikasi khusus National Bureau of Standards “Pengujian Terstruktur: Metodologi Pengujian Perangkat Lunak Menggunakan Metrik Kompleksitas Siklomatik. [McCabe-82]
- Berat Tautan Dapat mengaitkan angka dengan setiap entri tepi (edge). Dalam bentuk yang paling sederhana, file link weight adalah 1 (koneksi ada) atau 0 (koneksi tidak ada).
- Gunakan nilai 1 (menunjukkan bahwa ada koneksi) untuk menghitung kompleksitas siklomatik:
  - Untuk setiap baris, jumlahkan nilai baris dan kurangi 1
  - Jumlahkan nilai total kemudian tambahkan 1

Connected to node

Node	1	2	3	4	5	6	7	8	
1	0	1	0	0	0	0	0	0	1 - 1 = 0
2	0	0	1	0	1	0	0	0	2 - 1 = 1
3	0	0	0	1	0	1	0	0	2 - 1 = 1
4	0	1	0	0	0	0	0	0	1 - 1 = 0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	1	2 - 1 = 1
7	0	0	0	1	0	0	0	0	1 - 1 = 0
8	0	0	0	0	0	0	1	0	1 - 1 = 0

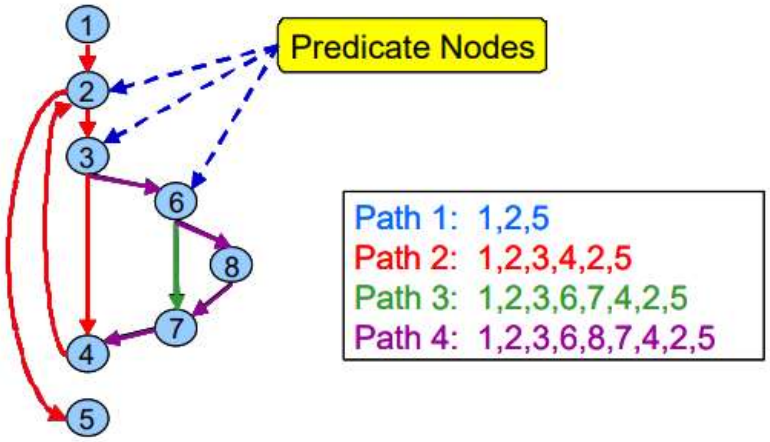
3 + 1 = 4



Step 3: Pilih Kumpulan Jalur Dasar (Basis Set Of Path)

Step 3 Pilih Jalur Kumpulan Dasar (Basis Set Of Path) - menentukan **node predikat** dapat membantu **mengidentifikasi satu set jalur basis**. Jika **test case** dapat dirancang untuk mencakup set jalur dasar, maka akan **menghasilkan cakupan keputusan (dan pernyataan) kode yang lengkap**. Setiap jalur basis yang Anda pilih harus menguji setidaknya satu tepi (edge) baru yang belum diuji, dengan kata lain harus melintasi setidaknya satu tepi baru. Jika tidak, ini dianggap jalur yang berlebihan dan tidak termasuk dalam kumpulan basis. Ini membantu dalam menghilangkan pengujian yang berlebihan dan memastikan validitas setiap kasus pengujian. Untuk setiap jalur basis berikutnya yang dipilih, coba pertahankan jumlah tepi baru yang ditambahkan serendah mungkin, teruskan hingga Anda telah mencakup semua jalur basis yang memungkinkan.

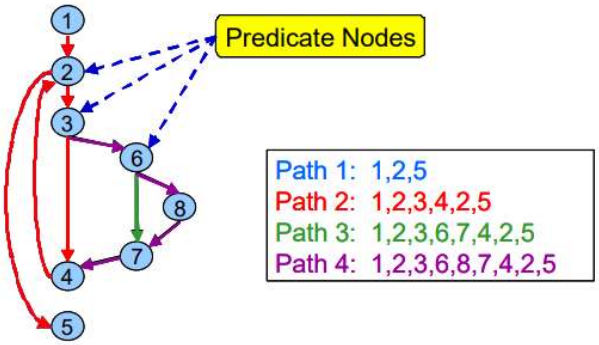
Uji Kasus - Step 3: Pilih Kumpulan Jalur Dasar (Basis Set Of Path)



FAKULTAS ILMU KOMPUTER
PROGRAM STUDI
TEKNIK INFORMATIKA
MATA KULIAH
SOFTWARE QUALITY & TESTING

Step 4: Buat kasus uji (test case) untuk melatih setiap jalur

Buat kasus uji (test case) yang akan memaksa eksekusi setiap jalur dalam set basis - misalnya:



Jalur 1: 1,2,5
Kasus uji 1
Jalur 1 tidak dapat diuji berdiri sendiri & harus diuji sebagai bagian dari jalur 2, 3, atau 4

Jalur 2: 1,2,3,4,2,5
Kasus uji 2
Tekan batal sebagai tanggapan atas permintaan "Masukkan Nomor PIN"

Jalur 3: 1,2,3,6,7,4,2,5
Kasus uji 3
Masukkan nomor PIN yang valid pada percobaan pertama

Jalur 4: 1,2,3,6,8,7,4,2,5
Kasus uji 4
Masukkan PIN yang tidak valid pada percobaan pertama & PIN yang valid pada percobaan kedua

Perlu dicatat bahwa untuk mengikuti jalur 4 kasus uji ini bisa mengeksekusi loop untuk kedua kalinya sehingga jalurnya sebenarnya 1,2,3,6,8,7,4,2,3,6,7,4, 2,5.

Kesimpulan

Manfaat pengujian jalur dasar/ Basis Path:

- Mendefinisikan jumlah jalur independen sehingga jumlah kasus uji yang diperlukan untuk memastikan:
  - Setiap pernyataan akan dieksekusi setidaknya satu kali
  - Setiap kondisi akan dieksekusi di sisi benar & salahnya
- Memusatkan perhatian pada logika program
- Memfasilitasi desain kasus uji analitis versus arbitrer