Prova – CE-237 – 2º Semestre - 1º BIMESTRE de 2020

Aluno:

- Lucas Barioni Toma (ELE-21)

Professores:

- L. A. Vieira Dias
- Adilson M. Cunha
- Lineu F. S. Mialaret

Hora:

- inicial: `8:00`
- final: `9:38`

Primeira questão

1. O objetivo do teste é entender se o software irá funcionar corretamente ou se irá falhar total ou parcialmente. O teste de software também é feito para verificar seu comportamento sub condições diversas.
2. A adoção de um sistema numeral escrito, o sistema numeral arábico.
3. Apesar das linguagens compiladas terem surgido nessa década, a maior parte do software desenvolvido era feito em assembly, então, o teste de software era restrito à analise dos fluxogramas de programa. Outra característica que dificultava o teste de software era o uso de entradas e saídas caras e difíceis de manusear (como cartões pré perfurados e impressoras).
4. Flexível (multiparadígma) e Fácil de aprender/utilizar.

Segunda questão

1. Utilizaria algum algorítimo para a obtenção das soluções numéricas, como o método de Newton-Raphson.
2. Para testar se o resultado está correto, basta calcular o valor da função $y(x)$ para as raízes encontradas e verificar se $y(x) = 0$.
3. Verificaria os resultados produzidos pelas entradas as agruparia de acordo com o resultado obtido.
4. Utilizaria os valores das fronteiras das classes de equivalência e valores um pouco acima e um pouco abaixo, a separação entre os valores provavelmente não seria linear.
5. Pois as falhas normalmente ocorrem em torno dos valores de fronteira.

Terceira questão

1. Utilizando a regra do produto:
   - (6 Dispositivos).(5 Teclados).(3 Mouses).(7 Linguagens) = **630 casos de teste**
   - Observação: considerado que possa ser utilizado apenas uma linguagem de programação para cada caso
2. O tempo médio se dá pela soma das esperanças do tempo levado para abrir cada cadeado (que são iguais):
   $T_{med} = 4.\sum_{i=0}^{10000} t_i.p_i =$
   $4.(0,5.\frac{1}{10000} + 1,0.(\frac{9999.1}{10000.9999}) + 1,5.\frac{9999.9998.1}{10000.9999.9998} + \ldots + 5000.\frac{9999.9998\ldots2.1}{10000.9999\ldots2}) =$
   $4.\frac{(0,5+1,0+\ldots5000)}{2.10000} = 4.\frac{5000,5.10000}{20000} = 10001s$
3. O tempo máximo é dado pelo tempo do pior caso (10000 tentativas por cadeado):
   $T_{max} = (4).(10000).(0,5) = 20000s$
4. O tempo mínimo é dado pelo tempo do melhor caso (1 tentativa por cadeado):
   $T_{min} = (4).(1).(0,5) = 2s$

Quarta questão

1. Segundo a tabela de Tagushi (10 parâmetro e 5 níveis), 50 casos de teste
2. Teste com todas as combinações: $N = 5^{10} = 9765625 \text{ casos de teste}$
3.
   - n a n: R$ 97.656.250,00
   - 2 a 2: R$ 500,00
4. Pois é necessário resolver um sistema de N equações e N variáveis, o que exige muito mais operações algébricas (escalonamento e cálculos de determinantes) que apenas verificar se um solução é válida (multiplicar as matrizes e ver se o resultado é a matriz unitária).

Quinta questão

1. Pois em alguns casos nem sempre é possível encontrar um OA “perfeita”, sendo necessário utilizar de OAs maiores, que introduzem mais testes que o necessário (maior balanceamento), ao passo que o algorítimo de Todos os Pares sempre testa apenas o necessário.
2. Caso seja possível identificar classes de equivalência e valores de fronteira, é possível obtermos níveis discretos que representam os demais valores contínuos, tornando as técnicas aplicáveis.
3. Nesse caso, não é possível recorrer às técnicas de pairwise ou todos os pares, sendo necessário testar todas as combinações possíveis (n a n) ou, caso isso seja inviável, utilizar softwares que identificam e enumeram os casos de teste de triplets.
4. Quanto maior o número de parâmetros ou níveis, maior é o tamanho dos OAs (maior dificuldade para encontrar um OA) ou maior o esforço computacional para se utilizar os algorítimos de allpairs, o que inviabiliza o uso das técnicas de Pairwise Testing para um grande número de parâmetros ou níveis.