Tutorial 6

MATH 10017

8-9 Dec, 2022

Contents

1 Finding extreme values

1.1 Find max/min

Write a function to find the maximum element of an array by looping over the elements of the array.

- Decide how to initialize the value of `max`
- On each iteration of the loop, you need to decide whether to update max.

```
int find_max(int arr[], int len){
  int max;
  // YOUR CODE HERE
  return max;
}

int find_min(int arr[], int len){
  // YOUR CODE HERE
}
```

1.2 Testing

- Write code in `main` to test these functions.
- Use a few different arrays with different features (repeated values, positive and negative values, length = 1)

1.3 Find the second smallest number in an array

To find the second smallest number in an array, there are (at least) two different strategies:

1. modify the strategy from the previous section by keeping track of `min1` and `min2`; if `arr[i] > min1`, set `min2` to `min1` and set `min1` to `arr[i]`.
2. write a function `find_min_idx` that return the *index* of the smallest value in the array; set this value to `INT_MAX`, which is the largest possible value of an `int` (you need to import `limits.h` to use `INT_MAX` and `INT_MIN`). Now search the array for the minimum element using `find_min`.

Try the first strategy here:

```
int find_second_smallest(int arr[], int len){
  int min1, min2;
  // YOUR CODE HERE
  return min2;
}
```

1.4 Discussion

- How could you adapt these function to work for a vector structure? (e.g. like that defined in the next section)

- How could you modify these function to return the *index* of the maximum or minimum value?
- If you want to find the k smallest numbers in an array, then it might be more efficient to *sort* the array; we will talk about sorting the array later in the course. (If n is the length of the array then sorting will be better when $k$is much larger than $\log n$.)
- Can you write a function `void find_smallest(int arr[], int len, int vals[], int k)` that finds the k smallest values in `arr` and stores them in `vals`?

2 Gettings rows from row-major matrices

Remember that *row-major order* is a way to store a 2D matrix in a 1D array:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \implies (\underbrace{a_{11}, a_{12}, a_{13}}_{\text{row 1}}, \underbrace{a_{21}, a_{22}, a_{23}}_{\text{row 2}}, \underbrace{a_{31}, a_{32}, a_{33}}_{\text{row 3}})$$

The elements in each **row** form a *continguous block* of memory, meaning they are stored next to each other in memory. On the other hand, the elements of a **column** are not contiguous.

Thus, if a matrix is stored in row-major order, row operations will be more efficient than column operations.

2.1 Vector and Matrix structures

Here is some code you can use for the problems in the next section:

```
struct vector {
  int len;
  int *elements;
};
typedef struct vector Vector;

void print_vec(Vector v){
  for(int i = 0; i < v.len; i++)
    printf("%d ", v.elements[i]);
  printf("\n");
}
```

```
struct matrix {
  int nrow;
  int ncol;
  int *elements;
};
typedef struct matrix Matrix;

int idx(Matrix M, int i, int j){
  return i * M.ncol + j;
}

int get_elem(Matrix M, int i, int j){
  return M.elements[idx(M, i, j)];
}

void set_elem(Matrix M, int i, int j, int val){
  M.elements[idx(M, i, j)] = val;
}

void print_mat(Matrix M){
  for(int i = 0; i < M.nrow; i++){
    for(int j = 0; j < M.ncol; j++){
      printf("%d ", get_elem(M, i, j));
    }
    printf("\n");
  }
}
```

2.2 Get row

2.2.1 Write `get_row`

Write a function `get_row` that returns a `Vector` whose elements are the ith row of a `Matrix` M.

```
/*
 * get_row: return a vector whose elements are the i-th row of Matrix M
 */
Vector get_row(Matrix M, int i){
  Vector v;
```

```
  // YOUR CODE HERE
  // set length
  // set v's elements pointer
  return v;
}
```

Hints:

- To define a `Vector` you need an `int` for len and an integer pointer for elements.
- What should the length be?
- What should the first element in `v.elements` be? How can you get a pointer to this element?

2.2.2 Test `get_row`

In `main`:

1. Create a 2×3 matrix whose elements are $1, 2, 3, 4, 5, 6$.
2. Use `get_row` to create a vector whose elements are the 0-th row of this matrix.
3. Print the matrix.
4. Change the value of an element of the vector.
5. Print the matrix again.

Changing the vector should change the matrix. Since the elements of the vector are accessed by a pointer to the elements of the matrix, we say that the vector is a **view** of the row of the matrix.

We discuss this more below; it has practice implications for working with pandas in Python.

2.2.3 Use `get_row` to simplify some code

Use `get_row` and `print_vec` to print a `Matrix` using a single loop over the rows of the matrix:

```
void print_mat2(Matrix M){
  // YOUR CODE HERE
}
```

Test this function in `main` by printing out a matrix.

2.3 Get column (optional)

Write a function that returns a vector whose elements are the elements in the j-th column of a matrix:

```
Vector get_col(Matrix M, int j){
  Vector v = {M.nrow, malloc(M.nrow * sizeof(int))};
  // YOUR CODE HERE
  return v;
}
```

Another option would be passing a `Vector` along with the `Matrix` and column number.

```
/*
 * copy_col: copies the elements of the j-th column of Matrix M to the elements of Vec
 * Vector v: must have the correct length and have memory allocated.
 */
void copy_col(Matrix M, int j, Vector v){
  for(int i = 0; i < M.nrow; i++){
    v.elements[i] = get_elem(M, i, j);
  }
}
```

2.4 Discussion

- Our `get_row` function returns a pointer to the elements array of a matrix, so modifying a row obtained by `get_row` modifies the matrix. We say that `get_row` returns a **view** to the i-th row of a matrix.
- Why shouldn't you free the memory for a vector obtained by `get_row`?
- Should you free the memory for elements of a vector created by `get_col`?
- Modifying a vector created by `get_col` doesn't change the the corresponding values in the matrix. (Why?)

- Check out the first section of this jupyter notebook for a discussion of the effects of column-major order in pandas (a popular package for data manipulation in Python). `https://github.com/chiphuyen/just-pandas-things`