|  |  |  |  |
| --- | --- | --- | --- |
|  | **CENTRO UNIVERSITÁRIO GERALDI DI BIASE**  **FUNDAÇÃO EDUCACIONAL ROSEMAR PIMENTEL**  **PRÓ-REITORIA DE ASSUNTOS ACADÊMICOS - PROAC**  **CURSO: Sistemas de Informação**  **DISCIPLINA: Complexidade de Algoritmos – Trabalho 2 – Bimestre 1 – Valor: 2,0**  **PROFESSOR: Marcelo Arantes** | | |
| **PERÍODO: 3º - Sistemas de Informação** | | **TURNO: Noturno** | **Data de Entrega:** 29/09/2024 |
| **ALUNO (A):** | | | **MATRÍCULA:** |

Formato de Entrega: arquivo em formato .pdf – Equipes de até 6 componentes.

1. Considere alg1a e alg1b, algoritmos para calcular o Fibonacci de um número. Determine a complexidade assintótica de tempo de cada algoritmo fazendo uma comparação entre eles. (1,0)

def alg1a(n):  
 if n==1 or n==2:  
 return 1  
 else:  
 return alg1a(n-2)+alg1a(n-1)

def alg1b(n):  
 ant = 0  
 pos = 1  
 n = n-1  
 while n>=1:  
 print(n)  
 t = ant  
 ant = pos  
 pos = pos + t  
 n = n-1  
 return pos

1. Um aluno de Inteligência Artificial escreveu o algoritmo abaixo (função bt), como variação da busca em profundidade, e afirma ter reduzido a complexidade assintótica de espaço. Verifique se ele diz a verdade explicando o porquê e determine a complexidade assintótica de espaço desse algoritmo (m é um grafo, e c uma lista para mostrar o caminho – implementações em <https://drive.google.com/drive/folders/1lRe1NaIQ0VmR3lQwmYY__6IlqMzGjTRj?usp=sharing> ). (1,0)

def bt(self, m, o, d, a, v, c):  
 if d == a:  
 return True  
 else:  
 if not a in v:  
 v.append(a)  
 l = m.getVizinhos(a)  
 for i in l:  
 x = self.bt(m,o,d,i,v,c)  
 if x:  
 c.append(i)  
 return x  
 return False  
  
def busca(self, m, o, d):  
 v = list()  
 c = list()  
 if self.bt(m, o, d, o, v, c):  
 print("\nEncontrei: \n")  
 c.append(o)  
 self.caminho(c)  
 else:  
 print("Não encontrei o caminho. ")  
  
def caminho(self, c):  
 if len(c)>0:  
 l = c.pop()  
 print(l.getNome())  
 self.caminho(c)