Questão 1 (6 pontos). Considere o algoritmo abaixo.

Algorithm 1

Entrada: Lista $A = A[1 \cdots n]$, com n inteiros

n = A.length;
1 **for** $i = 1$ *to* $n - 1$ **do**
  2 **for** $j = n$ *downto* $i + 1$ **do**
    if $A[j] < A[j-1]$ **then**
      swap = A[j-1];
      A[j-1] = A[j];
      A[j] = swap.

(a) O que este algoritmo realiza na lista A?

(b) Para se demonstrar formalmente que este Algoritmo 1 de fato funciona, é preciso demonstrar que ambos os loops funcionam corretamente. Para o loop interno, utilizou-se o seguinte loop invariante.

No começo de cada iteração do loop interno, $A[j] = \min\{A[k] : j \leq k \leq n\}$, *e ademais, a sublista* $A[j..n]$ *é uma permutação dos valores que estavam originalmente nestas posições antes do loop iniciar.*

Imagine que a corretude deste loop já foi demonstrada. Qual loop invariante deve ser usado para o loop externo? (não precisa demonstrar a sua corretude.)

Para a próximas questões, é preciso ter alguma justificativa. Use o verso.

(c) Considere agora todas as possíveis sequências de n números distintos de 1 a n, geradas com probabilidade uniforme. Seja s uma dessas sequências. Definimos a variável aleatória $X_{ij}(s) = 1$ se $s[i] > s[j]$, e $= 0$ caso contrário. Qual o valor esperado de X_{ij} ?

(d) Qual o valor esperado da quantidade de vezes que a condição do "if" no Algoritmo 1 irá testar verdadeira se a entrada for uma das sequências da letra c?

Questão 2 (9 pontos).

(a) Seja $T(n)$ uma função definida recursivamente, com $T(1) = 1$, e $T(n) = T(n-1) + 3n$. Demonstre, por indução, que
$$T(n) \in O(n^2).$$

(b) Seja $S(n)$ uma função definida recursivamente por $S(0) = S(1) = 1$, e $S(n) = 3S(n-4) + 2$.

(i) Esboce a árvore de recorrência desta função no verso.

(ii) Quantos níveis esta árvore possui?

(iii) Qual o custo do k-ésimo nível desta árvore? (conte de cima para baixo, com o primeiro nível sendo $k = 0$)

(iv) Encontre uma função $f(n)$, definida explicitamente, tal que $S(n) \in \Theta(f(n))$ (mas não é necessário mostrar por indução que seu chute funciona).

Questão 3 (6 pontos). Deseja-se implementar uma pilha usando listas. Há portanto uma lista A de tamanho n, e uma variável k que guarda a última posição alocada na lista. Ao solicitar a operação `Push(x)`, verifica-se se $k < n$. Caso sim, $k \leftarrow k+1$, e $A[k+1] = x$. Caso $k = n$, é preciso criar uma nova lista maior, copiar todos os elementos da antiga, e então adicionar o novo elemento. A operação `Pop` simplesmente retorna $A[k]$ se $k > 0$, e executa $k \leftarrow k - 1$.

Assuma custo constante igual a C para ler, copiar, ou gravar um elemento na lista, e custo irrelevante para alocar memória para a lista nova. Ou seja, o custo de aumentar a lista é essencialmente o custo de copiar os elementos da anterior.

(a) Suponha que ao ser necessário criar uma lista nova pois a anterior ficou cheia, decide-se criar uma lista de tamanho 1 unidade maior que a anterior. Supondo que a lista A inicia com tamanho 1, encontre $f(n)$ tal que o custo de n operações `Push` esteja em $\Theta(f(n))$ (justifique brevemente).

(b) Suponha agora que ao ser necessário criar uma lista nova, cria-se uma lista de tamanho o dobro da anterior. Suponha que A inicia com tamanho 1. Observe a tabela abaixo com o custo aproximado (supondo $C = 1$) de executar n operações `Push`.

n	1	2	3	4	5	6	7	8	9	10
custo acumulado	1	3	6	7	12	13	14	15	24	25

Separando o custo de inserir um novo elemento e o custo de dobrar a lista copiando os anteriores, escreva uma fórmula para quanto custa executar n operações do tipo `Push` e mostre que é $O(n)$.

(c) Aponte quanto seria um custo amortizado razoável para cada operação `Push`, e explique mais ou menos o que cada unidade deste custo estaria pagando.