Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Thiago do Nascimento

Implementação do array dinâmico

Natal – RN

2019

ali_check_type – O(1)

Recebe um array_list_int e checa, usando comparando o magic, se é um struct válido.

Retornos (int):

1 – Caso não seja válido.

0 – Caso seja válido

```
int ali_check_type(array_list_int ali){
  if (ali->magic!=MAGIC){
    return 0;
  } else {
    return 1;
  }
}
```

ali_realloc

Recebe um array_list_int, realoca a sua memória e aumenta a sua capacidade em 1

Retornos (int):

Nova capacidade – Caso tenha sido realocado com sucesso.

0 – Caso realocação falhe

```
int ali_realloc(array_list_int ali){
  int* tmp = (int *) realloc(ali->a, (ali->size + 1)*sizeof(int));
  if (tmp != NULL){
    ali->a = tmp;
    ali->capacity++;
    return ali->capacity;
  }
  return 0; /* Realloc could not allocate new memory */

}
```

ali_create – O(1)

Cria um novo array com capacidade de 4 elementos.

Retornos (array_list_int):

Struct do array

```
array_list_int ali_create(){
  array_list_int ali= (array_list_int)malloc(sizeof(struct array_list_int));
  ali->a = (int*)malloc(sizeof(int)*4);
  ali->size=0;
  ali->capacity=4;
  ali->magic=MAGIC;
  return ali;
}
```

ali_get – O(1)

Recebe um array_list_int e um índice *i*.

Retornos (int):

Iésimo elemento – caso seja um índice e array válido

0 – Caso índice ou array sejam inválidos

```
int ali_get(array_list_int ali, int index){
  if (!ali_check_type(ali))
    return 0;
  if (index<0 || index>=ali->size){
    return 0;
  }
  return ali->a[index];
}
```

ali_push_back – O(N)

Recebe um array_list_int e um elemento, adicionando-o ao final do array, e realocando caso array esteja em capacidade máxima

Retornos (unsigned int):

-1 – Caso array esteja na capacidade máxima e não exista memória para realocar

0 – Caso não seja um array válido

Novo tamanho do array - Caso elemento tenha sido adicionado com sucesso

```
unsigned int ali_push_back(array_list_int ali, int value){
  if (!ali_check_type(ali))
    return 0;
  if (ali->size == ali->capacity)
    if (ali_realloc(ali) == 0)
      return -1;
  ali->a[ali->size++]=value;
  return ali->size;
}
```

ali_pop_back – O(1)

Recebe um array_list_int e diminui seu tamanho em 1

Retornos (unsigned int):

0 – Caso não seja um array válido ou tenha tamanho igual a 0

Novo tamanho do array - Caso seja um array válido e tenha tamanho maior que 0

```
unsigned int ali_pop_back(array_list_int ali){
  if (!ali_check_type(ali))
    return 0;
if (ali->size == 0)
    return 0;
  ali->size--;
  return ali->size;
}
```

ali_size – O(1)

Recebe um array_list_int e retorna seu tamanho.

Retornos (unsigned int):

0 – Caso não seja um array válido.

Tamanho do array - Caso seja um array válido

```
unsigned int ali_size(array_list_int ali){
  if (!ali_check_type(ali))
    return 0;
  return ali->size;
}
```

ali_find – O(n)

Recebe um array_list_int e e um valor inteiro e retorna o índice do valor – se existir – no array

Retornos (int):

-1 – Caso o valor não seja encontrado.

Indíce do elemento - Caso exista no array

```
int ali_find(array_list_int ali, int value){
  int i = 0;
  for (i=0; i<ali->size; i++){
    if (ali->a[i] == value){
      return i;
    }
  }
  return -1;
}
```

ali_insert_at – O(n)

Recebe um array_list_int, um valor inteiro e um índice, onde o elemento é inserido. Os elementos restantes são deslocadas para a direita.

Retornos (int):

0 – Caso a realocação tenha falhado ou não seja o array válido

Indíce do elemento - Caso a inserção tenha sido sucedida

```
int ali_insert_at(array_list_int ali, int index, int value){
  int i;
  if (!ali_check_type(ali))
    return 0;
  if (ali->size == ali->capacity)
    if (!ali_realloc(ali))
      return 0;
  ali->size++;
  for (i = (ali->size)-1; i > index; i--){
    ali->a[i] = ali->a[i-1];
  }
  ali->a[index] = value;
  return index;
}
```

ali_remove_from – O(n)

Recebe um array_list_int e um índice, o elemento deste é removido. Os elementos restantes são deslocados para a esquerda.

Retornos (int):

-1 – Caso não seja um índice valido

Indíce do elemento - Caso a remorção tenha sido sucedida

```
int ali_remove_from(array_list_int ali, int index){
  if (index < 0 || index >= ali->size) return -1;
  int i;
  for (i = index; i < ali->size-1; i++){
    ali->a[i] = ali->a[i+1];
  }
  ali->size--;
  return ali->size;
}
```

ali_capacity – O(1)

Recebe um array_list_int e retorna sua capacidade.

Retornos (unsigned int):

Capacidade do array

```
unsigned int ali_capacity(array_list_int ali){
  return ali->capacity;
}
```

ali_percent_occuped– O(1)

Recebe um array_list_e retorna a porcentagem ocupada.

Retornos (double):

De 0 a 1 – Ocupação do array

```
double ali_percent_occuped(array_list_int ali){
  return ((double)ali->size / (double)ali->capacity);
}
```