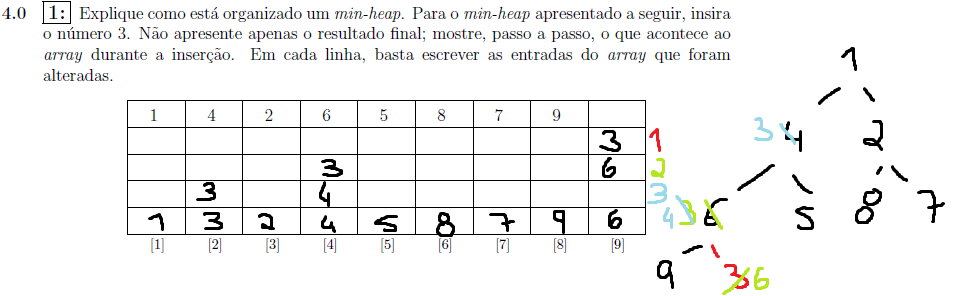
TESTE 2021\_T2\_AED

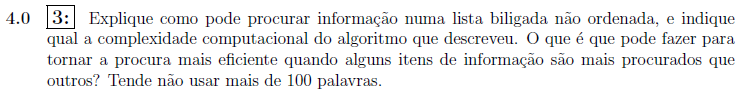
1)

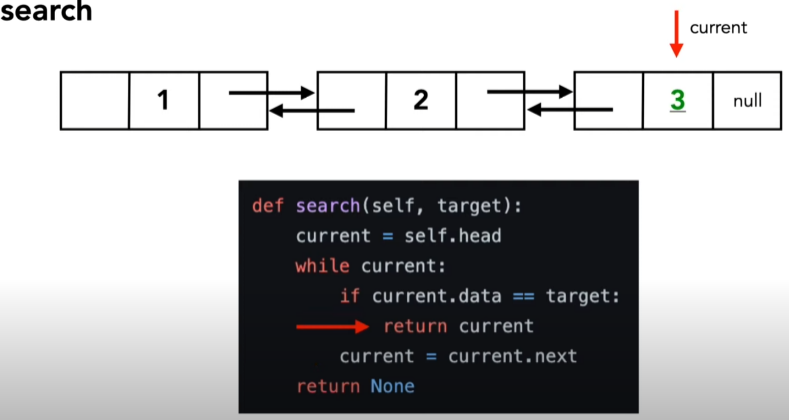


2)



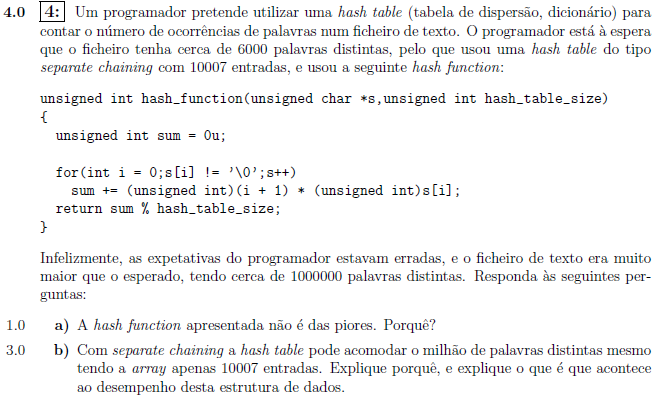
Insertion sort corre a lista elemento a elemento, caso i-1< i então troca as posições dos elementos. O melhor caso acontece quando a lista é percorrida apenas vez porque todas as posições já estão corretas. O pior caso é a lista estar ao contrário, isto é, caso o sort esteja a correr com o intuito de deixar os valores na ordem crescente, o pior caso seria estar de forma decrescente. Complexidade O(n2).

3) 



Para fazer search de uma lista biligada não ordenada é necessário correr todos os elementos da lista até chegar ao objetivo, pelo que tem uma complexidade de O(n). Para optimizar esta procura aplicaria um mecanismo de sort, que coloca a informação mais procurada mais perto da head da lista.

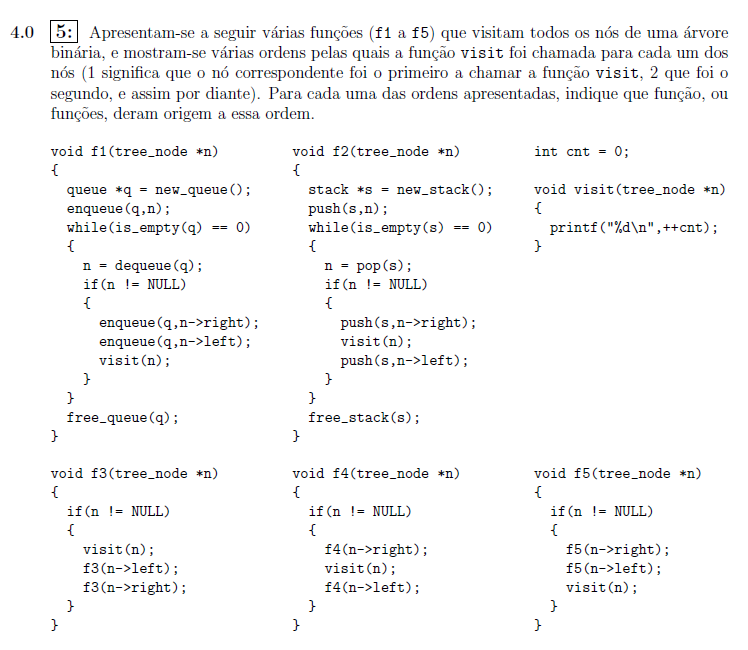
4)

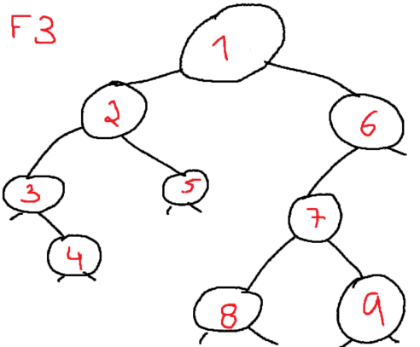
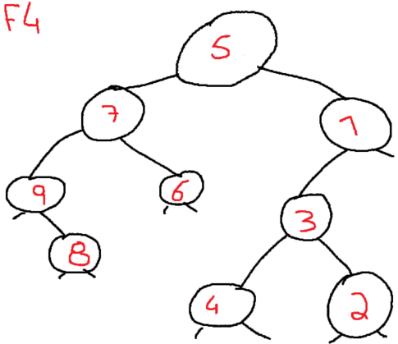
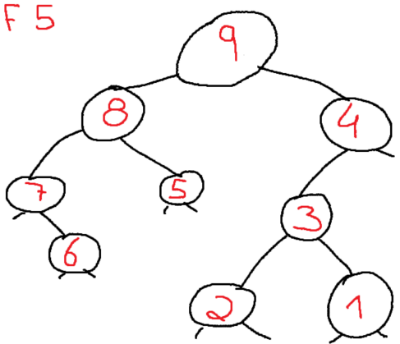


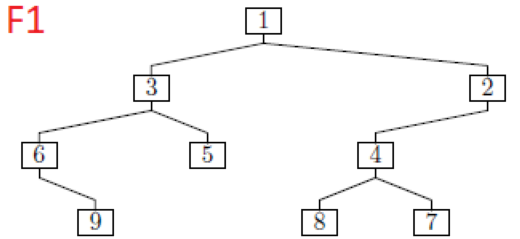
a) Para alem de usar o ascii usa também a posição do carater atual para o cálculo, logo evita produzir o mesmo resultado para diferentes inputs. Ex./ ABA e ABB terão diferentes outputs. Se apenas depende-se do ascii teriam o mesmo.

b) Ao invés de guardar valores, guarda pointers para linked-lists. Diminui significativamente a performance da hash table. Com a inserção de 10M elementos numa table com apenas 10007 entradas, cada entrada pode ter cerca de 999 conflitos (assumindo distribuiçao uniforme). Uma hash table devia ser O(1), mas com estes eventos, a procura de um elemento torna-se num processo iterativo potencialmente grande. O melhor caso é O(1), o pior é O(999) e o average O(500)

5)





Enqueue -> é da direção mencionada para a oposta, constrói uma min-heap, p/ex Enqueue right, é pela direita logo F1 é

Push -> empurra parede logo anda na direção contrária, F2=F3, p/ex push right vai pela esquerda primeiro