Análise do código

Uma análise de trechos do código de alocação de horários e salas de aulas para disciplinas do curso DCC.

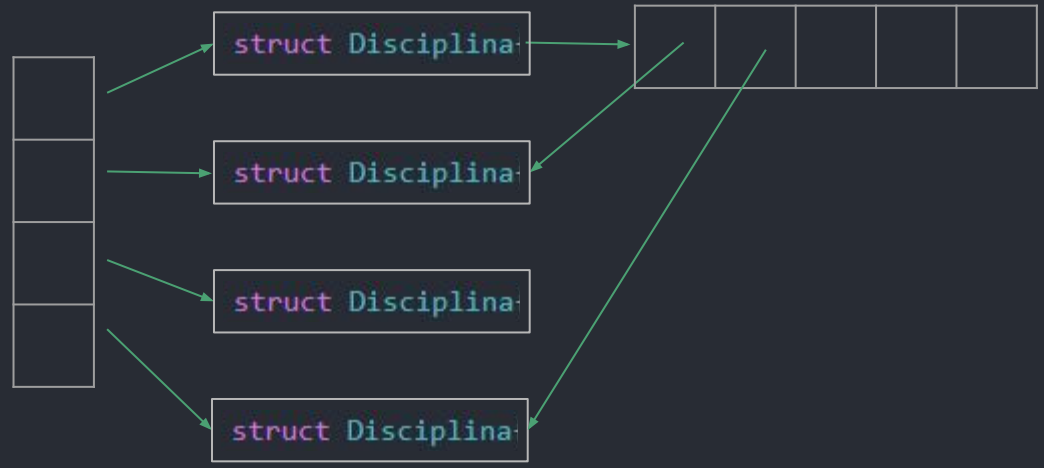
Estrutura utilizada para as disciplinas

```c
typedef struct Disciplinas{
  Disciplina **listadisciplinas;
  int quantidade;
} Disciplinas;
```

```c
typedef struct Disciplina{
  char codigo[7];
  char nome[60];
  int cargahoraria;
  char prerequisitos[18];
  char docente[30];
  int turma;
  int qtd_alunos;
  int recursos[3];
  int nConexoes;
  int horario;
  struct Disciplina **conexoes;
} Disciplina;
```

Estrutura utilizada para as disciplinas

```
typedef struct Disciplinas{
  Disciplina **listadisciplinas;
  int quantidade;
} Disciplinas;
```



Estrutura utilizada para as salas

```c
typedef struct Salas{
  Sala **listasalas;
  int quantidade;
} Salas;
```



```c
typedef struct Sala{
  char codigo[5];
  int capacidade;
  int recursos[3];
} Sala;
```

Análise da função lerDisciplinas( )

```
int lerDisciplinas(Disciplinas *disciplinas, char *url){
  int qtddisciplinas = contaArquivo(url);
  disciplinas->listadisciplinas = (Disciplina **) malloc(qtddisciplinas*(sizeof(Disciplina)));

  FILE *arquivo;
  Disciplina *disciplina;
  arquivo = fopen(url, "r");
  fflush(stdin);

  if(arquivo == NULL){…
  }else{…}
  fclose(arquivo);
  return 1;
}
```

O(n)

Análise da função lerDisciplinas( )

```
if(arquivo == NULL){
}else{
  while(!feof(arquivo)){
    disciplina = alocaDisciplina();
    fscanf(arquivo,"%[^,]s", disciplina->codigo);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%[^,]s", disciplina->nome);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d", &disciplina->cargahoraria);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%[^,]s", disciplina->prerequisitos);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%[^,]s", disciplina->docente);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d", &disciplina->turma);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d", &disciplina->qtd_alunos);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d", &disciplina->recursos[0]);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d", &disciplina->recursos[1]);
    fseek(arquivo, +1, SEEK_CUR);
    fscanf(arquivo,"%d\n", &disciplina->recursos[2]);
```

O(n)

Análise da função lerDisciplinas( )

```
void criaConexao(Disciplinas *disciplinas, Disciplina *disciplina){
  Disciplina *aux;
  int token;

  for(token = 0; token < disciplinas->quantidade; token++){
    aux = disciplinas->listadisciplinas[token];
    if(verifConexao(disciplina, aux)){
      disciplina->conexoes[disciplina->nConexoes] = aux;
      disciplina->nConexoes += 1;

      aux->conexoes[aux->nConexoes] = disciplina;
      aux->nConexoes += 1;
    }
  }
}
```

O(1)

Análise da função lerDisciplinas( )

```
int lerDisciplinas(Disciplinas *disciplinas, char *url){
  int qtddisciplinas = contaArquivo(url);
  disciplinas->listadisciplinas = (Disciplina **) malloc(qtddisciplinas*(sizeof(Disciplina)));

  FILE *arquivo;
  Disciplina *disciplina;
  arquivo = fopen(url, "r");
  fflush(stdin);

  if(arquivo == NULL){
  }else{}
  fclose(arquivo);
  return 1;
}
```

$$\sum_{i=0}^{n} 1 + \sum_{i=0}^{n} \sum_{j=0}^{i} 1 = n^2 + 5n + 4$$

n = Tamanho do arquivo

Análise da função coloreGrafo( )

```
void coloreGrafo(Disciplinas *disciplinas){
  Disciplina *disciplina, *aux;
  int token, i, ncoresatual, ncoresaux, cor;
  for(token = 0; token < disciplinas->quantidade; token++){
    disciplina = disciplinas->listadisciplinas[token];

    // ------ SAIO DESTE FOR COM O PRÓXIMO A SER COLORIDO
    for(i=0; i < disciplinas->quantidade; i++){
      aux = disciplinas->listadisciplinas[i];
      if(aux->horario != 0 || disciplina == aux){
        continue;
      }
      if(disciplina->horario != 0 && aux->horario == 0){
        disciplina = aux;
      }
      if(disciplina->nConexoes < aux->nConexoes){
        disciplina = aux;
      }else if(disciplina->nConexoes == aux->nConexoes){
        verifCoresAdj(disciplina, &ncoresatual);
        verifCoresAdj(aux, &ncoresaux);
        if(ncoresatual < ncoresaux){
          disciplina = aux;
        }
      }
    }
    // ------- FIM DA SELEÇÃO
    disciplina->horario = verifCoresAdj(disciplina, &ncoresatual);
  }
}
```

$$\sum_{i=0}^{n} G_i + \sum_{j=0}^{n} G_i + G_j = n^2 + n + G_i n + G_i$$

n = disciplinas
n[i] = disciplina
G_i = Grau de i

Análise da impressão do grafo

```
for(i=1 ; i<9 ; i++){
  for(j=0; j<disciplinas->quantidade;j++){
    if(disciplinas->listadisciplinas[j]->horario == i){
      printf("HORARIO: %s | DISCIPLINA: %s | ", cores[i], disciplinas->listadisciplinas[j]->codigo);
      for(k=0; k<salas->quantidade; k++){
        if(disciplinas->listadisciplinas[j]->qtd_alunos == salas->listasalas[k]->capacidade){
          if(disciplinas->listadisciplinas[j]->recursos[0] == salas->listasalas[k]->recursos[0]
             &&
             disciplinas->listadisciplinas[j]->recursos[1] == salas->listasalas[k]->recursos[1]
             &&
             disciplinas->listadisciplinas[j]->recursos[2] == salas->listasalas[k]->recursos[2]){
              printf("SALA: %s\n", salas->listasalas[k]->codigo);
            }
        }
      }
    }
  }
}
```

$$\sum_{i=0}^{9} \sum_{j=0}^{n} \sum_{k=0}^{m} 1 = 9(n + n*m)$$

n = disciplinas
m = salas