Exame: Quiz #08 **Aluno:** Rodrigo João Bastos Mendes

Resultado obtido não conta para a sua avaliação.
Apenas uma das respostas está correcta por pergunta.
Podem ser escolhidas várias respostas a uma mesma pergunta. Pode tornar a responder quantas vezes quiser ao questionário.
O tempo mostrado é só para uma tentativa de resposta a todo o questionário.
Para efeitos informativos, é mostrada uma pontuação: Cada questionário estará online para efeitos de frequência durante um curto perí
1 resposta correta: 1 pontos
1 resposta errada: -0.5 pontos

Valorização: 1.0
Correcção: as afirmações correctas estão marcadas com C e as erradas estão marcadas com E

Sair

1. Ciclos e Somatórios I

Qual o valor da variável `count` no final deste programa?

```
int count = 0;
for (int i=1; i<=n; i++)
    for (int j=i+1; j<=n; j++)
        count++;
```

[ ] E $n(n+1)/2$
[x] C $n(n-1)/2$
[ ] E $n^2/2$

2. Ciclos e Somatórios II

Qual é a complexidade assintótica deste programa? Note que $/$ é divisão inteira.

```
int count = 0;
for (i=1; i <= n/2; i++)
    for (j=1; j <= n/2; j++)
        count++;
```

[x] E $\Theta(n)$
[ ] E $\Theta(n \log n)$
[ ] C $\Theta(n^2)$

3. Dividir para Conquistar I

No uso da técnica de desenho algorítmico de dividir para conquistar, a parte onde se juntam as soluções de subproblemas mais pequenos para formar uma solução para o problema original é conhecida como:

[ ] E ~~Dividir~~
[ ] C **Combinar**
[x] E ~~Conquistar~~

4. Dividir para Conquistar II

No algoritmo `MergeSort`, a parte de juntar dois arrays ordenados (o *merge*) tem complexidade:

[x] C **linear**
[ ] E ~~constante~~
[ ] E ~~linearítmica~~

5. Recorrências I

Qual das seguintes recorrências corresponde a uma pesquisa binária? (partindo de um array ordenado, dividir sucessivamente o espaço de busca comparando com o elemento do meio)

[ ] E $T(n) = 2T(n/2) + \Theta(n)$
[x] C $T(n) = T(n/2) + \Theta(1)$
[ ] E $T(n) = 2T(n/2) + \Theta(1)$

6. Recorrências II

Qual a complexidade da recorrência $T(n) = 2T(n/2) + \Theta(1)$

[ ] C $\Theta(n)$
[ ] E $\Theta(\log n)$
[x] E $\Theta(n \log n)$

7. Recursividade I

Existem algoritmos que se podem fazer somente de forma recursiva. Esta afirmação é?

[ ] E ~~Não é possível determinar a veracidade da afirmação sem saber mais sobre o algoritmo em causa~~
[ ] E ~~Verdadeira, porque alguns problemas são apenas resolúveis através da recursão~~
[x] C **Falsa, porque qualquer algoritmo recursivo pode ser transformado num algoritmo iterativo equivalente**

8. Recursividade II

Quande podemos dizer que uma recursão não converge?

[x] E ~~Quando a recursão não é eficiente e leva a uma complexidade temporal exponencial~~
[ ] C **Quando as chamadas recursivos não são sempre aplicadas a casos mais pequenos que o inicial, ou quando não levam a recursão para mais perto de um caso base**
[ ] E ~~Quando a recursão diverge por estar dispersa pela memória~~

9. Recursividade III

Considere o seguinte método recursivo que tem como intenção descobrir o mínimo de um array:

```
int minRec(int v[], int start, int end) {
    int min = minRec(v, start + 1, end);
    return Math.min(v[start], min);
}
```

[x] E ~~O método está incorrecto porque a chamada recursiva não está correcta~~
[ ] E ~~O método está correcto~~
[ ] C **O método está incorrecto porque não tem nenhum caso base.**

10. Recursividade IV

Imagine que `count` é uma variável global que foi inicializada a 0. Se for feita uma chamada a maxRec2(v, 0, 7), com v[] a ser um array de tamanho 8, qual o valor de count quando essa chamada inicial termina? O método maxRec2 é o seguinte:

```
int maxRec2(int v[], int start, int end) {
    count++;
    if (start == end) return v[start];
    int middle = (start + end) / 2;
    int max1 = maxRec2(v, start, middle);
    int max2 = maxRec2(v, middle+1, end);
    return Math.max(max1, max2);
}
```

[x] E ~~8~~
[ ] E ~~1~~
[ ] C **15**