Algoritmos e Estruturas de Dados II
EC4

Nome: Lucas Gonçalves Santos
Matrícula: 202120303511

Questão 1. [Maratona de Programação IME - 2022]
Escreva um programa em C, C++ ou Python, usando o Método Guloso, para o problema a seguir:
Alice tem identificado um comportamento estranho na rede de computadores de sua casa e suspeita que esteja sendo vítima de um ataque cibernético. Ela pesquisou um pouco sobre o tema na Internet e viu que uma prática comum para identificar e responder a tais ataques consiste no monitoramento do que acontece na rede. Depois disso, Alice viu que a maior parte dos dispositivos e aplicações que usa em sua rede geram arquivos de log, onde são registrados diversos eventos, como erros, avisos, e informações variadas sobre o funcionamento destes dispositivos e aplicações. Ela resolveu então fazer a coleta e o gerenciamento dos arquivos que contém esses logs para tentar identificar se, de fato, está sofrendo um ataque.
O problema é que Alice não possui recursos computacionais para realizar o gerenciamento de todos os arquivos de log produzidos. Para facilitar sua escolha de quais arquivos tratar, Alice criou um esquema de classificação dos arquivos de acordo com a criticidade, no qual atribuiu rótulos aos arquivos de acordo com os seguintes níveis: “1” - crítico, “2” – alto e “3” – moderado, tal que “1” >crit “2” >crit “3”, onde “>crit” significa “mais crítico que”. Cada arquivo possui também um tamanho, que representa o número de registros de eventos no arquivo. Alice quer processar a maior quantidade de eventos, respeitando o espaço disponível e priorizando os arquivos mais críticos. Ajude Alice a identificar o maior número de arquivos que ela consegue tratar.

Entrada
A entrada consiste de vários casos de teste. A primeira linha representa o número de casos de teste. Cada caso de teste consiste de uma linha com dois inteiros, sendo k correspondente ao espaço máximo para armazenamento dos arquivos e m o número de arquivos (1 <= m <= 10^5), seguida de m linhas correspondentes aos arquivos a serem processados com dois inteiros: criticidade (“1 – crítico, “2” – alto e “3” – moderado) e tamanho n (1 <= n <= 10^5).

Saída
A saída corresponde ao número máximo de arquivos que podem ser tratados por Alice, respeitando sua criticidade e o espaço máximo de armazenamento dos arquivos.

Foram realizados 2 mergeSort, um para o tamanho dado e outro para a criticidade dada.

```python
class Arquivo:
    def __init__(self, crit, tamanho):
        self.crit = crit
        self.tamanho = tamanho

def mergeSortTamanho(arr):
    if len(arr) <= 1:
        return arr
    meio = len(arr) // 2
    esq = mergeSortTamanho(arr[:meio])
    dir = mergeSortTamanho(arr[meio:])
    return mergeTamanho(esq, dir)

def mergeTamanho(esq, dir):
    merged = []
    i = j = 0
    while i < len(esq) and j < len(dir):
        if esq[i].tamanho <= dir[j].tamanho:
            merged.append(esq[i])
            i += 1
        else:
            merged.append(dir[j])
            j += 1
    merged.extend(esq[i:])
    merged.extend(dir[j:])
    return merged

def mergeSortCriticidade(arr):
    if len(arr) <= 1:
        return arr

    meio = len(arr) // 2
    esq = mergeSortCriticidade(arr[:meio])
    dir = mergeSortCriticidade(arr[meio:])

    return mergeCriticidade(esq, dir)
```

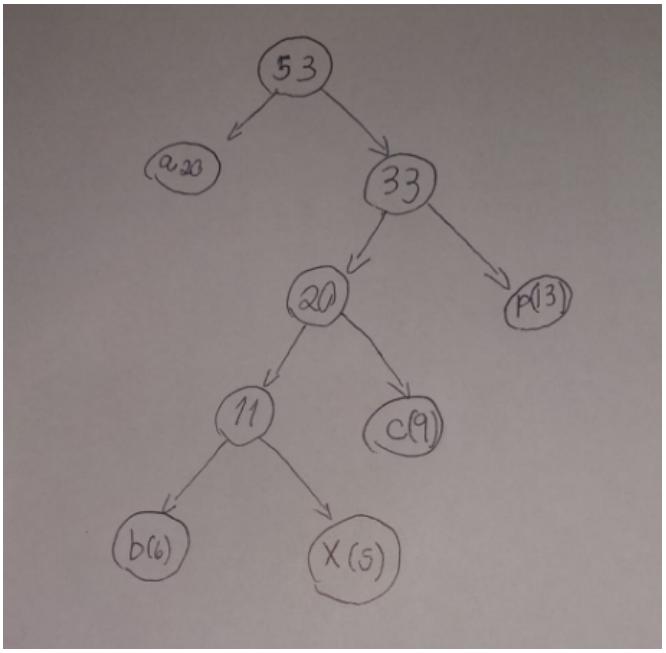
```python
def mergeCriticidade(esq, dir):
    merged = []
    i = j = 0
    while i < len(esq) and j < len(dir):
        if esq[i].crit <= dir[j].crit:
            merged.append(esq[i])
            i += 1
        else:
            merged.append(dir[j])
            j += 1
    merged.extend(esq[i:])
    merged.extend(dir[j:])
    return merged

t = int(input("Número de testes: "))
while t > 0:
    k = int(input("Espaço máximo: "))
    m = int(input("Quantidade de arquivos"))
    vetor = []
    for i in range(m):
        crit, tamanho = map(int, input("Crit/Tam ").split())
        vetor.append(Arquivo(crit, tamanho))
    mergeSortCriticidade(vetor)
    mergeSortTamanho(vetor)
    soma = 0
    contador = 0
    i = 0
    while k >= soma and contador < m:
        soma += vetor[i].tamanho
        contador = contador + 1
        i = i + 1
        if k < soma:
            contador = contador -1

    print(contador)
    t = t -1
```

Questão 2. Construa uma árvore de Huffman para os seguintes símbolos com suas respectivas frequências, onde NLPN = número de letras do seu primeiro nome, NLUS = número de letras do seu último sobrenome, NLNC = número de letras do seu nome completo: (X, NLPN), (P, 13), (A, NLNC), (B, NLUS) e (C, 9). Calcule o comprimento médio da codificação.

(X,5) = lucas
(P,13)
(A,20) = lucasgoncalvessantos
(B,6) = santos
(C,9)



Tamanho:
((20 * 1)+(13*2)+(9*3)+(6*4)+(5*4))/53 = 2,207