**UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC**

**CENTRO DE CIÊNCIAS E TECNOLOGIAS – CCT**

**ESTRUTURA DE DADOS I – EDA1**

**JOÃO GUILHERME VARGAS**

**TAREFA 2**: FDE COM PRIORIDADE

**JOINVILLE**

**2023**

**JOÃO GUILHERME VARGAS**

**TAREFA 2**: FDE COM PRIORIDADE

Relatório sobre a tarefa de Estrutura de Dados I.

Professor: Gilmário Barbosa Dos Santos

**FLORIANÓPOLIS**

**2023**

**Sumário**

[**1 Explicação da Metodologia 15**](#_30j0zll)

[1.1 Preparando os conjuntos de sequências 15](#_atqul6rawhkq)

[1.2 Lendo e armazenando o CSV 16](#_2edmmjyaqudh)

[1.3 Inserindo na FDE s/ RM 17](#_62z1h2r0hmr2)

[**2 Resultados 19**](#_ttj43tah6pk3)

# Explicação da Metodologia

## Preparando os conjuntos de sequências

Primeiramente decidi começar preparando os casos base aleatórios, para isso eu criei um vetor de inteiros de tamanho 10.000 e habitei ele com o inteiro que caia na devida posição. Logo em seguida eu usei a função rand() para aleatorizar as posições dele.

Como precisava de 18 sequências de casos de tamanhos diferentes eu criei uma matriz 18xN, onde N é o número de casos de cada conjunto, e habitei essa matriz com os N primeiros números da minha lista antes criada. Além disso, sempre depois de pegar os casos eu re-aleatorizava a lista de 10.000 itens, assim no final eu fiquei com uma matriz com as posições dos casos que não se repetia.

| int listaNumCasos[TAMANHO\_NUM\_CASOS] = {500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000}; int sequencia10kBase[10000]; int \*\*sequenciaPorNumCasos; srand(time(NULL));  sequenciaPorNumCasos = (int \*\*) malloc(TAMANHO\_NUM\_CASOS \* sizeof(int \*)); // Matriz de tamanho baseado no num de conjuntos de casos  for (size\_t i = 0; i < 10000; i++){ //Preenchendo o array 10k com a sequencia  sequencia10kBase[i] = i; }  for (size\_t i = 0; i < 10000; i++) { // Randomizando a sequencia do array 10k  int j = rand() % 10000;  int aux;   aux = sequencia10kBase[i];  sequencia10kBase[i] = sequencia10kBase[j];  sequencia10kBase[j] = aux; }  for (size\_t i = 0; i < TAMANHO\_NUM\_CASOS; i++) { // Habitando a matriz  sequenciaPorNumCasos[i] = (int \*) malloc(listaNumCasos[i] \* sizeof(int));   for (size\_t t = 0; t < listaNumCasos[i]; t++) { //Preencendo cada sequecnia de casos aleatoriamente  sequenciaPorNumCasos[i][t] = sequencia10kBase[t];  }   for (size\_t t = 0; t < 10000; t++) { // Re-randomizando a sequencia 10k  int j = rand() % 10000;  int aux;   aux = sequencia10kBase[t];  sequencia10kBase[t] = sequencia10kBase[j];  sequencia10kBase[j] = aux;  } } |
| --- |

## Lendo e armazenando o CSV

Como eu iria fazer a consulta já aleatorizada eu decidi armazenar o CSV bruto num vetor de strings de tamanho 10.000.

| char dataset[10000][256]; char linha[256];  //LOCALIZAÇÃO DO DATASET FILE \*datasetCSV = fopen("/home/jotausr/Projetos/UDESC/2023.2.EDA/Atividade 2/Atividade 2 - João Guilherme Vargas/Apresentacao/Aux/dataset\_v1.csv", "r");  fgets(linha, 256, datasetCSV);// Ignorar a primeira linha  for (size\_t i = 0; i < 10000; i++){  if (fgets(linha, 256, datasetCSV)) {  linha[strcspn(linha, "\n")] = 0; // Remove a quebra de linha   strcpy(dataset[i], linha);  strcpy(linha, "");  } else {  break;  } } |
| --- |

## Inserindo na FDE s/ RM

Para cada conjunto de casos eu usei um descritor vazio e inseri os itens do tipo Info seguindo a ordem das sequências previamente criadas, cada vez que um item era inserido na fila eu salvei o número de iterações necessárias.

Ao fim de cada loop eu fiz uma média da soma de iterações de um conjunto com o tamanho dele e salvei no vetor de médias. Além disso, eu decidi refazer todas as inserções 50 vezes e traçar uma média disso na saída para poder

| infoNode transformaCSV(char \*linha){ // Tokenização dos itens do csv e salvar como info  infoNode infoAux;  char \*token;   char \*teste = (char \*) malloc(256 \* sizeof(char));  strcpy(teste, linha);    token = strtok(teste, ",");  strcpy(infoAux.nome, token);   token = strtok(NULL, ",\n");  infoAux.matricula = atoi(token);   token = strtok(NULL, ",\n");  infoAux.ranking = atoi(token);   token = strtok(NULL, ",\n");  strcpy(infoAux.curso, token);   free(teste);    return infoAux; } |
| --- |

| int comparaSRM(infoNode \*a, infoNode \*b) { // Função de comparaçao  if (a->ranking < b->ranking)  return MAIOR;  else if (a->ranking > b->ranking)  return MENOR;  else  return IGUAL; } |
| --- |

| descFDE \*descritor = NULL; infoNode auxInfo; int totalIteracoes = 0; int listaMediaPorNumCasos[TAMANHO\_NUM\_CASOS];  for (size\_t i = 0; i < TAMANHO\_NUM\_CASOS; i++) { // Zerando a lista de medias por numero de casos  listaMediaPorNumCasos[i] = 0; }   for (size\_t k = 0; k < 50; k++) { // Loop para repetir o estudo 50 vezes e pegar uma media   for (size\_t i = 0; i < TAMANHO\_NUM\_CASOS; i++) { // Loop inicial percorre o numero de conjuntos aka 18  if ((descritor = criaSRM(sizeof(infoNode))) == 0) { // Cria um descritor de FDE S/ RM  printf("ERRO: Criacao do FDE s/ RM");  }   for (size\_t t = 0; t < listaNumCasos[i]; t++) { // Loop secundario percorre o numero de casos de cada conjunto  auxInfo = transformaCSV(dataset[sequenciaPorNumCasos[i][t]]); // Funçao devlve um struct tipo info a partir do csv armazenado  insereSRM(&auxInfo, descritor, comparaSRM, &totalIteracoes); // Insere info na fila e recebe o numero de iteraçoes  }   descritor = destroiSRM(descritor); // Ao fim destroi o descritor  listaMediaPorNumCasos[i] += totalIteracoes / listaNumCasos[i]; // Salva no vetor as medias por caso  totalIteracoes = 0;  }  }    for (size\_t i = 0; i < TAMANHO\_NUM\_CASOS; i++) { // Printar os resultados na tela  printf("%d,%d\n", listaNumCasos[i], listaMediaPorNumCasos[i] / 50); } |
| --- |

# Resultados

Analisando os resultados gerados nota-se que o número de iterações pelo número de casos se mostra proporcional ao número de casos, mantendo uma quase constante de 25% do número de casos.

Entretanto, se for analisar apenas o número total de iterações pelo número de casos é possível notar uma curva crescente similar a uma curva exponencial.

| **Número de Casos** | **Média de Iterações por**  **Número de Casos** | **Relação entre Número de Casos e Iterações** |
| --- | --- | --- |
| 500 | 131 | 26,20% |
| 1.000 | 243 | 24,30% |
| 1.500 | 377 | 25,13% |
| 2.000 | 505 | 25,25% |
| 2.500 | 618 | 24,72% |
| 3.000 | 775 | 25,83% |
| 3.500 | 887 | 25,34% |
| 4.000 | 1.010 | 25,25% |
| 4.500 | 1.134 | 25,20% |
| 5.000 | 1.285 | 25,70% |
| 5.500 | 1.381 | 25,11% |
| 6.000 | 1.531 | 25,52% |
| 6.500 | 1.643 | 25,28% |
| 7.000 | 1.774 | 25,34% |
| 7.500 | 1.891 | 25,21% |
| 8.000 | 2.056 | 25,70% |
| 8.500 | 2.150 | 25,29% |
| 9.000 | 2.305 | 25,61% |

| **Num de Casos** | **Média de Iterações**  **Totais** |
| --- | --- |
| 500 | 64.671 |
| 1.000 | 248.819 |
| 1.500 | 559.737 |
| 2.000 | 1.004.486 |
| 2.500 | 1.568.265 |
| 3.000 | 2.278.843 |
| 3.500 | 3.065.819 |
| 4.000 | 4.148.661 |
| 4.500 | 5.112.397 |
| 5.000 | 6.408.876 |
| 5.500 | 7.705.961 |
| 6.000 | 9.218.101 |
| 6.500 | 10.697.549 |
| 7.000 | 12.351.775 |
| 7.500 | 14.267.664 |
| 8.000 | 15.941.204 |
| 8.500 | 18.422.508 |
| 9.000 | 20.297.577 |

