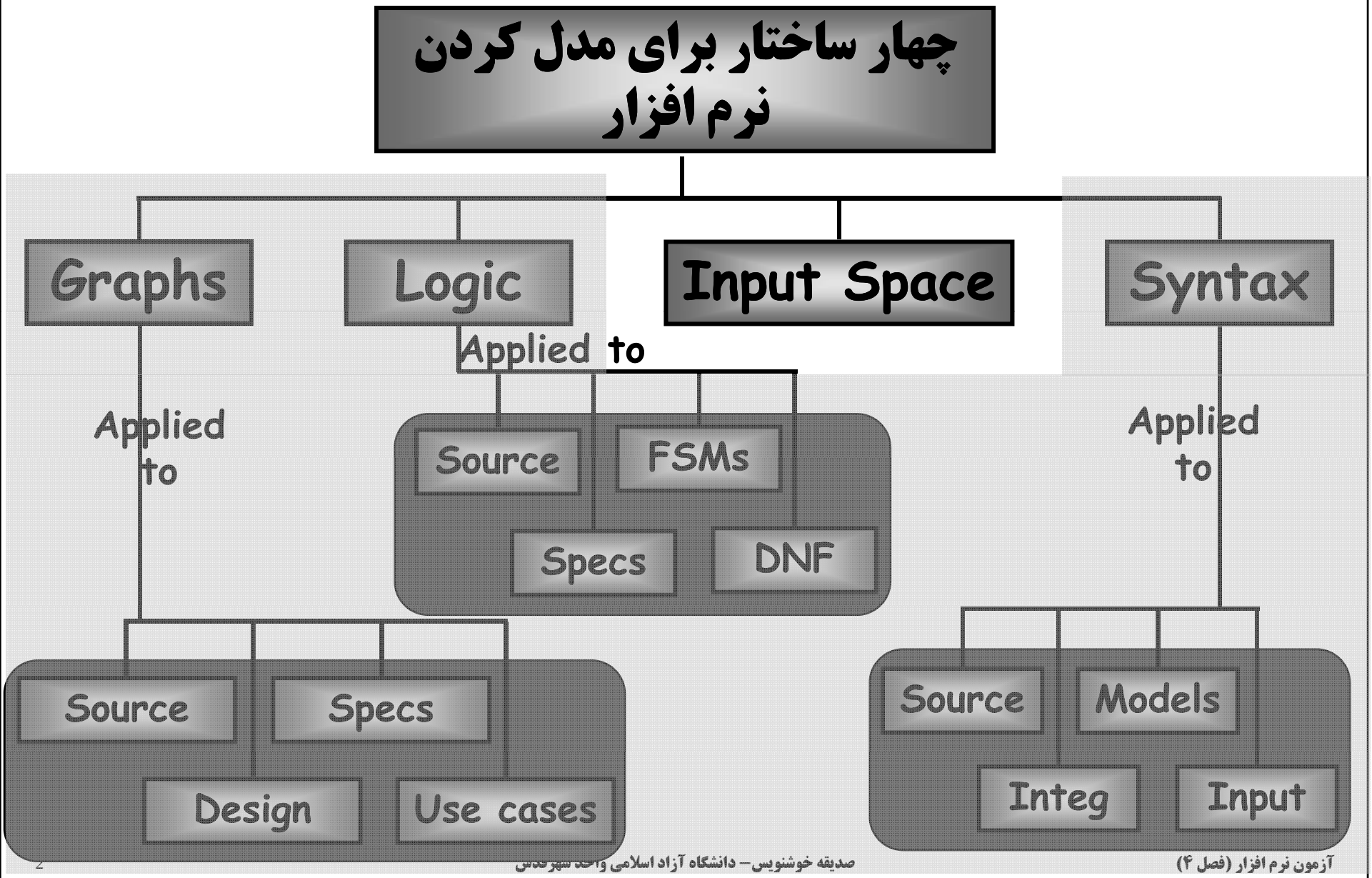
آزمون نرم افزار – فصل۴

افراز فضای ورودی

صدیقه خوشنویس
دانشگاه آزاد اسلامی – واحد شهرقدس

پوشش فضای ورودی



دامنه ورودی

- دامنه ورودی برنامه شامل همه ورودیهای ممکن آن برنامه است.
  - حتی برای برنامه های کوچک هم بسیار بزرگ و حتی نامتناهی است.

- آزمایش به این روش با انتخاب مجموعه های متناهی از مقادیر دامنه ورودی سرو کار دارد.

- پارامترهای ورودی، حیطه دامنه ورودی را تعریف می کنند:
  - پارامترهای متد
  - داده های خوانده شده از یک فایل
  - متغیرهای سراسری
  - متغیرهای سطح کاربر

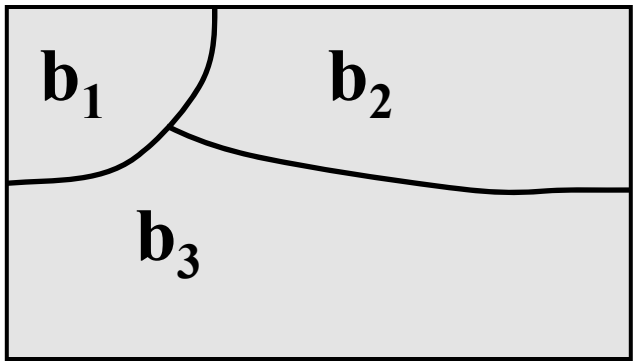
- دامنه هر پارامتر ورودی به چند ناحیه افراز می شود.
- از هر ناحیه حداقل یک مقدار انتخاب می شود.

مزایای روش ISP (افراز دامنه ورودی)

- می تواند برای سطوح مختلفی از آزمایش استفاده شود:
  - تست واحد
  - تست یکپارچه سازی
  - تست سیستم
- انجام آن بدون خودکار سازی نسبتاً آسان است.
- می توان آن را به راحتی برای تستهای کمتر یا بیشتر تنظیم کرد.
- دانش پیاده سازی خاصی نیاز ندارد (و اصلاً با پیاده سازی سر و کار ندارد)؛ فقط دانش درباره فضای ورودی لازم است.

مفهوم افراز (partitioning)

- دامنه D
- طرح افراز q روی D
- افراز q موجب تعریف یک سری ناحیه به نام بلاک (block) می شود:
- $B_q=b_1,b_2,\ldots,b_Q$
- افراز باید دو ویژگی را برآورده کند:
  - بلاکها باید دو به دو گسسته (منفصل) باشند (همپوشانی یا اشتراک نداشته باشند) (disjoint)
  - بلاکها با هم (اجتماع آنها) باید کل دامنه D را بسازد (باید کامل باشند) (complete)



$$b_i \cap b_j = \Phi, \forall i \neq j, b_i, b_j \in B_q$$

$$\bigcup_{b \in Bq} b = D$$

فرضها و مراحل

- قرار است از هر بلاک، یک مقدار انتخاب کنیم.
- فرض بر آن است که در یک بلاک همه مقدارها به یک اندازه مهم و مفید هستند.

- مراحل کار برای تست ISP:
  - ۱) یک یا چند خصوصیت (characteristic) از ورودی را پیدا می کنیم.
  - ۲) هر خصوصیت را به چند بلاک، افراز (partition) می کنیم.
  - ۳) با ترکیب مقادیر از بلاکهای خصوصیتها، آزمونها را انتخاب می کنیم.

- مثال از خصوصیت ورودی:
  - پوچ بودن X (پوچ، غیرپوچ)
  - وضعیت مرتب بودن فیلدهای فایل F (مرتب صعودی، مرتب نزولی، نامرتب)
  - فاصله میان دو هواپیما (نرمال، کمتر از حد مجاز)
  - و ...

انتخاب خصوصیت (characteristic)

- انتخاب یا تعریف افراز (یعنی خصوصیتها و بلاکهایشان) آسان است.
- اما اشتباه کردن در آن نیز آسان است.
- ترتیب فایلی به نام F را در نظر بگیرید:

خصوصیت ۰ = وضعیت مرتب بودن فایل

b_1 = مرتب صعودی

b_2 = مرتب نزولی

b_3 = نامرتب

اما این بلاک بندی ایراد دارد

اگر فایل فقط یک عنصر داشته باشد؟

هم در بلاک ۱ قرار می گیرد و هم در بلاک۲

پس شرط گسستگی (انفصال) (disjoint) نقض می شود!

راه حل:

هر خصوصیت تنها به یک ویژگی اشاره کند.

خصوصیت ۱ = ترتیب صعودی فایل F

- b1 = true
- b2 = false

خصوصیت ۲ = ترتیب نزولی فایل F

- b1 = true
- b2 = false

مثال های بیشتر

- برای یک پشته (stack) از اشیاء مثلاً از نوع string:
- خصوصیت Q1: خالی بودن پشته
  - بلاک ۱: true (مثلاً stack=[])
  - بلاک ۲: false (مثلاً stack=["red"] یا stack=["red' , "blue"])

- خصوصیت Q2: سایز پشته
  - بلاک ۱: سایز = ۰(مثلاً stack=[])
  - بلاک ۲: سایز = ۱ (stack=["green"] یا stack=[null])
  - بلاک ۳: سایز >۱ (مثلاً stack=["orange' , "green" , "yellow"] یا stack=["red' , null])

- خصوصیت Q3: وجود مقدار null در پشته
  - بلاک ۱: true (مثلاً stack=[null] یا stack=["red' , null])
  - بلاک ۲: false (مثلاً stack=["red"] یا stack=[])

مثال های بیشتر...

- برای اشیاء x درون پشته:
- خصوصیت Q1: پوچ بودن شیء x
  - بلاک ۱: true (x=null)
  - بلاک ۲: false (مثلاً x="white")

- برای ترکیب اشیاء و پشته:
- خصوصیت Q1: وجود شیء x در پشته stack
  - بلاک ۱: true (مثلاً x="red" و stack=["red' , null])
  - بلاک ۲: false (مثلاً x="blue" و stack=["red"])

مدلسازی دامنه

- مراحل:
  - ۱) تعیین آنچه می خواهیم تست کنیم (متد، کلاس، برنامه، کل سیستم، ...)
  - ۲) پیدا کردن همه پارامترهای ورودی
  - ۳) مدلسازی دامنه ورودی (تعیین خصوصیتها و بلاکها)
  - ۴) اعمال یک معیار آزمون برای تعیین ترکیبی از مقادیر
    - ورودی آزمون باید برای هر پارامتر یک مقدار داشته باشد
    - همه ترکیب ها از همه بلاکها و همه خصوصیتها معمولاً ناممکن است؛
    - و با استفاده از معیار، زیرمجموعه ای از مقادیر را تعیین می کنیم.
  - ۵) استخراج ورودیهای آزمون بر اساس ترکیب خصوصیتها و بلاکها و انتخاب مقادیر از آنها

دو رویکرد اصلی در مدلسازی دامنه ورودی (IDM)

- ۱) مبتنی بر واسط (interface-based)
  - ساده ترین روش
  - تعیین خصوصیت مبتنی بر هر یک از پارامترهای ورودی (در واسط برنامه)
  - تاحدودی قابل خودکار سازی
  - مبتنی بر نحو (syntax)

- ۲) مبتنی بر عملکرد (functionality-based)
  - کمی سخت تر است
  - تعیین خصوصیت مبتنی بر رفتار برنامه
  - آزمونهای بهتر و با تعداد کمتر و مؤثرتری ایجاد می کند
  - مبتنی بر معنا (semantics)

۱. رویکرد مبتنی بر واسط

- هر پارامتر را به صورت منفرد در نظر بگیرید.
- ارتباط های میان پارامترها را نادیده بگیرید.

- مثال:
  - فرض کنید برنامه **TriType** برنامه تعیین نوع مثلث است و سه ورودی **int** (طول سه ضلع) می گیرد.
  - **IDM** برای هر سه ورودی مثل هم است
  - مثلاً:
  - خصوصیت **Q1**: ارتباط طول ضلع با صفر (مثبت، منفی یا صفر بودن)
    - بلاک ۱: مثبت
    - بلاک ۲: منفی
    - بلاک ۳: صفر

۲. رویکرد مبتنی بر عملکرد

- خصوصیتها را با توجه به عملکرد برنامه تعیین کنید
- ارتباط های میان پارامترها را می توانید در نظر بگیرید
- نیازمندیهای برنامه را نیز می توانید در نظر بگیرید
- یک پارامتر ممکن است در خصوصیتهای مختلفی به کار رود

- مثال: برای برنامه TriType : سه ورودی آن تشکیل یک «مثلث» می دهند
- IDM را می توانیم برای مثلث در نظر بگیریم
- مثلاً:
- خصوصیت Q1: نوع مثلث:
  - بلاک ۱: متساوی الاضلاع
  - بلاک ۲: مختلف الاضلاع
  - بلاک ۳: ... (بعداً برای این مثال بیشتر بحث می کنیم)

تفاوت دو رویکرد «مبتنی بر واسط» و «مبتنی بر عملکرد»

مثال:

```
public boolean findElement (List list, Object element)
// اگر list یا element برابر با پوچ بود استثنای NullPointerException ایجاد می کند
// در غیر این صورت اگر element در list باشد true برمی گرداند
// و در غیر این صورت false بر می گرداند
```

با رویکرد مبتنی بر واسط

دو پارامتر : list و element

خصوصیت ها:

Q1 : list پوچ است (بلاک۱= true ، بلاک ۲=false)

Q2 : list خالی است (بلاک۱= true ، بلاک ۲=false)

با رویکرد مبتنی بر عملکرد

دو پارامتر : list و element

خصوصیت ها:

Q1 : تعداد وقوع element در list (بلاک۱= 0 ، بلاک ۲=1 ، بلاک ۳= بیشتر از ۱)

Q2 : element در ابتدای list است (بلاک۱= true ، بلاک ۲=false)

Q3 : element در انتهای list است (بلاک۱= true ، بلاک ۲=false)

چند نکته درباره IDM

- هرچه تعداد بلاکها بیشتر باشد یعنی تعداد آزمونها بیشتر خواهد بود.
- افراز معمولاً به تعریف خصوصیت وابسته است. ولی باید هر دو را با هم در نظر بگیریم. زیرا گاهی می توان خصوصیتهای کمتر با بلاکهای بیشتر تعریف کرد و برعکس

- چند استراتژی برای یافتن مقادیر:
  - مقادیر معتبر، نامعتبر، یا خاص
  - ایجاد افراز هایی در درون بلاکها (زیرافراز sub-partition)
  - توجه به مرزهای دامنه
  - مقادیر نرمال و معمولی
  - سعی برای متعادل نگه داشتن تعداد بلاکها در هر خصوصیت
  - بررسی دو شرط کامل بودن و گسستگی (انفصال)

مثال: IDM برای TriType با رویکرد مبتنی بر واسط

- **این برنامه یک متد است که سه ورودی int می گیرد.**

سری اول تعیین خصوصیات ورودیهای TriType

خصوصیت	b_1	b_2	b_3
q_1 = "ارتباط ضلع ۱ با صفر"	بزرگتر از صفر	مساوی صفر	کوچکتر از صفر
q_2 = "ارتباط ضلع ۲ با صفر"	بزرگتر از صفر	مساوی صفر	کوچکتر از صفر
q_3 = "ارتباط ضلع ۳ با صفر"	بزرگتر از صفر	مساوی صفر	کوچکتر از صفر

- **تعداد تستها حداکثر = ۳*۳*۳ = ۲۷**
- **برخی ورودیها مثلثهای معتبر را تشکیل می دهند و بعضی نامعتبر هستند (مثلث تشکیل نمی شود)**
- **پالایش این نوع تعیین خصوصیت ممکن است منجر به آزمونهای بیشتری شود ... (اسلاید بعد)**

مثال: IDM برای TriType با رویکرد مبتنی بر واسط..

سری دوم تعیین خصوصیات ورودیهای TriType

خصوصیت	b_1	b_2	b_3	b_4
q_1 = "ارتباط ضلع ۱ با صفر و یک"	بزرگتر از یک	مساوی یک	مساوی صفر	کوچکتر از صفر
q_2 = "ارتباط ضلع ۲ با صفر و یک"	بزرگتر از یک	مساوی یک	مساوی صفر	کوچکتر از صفر
q_3 = "ارتباط ضلع ۳ با صفر و یک"	بزرگتر از یک	مساوی یک	مساوی صفر	کوچکتر از صفر

- **تعداد تستها حداکثر = ۴*۴*۴ = ۶۴**
- **این افراز دارای شرط کامل بودن هست زیرا ورودیها int هستند.**
- **(اگر float بودند به خاطر در نظر نگرفتن مقادیر بین ۰ و ۱ کامل نبود)**

مثال: IDM برای TriType با رویکرد مبتنی بر عملکرد

- در این حالت، معنا و عملکرد و نیازمندی و ارتباط ورودیها را با هم در نظر می گیریم: این ۳ ورودی قرار است اضلاع یک مثلث باشند (از دیدگاه هندسه به موضوع نگاه کنیم)

تعیین خصوصیات ورودیهای TriType از دیدگاه هندسی

خصوصیت	b_1	b_2	b_3	b_4
q_1 = "تعیین نوع هندسی مثلث"	مختلف الاضلاع	متساوی الساقین	متساوی الاضلاع	نامعتبر

- یک ایراد در افراز بالا هست: هر مثلث متساوی الاضلاع، متساوی الساقین هم هست! پس این بلاکها گسسته (منفصل) نیستند و اشتراک دارند. پس باید آن را اصلاح کنیم:

تعیین خصوصیات ورودیهای TriType از دیدگاه هندسی (اصلاح شده)

خصوصیت	b_1	b_2	b_3	b_4
q_1 = "تعیین نوع هندسی مثلث"	مختلف الاضلاع	متساوی الساقین و غیر متساوی الاضلاع	متساوی الاضلاع	نامعتبر

مثال: IDM برای TriType با رویکرد مبتنی بر عملکرد

- مقادیر برای این بلاکها می توانند به صورت زیر باشند

تعیین خصوصیات ورودیهای TriType از دیدگاه هندسی

خصوصیت	b_1	b_2	b_3	b_4
"تعیین نوع هندسی مثلث" = q_1	مختلف الاضلاع	متساوی الساقین غیر متساوی الاضلاع	متساوی الاضلاع	نامعتبر
مثلث	(4, 5, 6)	(3, 3, 4)	(3, 3, 3)	(3, 4, 8)

مثال: IDM برای TriType با رویکرد مبتنی بر عملکرد

- یک نوع دیگر از تعیین خصوصیتها برای مثال **TriType** می تواند به شکل زیر باشد:

تعیین خصوصیات ورودیهای TriType (چهار خصوصیت مجزای هندسی)

خصوصیت	b_1	b_2
q_1 = "مختلف الاضلاع"	True	False
q_2 = "متساوی الساقین"	True	False
q_3 = "متساوی الاضلاع"	True	False
q_4 = "معتبر"	True	False

- قواعد محدودیتی هم وجود دارد که خوب است آنها را چک کنیم:
- اگر متساوی الاضلاع = **True** آنگاه متساوی الساقین = **True**
- اگر معتبر = **False** آنگاه مختلف الاضلاع = متساوی الساقین = متساوی الاضلاع = **False**

استفاده از بیشتر از یک IDM

- برخی برنامه ها ممکن است دهها یا حتی صدها پارامتر داشته باشد
- در این حالت چندین IDM کوچک ایجاد کنید. (رویکرد تقسیم و غلبه)
- ایرادی ندارد اگر IDM ها همپوشانی داشته باشند
  - (یعنی یک متغیر در بیش از یک IDM وجود داشته باشد)

انتخاب ترکیبی از مقادیر

- پس از تعیین خصوصیتها و بلاکها باید مقادیر آزمون را تعیین کنیم.
- برای این کار از معیارها استفاده می کنیم. (تا مقادیر مؤثر و مناسبی انتخاب شوند)

- ساده ترین معیار: همه ترکیبها

پوشش همه ترکیبها (ACoC): همه ترکیبهای همه بلاکها از همه خصوصیتها باید استفاده شود

- تعداد آزمونها = حاصلضرب تعداد بلاکهای خصوصیتها = $\prod_{i=1}^{Q}(B_i)$

- دومین تعیین خصوصیت برای TriType (در رویکرد مبتنی بر واسط) منجر به ۴*۴*۴ یعنی ۶۴ آزمون می شود! که نه منطقی است و نه اقتصادی و نه عملی!

مثال انتزاعی (کلی)

- فرض کنید سه خصوصیت داریم با بلاک بندی های زیر:
- Q1: [A,B] Q2: [1,2,3] Q3: [x,y]

- پوشش همه ترکیبها (ACoC) منجر به ۱۲ آزمون خواهد شد با انتخاب مقادیری از بلاکها به شکل زیر:

(A,1,x)	(B,1,x)
(A,1,y)	(B,1,y)
(A,2,x)	(B,2,x)
(A,2,y)	(B,2,y)
(A,3,x)	(B,3,x)
(A,3,y)	(B,3,y)

- در این لیست مثلاً (A,1,x) به معنای آن است که
  - یک مقدار باید از بلاک A از خصوصیت Q1 انتخاب شود
  - یک مقدار باید از بلاک 1 از خصوصیت Q2 انتخاب شود
  - یک مقدار باید از بلاک x از خصوصیت Q3 انتخاب شود

معیارهای ISP – هر گزینه

- ۶۴ آزمون برای TriType خیلی زیاد است!
- یک معیار دیگر می گوید: باید از هر بلاک حداقل یک مقدار را امتحان کنیم.

پوشش هر گزینه (Each Choice Coverage) (ECC): از هر بلاکِ هر خصوصیت، یک مقدار باید حداقل در یک آزمون استفاده شود.

- تعداد آزمونها = تعداد بلاکها در بزرگترین افراز (خصوصیت با بیشترین بلاک) = $\mathbf{Max}_{i=1}^{Q}(B_i)$

برای TriType :

2, 2, 2

1, 1, 1

0, 0, 0

-1, -1, -1

برای مثال انتزاعی:

(A,1,x)

(B,2,y)

(A,3,x)

معیارهای ISP – دو به دو (Pair-wise)

- معیار هر گزینه منجر به آزمونهای کمتری می شود و ارزان است؛ ولی ناموثر هم هست!
- رویکرد دیگر می گوید: مقادیر را با هم ترکیب کنید...

پوشش دو به دو (Pair-wise Coverage) (PWC): هر مقدار از هر بلاکِ هر خصوصیت، باید با مقداری از هر بلاکِ هر خصوصیت دیگر ترکیب شود.

- حداقل تعداد آزمونها در این روش = حاصلضرب تعداد بلاکهای دو خصوصیت بزرگتر =

$$(\text{Max}_{i=1}^{Q}(B_i)) * (\text{Max}_{j=1,\, j!=i}^{Q}(B_j))$$

- حداکثر تعداد آزمونها = $\text{Max}(Bi)^2$

برای TriType:

2, 2, 2	2, 1, 1	2, 0, 0	2, -1, -1
1, 2, 1	1, 1, 0	1, 0, -1	1, -1, 2
0, 2, 0	0, 1, -1	0, 0, 2	0, -1, 1
-1, 2, -1	-1, 1, 2	-1, 0, 1	-1, -1, 0

مثال انتزاعی – پوشش دو به دو

(A,1)	(B,1)	(1,x)
(A,2)	(B,2)	(1,y)
(A,3)	(B,3)	(2,x)
(A,x)	(B,x)	(2,y)
(A,y)	(B,y)	(3,x)
		(3,y)

(A,1,x)
(A,2,y)
(A,3,x)
(B,1,y)
(B,2,x)
(B,3,y)

- در مثال انتزاعی داشتیم:
- Q1: [A,B] Q2: [1,2,3] Q3: [x,y]

- پوشش دو به دو به ۱۶ ترکیب روبرو نیاز دارد:

- در هر آزمون می توان بیش از یک ترکیب را تست کرد.
- مثلاً اگر ترکیب (A,1,x) را آزمایش کنیم، ترکیبهای (A,1) و (1,x) و (A,x) آزمایش می شوند.
- پس با آزمونهایی مثلاً آزمونهای زیر می توان پوشش دو به دو را برآورده کرد:

- در جدول روبرو علامت «-» به معنای don’t care است.
  - (هر بلاکی می تواند انتخاب شود)

معیارهای ISP – T تایی (T-wise)

- یک گسترش معمولی روی پوشش دو به دو (۲ تایی بود)، پوشش T تایی است.
- یعنی ترکیب T مقدار به جای دو مقدار.

پوشش T تایی (T-wise Coverage) (TWC): یک مقدار از هر بلاک، برای هر گروه از T خصوصیت باید ترکیب شود.

- تعداد آزمونها در این روش، حداقل = حاصلضرب تعداد بلاکهای t خصوصیت بزرگتر
- اگر همه خصوصیتها دارای یک تعداد بلاک باشند آنگاه: تعداد آزمونهای مورد نیاز، حداکثر =

$$(\text{Max}_{i=1}^{Q}(B_i))^t$$

- اگر t = تعداد خصوصیتها (یعنی Q) باشد، آنگاه این آزمون برابر است با آزمون همه ترکیبها
- یعنی Q-wise = ACoC
- آزمون t-wise گران است و مزایای آن روشن نیست!
- برای مثال انتزاعی و مثال TriType : اگر t=2 باشد آنگاه TWC=PWC که در اسلایدهای قبل دیدیم ؛
- و اگر t=3 باشد، چون Q=3، پس t=Q، و بنا بر این، TWC=ACoC که باز هم در اسلایدهای قبل ارائه شد.

معیارهای ISP –گزینه پایه

- گاهی آزمونگران برخی مقادیر را (بر اساس دانش خود از دامنه) مهم می دانند.

پوشش گزینه پایه (Base-Choice Coverage) (BCC): یک بلاک گزینه پایه برای هر خصوصیت انتخاب می شود، و با استفاده از هر گزینه پایه برای هر خصوصیت، یک آزمون پایه شکل می گیرد . برای یافتن آزمونهای بعدی هر بار همه گزینه های پایه را بجز یکی ثابت نگه می داریم و از گزینه های غیرپایه در هر خصوصیت دیگر استفاده می کنیم.

- تعداد آزمونها برابر است با یک آزمون پایه + برای هر بلاک باقی مانده دیگر یک آزمون؛
- یعنی:

$$1 + \sum_{i=1}^{Q} (B_i - 1)$$

برای مثال TriType : بلاک «بزرگتر از ۱» را به عنوان بلاک پایه برای همه خصوصیتها انتخاب کرده ایم (می توان بلاکهای دیگری را به دلخواه به عنوان بلاک پایه انتخاب کرد)

آزمون پایه: 2, 2, 2 سایر آزمونها: 2, 2, 1 2, 1, 2 1, 2, 2

2, 2, 0 2, 0, 2 0, 2, 2

2, 2,-1 2,-1,2 -1, 2, 2

مثال انتزاعی – پوشش گزینه پایه

- در مثال انتزاعی داشتیم:
- Q1: [A,B] Q2: [1,2,3] Q3: [x,y]

- فرض کنید بلاکهای پایه برابرند با:
  - بلاک A برای Q1
  - بلاک 1 برای Q2
  - بلاک x برای Q3

- پس گزینه پایه برابر است با: (A,1,x)
- آزمونهای مورد نیاز برای پوشش گزینه پایه به شکل روبه رو است:

آزمون پایه	(A,1,x)
سایر آزمونها	(B,1,x)
	(A,2,x)
	(A,3,x)
	(A,1,y)

معیارهای ISP – چندین گزینه پایه

- گاهی آزمونگران منطقاً بیش از یک گزینه پایه دارند.

پوشش چندین گزینه پایه (Multiple Base-Choice Coverage) (MBCC): یک یا چند بلاک گزینه پایه برای هر خصوصیت انتخاب می شود، و با استفاده از هر گزینه پایه برای هر خصوصیت، آزمون های پایه شکل می گیرد . برای یافتن آزمونهای بعدی هر بار همه گزینه های پایه را بجز یکی ثابت نگه می داریم و از گزینه های غیرپایه در هر خصوصیت دیگر استفاده می کنیم.

- اگر M آزمون پایه و برای هر Q، تعداد mi گزینه پایه داشته باشیم آنگاه تعداد آزمونها =

$$M + \sum_{i=1}^{Q} (M * (B_i - m_i))$$

برای مثال TriType : بلاک های «بزرگتر از ۱» و «برابر با ۱» را به عنوان بلاک پایه برای همه خصوصیتها انتخاب کرده ایم (می توان بلاکهای دیگری را به دلخواه به عنوان بلاک پایه انتخاب کرد)

آزمون پایه: 2, 2, 2	سایر آزمونها: 2, 2, 0	2, 0, 2	0, 2, 2
	2, 2,-1	2,-1, 2	-1, 2, 2
آزمون پایه: 1, 1, 1	سایر آزمونها: 1, 1, 0	1, 0, 1	0, 1, 1
	1, 1,-1	1,-1, 1	-1, 1, 1

مثال انتزاعی– پوشش چندین گزینه پایه

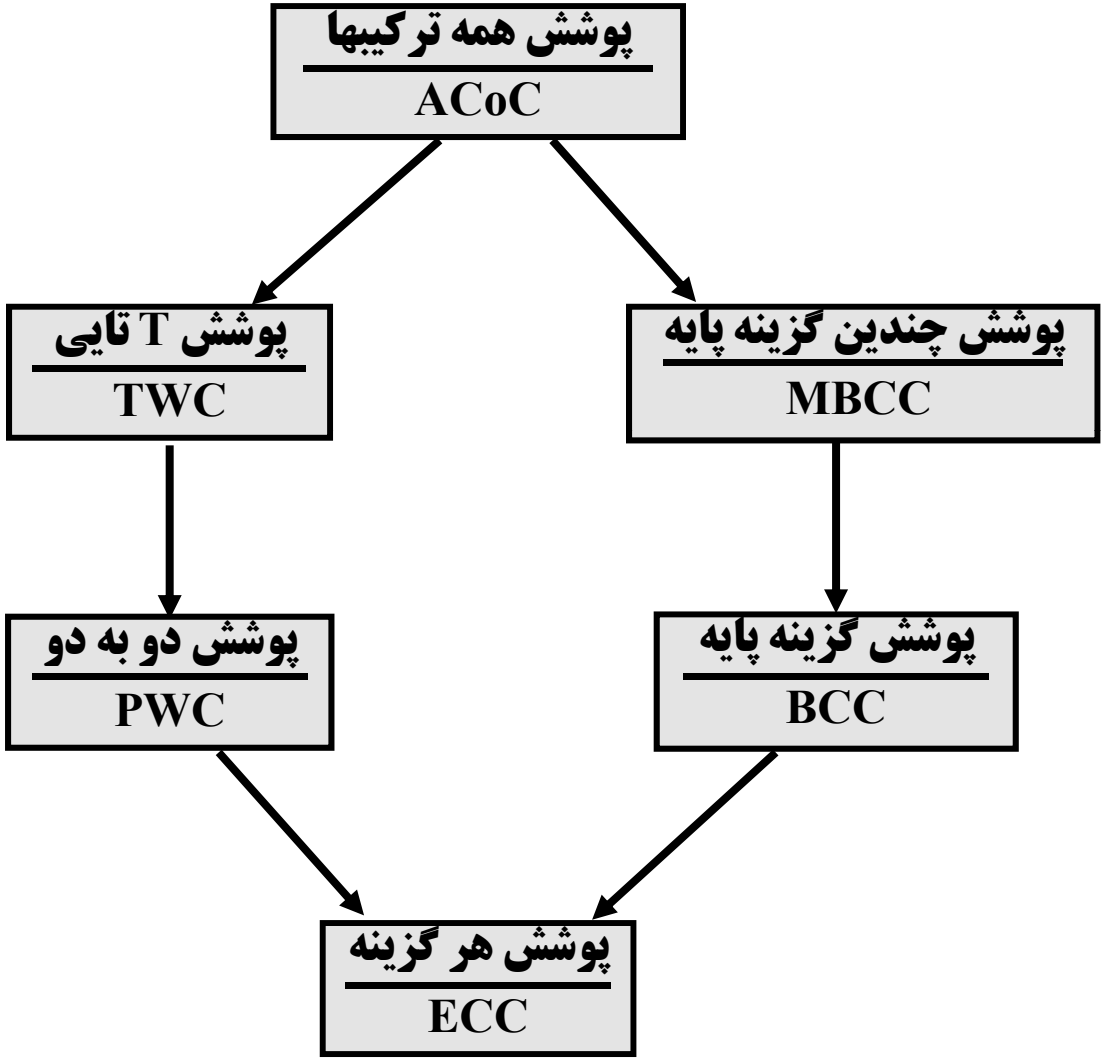
- در مثال انتزاعی داشتیم:
- Q1: [A,B] Q2: [1,2,3] Q3: [x,y]

- فرض کنید بلاکهای پایه برابرند با:
  - بلاک A و B برای Q1 (m1=2)
  - بلاک 1 و 2 برای Q2 (m2=2)
  - بلاک x و y برای Q3 (m3=2)

- پس تست های پایه برابر است با: (A,1,x) و (B,2,y) (یعنی M=2)
- آزمونهای مورد نیاز برای پوشش گزینه پایه به شکل روبه رو است:

آزمون های پایه	(A,1,x)
	(B,2,y)
سایر آزمونها	(A,3,x)
	(B,3,y)

رابطه در بر داشتن میان معیارهای ISP



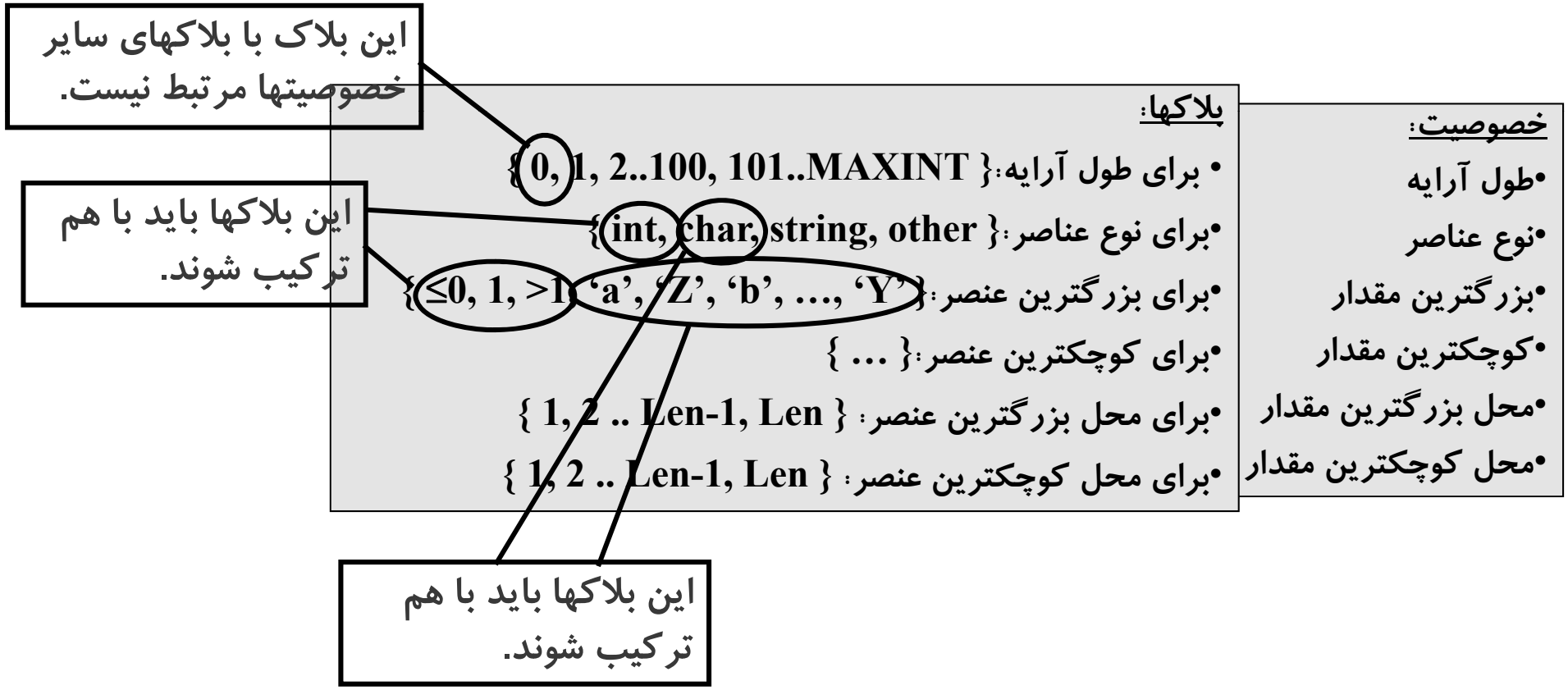
محدودیتهای میان خصوصیتها

- برخی ترکیبهای میان بلاکها ناممکن است:
  - مثلاً «کمتر از صفر» و «مختلف الاضلاع» با هم نمی توانند اتفاق بیفتند!
- این موارد را باید به عنوان محدودیتهایی (constraint) میان بلاکها تعریف کنیم.
- به طور کلی دو نوع محدودیت داریم:
  - یک بلاک از یک خصوصیت نمی تواند با بلاک معینی از خصوصیت دیگر ترکیب شود.
  - یک بلاک از یک خصوصیت فقط می تواند با بلاک معینی از خصوصیت دیگر ترکیب شود.

- مدیریت محدودیتها وابسته به معیار مورد استفاده است
  - برای ACoC، PWC و TWC: ترکیبهای ناممکن را حذف می کنیم
  - برای BCC و MBCC: یک مقدار را به مقدار غیرپایه دیگری تغییر می دهیم تا ترکیب ممکنی را پیدا کنیم

مثال از مدیریت محدودیتهای میان خصوصیتها

- مرتب سازی آرایه
  - ورودی: آرایه با طول متغیر با داده هایی از نوع دلخواه
  - خروجی: آرایه مرتب شده، بزرگترین و کوچکترین مقدار آن



خلاصه ISP

مثال TriType: Q=3، B1=B2=B3=4	مثال انتزاعی: Q=3، B1=B3=2، B2=3

معیار	شرح	فرمول تعداد آزمون	حاصل فرمول برای مثال TriType	حاصل فرمول برای مثال انتزاعی
همه ترکیبها (ACoC)	همه ترکیبهای همه بلاکهای همه خصوصیتها	$\prod_{i=1}^{Q}(B_i)$	4*4*4 = 64	2*3*2=12
هر گزینه (ECC)	از هر بلاک حداقل یک مقدار تست شود	$\underset{i=1}{\overset{Q}{Max}}(B_i)$	Max(4,4,4) = 4	Max(2,3,2) = 3
دو به دو (PWC)	هر بلاک از هر q با هر بلاک دیگر از هر q دیگر ترکیب شود	حداقل: $\underset{i=1}{\overset{Q}{Max}}(B_i) * \underset{j=1\neq i}{\overset{Q}{Max}}(B_j)$	4*4=16	3*2=6
T تایی (TWC)	هر بلاک از هر q با t بلاک دیگر از هر q دیگر ترکیب شود	حداکثر: $(\underset{i=1}{\overset{Q}{Max}}(B_i))^t$	اگر t=2 ، TWC=PWC اگر t=3، چون Q=3، پس: TWC=ACoC	اگر t=2 ، TWC=PWC اگر t=3، چون Q=3، پس: TWC=ACoC
گزینه پایه (BCC)	یک تست پایه و برای هر q، یک بلاک پایه داریم. هربار یک بلاک را به بلاکی غیرپایه تغییر می دهیم	$1 + \sum_{i=1}^{Q}(B_i - 1)$	1 + (3+3+3) = 10	1+ (1+2+1) = 5
چندین گزینه پایه (MBCC)	M تست پایه و برای هر qi، mi بلاک پایه داریم. برای هر تست پایه مثل BCC عمل می کنیم.	$M * (1 + \sum_{i=1}^{Q}(B_i - m_i))$	اگر M=2 و هر mi=2 باشد: 2*(1+(2+2+2))=14	اگر M=2 و هر mi=2 : 2*(1+(0+1+0))=4

خلاصه ISP برای مثال انتزاعی ...

در مثال انتزاعی داشتیم:

Q1: [A,B] Q2: [1,2,3] Q3: [x,y]

پوشش همه ترکیبها - ACoC

(A,1,x)	(B,1,x)
(A,1,y)	(B,1,y)
(A,2,x)	(B,2,x)
(A,2,y)	(B,2,y)
(A,3,x)	(B,3,x)
(A,3,y)	(B,3,y)

پوشش هر گزینه-ECC

(A,1,x)
(B,2,y)
(A,3,x)

(A,1)	(B,1)	(1,x)
(A,2)	(B,2)	(1,y)
(A,3)	(B,3)	(2,x)
(A,x)	(B,x)	(2,y)
(A,y)	(B,y)	(3,x)
		(3,y)

(A,1,x)
(A,2,y)
(A,3,x)
(B,1,y)
(B,2,x)
(B,3,y)

پوشش دو به دو-PWC

پوشش گزینه پایه-BCC

آزمون پایه

(A,1,x)

سایر آزمونها

(B,1,x)
(A,2,x)
(A,3,x)
(A,1,y)

پوشش چندین گزینه پایه-MBCC

آزمون های پایه

(A,1,x)
(B,2,y)

سایر آزمونها

(A,3,x)
(B,3,y)

پوشش t تایی-TWC:
اگر t=2 همان PWC است و اگر t=3=Q همان ACoC است.

پایان جلسه هفتم