Regresyon Testi

Motivasyon

Sistem kodunda veya sistem ortamında değişiklikler yapılmış olsun.

Daha önce olmayan yeni hatalar ortaya çıktı mı?

Sistemin yeni sürümünü **güvenle** dağıtabilir miyiz?

Regresyon Testi artık yazılım geliştirme endüstrisinde standarttır.

Tipik olarak bir **regresyon test paketi** üzerine kuruludur

Senaryolar

Regresyon testi **çok maliyetli** olabilir

- Yazılım büyüdükçe test paketleri de büyür
- **CI/CD** (Continuous Integration & Development) frameworkları yardımcı olabilir
- Regresyon test paketlerini gece boyunca çalıştırmak yaygın bir uygulamadır

Mevcut testlerin yeniden çalıştırılması **her zaman mümkün olmayabilir**; örneğin genel arayüzler değiştiyse

Gömülü sistemler için, örn. araç kontrolörleri için kullanılan yazılımın yeniden test edilmesi **aylar** sürebilir. Simülasyon yardımcı olabilir ancak kesin bir çözüm değildir.

Regresyon Test Teknikleri

Minimizasyon – Test paketlerimi küçültebilir miyim?

Bir kritere bağımlıdır, ör., … kriteri sabit kalırken minimize etmek

Önceliklendirme – Testleri hangi sıra ile çalıştırmalıyım?

Yeni kusurları olabildiği kadar hızlı tespit etme

Seçim – Hangi testleri koşmalıyım?

Bir dizi kod satırı verildiğinde…

Minimizasyon (Küçültme)

Minimizasyon

Yazılım büyüdükçe, test paketinin boyutu da büyür!

Testler **geçerliliğini yitirir** veya **gereksiz hale** gelebilir.

Test sürdürülebilirliği pratikte nadiren önceliklendirilir.

Gereksizleri Kaldırma

Aynı kodu kapsayan birçok benzer test!

CI/CD'den kapsam bilgisi alınır, ör., daha önce koşturulan testler

Yalnızca kodun değişen veya silinen parçalarını çalıştıran test kodlarını çalıştırın. Daha önce çalıştırılmış değişmeyen noktalar aait testleri tekrar çalıştırmaya gerek yoktur.

Minimizasyon Algoritması

Kod kapsamına odaklanın, ör., **branch coverage**

Diğer özellikleri korumak için alternatif yaklaşımlar mevcuttur.

Bir regresyon test paketi T verildiğinde, **T ve T'nin aynı kapsama sahip olduğu en küçük alt küme *T*'yi** bulun.

Kapsamı korumanın etkililiği koruduğu, yani T'nin *T* kadar etkili olması umulur ancak her zaman bu durum geçerli değildir

En küçük test kümesinin seçimi zorunlu değildir: Daha büyük bir test kümesinin çalıştırma maliyeti küçük olandan az ise **çalıştırma maliyeti küçük olan** seçilir.

Formalizasyon

Bir dizi kapsam hedefi verildiğinde $C = \{c_1, \ldots, c_k\}$(ör., satır)

a.k.a. **test gereksinimleri**

Ve hepsini karşılayan/kapsayan bir test paketi $T = \{t_1, \ldots, t_n\}$

Burada her $t_i \in T$ belirli bir C_{t_i} kapsam hedefini kapsar

Burada $\bigcup_{t_i \in T'} C_{t_i} = C$ olan en küçük $T' \in T$ bulunur.

Bu NP-Complete küme kapsamı problemidir.

Kesin Çözümler vs Sezgisel Yaklaşımlar

Test paketleri genellikle büyüktür – böyle olmasaydı küçültme ile uğraşmazdık ☺

Bu yüzden problem örnekleri de genellikle büyüktür.

Problemi tam olarak çözmek genellikle imkansızdır bu yüzden sezgisel yaklaşımlara başvururuz: Nasıl **iyi (yeterince küçük) bir test paketi elde edilir**?

Basit Bir Açgözlü (Greedy) Algoritma

Boş bir küme ile başlayın

En **fazla hedefi** kapsayan bir test t_i ekleyin ve test havuzundan çıkarın.

Yinelemeler: En fazla hedefi kapsayan bir sonraki testi ekleyin.

Test paketi tam kapsam sağladığında **sonlandırın**.

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Daha sonra **B** seçilir

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Daha sonra **B** seçilir

Son olarak da **C** ve **D**

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Daha sonra **B** seçilir

Son olarak da **C** ve **D**

Test Kümesi: **{A, E, B, C, D}**

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Daha sonra **B** seçilir

Son olarak da **C** ve **D**

Test Kümesi: **{A, E, B, C, D}**

Ancak yalnızca {C, D} ile de aynı şeyi yapabilirdik!

Basit Bir Açgözlü (Greedy) Algoritma

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A ve **E** ile başlar (maksimum kapsam)

Daha sonra **B** seçilir

Son olarak da **C** ve **D**

Test Kümesi: **{A, E, B, C, D}**

Ancak yalnızca {C, D} ile de aynı şeyi yapabilirdik!

Her zaman iyi bir yaklaşım değildir!

Ek Açgözlü Yaklaşım

Mevcut ulaşılan kapsam değerini dikkate alın!

Tekrar boş bir küme ile başlayın

En fazla hedefi kapsayan testlerden birini ekleyin

Yinelemeler: **Şu anda kapsanmayan** en fazla hedefi kapsayan bir test ekleyin.

Tam kapsam sağlanınca sonlandırın.

Küme kapsama yöntemine göre daha iyi bir yaklaşım.

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A veya **E ile başla** (ikisinden biri)

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A veya E ile başla (ikisinden biri)

Kapsanmamış hedefler: **b_4** ve **b_5**

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A veya E ile başla (ikisinden biri)

Kapsanmamış hedefler: **b_4** ve **b_5**

C ve **D**'yi ekle

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

A veya E ile başla (ikisinden biri)

Kapsanmamış hedefler: **b_4** ve **b_5**

C ve **D**'yi ekle

Her biri kapsanmamış bir hedefi kapsar

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

- A veya E ile başla (ikisinden biri)

  Kapsanmamış hedefler: **b_4** ve **b_5**

- **C** ve **D**'yi ekle

  Her biri kapsanmamış bir hedefi kapsar.

- Sonuç Seti: **{A, C, D}**

Ek Açgözlü Yaklaşım

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

- A veya E ile başla (ikisinden biri)

  Kapsanmamış hedefler: **b_4** ve **b_5**

- **C** ve **D**'yi ekle

  Her biri kapsanmamış bir hedefi kapsar.

- Sonuç Seti: **{A, C, D}**

Optimal değil ancak daha iyi!

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Harrold et al (1993)

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

b_5 yalnızca **D** tarafından kapsanız bu yüzden **D** eklenir.

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

b_5 yalnızca **D** tarafından kapsanız bu yüzden **D** eklenir.

Yinelemeler: Daha sonra 2, 3, 4 hedefi kapsayan testler sırasıyla ele alınır.

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

b_5 yalnızca **D** tarafından kapsanız bu yüzden **D** eklenir.

Yinelemeler: Daha sonra 2, 3, 4 hedefi kapsayan testler sırasıyla ele alınır.

Kapsam sayısında eşitlik varsa özel olarak ele alınır.

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

b_5 yalnızca **D** tarafından kapsanız bu yüzden **D** eklenir.

Yinelemeler: Daha sonra 2, 3, 4 hedefi kapsayan testler sırasıyla ele alınır.

Kapsam sayısında eşitlik varsa özel olarak ele alınır.

Sonlandırma: Tam kapsama erişildiğinde

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x
E	x	x	x			x	x	x

Fikir: **Benzersiz olarak kapsanan hedefler** önce eklenir

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C** eklenir.

b_5 yalnızca **D** tarafından kapsanız bu yüzden **D** eklenir.

Yinelemeler: Daha sonra 2, 3, 4 hedefi kapsayan testler sırasıyla ele alınır.

Kapsam sayısında eşitlik varsa özel olarak ele alınır.

Sonlandırma: Tam kapsama erişildiğinde

Sonuç Kümesi: {C, D}

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

B ve **D** arasında **eşitlik çözümü**:

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

B ve **D** arasında **eşitlik çözümü**:

3 test tarafından kapsana hedeflere bak

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

B ve **D** arasında **eşitlik çözümü**:

3 test tarafından kapsana hedeflere bak

B : **b_1, b_2, b_3** vs **D** : **b_6**; o yüzden **B ekle**

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

B ve **D** arasında **eşitlik çözümü**:

3 test tarafından kapsana hedeflere bak

B : **b_1, b_2, b_3** vs **D** : **b_6**; o yüzden **B ekle**

Daha sonra, **b_6**, **3** test tarafından kapsanır: **A**, **D**, **E**

Harrold et al (1993)

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x		x		x	x
C	-	-	-	x				
D					x	x	x	x
E	x	x	x			x	x	x

b_4 yalnızca **C** tarafından kapsanız bu yüzden **C eklenir**.

Daha sonra, **b_5 2 test** tarafından kapsanır: **B**, **D**

B ve **D** arasında **eşitlik çözümü**:

3 test tarafından kapsana hedeflere bak

B : **b_1, b_2, b_3** vs **D** : **b_6**; o yüzden **B ekle**

Daha sonra, **b_6**, **3** test tarafından kapsanır: **A**, **D**, **E**

Sonuç kümesi: **{C, B, A|D|E}**

Çok-Amaçlı Yaklaşım

Genetik algoritma gibi metasezgisel yaklaşımlar Greedy Algoritmaya göre yeterli iyileşmeyi sağlar mı?

Problem genelleştirme: T' ile aynı kapsamı sağlayan ancak daha küçük bir test paketi bulunmayan bir regresyon test paketi T' bulun.

Burada iki farklı amaç fonksiyonu vardır: kapsam **maksimize** & maliyet **minimize**

Greedy'den daha esnektir: diğer kapsam kriterleri de eklenebilir

Çok amaçlı optimizasyon algoritmaları trade-off bir çözüm kümesi döndürür.

Çok-Amaçlı Yaklaşım

Pareto Dominance: Aday çözümleri karşılaştırmaya yönelik bir yaklaşım
İki aday çözüm **x** ve **y** verildiğinde, **x** tüm amaçlar üzerinde en az **y** kadar iyiyse ve **en az bir amaç üzerinde y**'den kesinlikle daha iyiyse, **x y**'ye Pareto baskındır.

ör., Yani x varken **asla** y seçilmez

İdeal olarak, Pareto Cephesini bulmak istiyoruz: Başka hiçbir çözüm tarafından Pareto baskınlığı olmayan çözümler kümesi.

Birçok meta sezgisel algoritma bulunmaktadır. İçlerinde en ünlüsü Pareto Baskın Sıralama Genetik Algoritması II (NSGA-II)'dir.

Önceliklendirme (Prioritisation)

Önceliklendirme

Amaç: İyi bir **test yürütme sırası**

- Test gereksinimlerini olabildiği **kadar hızlı elde etme**, ör., kapsam
- İdeal olarak **herhangi bir test başarısızlığının** olabildiği kadar hızlı gerçekleşmesini isteriz.

Yazılım geliştirmeyi hızlandırır: **Bir test başarısız olursa testi durdur.**

Tüm testleri yürütmeyi planlasak bile, başarısızlıkları ne kadar erken bulursak, kodu düzeltmeye o kadar erken başlayabiliriz.

Problem

Problem: Hangi testlerin önceden başarısız olacağını **bilmiyoruz**!

Yani, En iyi sıralama **bilinmiyor**

Fikir: Hatalarla ilişkili metrikleri ve geçmiş bilgisi kullanılabilir.

Başarısızlıklara yol açma olasılığı daha yüksek görülen testleri önceliklendirin.

Bu sırada da kapsımı hızlı bir şekilde **maksimize** edin.

Umuyoruz ki: **Hatalar erken bulunur** ☺

Coverage Kullanımı

Sadece kapsamı dikkate alabiliriz

Mümkün olduğunca çabuk %100 kapsama ulaşmayı hedefleyin.

Belirli bir bütçe için kapsamı maksimize edin, örneğin test sayısı

Aramayı ne zaman durdurursak durduralım **iyi** bir kapsam elde etmek için **hızlı** bir şekilde kapsama ulaşın.

Greedy Algoritma

A ile başla, sonra **D**, ve sonra **C**

%100 kapsama daha **hızlı** ulaşmak için **C** sonra **D** veya tam tersi yapılmalıdır

Test\branch	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
A	x	x	x			x	x	x
B	x	x	x				x	x
C	x	x	x	x				
D					x	x	x	x

Hata Tespit Verilerini Kullanma

Hata Tespit Verilerini Kullanma

Mutasyon testi verilerini kullanın!

Hata Tespit Verilerini Kullanma

Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

Hata Tespit Verilerini Kullanma

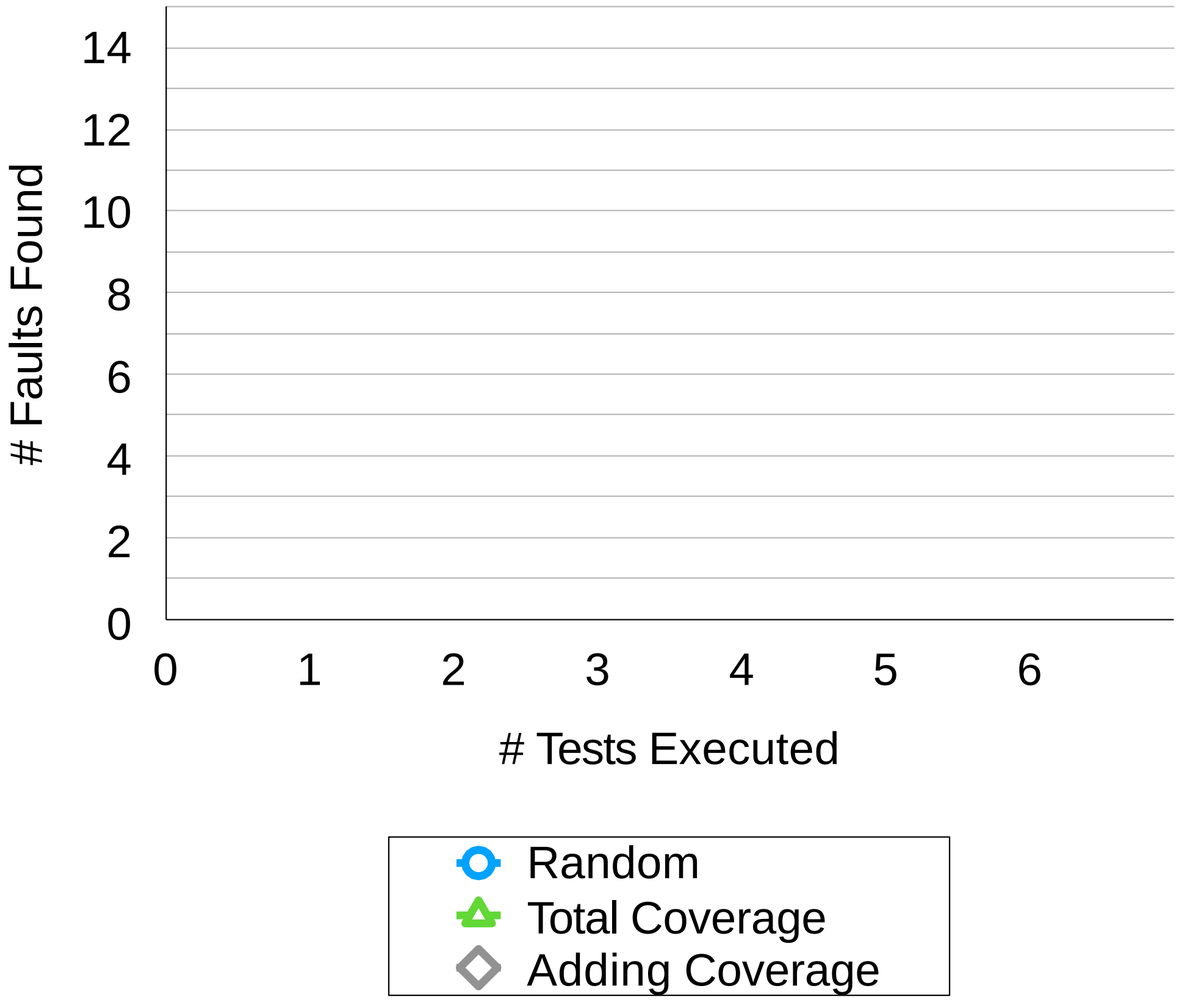
Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13’ünü

öldürmüş olsun

Hata Tespit Verilerini Kullanma



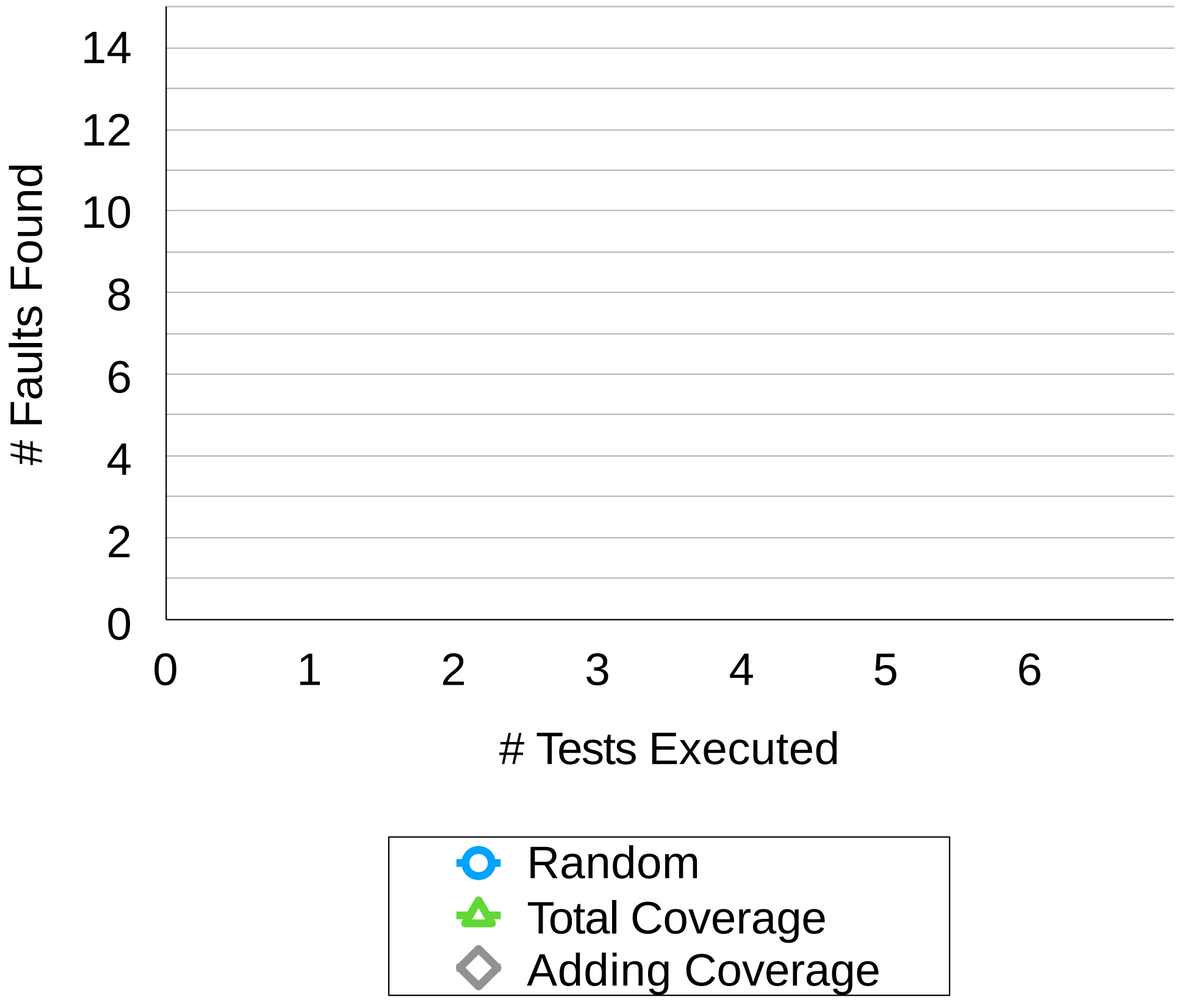
Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

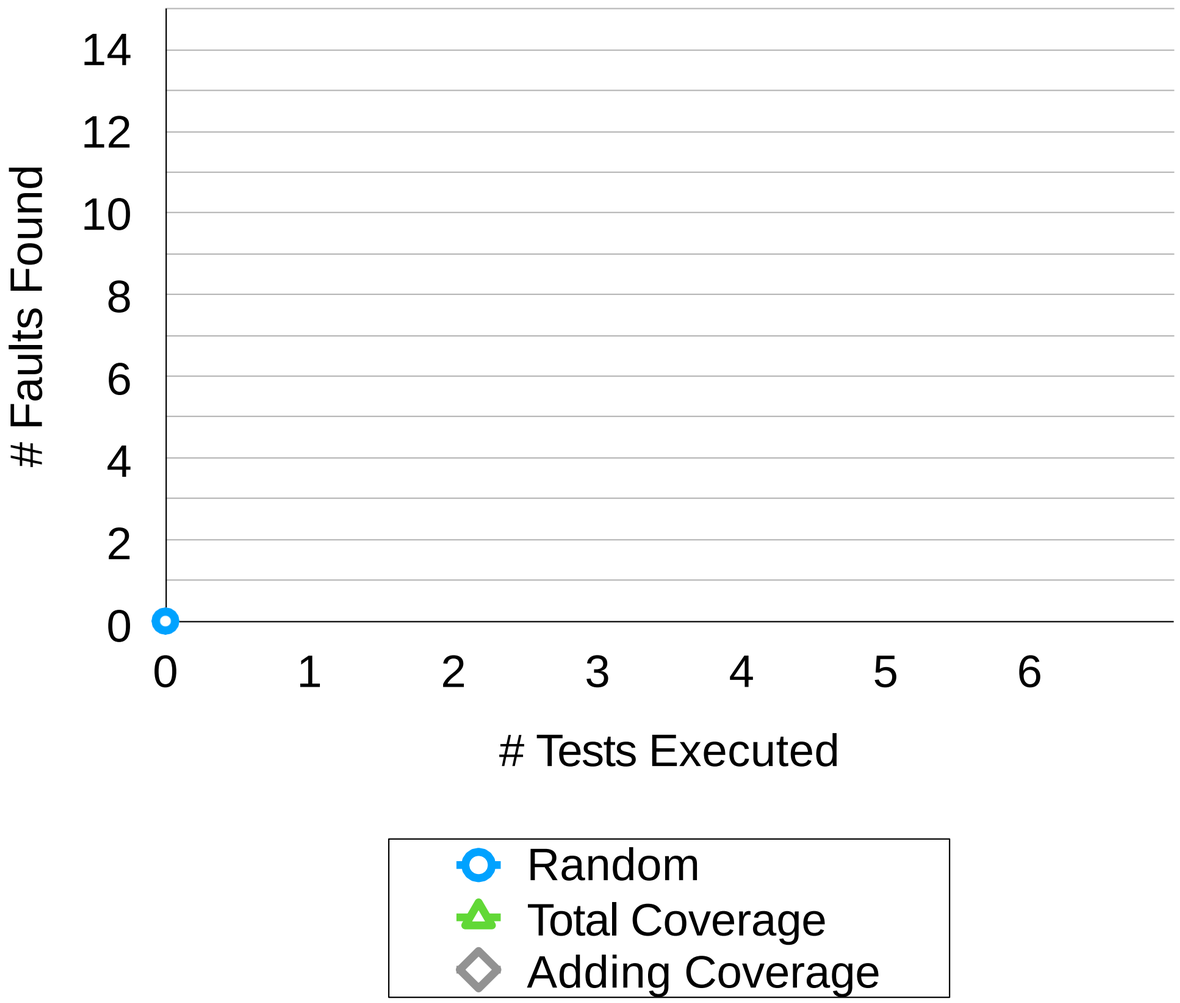
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

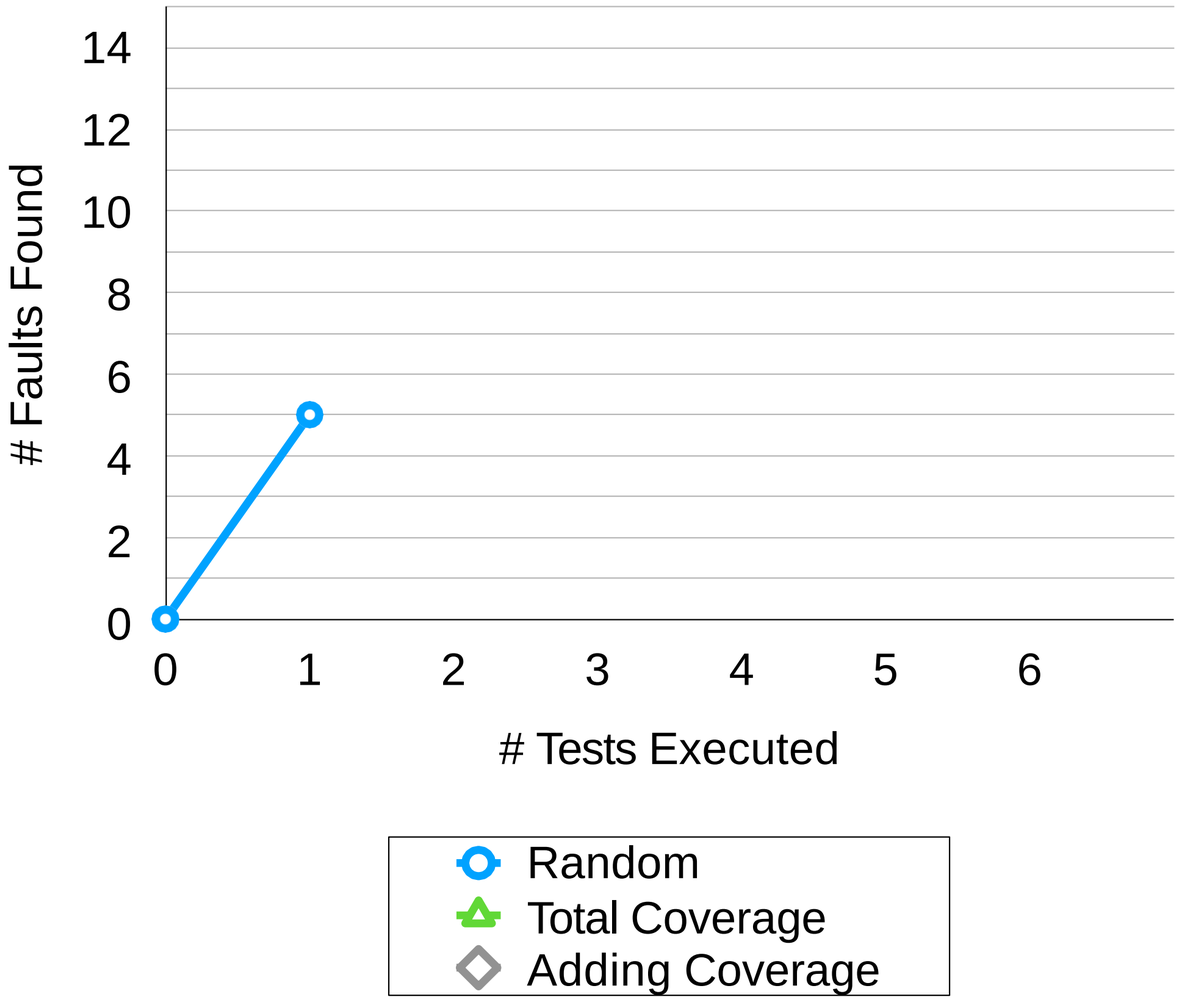
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

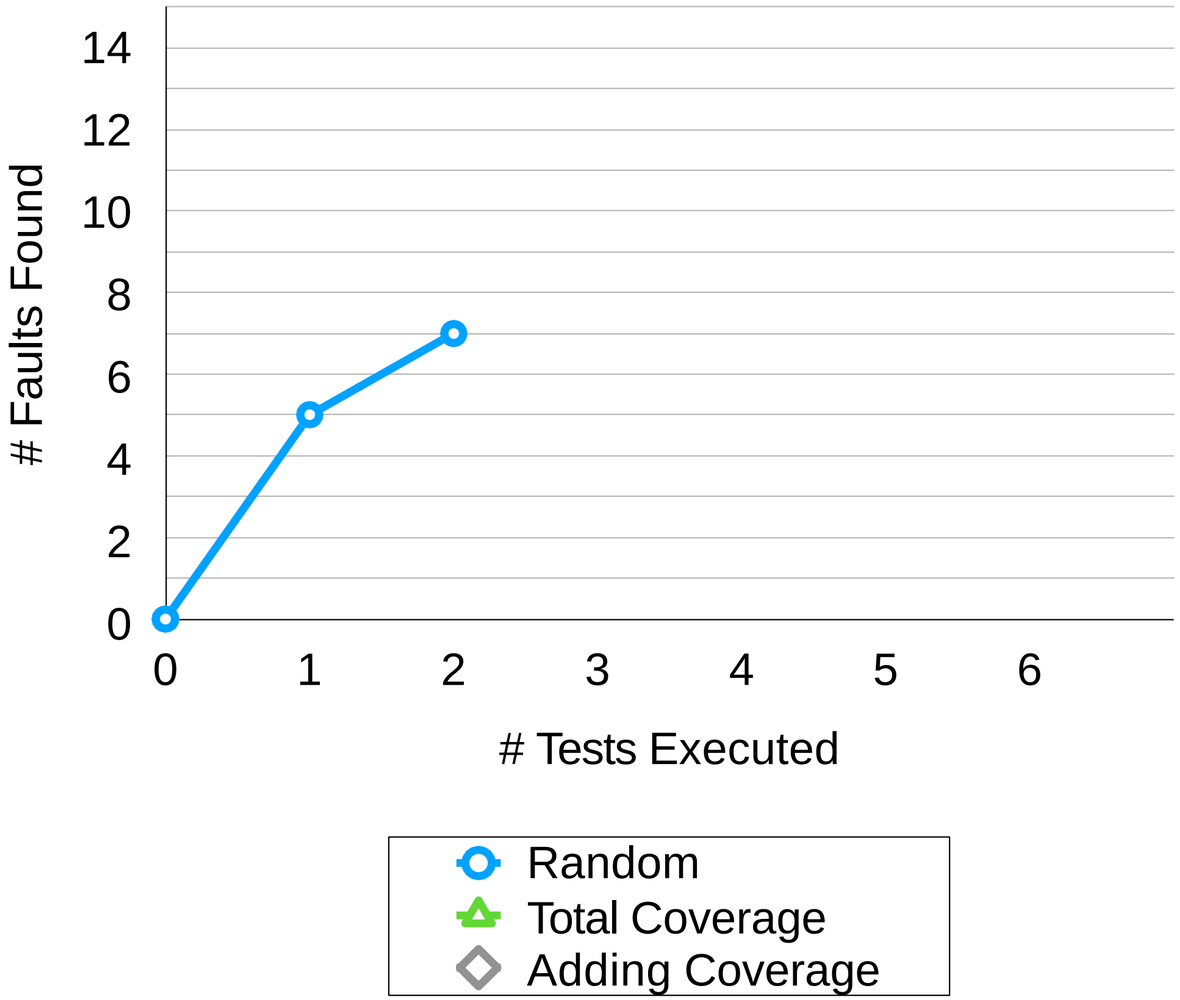
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

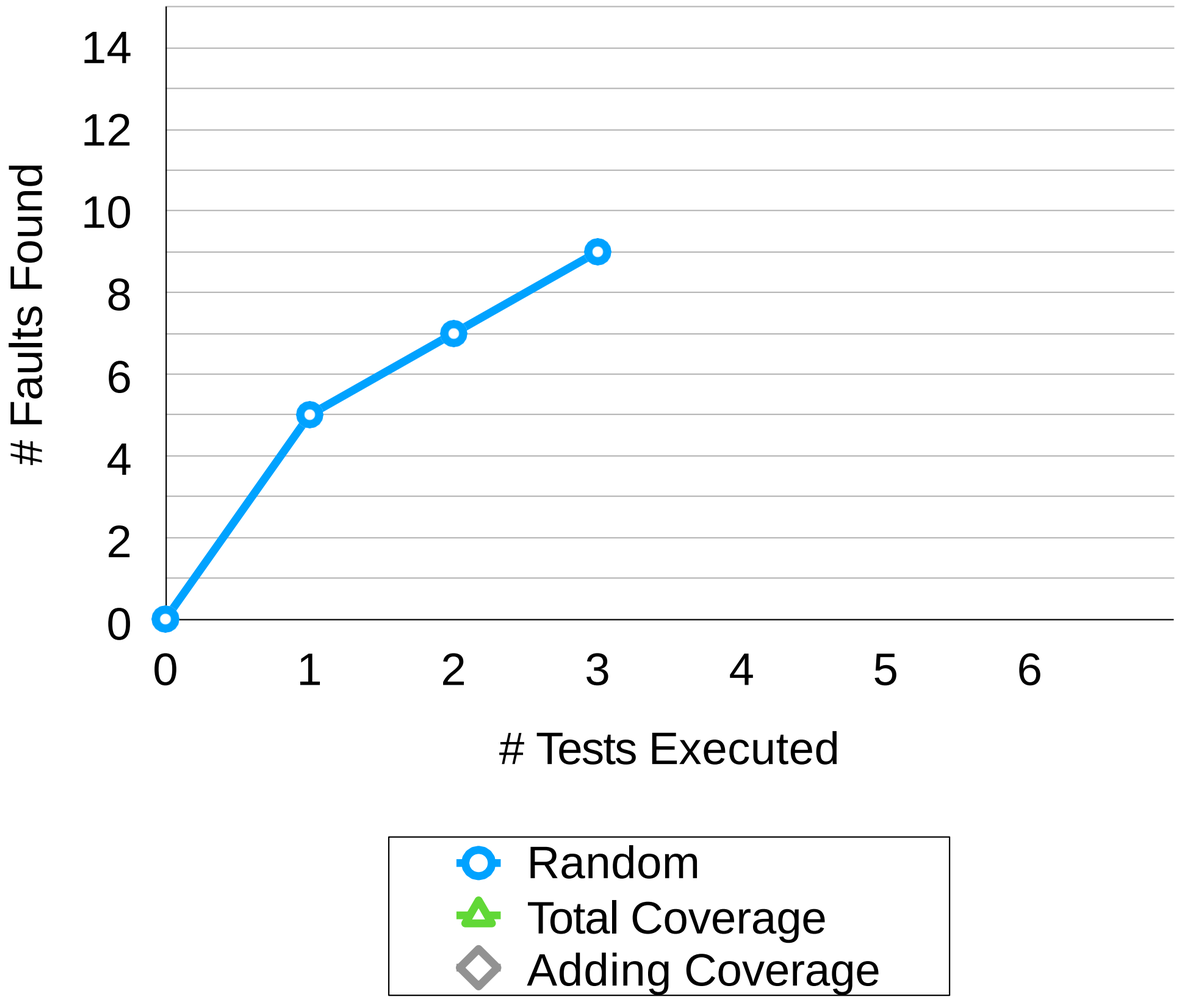
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

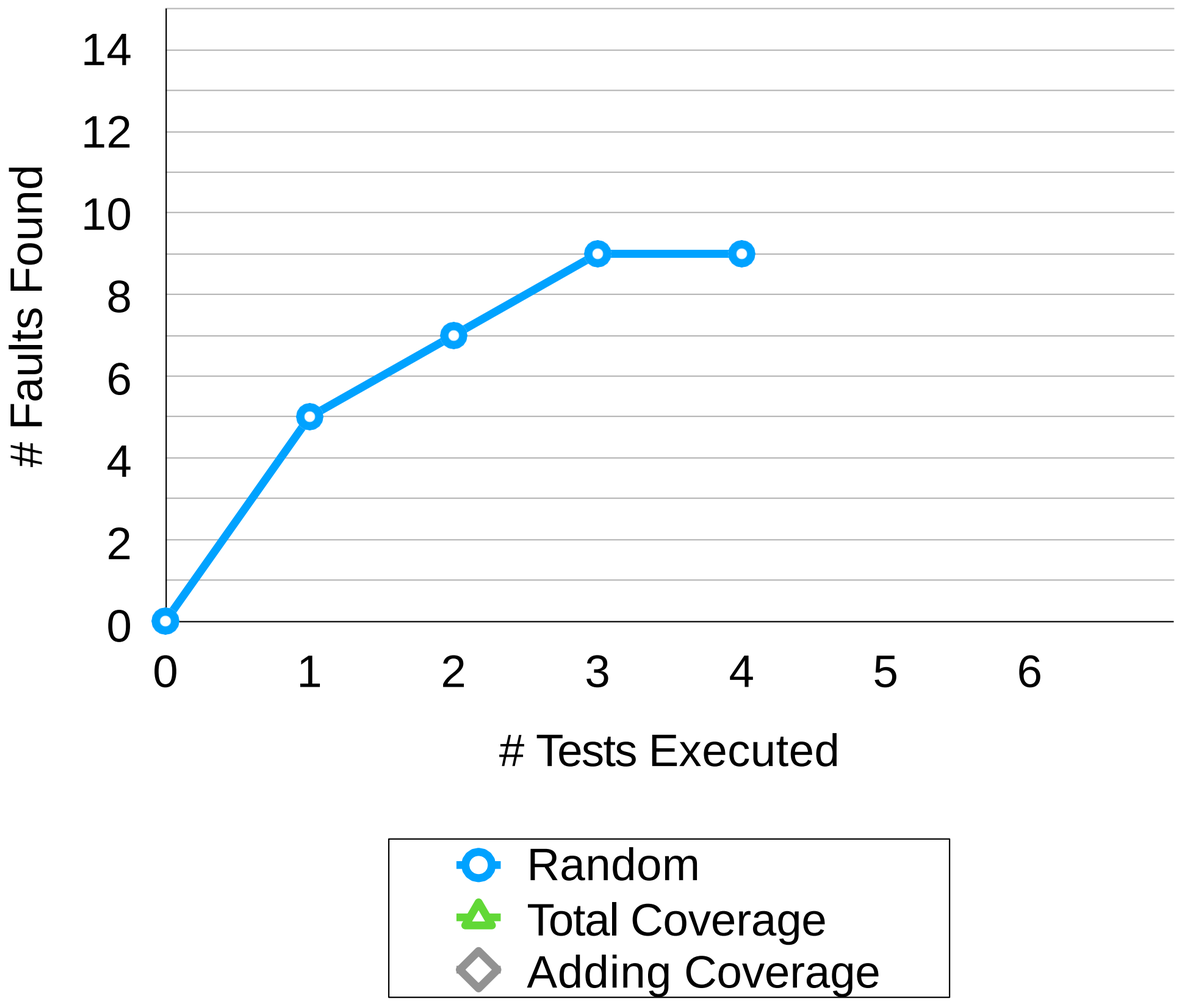
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

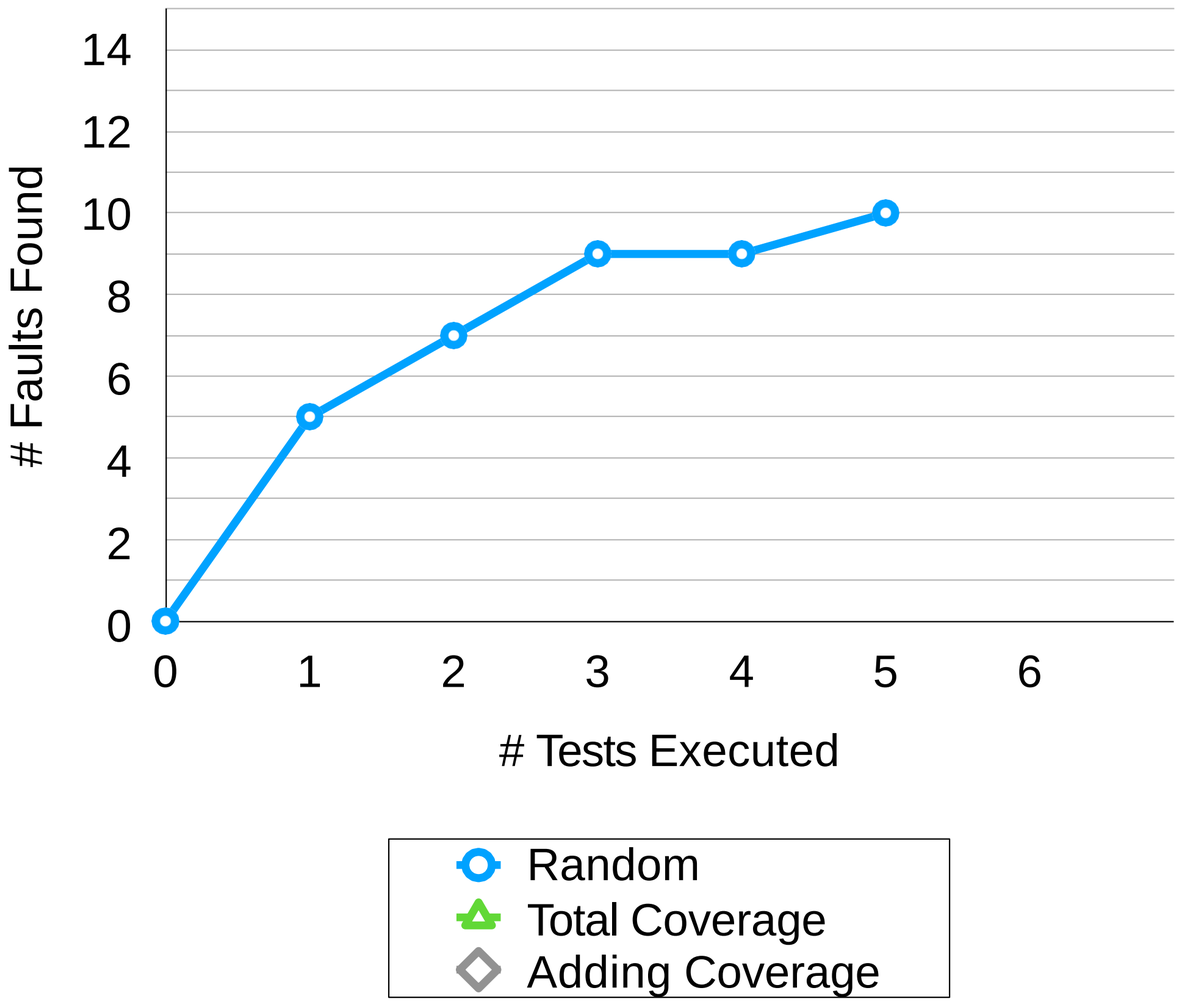
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

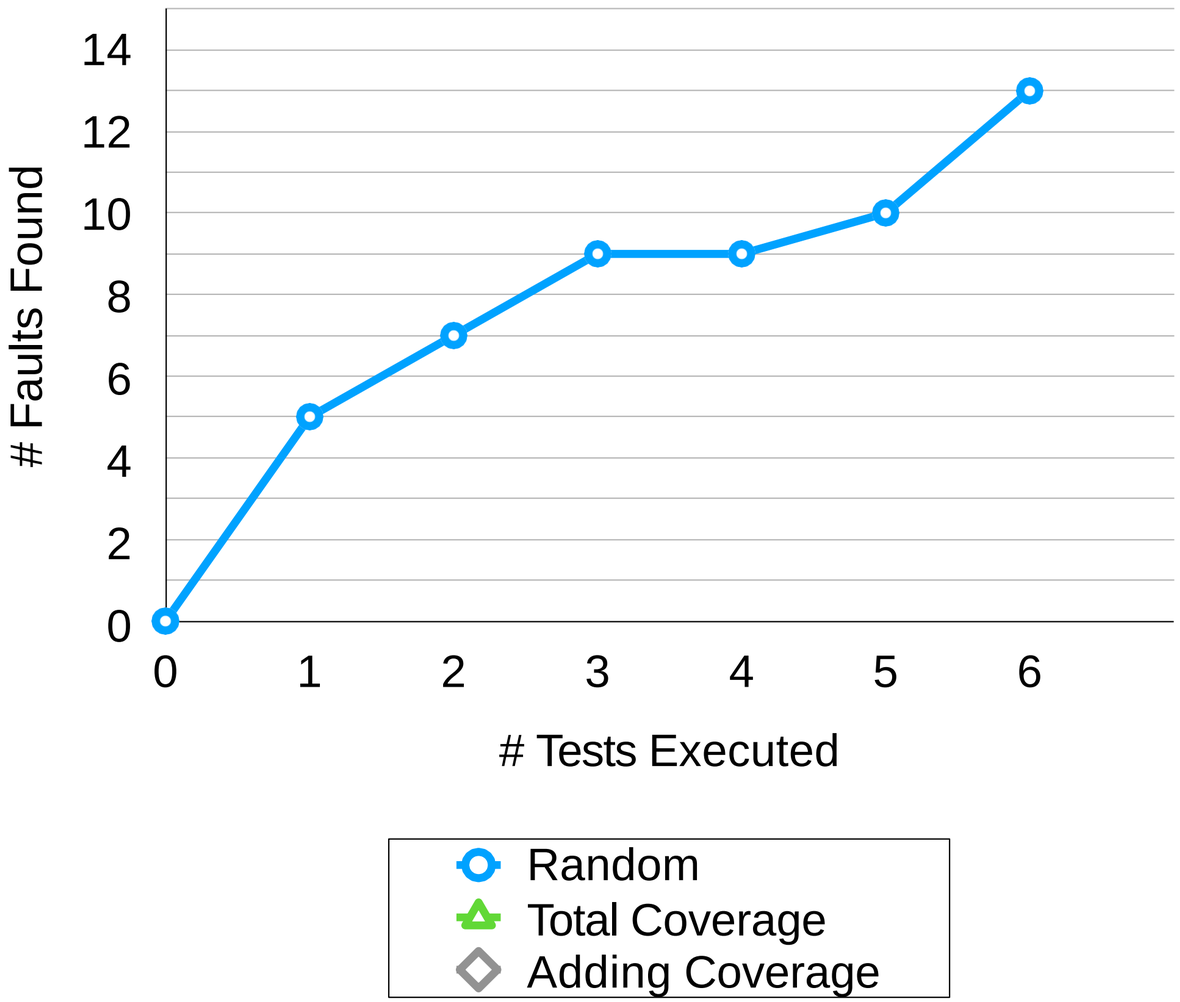
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

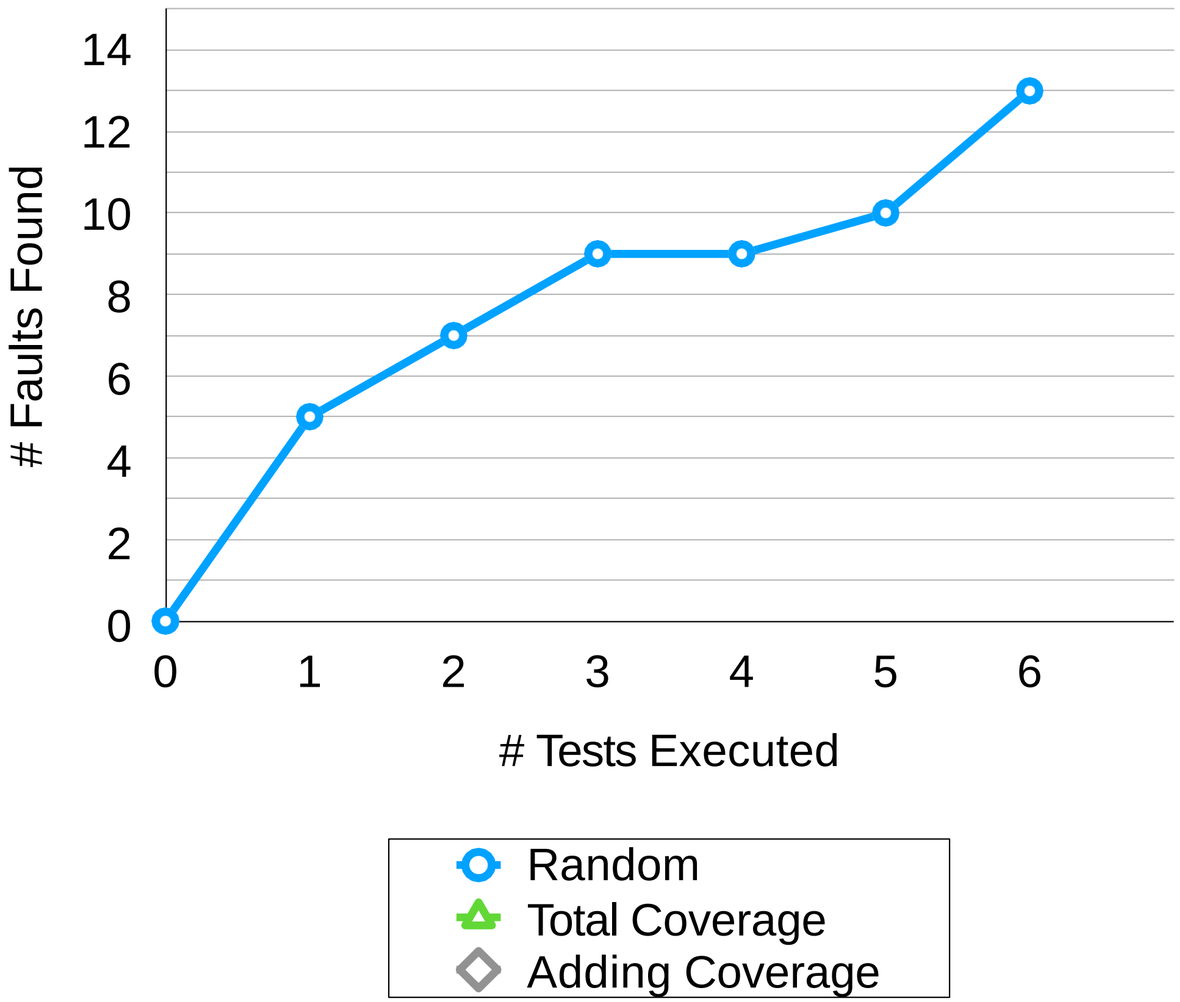
Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

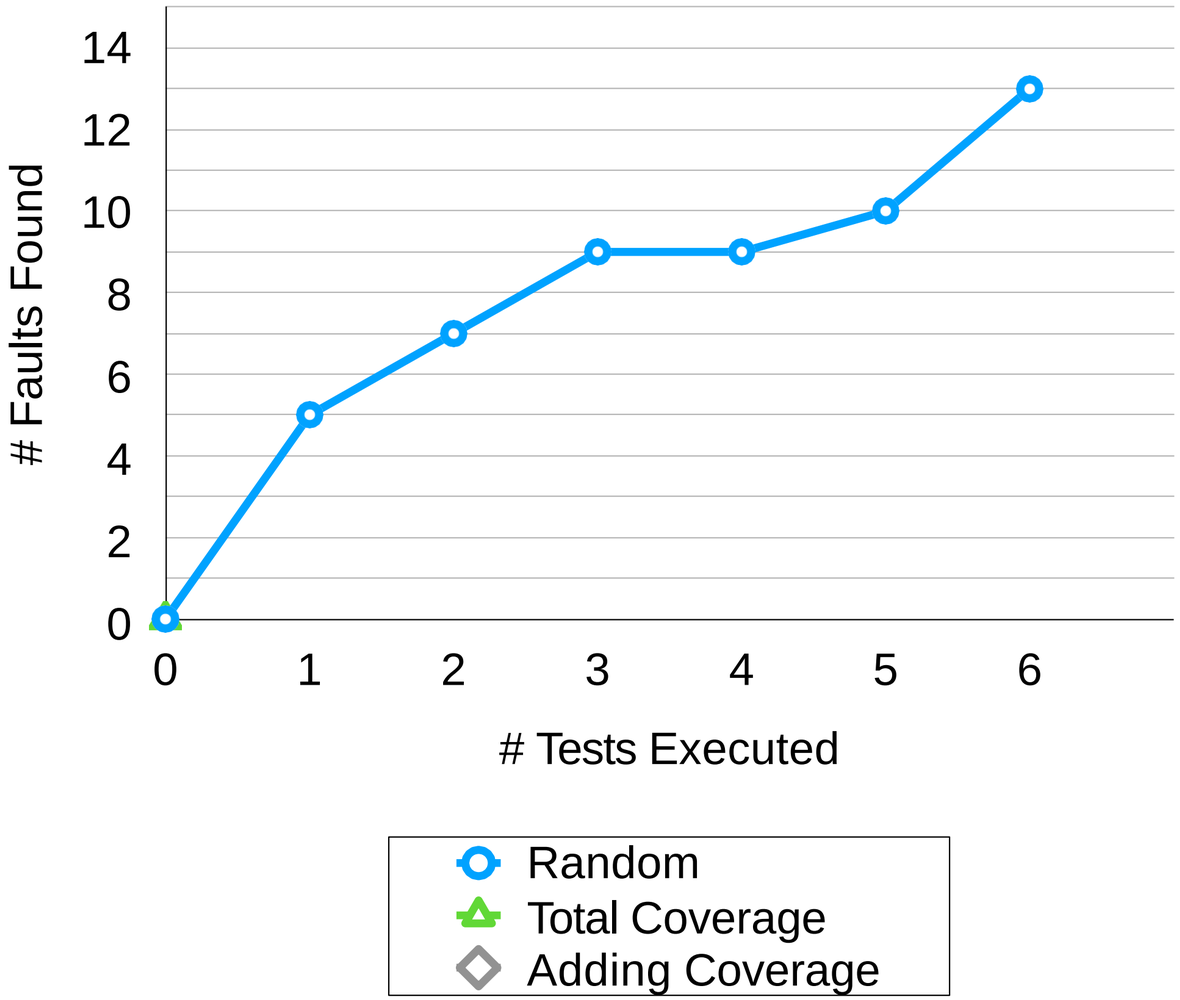
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

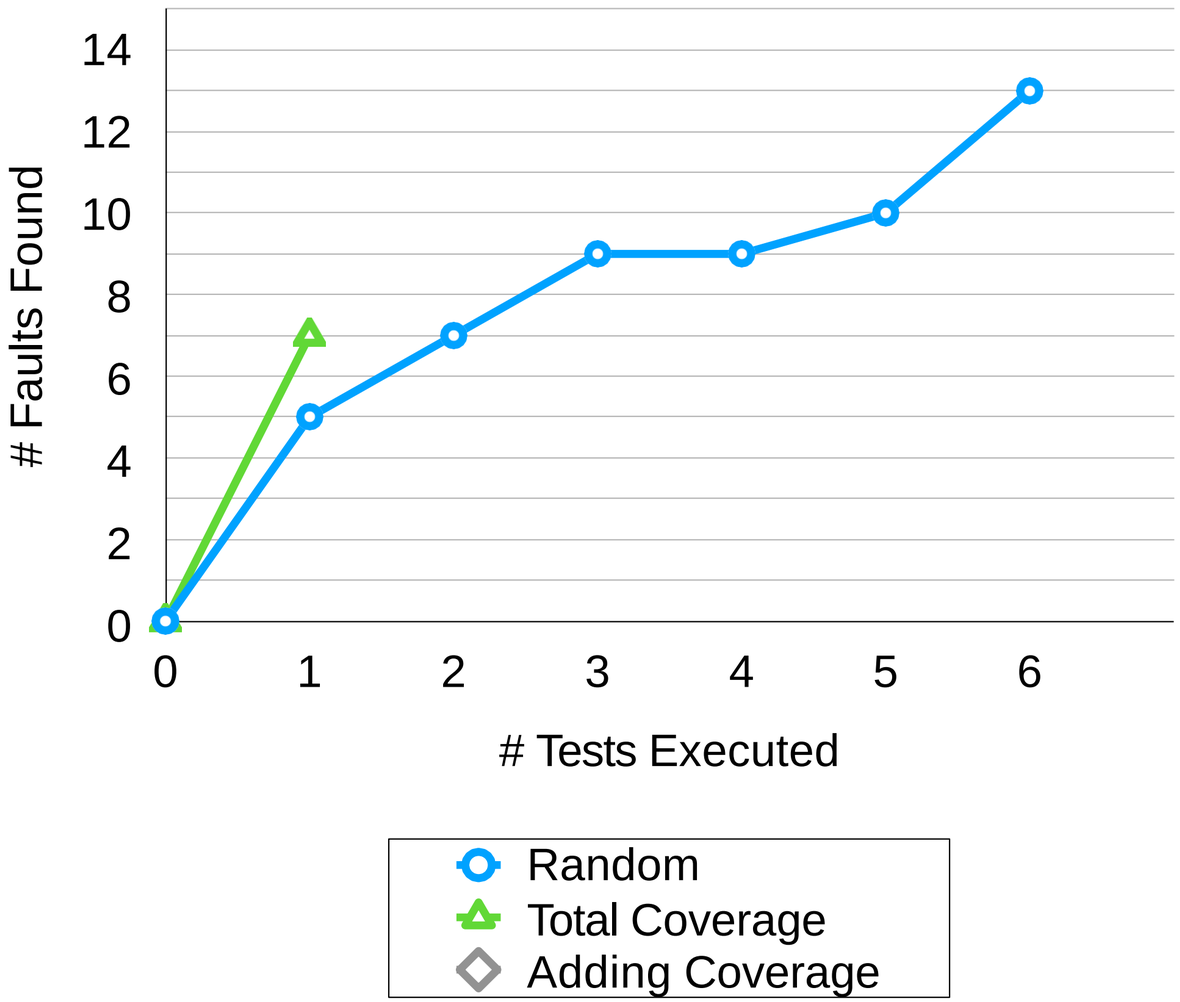
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

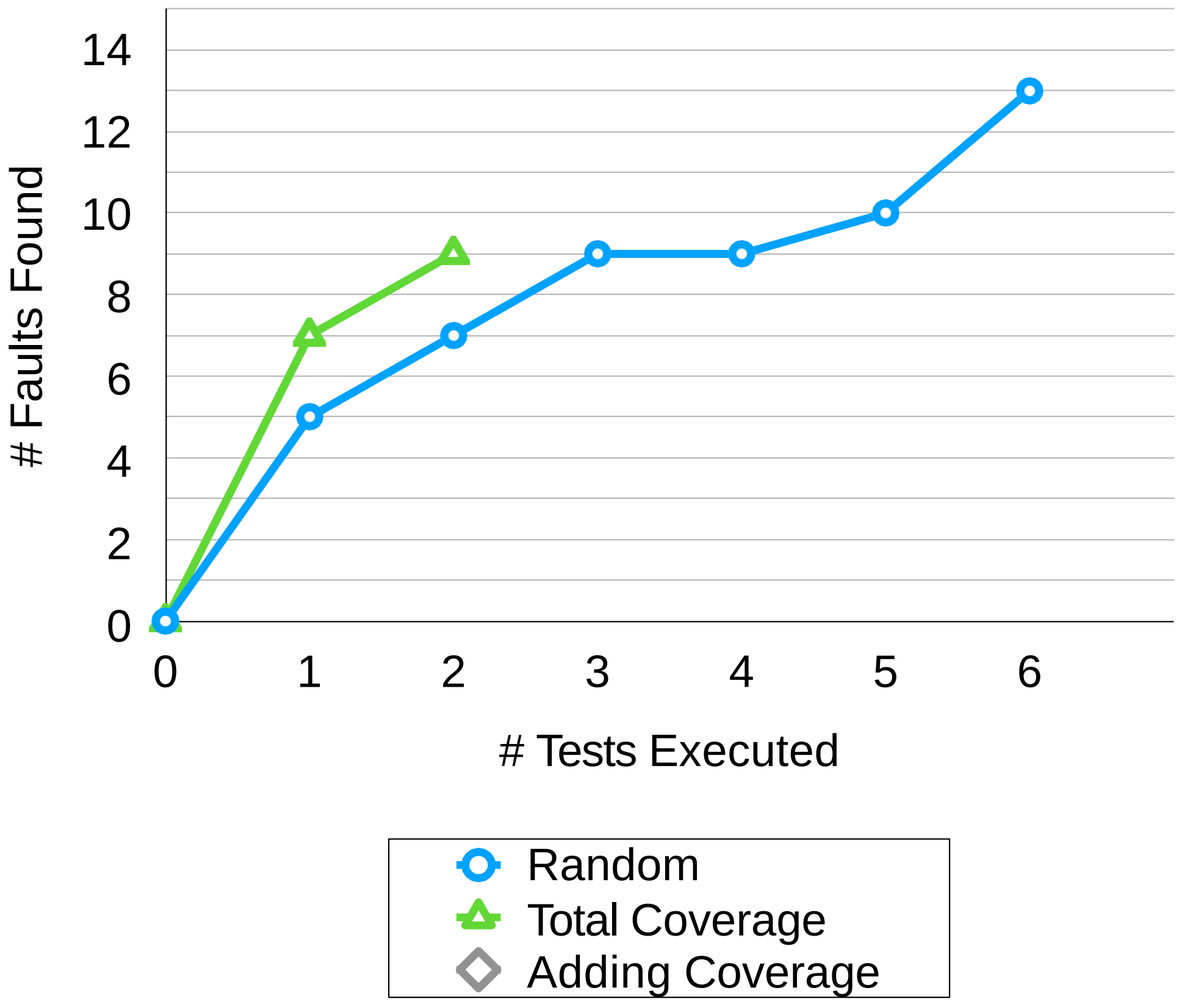
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

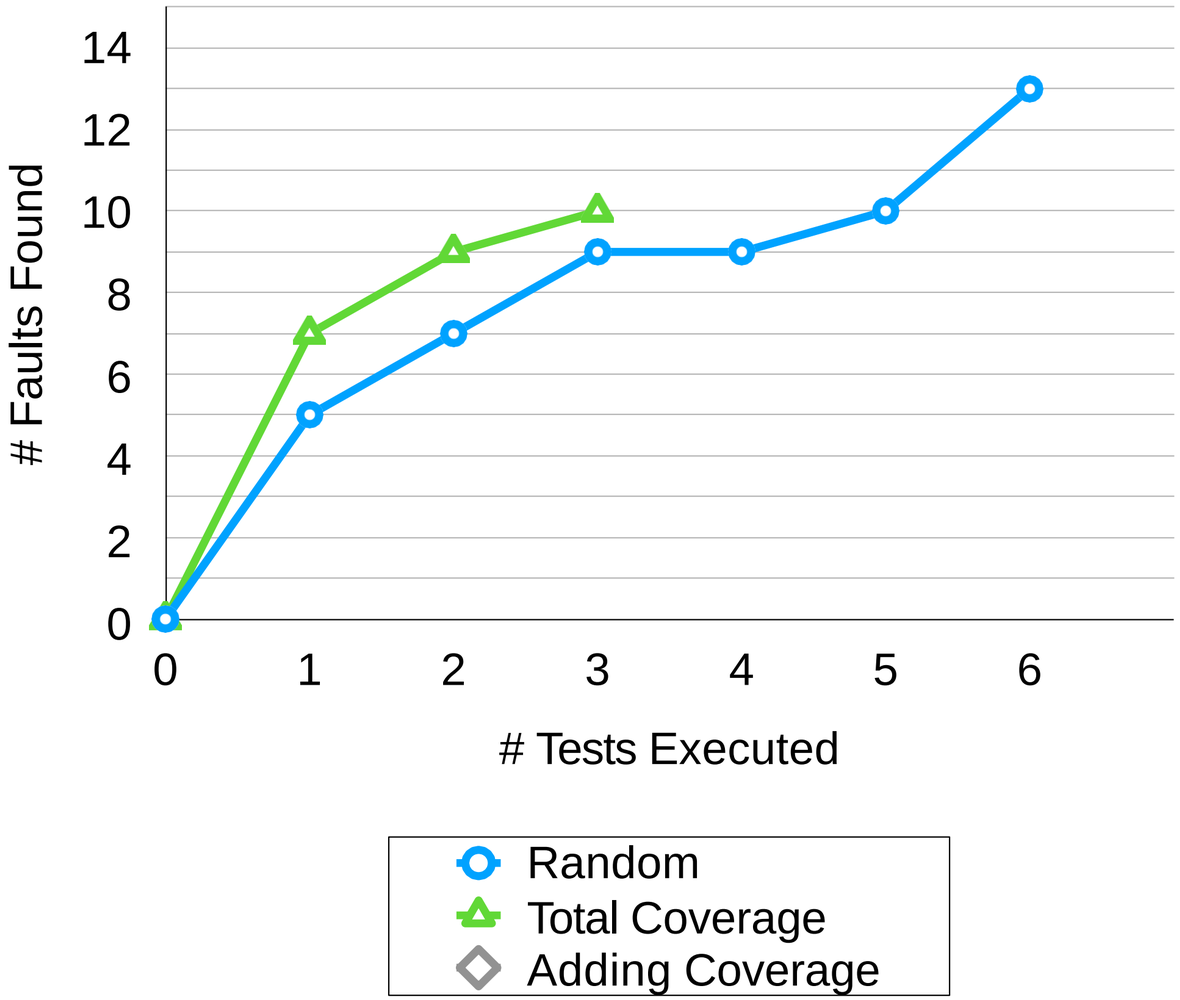
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

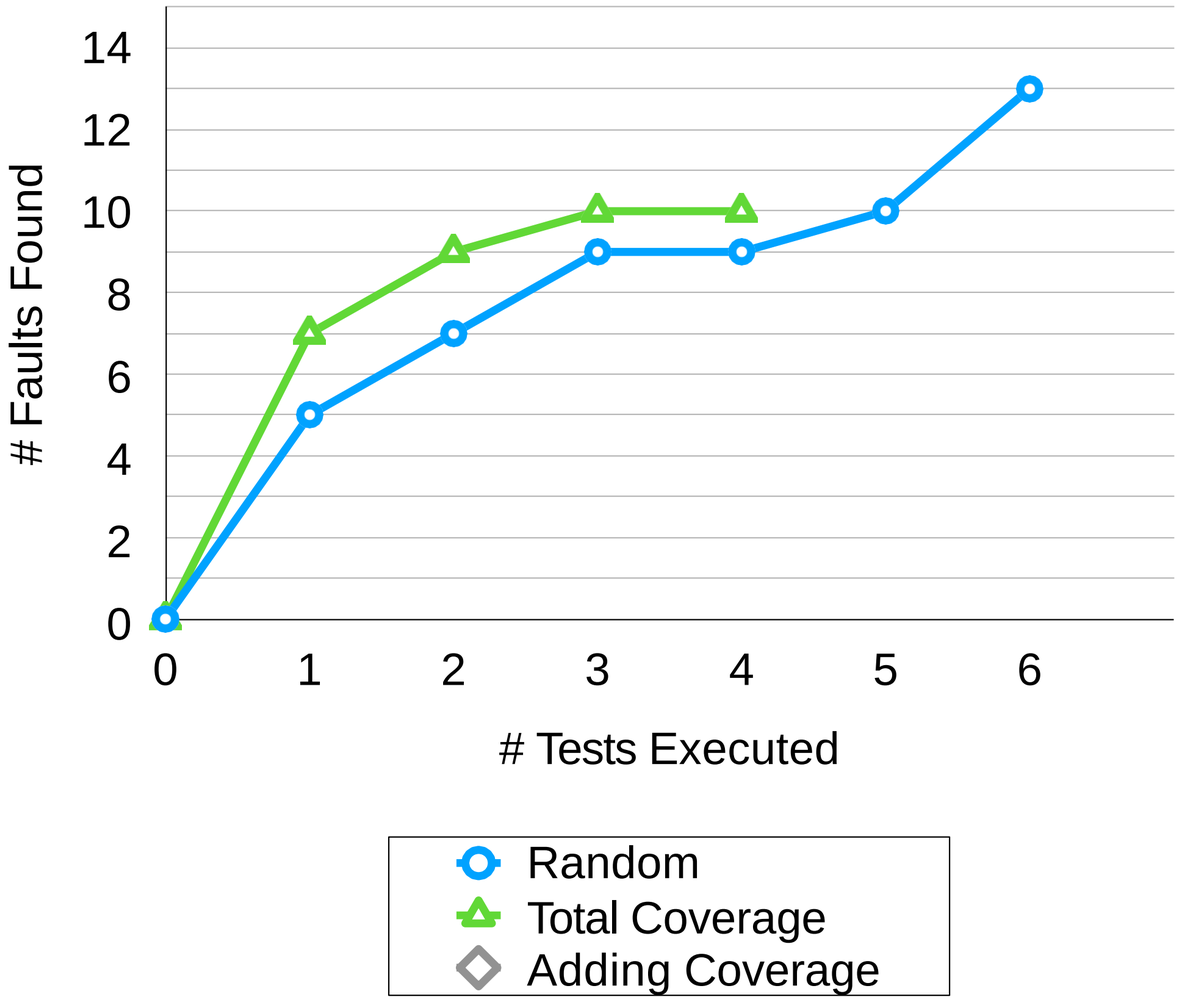
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

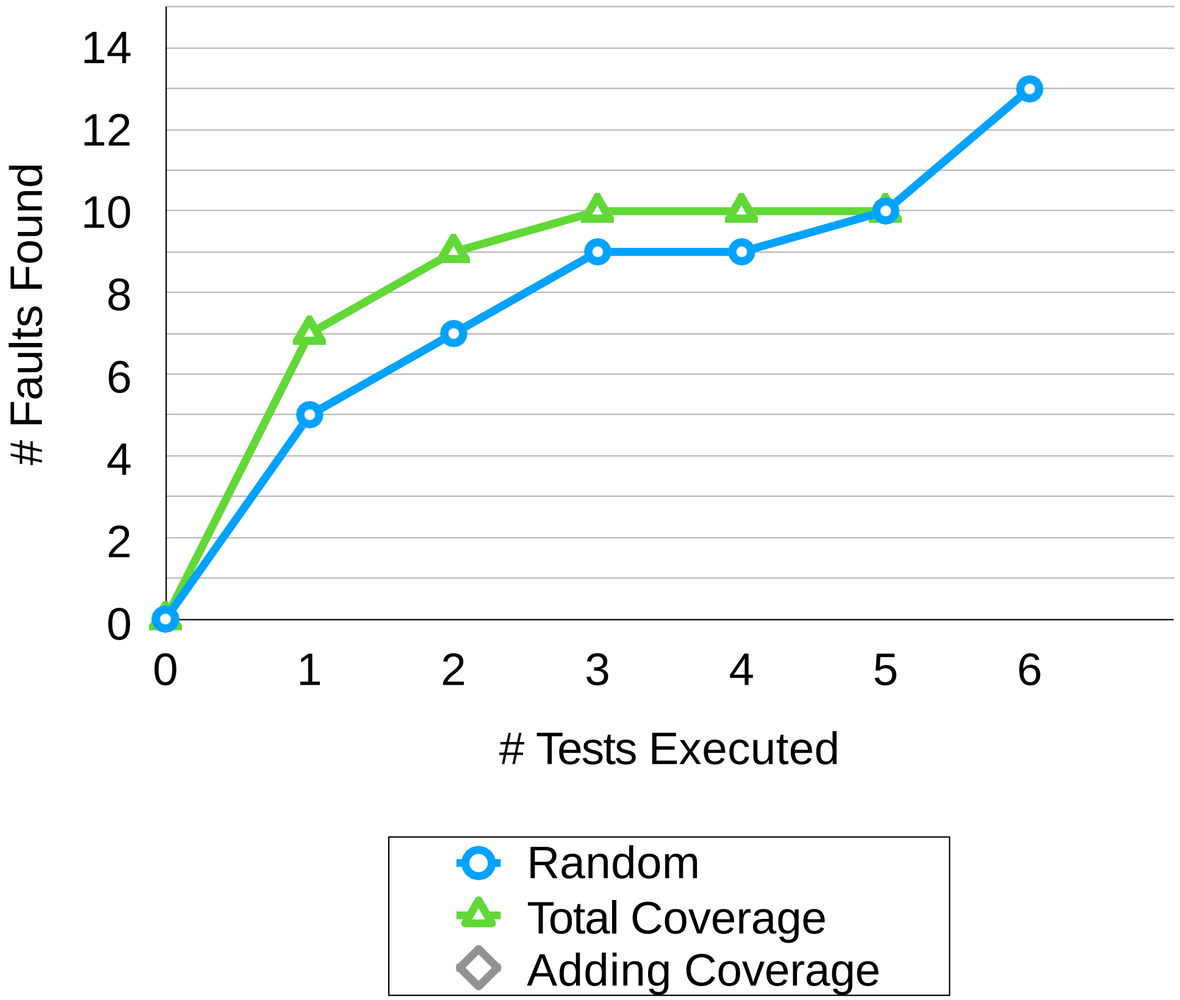
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

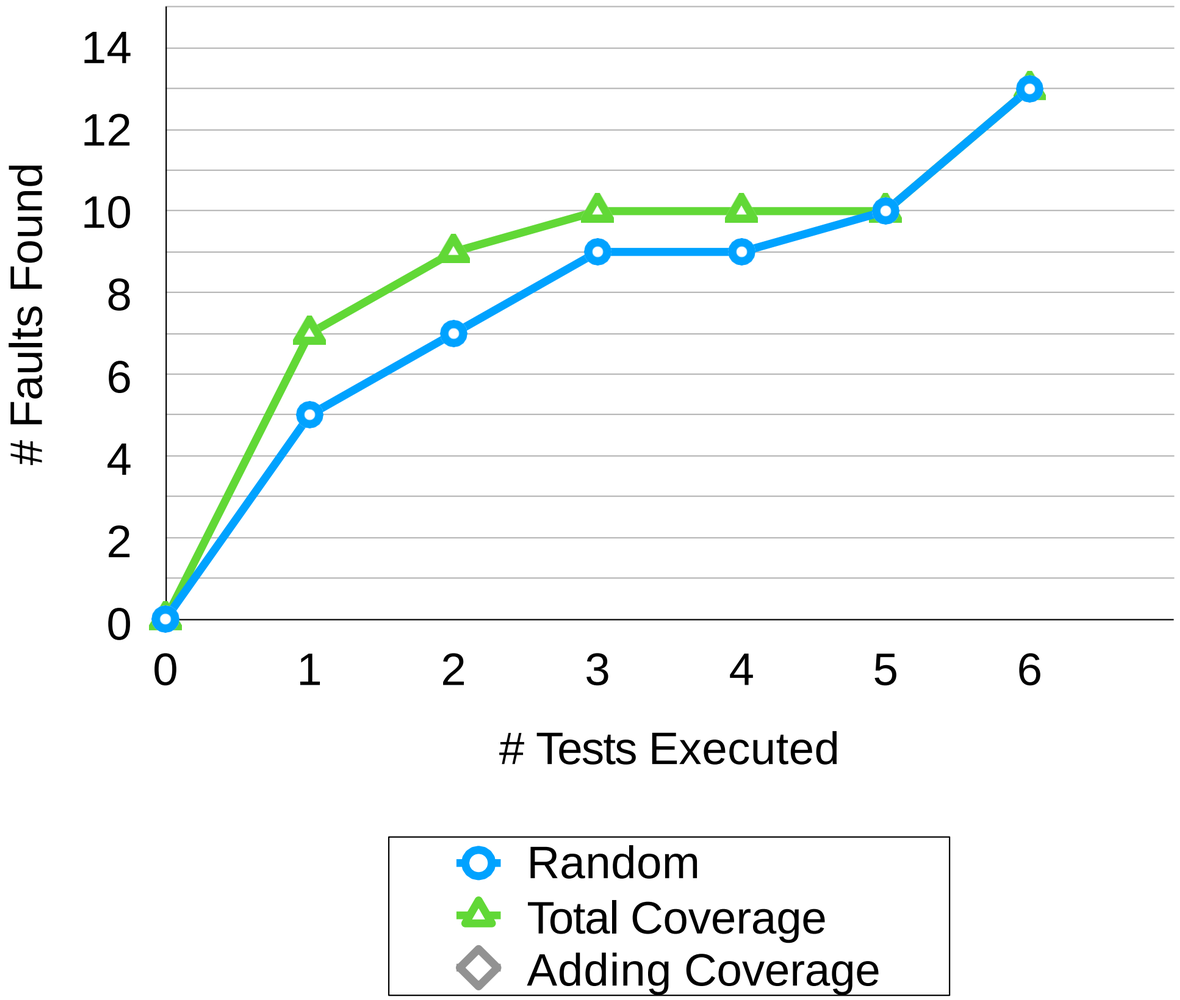
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

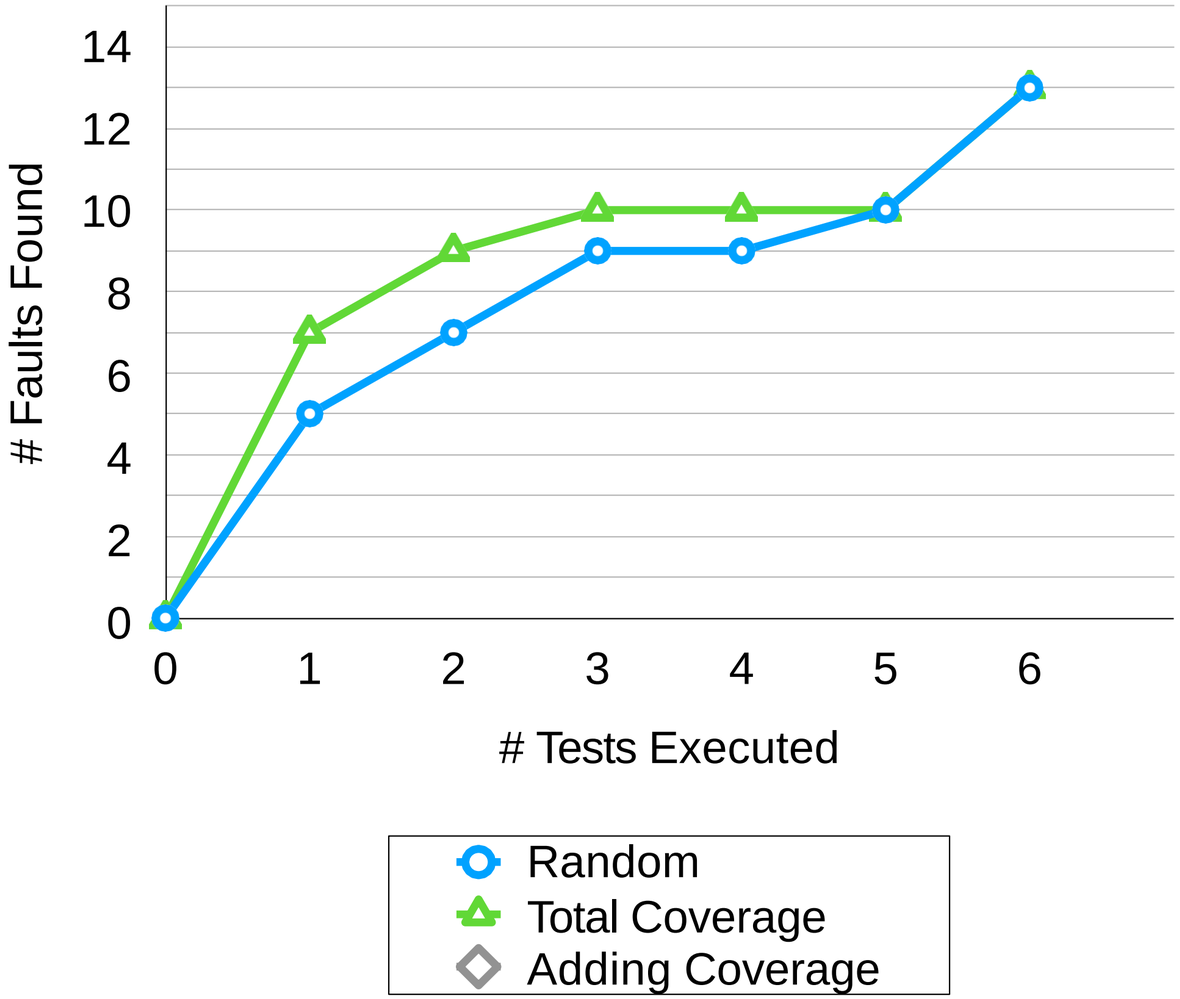
A-F testleri onlardan 13'ünü

öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

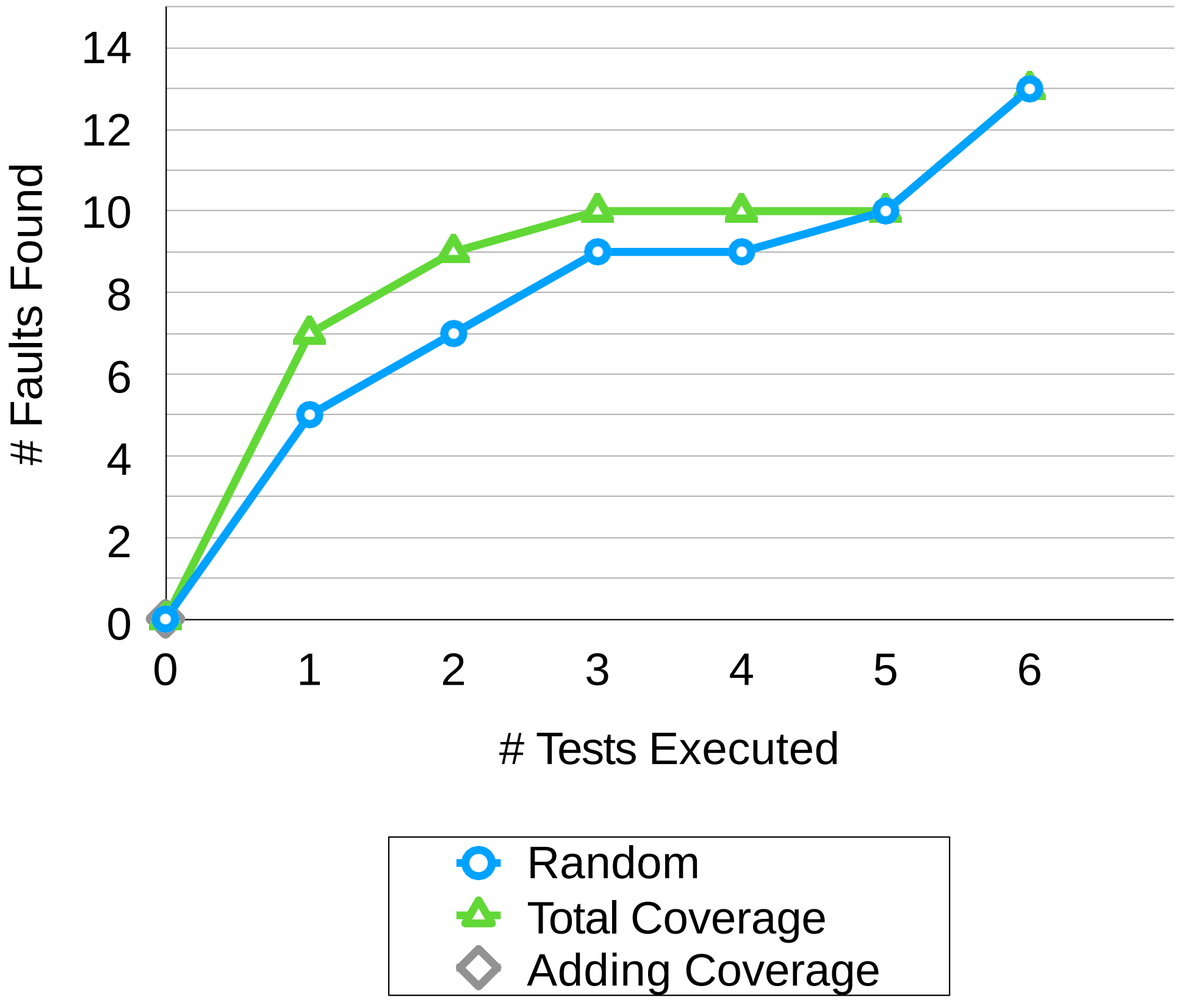
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

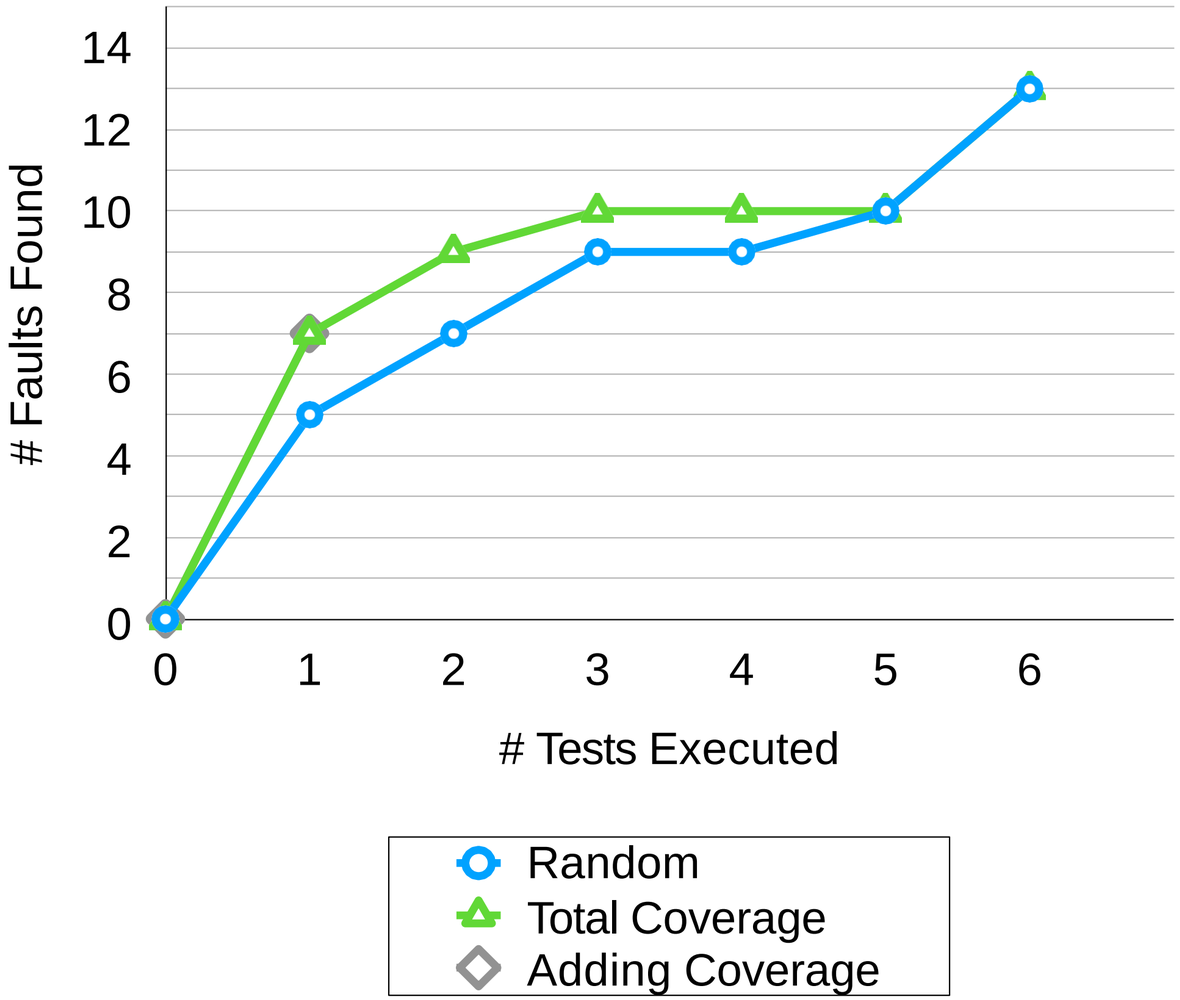
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

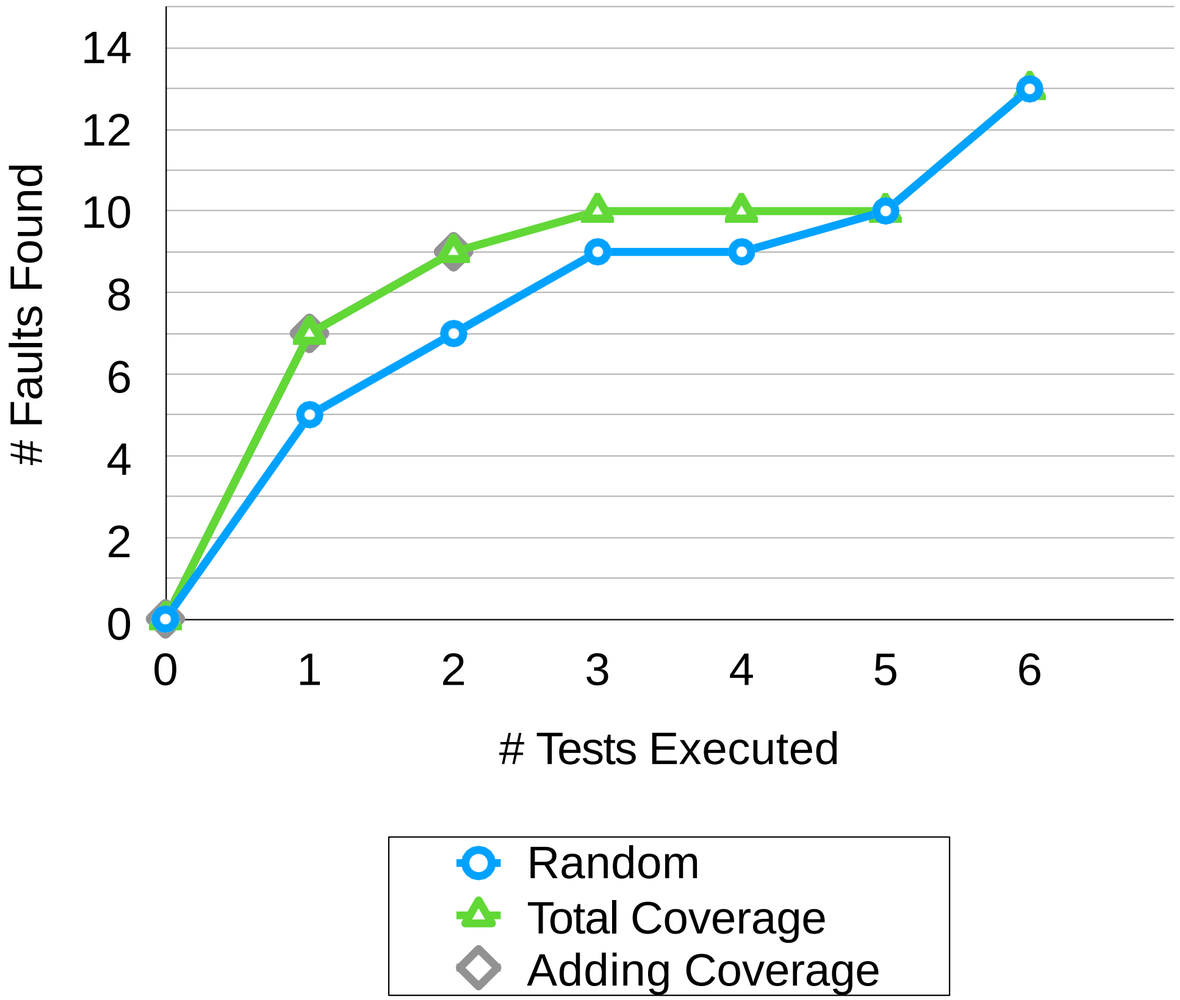
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

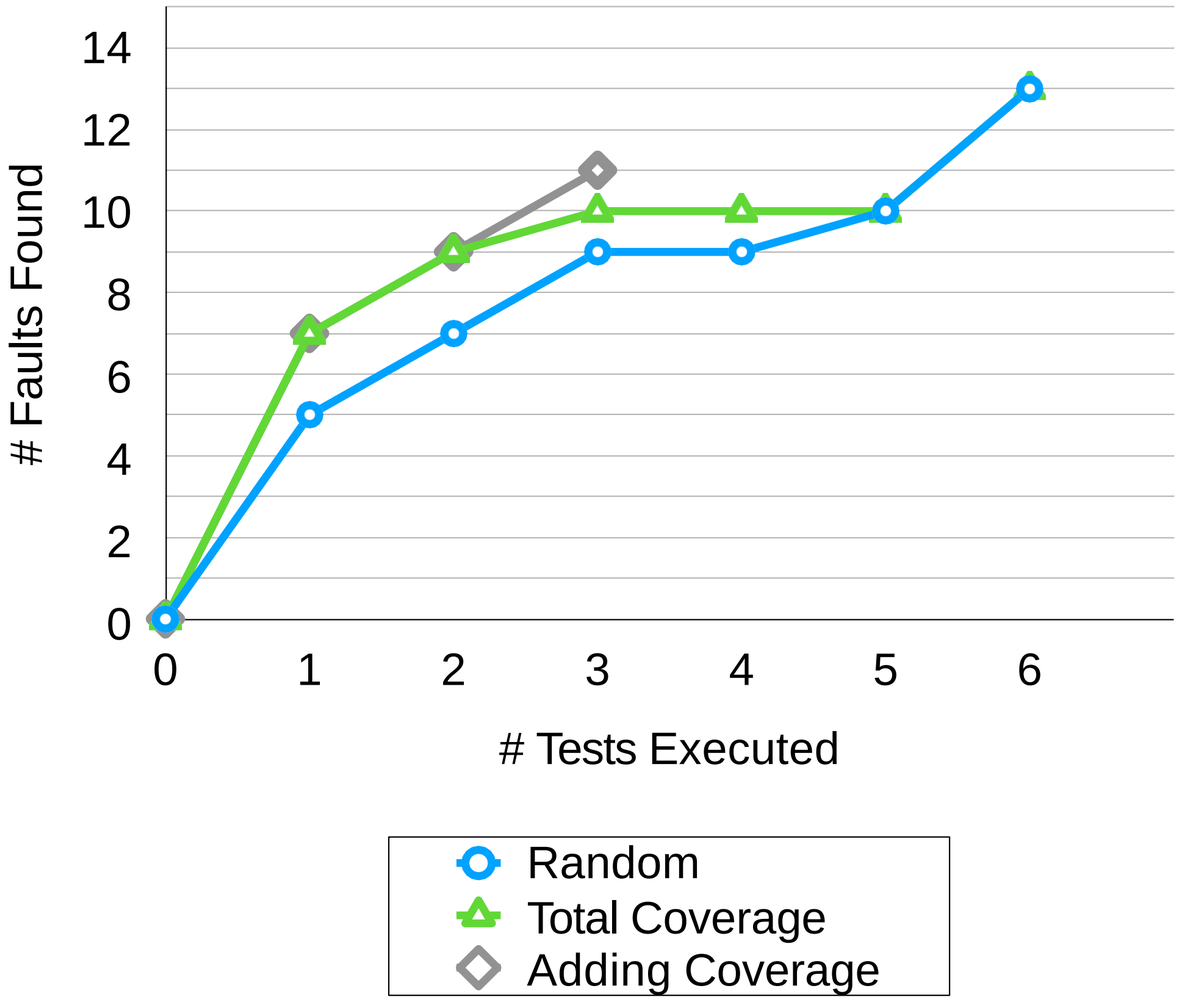
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

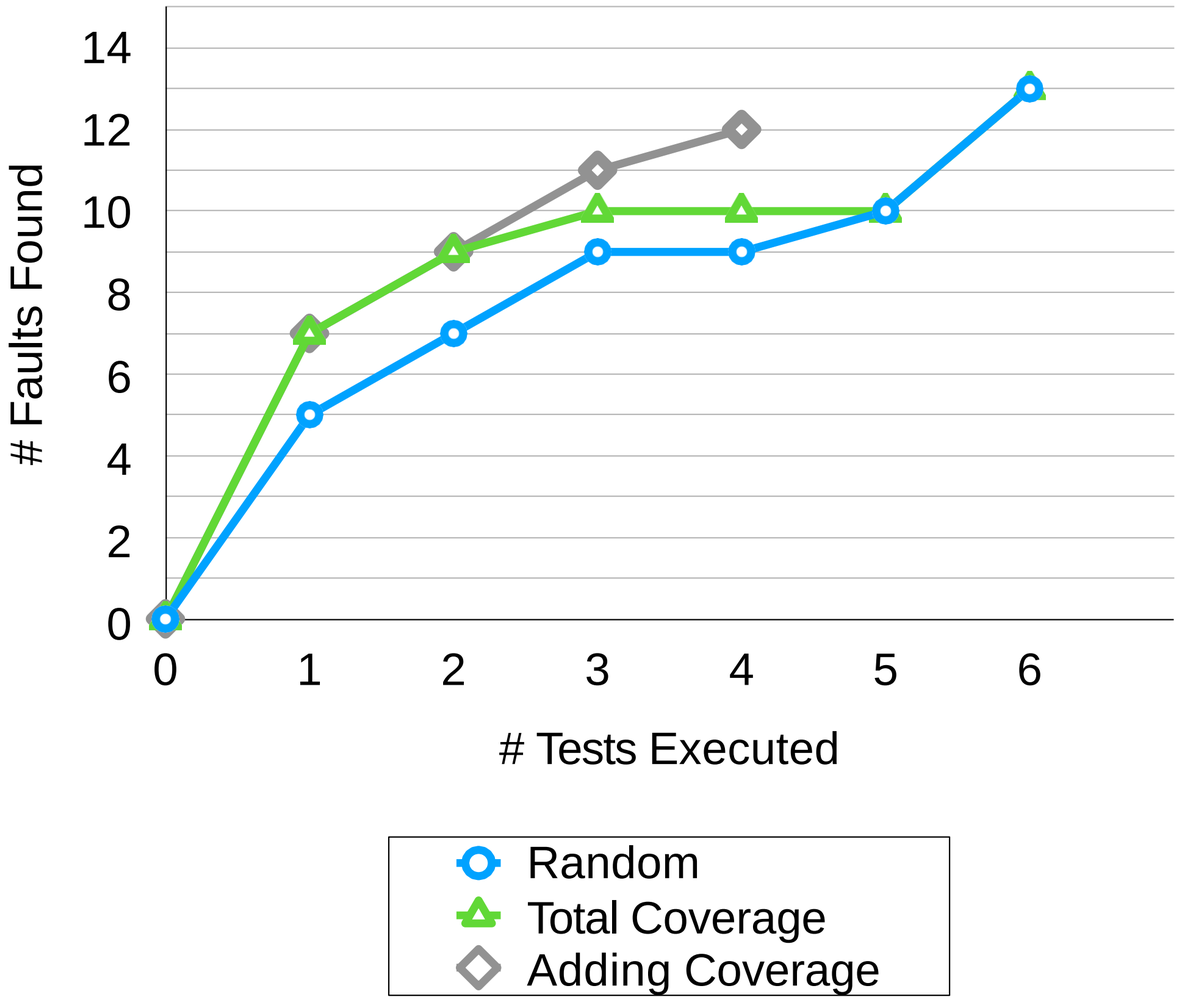
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

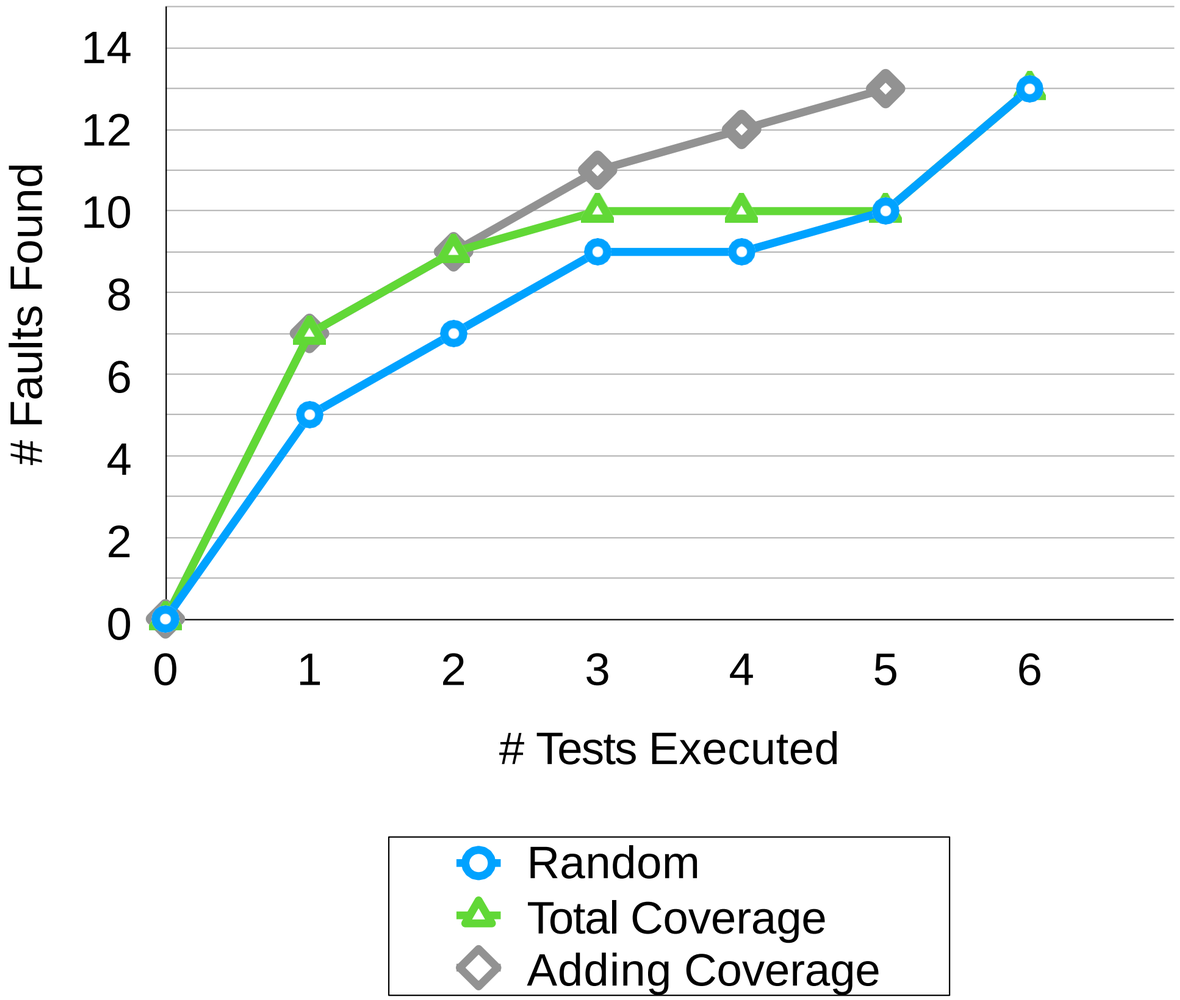
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

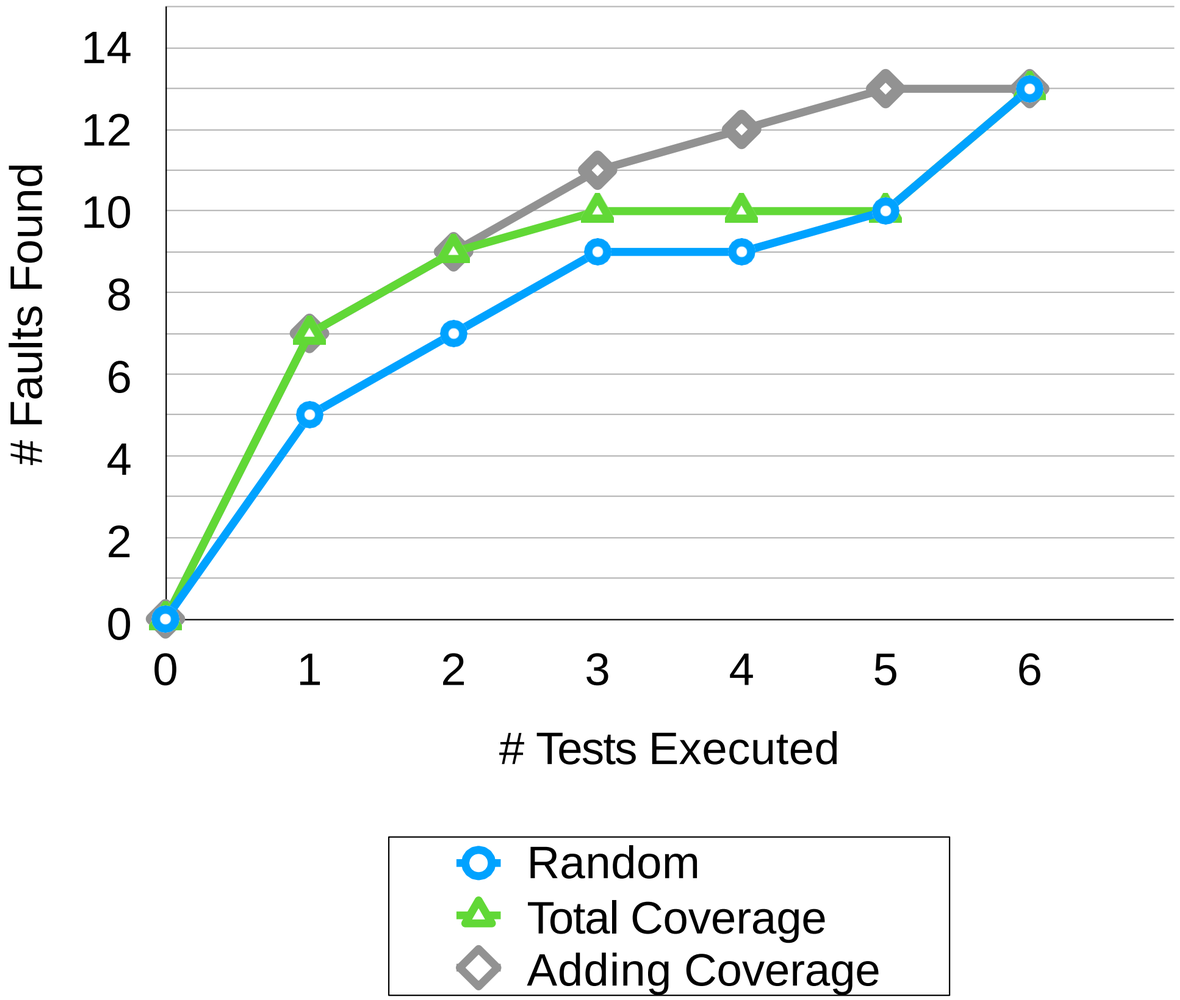
öldürmüş olsun

Rastgele: E-A-B-F-D-C

Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma



Mutasyon testi verilerini kullanın!

Toplamda 15 Mutant!

A-F testleri onlardan 13'ünü

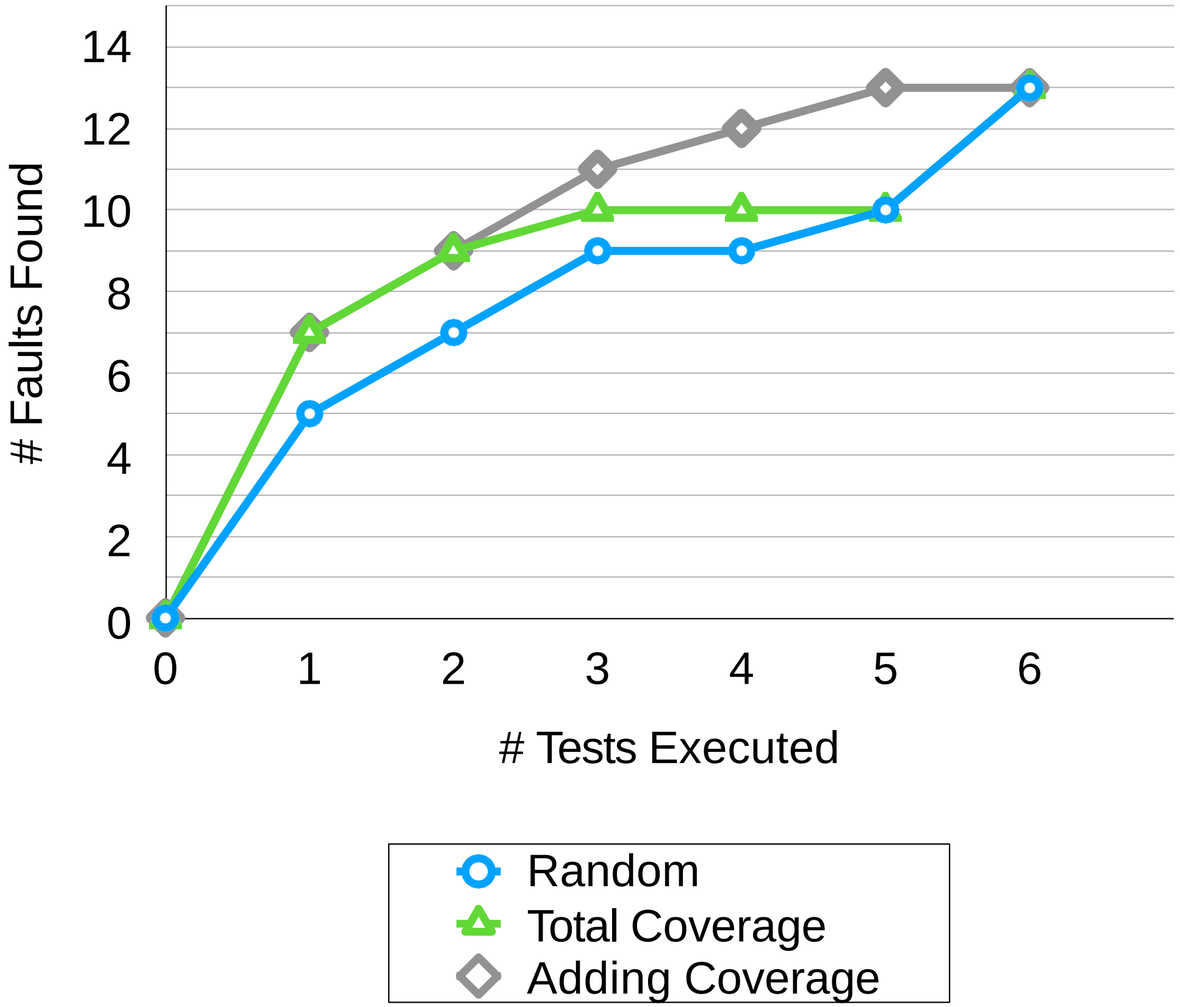
öldürmüş olsun

Rastgele: E-A-B-F-D-C

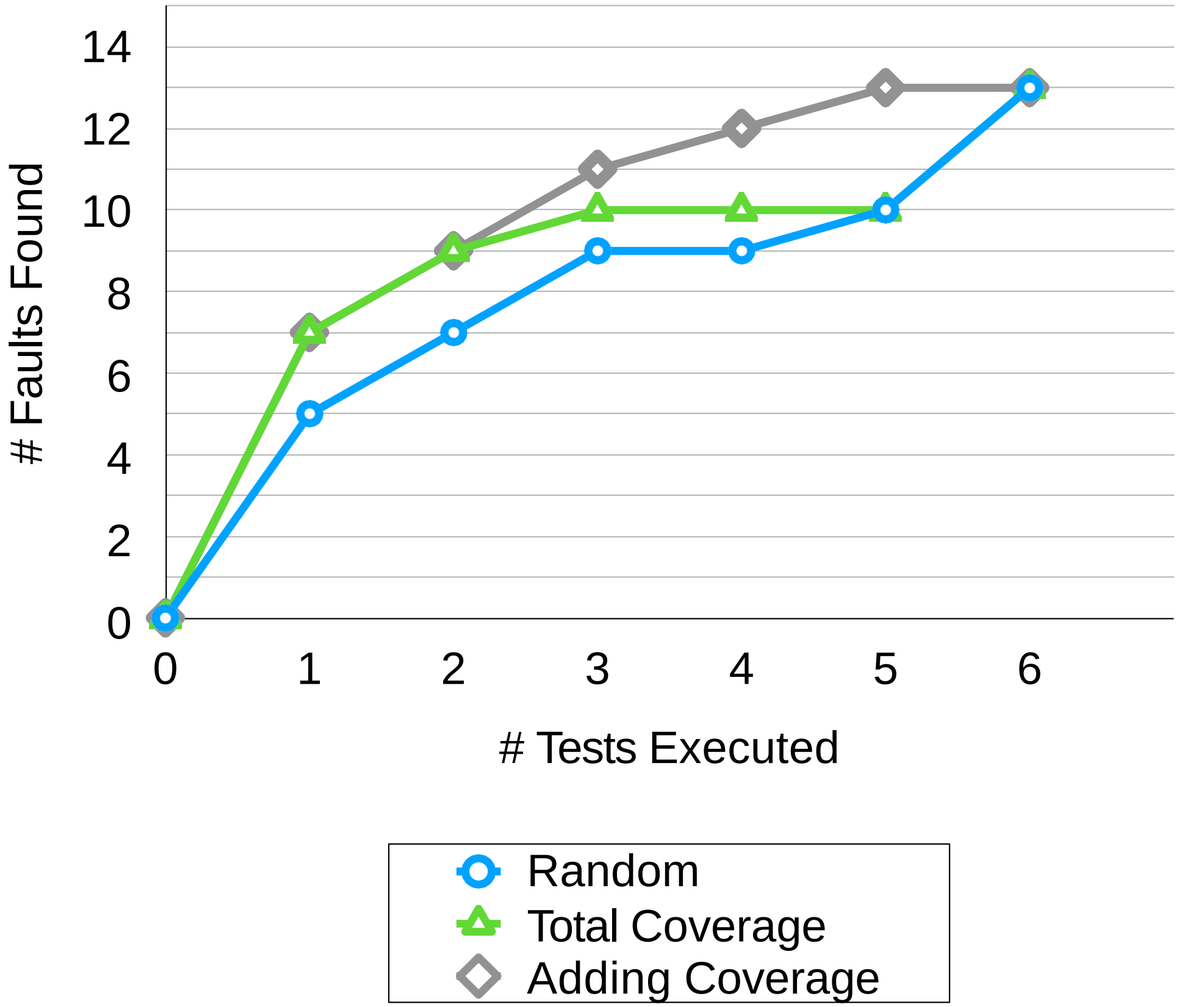
Total Cov: C-E-D-B-F-A

Adding Cov: C-E-A-F-D-B

Hata Tespit Verilerini Kullanma

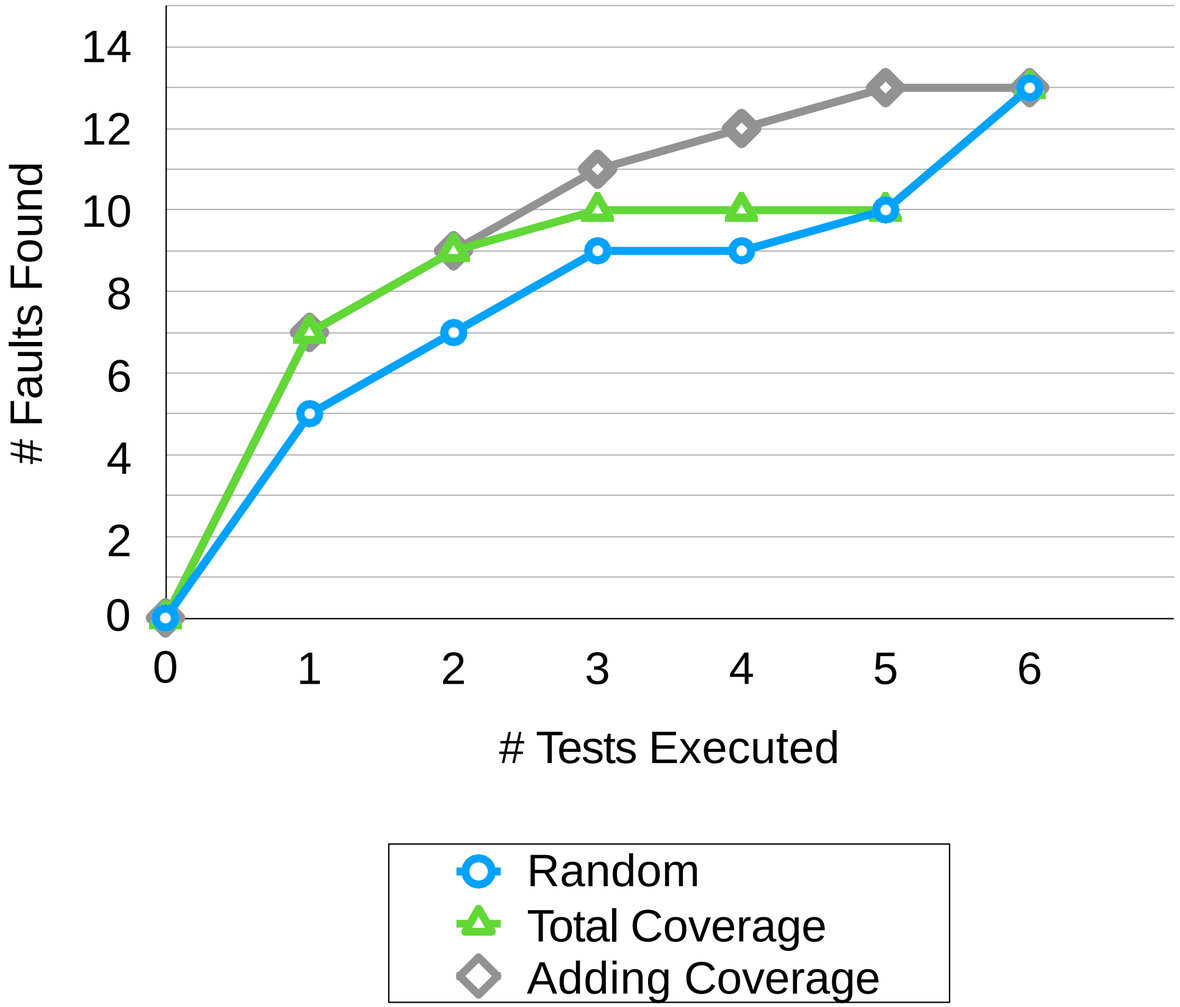


Hata Tespit Verilerini Kullanma



APFD: Average Percentage of Faults Detected (Tespiti Yapılan Hataların Ortalama Yüzdesi)

Hata Tespit Verilerini Kullanma



APFD: Average Percentage of Faults Detected (Tespiti Yapılan Hataların Ortalama Yüzdesi)

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n}$$

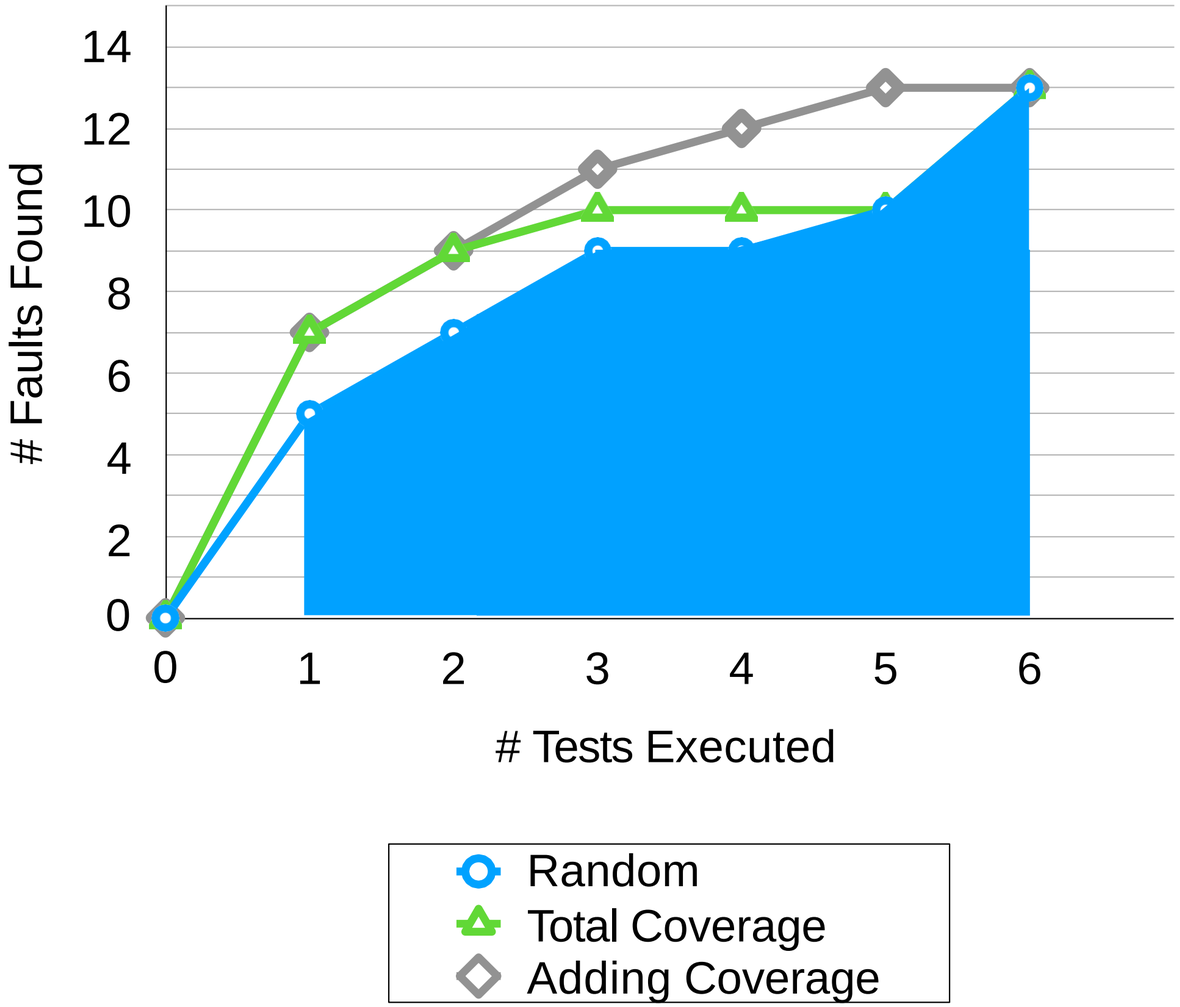
Burada

TF_i mutant I'i öldüren ilk testin konumu

m öldürülen mutant sayısı

n test sayısı

Hata Tespit Verilerini Kullanma



APFD: Average Percentage of Faults Detected (Tespiti Yapılan Hataların Ortalama Yüzdesi)

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n}$$

Burada

TF_i mutant I'i öldüren ilk testin konumu

m öldürülen mutant sayısı

n test sayısı

Kapsam için Uygunluk Fonksiyonu

Branch coverage dikkate alındığında (diğer metrikler için de geçerlidir.)

Belirli bir test sıralamasına nasıl bir puan/uygunluk atanır?

İyi olan sıralamaları ödüllendirin

Standart Yaklaşım: Test dizisi boyunca ağırlıklı ortalama kapsam yüzdesini kullanın.

Eğer n test senaryomuz, m dalımız varsa ve TB_i, i dalını yürüten ilk test senaryosunun sayısıysa, o zaman uygunluk şöyle olur:

$$APC = 1 - \frac{TB_1 + \ldots + TB_m}{nm} + \frac{1}{2n}$$

Optimizasyon

Minimizasyon problemine benzer, birden fazla hedefimiz olabilir.

- Farklı kapsam biçimleri.
- Geçmişte hatalı bileşenler için test önceliklendirme.
- Hata tahmin tekniklerine göre bileşenler için test önceliklendirme vs.

Çok amaçlı optimizasyon yöntemleri için potansiyel

Test Seçimi

Test Seçimi

Küçültme eylemi gereksiz testleri ortadan kaldırmayı amaçlar

Ancak bu uzun vadede iyi bir fikir midir?

Kaldırılan bir test, gelecekte bir hatayı ortaya çıkarmak için yararlı olabilir.

Seçim minimizasyon (küçültme) eyleminin alternatifidir.

Kod tabanında büyük testler kalmaya devam eder.

Ancak her değişiklikte hangi testlerin çalıştırılacağına dikkat edilir.

Etkilenen kodun belirlenmesi + çalıştırılacak testlerin seçimi

Metot İçi Test Seçimi

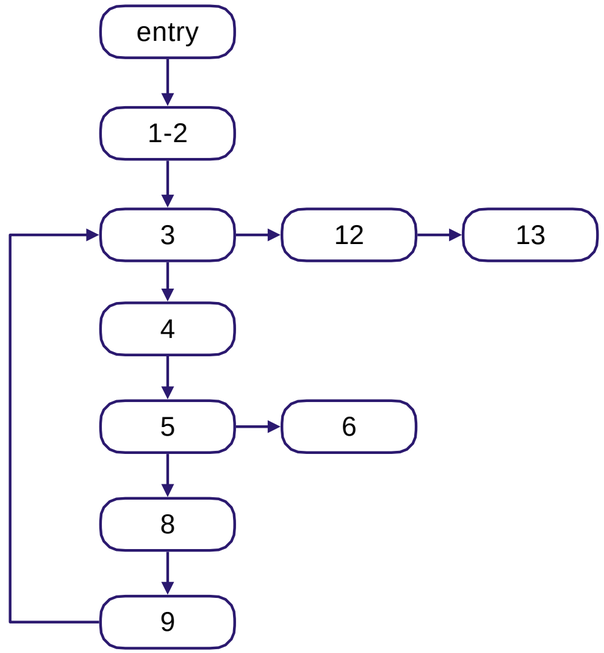
Etkilenen kodu, orijinal ve değiştirilmiş sürümlerin CFG'lerinin (kontrol akış grafikleri) **paralel olarak** taranmasına göre belirleyin.

Her adımda: Düğümleri **sözlüksel (lexicographical) eşdeğerlik** açısından karşılaştırın.

Fark var mı? Giriş düğümünden değiştirilen düğüme giden yolu geçen testleri **seçin**.

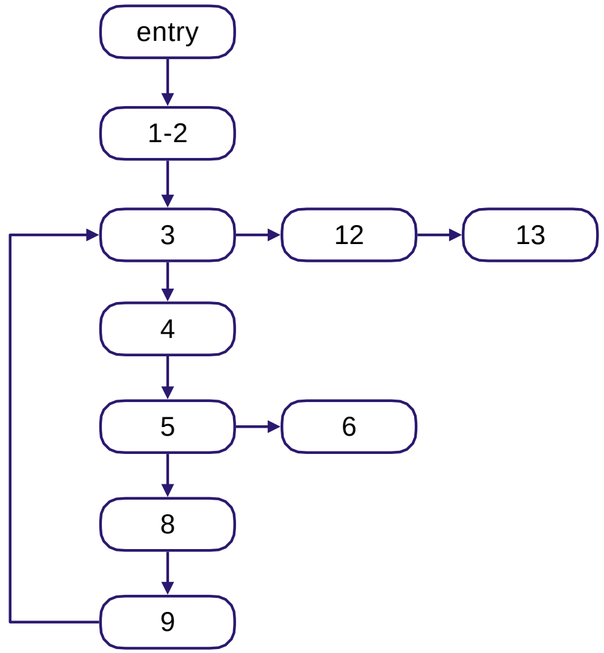
Average Örneği

```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```



Average Örneği

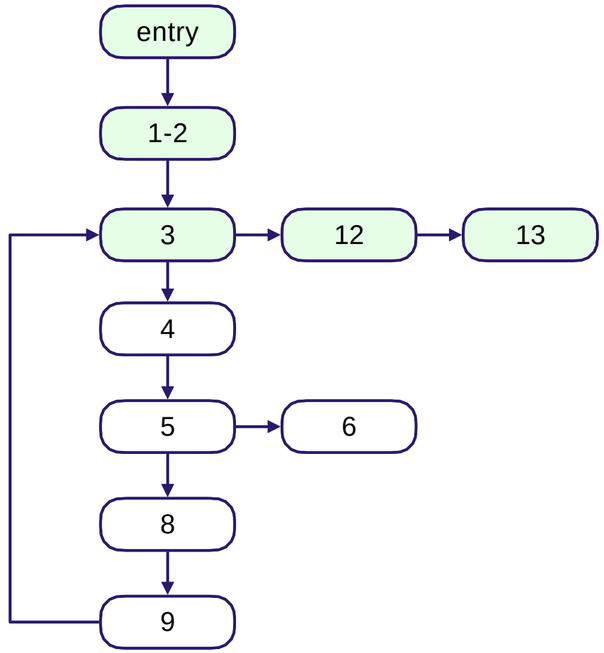
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}
```

Average Örneği

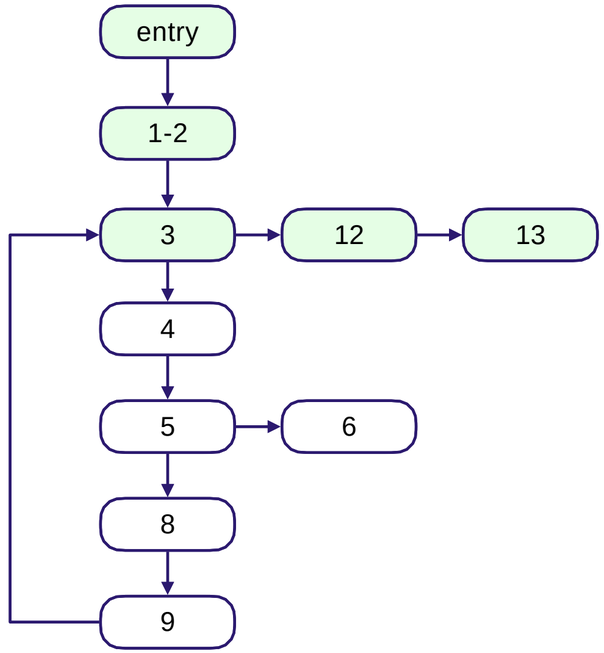
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}
```

Average Örneği

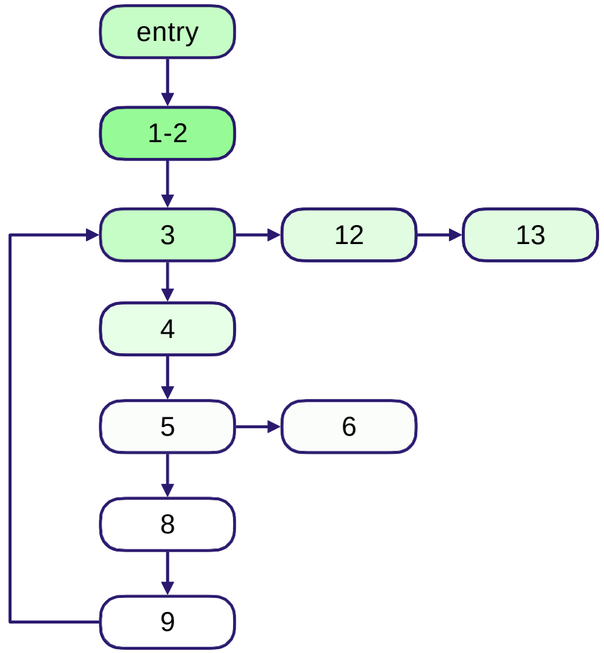
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}
```

Average Örneği

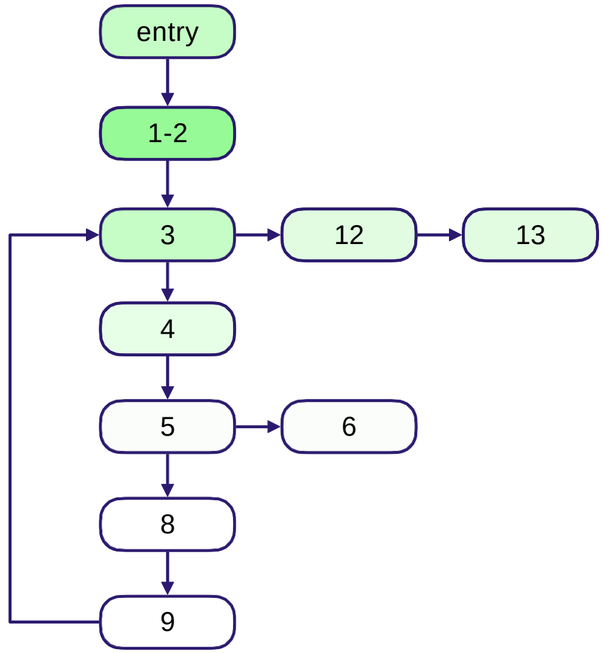
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}
```

Average Örneği

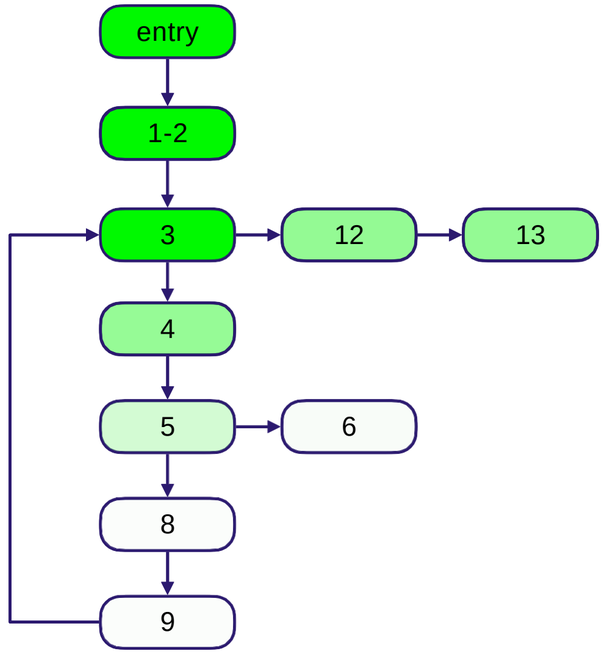
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

Average Örneği

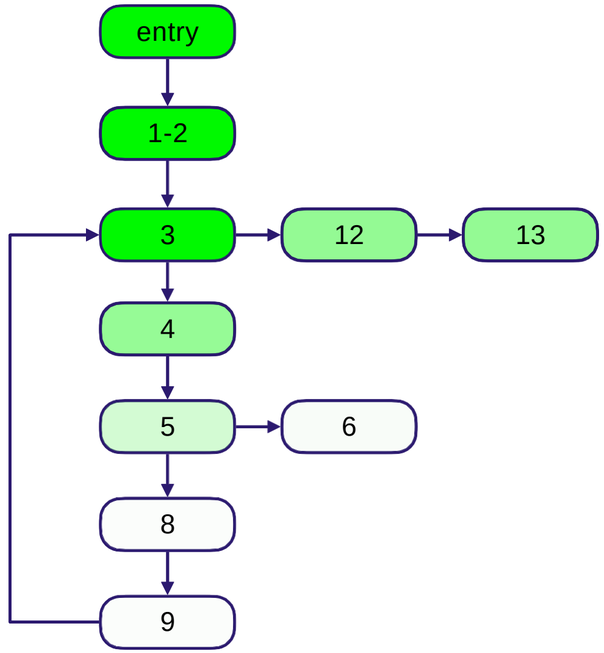
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```




```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

Average Örneği

```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```

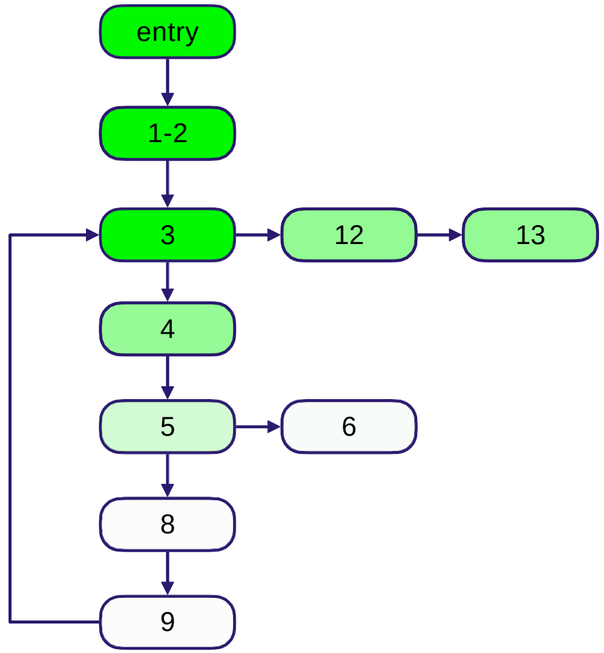



```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

Test	Edges Traversed

Average Örneği

```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```

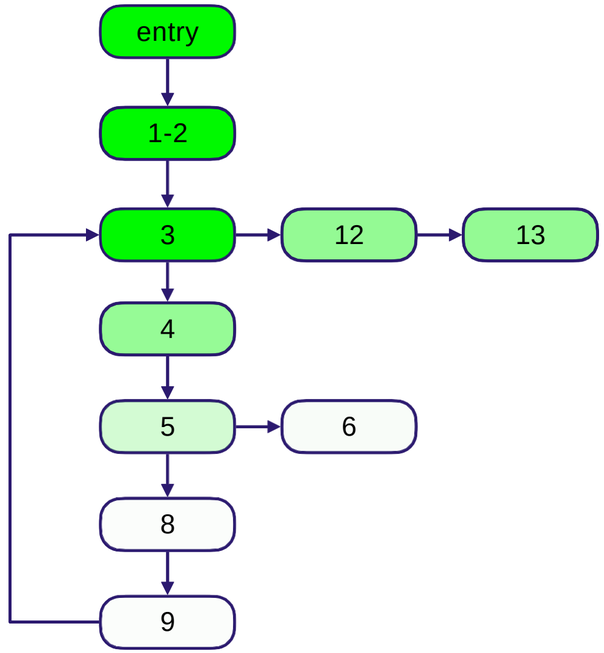



```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)

Average Örneği

```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```

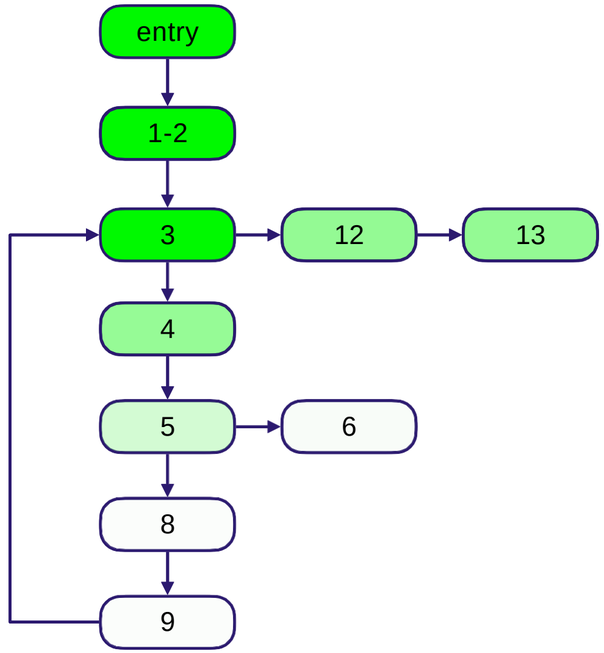



```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)

Average Örneği

```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new NumberFormatException();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```

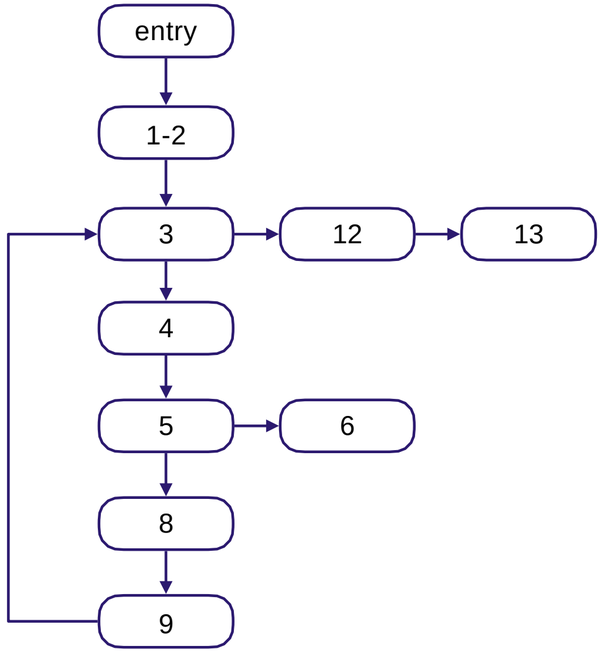



```
@Test
public void t1() throws Exception {
    assertEquals(0, TestSelection.avg(empty));
}

@Test
public void t2() throws Exception {
    assertThrows(NumberFormatException.class, () -> {
        TestSelection.avg(nonNumeric);
    });
}

@Test
public void t3() throws Exception {
    assertEquals(2, TestSelection.avg(numbers123));
}
```

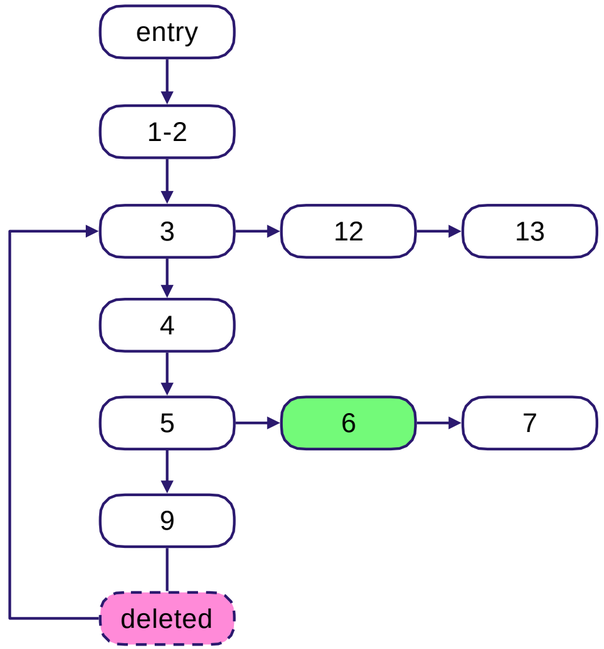
Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği

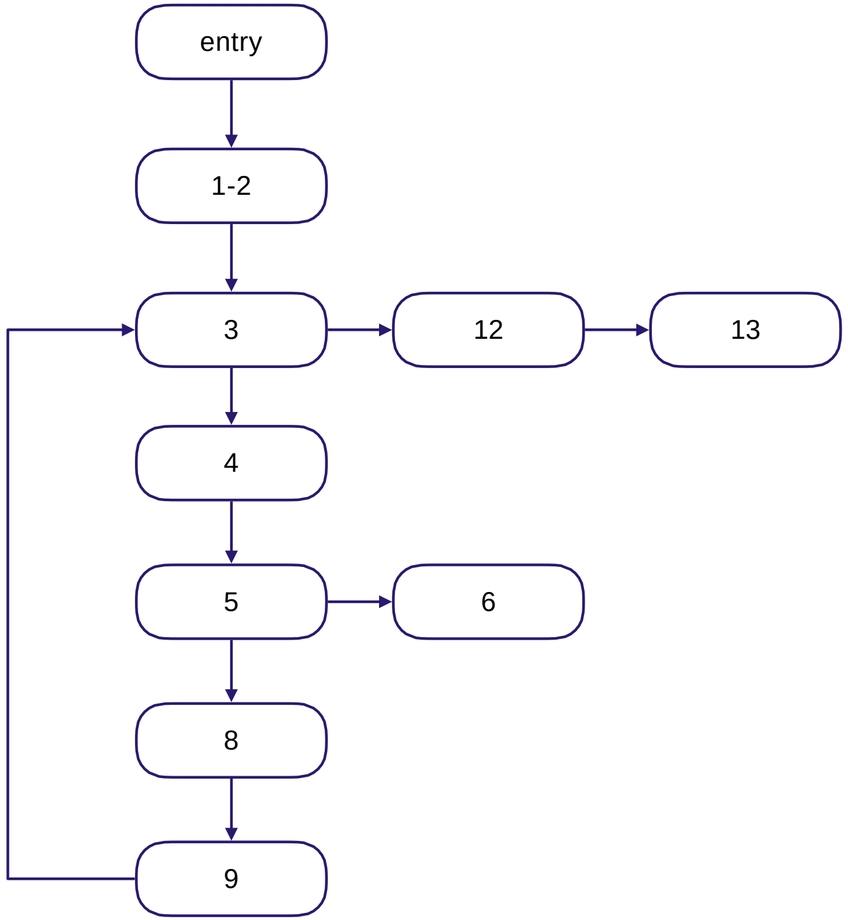
```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          throw new Exception();
 7      } else {
 8          sum += Integer.parseInt(line);
 9          count++;
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```

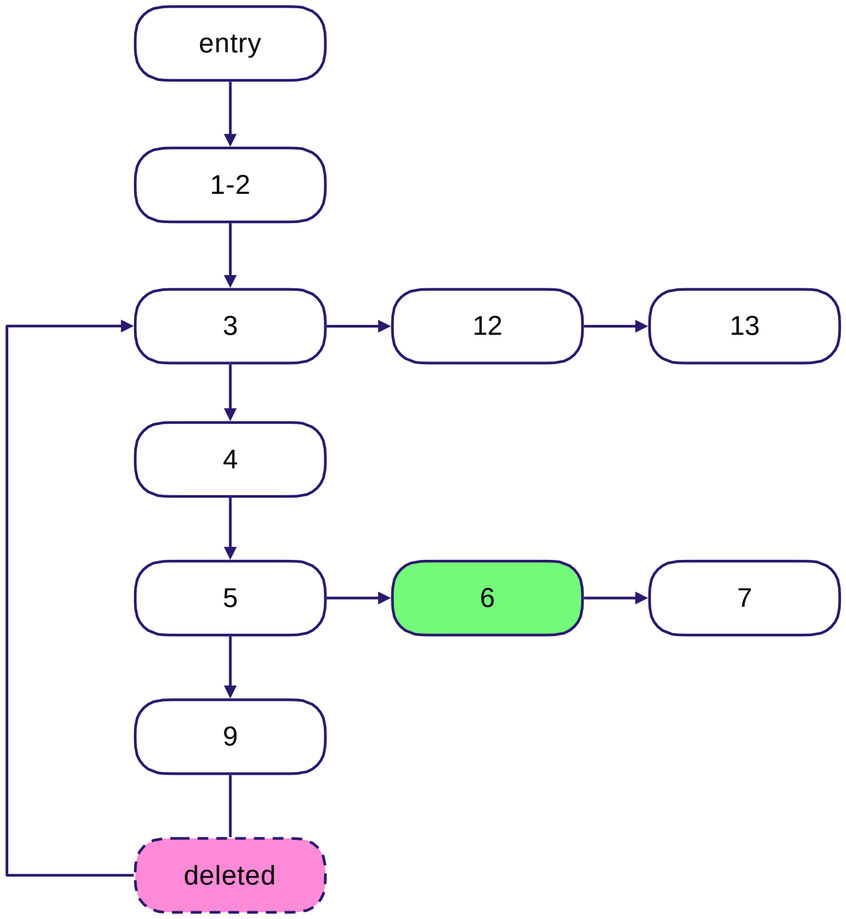



```
public static int avg(File inputFile) throws Exception {
 1  int count = 0; int sum = 0;
 2  Scanner in = new Scanner(new FileReader(inputFile));
 3  while(in.hasNext()) {
 4      String line = in.next();
 5      if (! StringUtils.isNumeric(line)) {
 6          System.err.println("Bad input");
 7          throw new Exception();
 8      } else {
 9          sum += Integer.parseInt(line);
10      }
11  }
12  in.close();
13  return count > 0 ? sum / count : 0;
}
```



Average Örneği

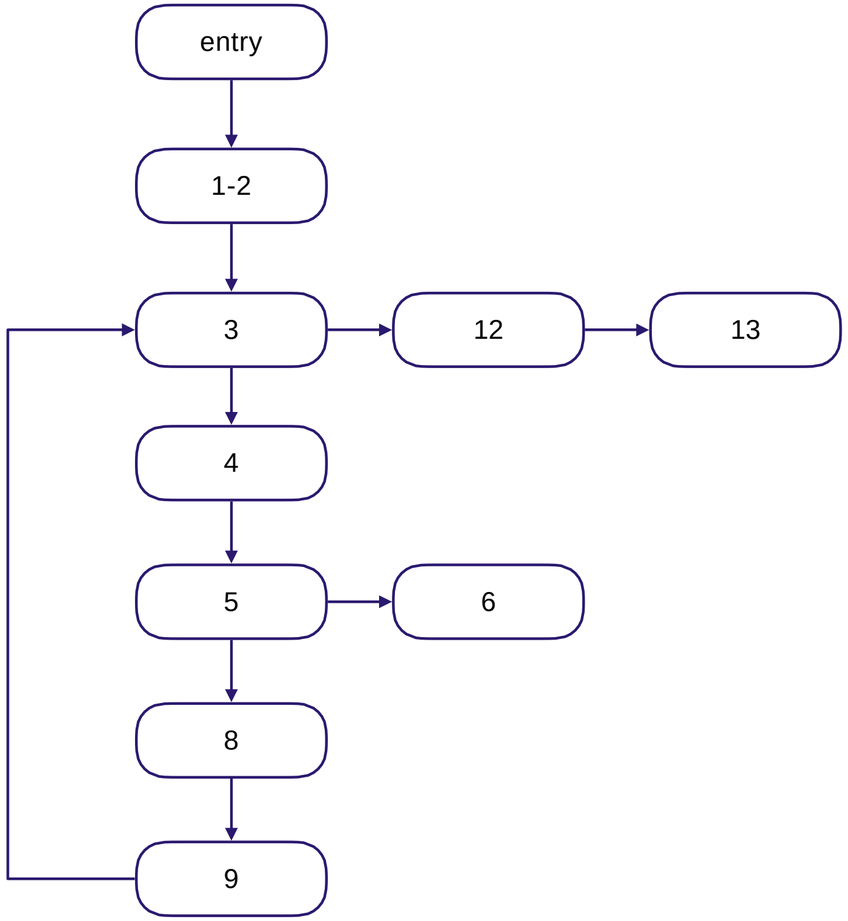


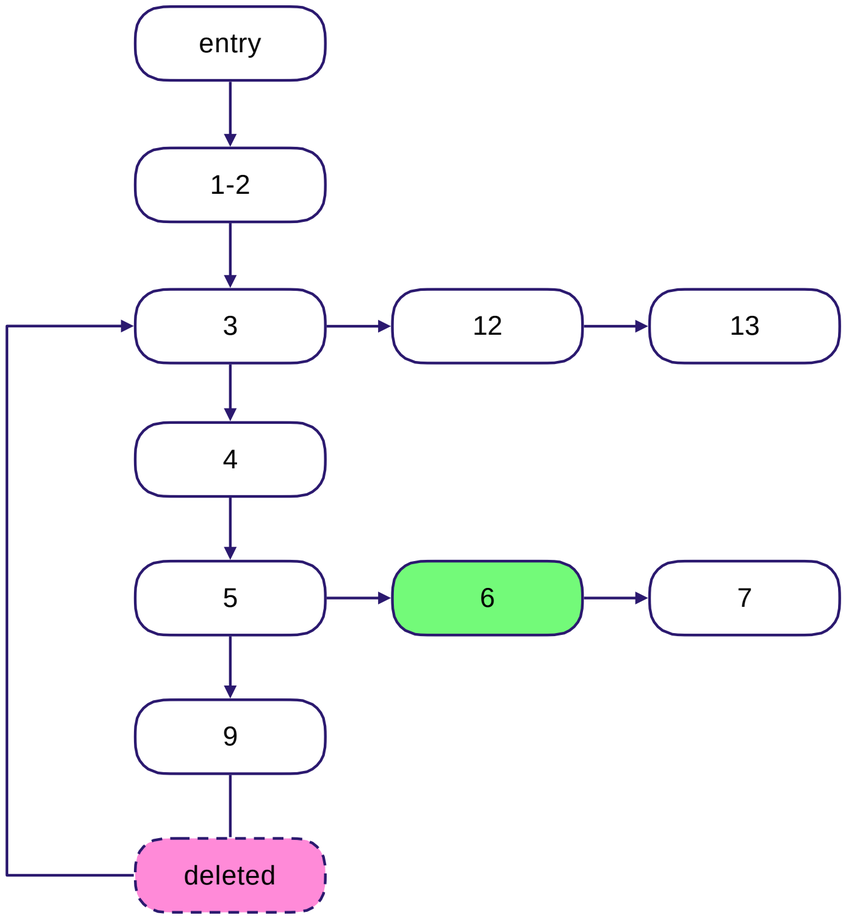


Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği

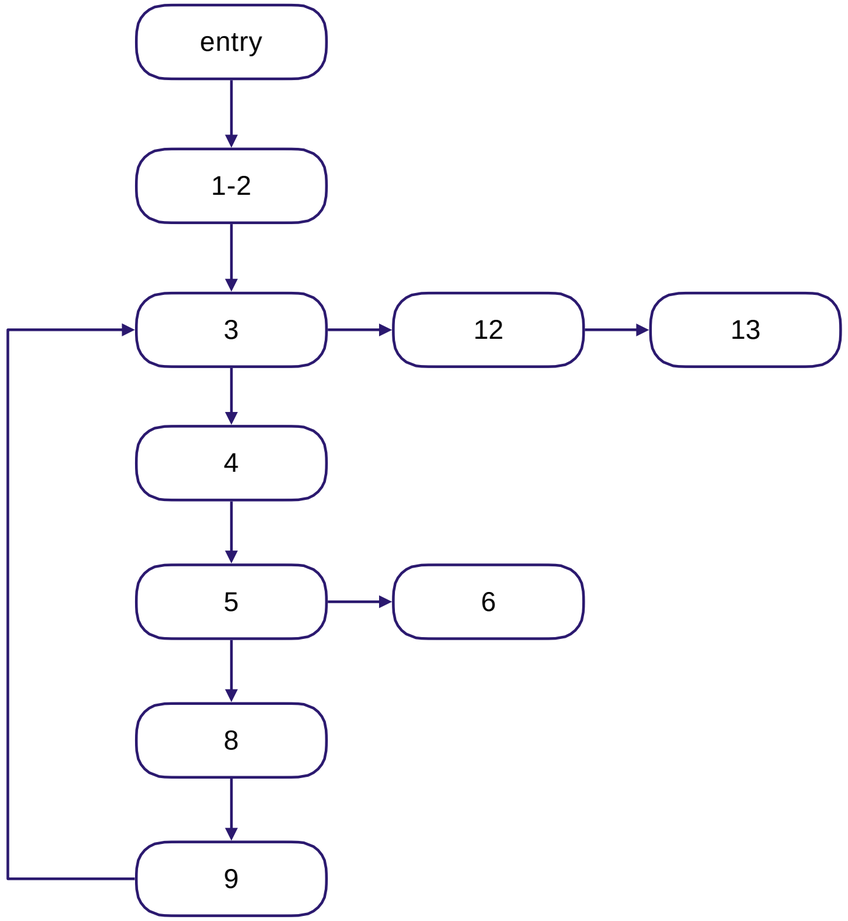
selection = {} ile başlayın

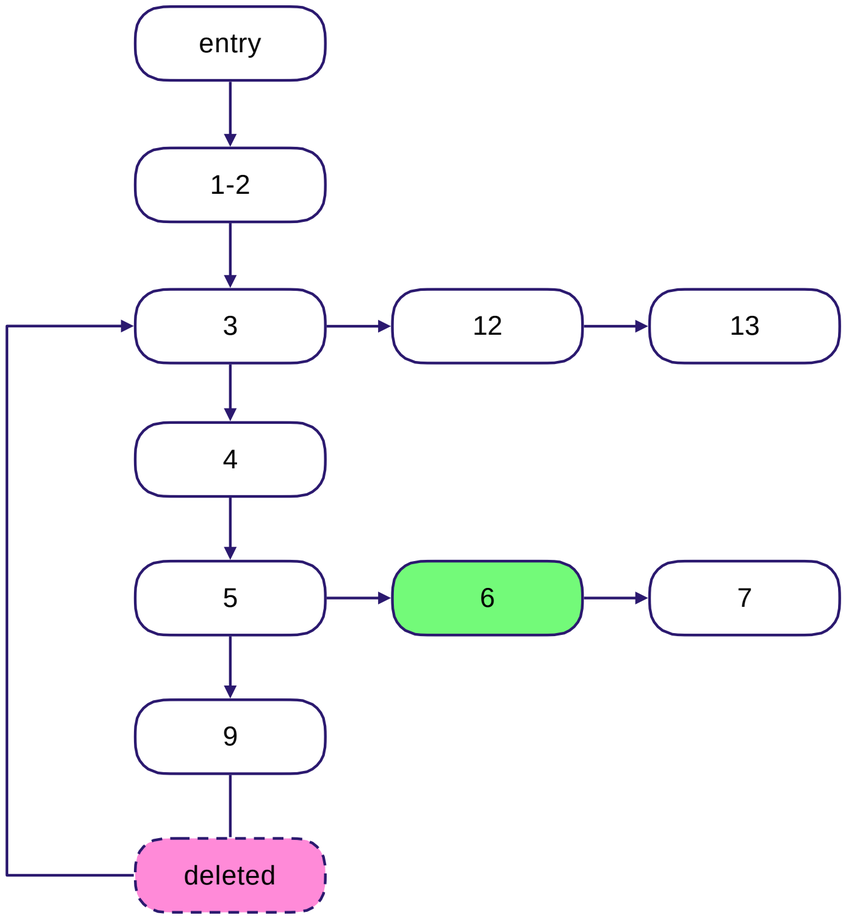




Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği



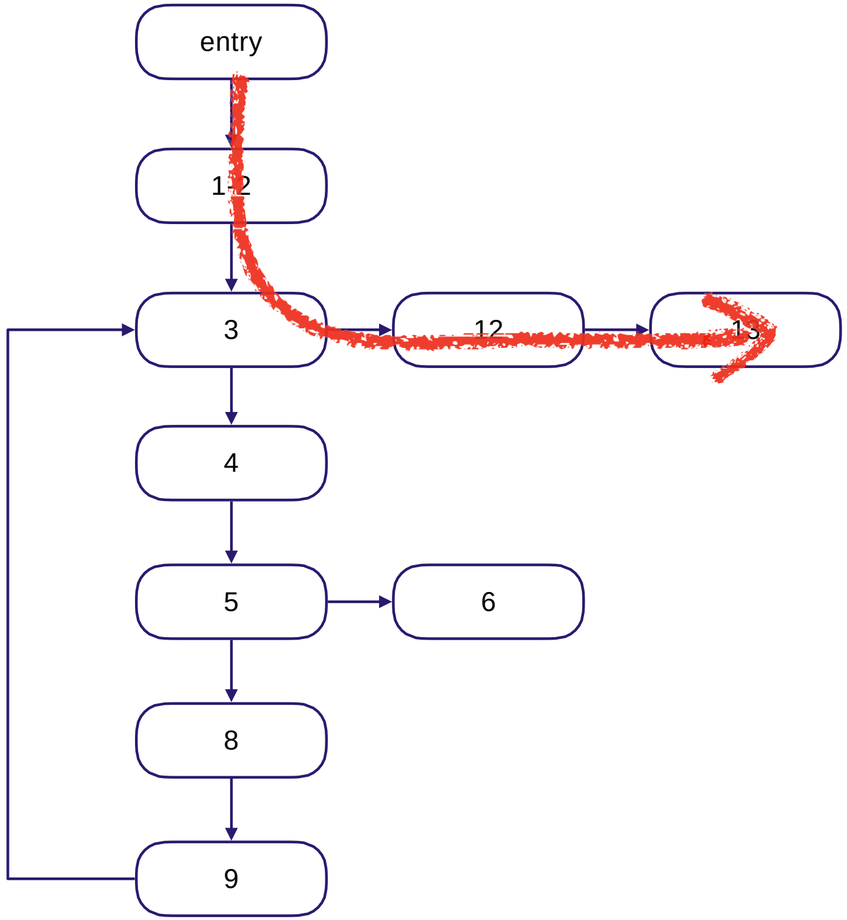


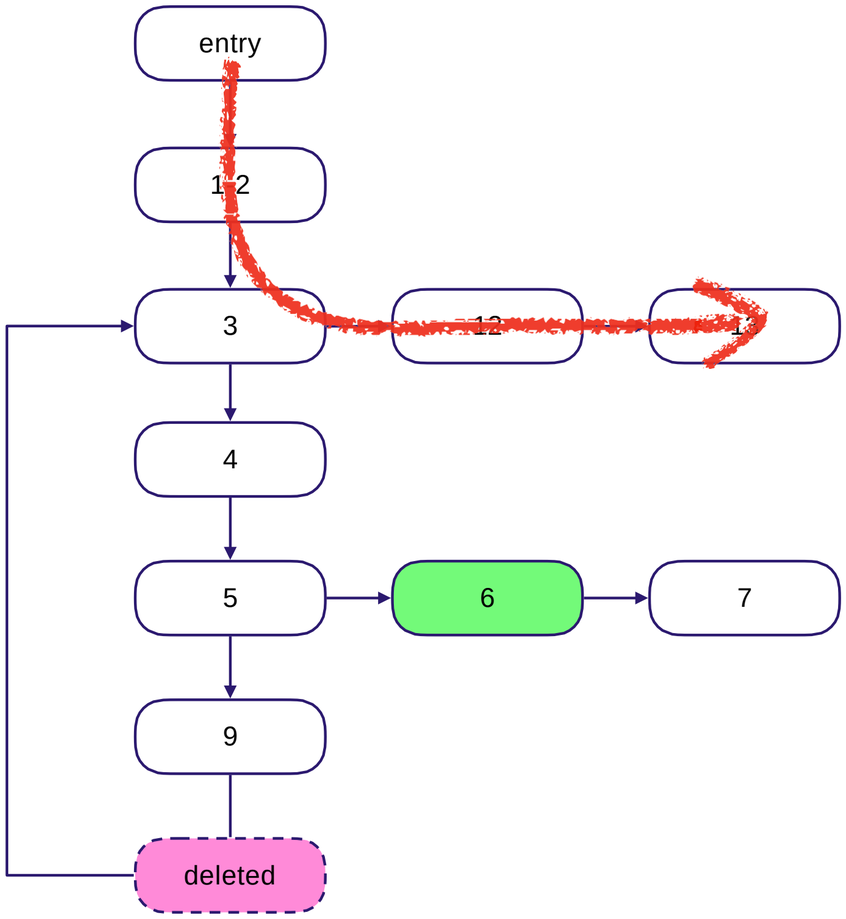
selection = {} ile başlayın

CFG'leri paralel olarak gezin

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği





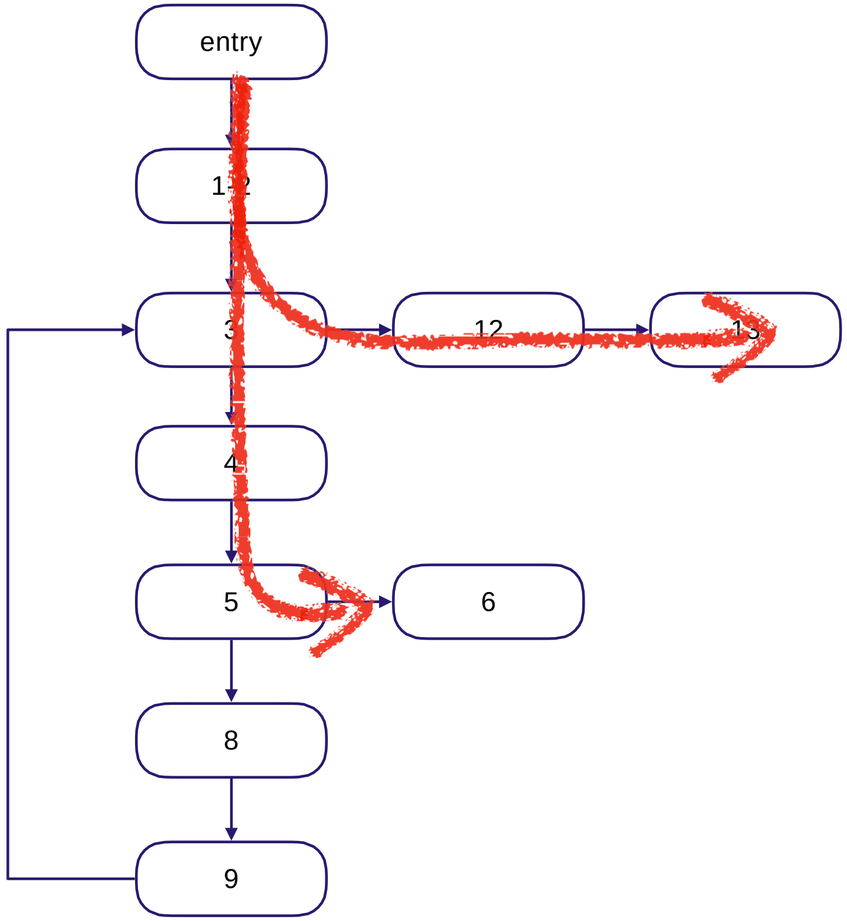
selection = {} ile başlayın

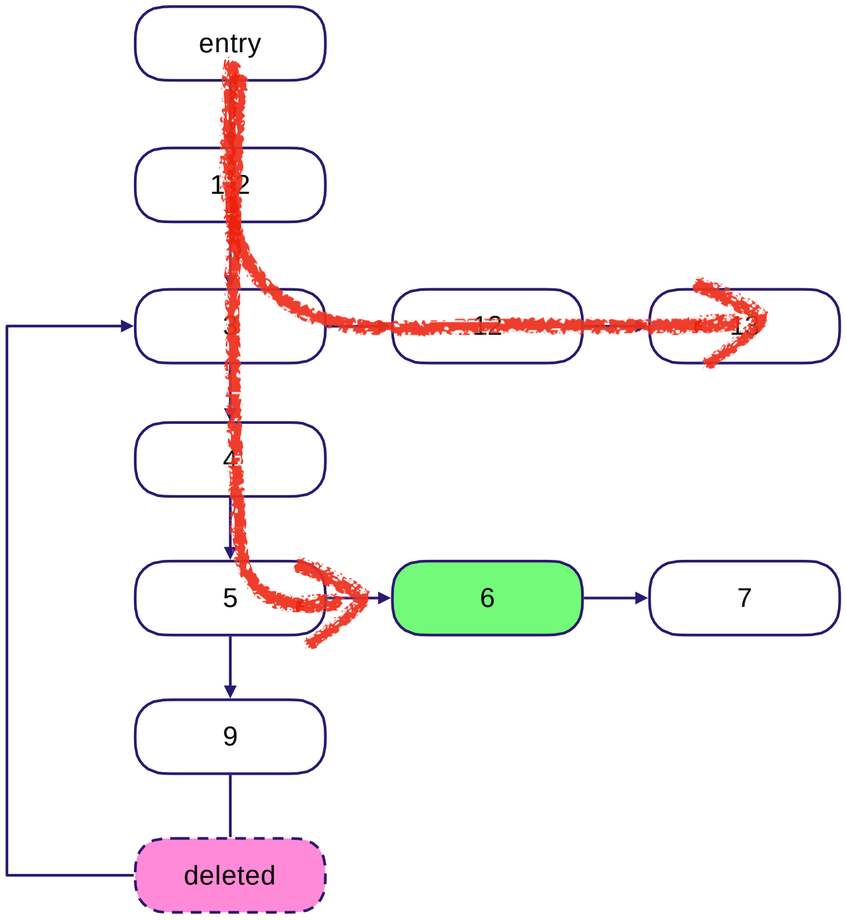
CFG'leri paralel olarak gezin

Farklılıkları belirleyin

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği





selection = {} ile başlayın

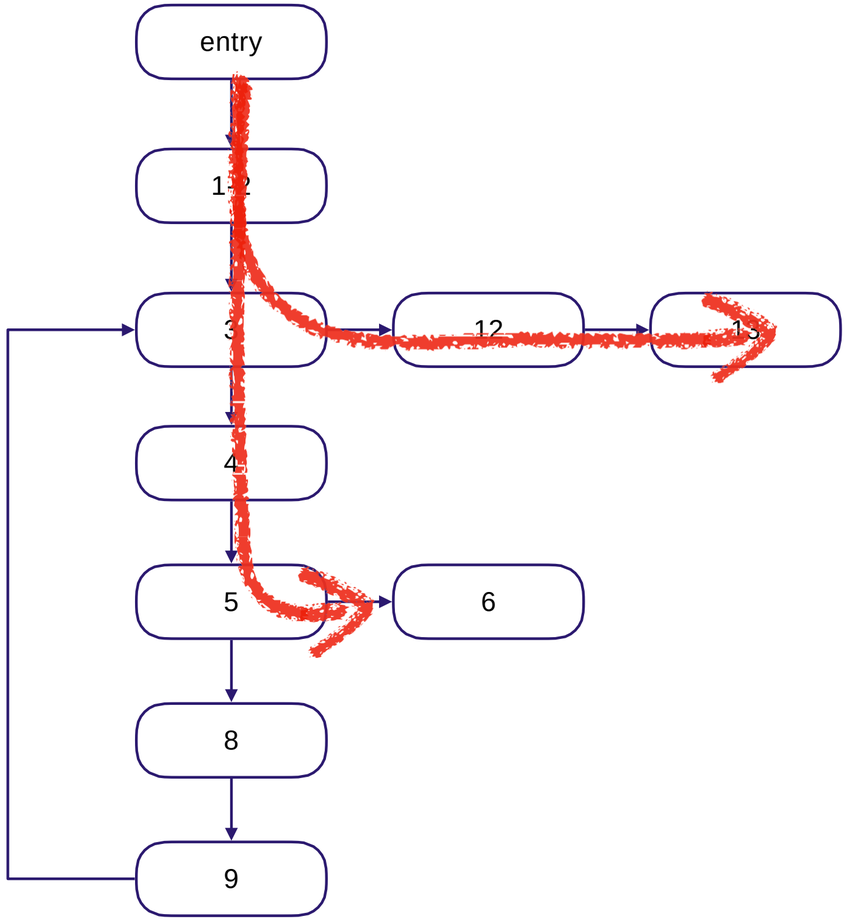
CFG'leri paralel olarak gezin

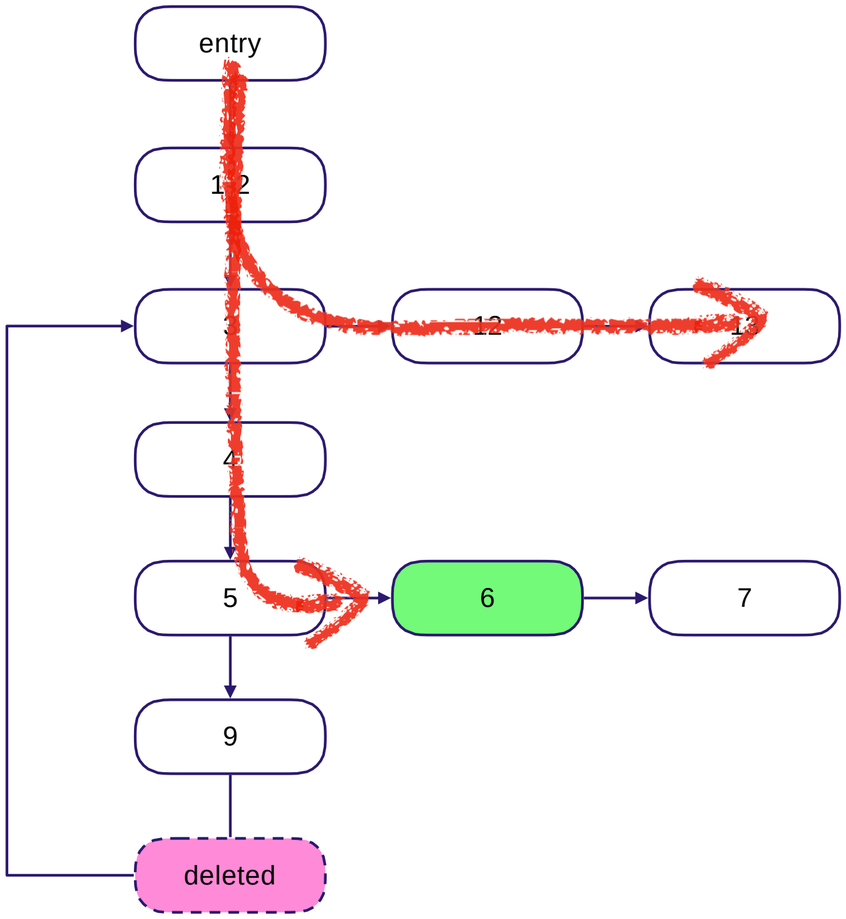
Farklılıkları belirleyin

İlk farklılık bulundu

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği





selection = {} ile başlayın

CFG'leri paralel olarak gezin

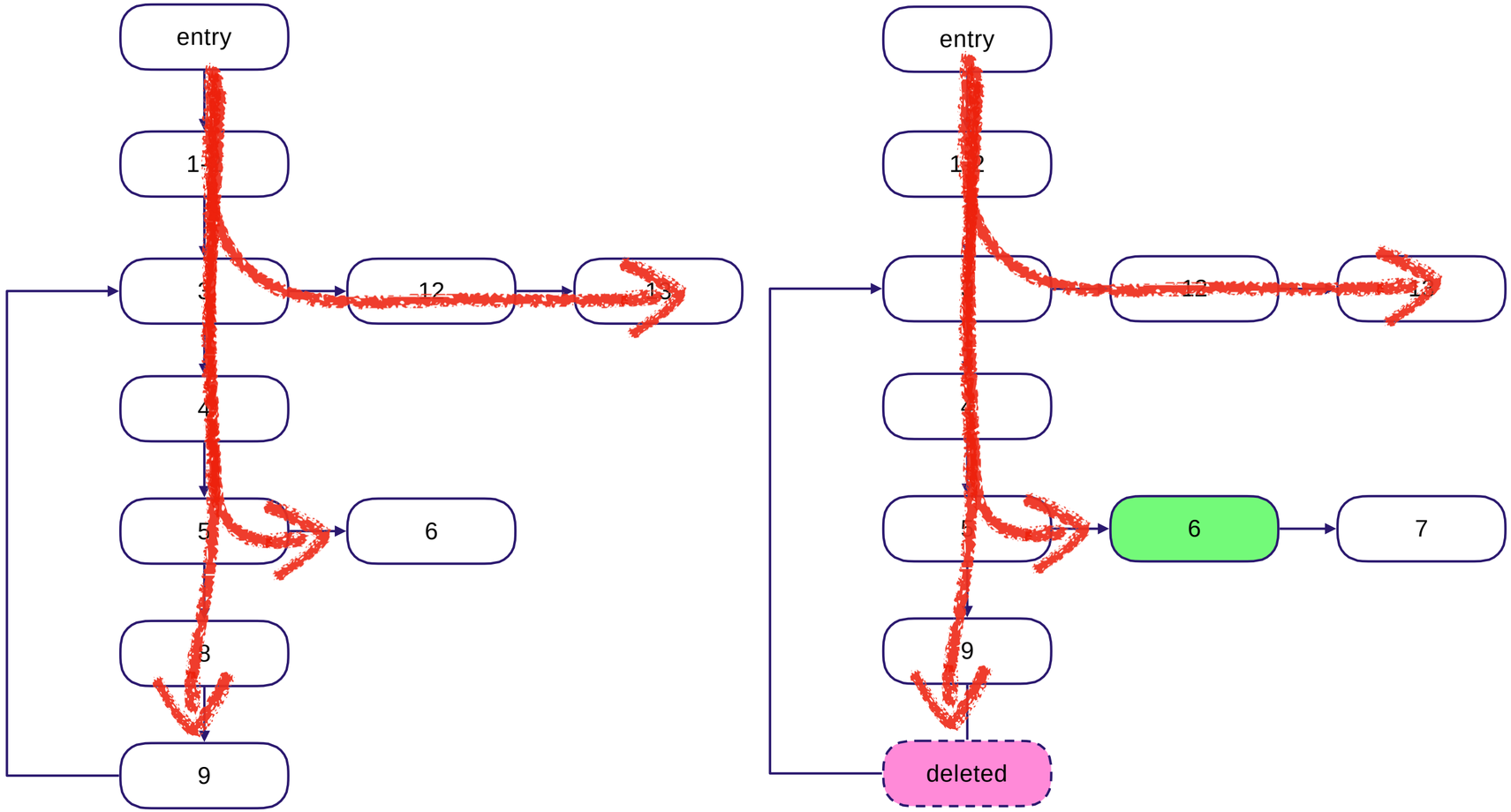
Farklılıkları belirleyin

İlk farklılık bulundu

selection = { t2 }'a testi ekleyin

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği



selection = {} ile başlayın

CFG'leri paralel olarak gezin

Farklılıkları belirleyin

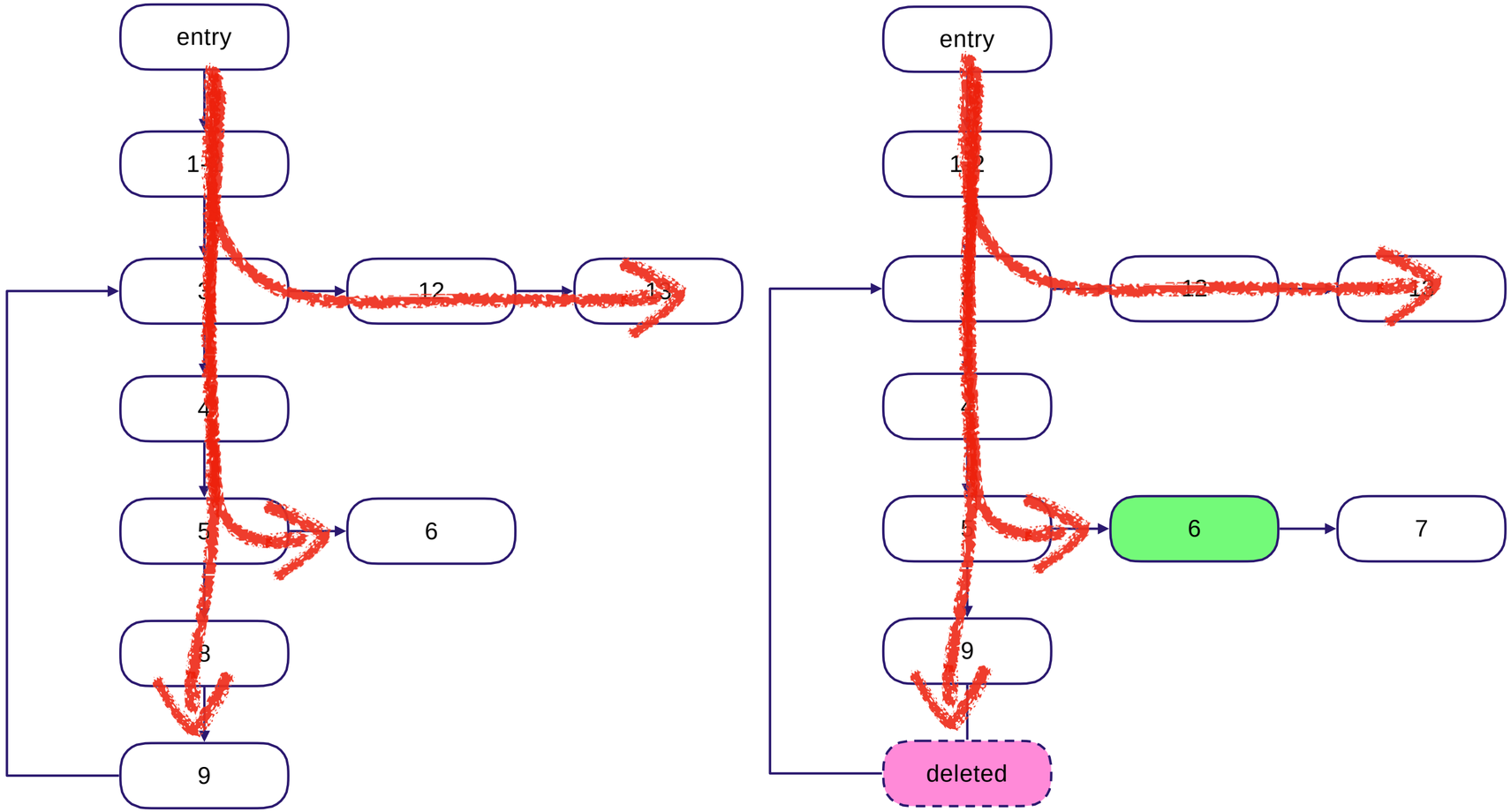
İlk farklılık bulundu

selection = { t2 }'a testi ekleyin

Yinelemeli olarak tekrarlayın

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği



selection = {} ile başlayın

CFG'leri paralel olarak gezin

Farklılıkları belirleyin

İlk farklılık bulundu

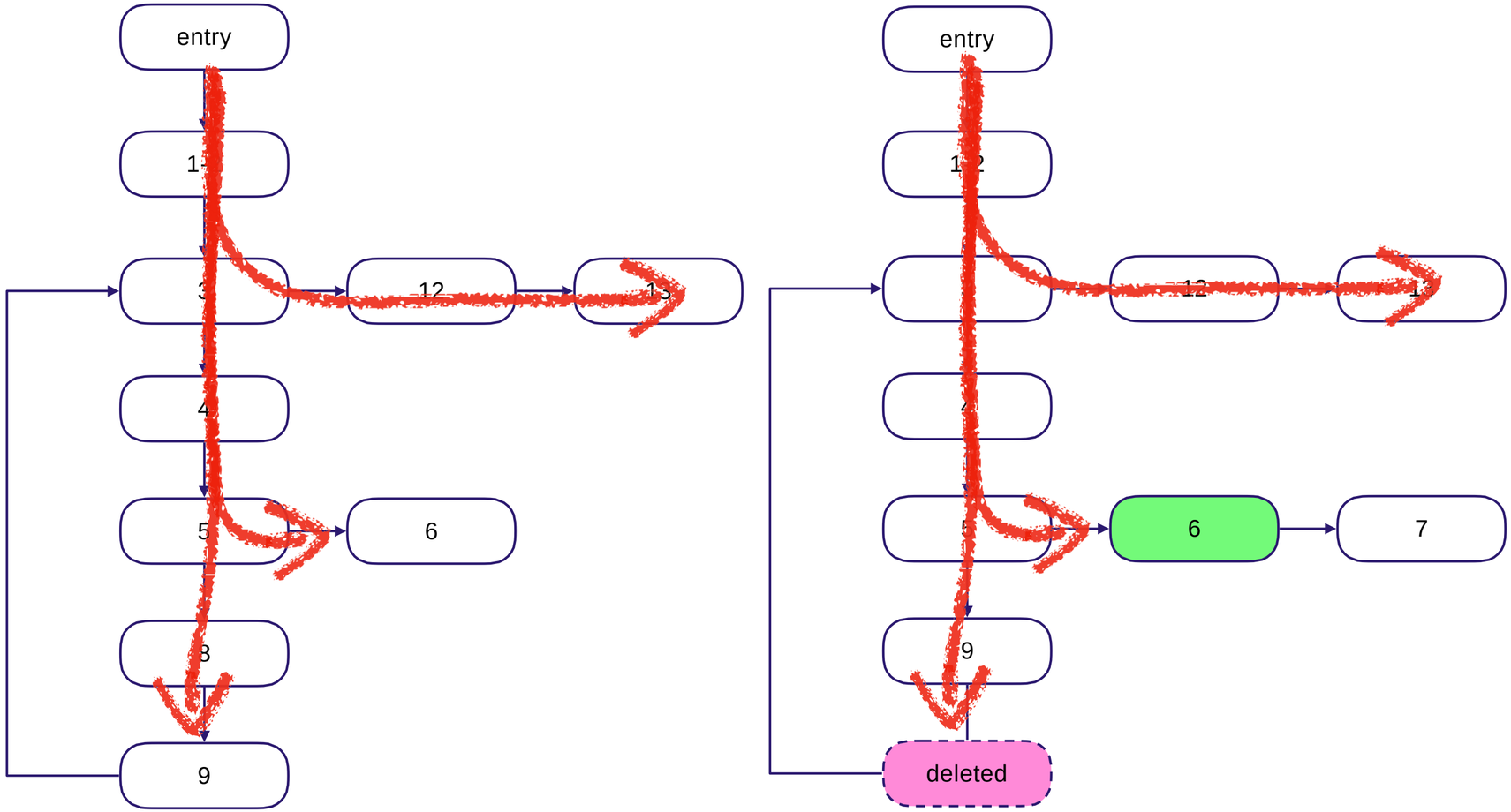
selection = { t2 }'a testi ekleyin

Yinelemeli olarak tekrarlayın

selection = { t2, t3 }

Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

Average Örneği



Test	Edges Traversed
t1	(entry, 1-2), (1-2, 3), (3, 12), (12, 13)
t2	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 6)
t3	(entry, 1-2), (1-2, 3), (3, 4), (4, 5), (5, 8), (8, 9), (9, 3), (3, 12), (12, 13)

selection = {} ile başlayın

CFG'leri paralel olarak gezin

Farklılıkları belirleyin

İlk farklılık bulundu

selection = { t2 }'a testi ekleyin

Yinelemeli olarak tekrarlayın

selection = { t2, t3 }

CFG tamamen gezildi.

Önemli Araştırmalar

Harrold MJ, Gupta R, Soffa ML. **A methodology for controlling the size of a test suite**. ACM Transactions on Software Engineering and Methodology (TOSEM) 2(3):270–285 (1993)

Rothermel G, Untch RH, Chu C, Harrold MJ. **Prioritizing test cases for regression testing**. IEEE Transactions on Software Engineering (TSE) 27(10):929-948 (2001)

Hadi Hemmati: **Chapter Four - Advances in Techniques for Test Prioritization**. Advances in Computing, vol 112: 185-221 (2019).

Rothermel G, Harrold MJ. **A safe, efficient regression test selection technique**. ACM Transactions on Software Engineering and Methodology (TOSEM) 6(2):173–210 (1997)

Özet

Hangi test senaryolarının kullanılacağının belirlendiği çoklu senaryolar

- Sürekli entegrasyon, test yürütmesinin otomasyonuna yardım eder.
- **Regresyon testi bugların** erken tespitine fayda sağlar
- Minimizasyon, önceliklendirme ve seçime fayda sağlar
- Basit sezgiseller – **Optimizasyon problemi** olarak ifade edilebilir
- Basit aç gözlü yaklaşımdan çok amaçlı evrimsel algoritmalara kadar çok farklı yöntemler