Program Assignment 2 Application of the Heap Structure

Oct 10, 2018

By *Jing-Jia Liou*

Contents

1 Problems

- There are two parts of this assignment:
  1. Reuse your assembly code and write some C code to find the k-th smallest element from an unsorted array.
  2. Prepare an unsorted array with a value K as a test case.

1.1 Penalty of plagiarism

- Each time you submit a plagiarized code, your grade of the homework will be discounted to 90%. The check is done in batch every hour. The check is based on code similarity at a 30% threshold (meaning 30% of your code is structurally identical to other's codes).
- The deduction will be **accumulated each time** you submit a plagiarized code. Please do not use trial-and-error approach to adjust your codes.
- If your final version (last submission before due date) is a plagiarized code, no credit will be given.

1.2 Part I Finding K-th smallest element from an unsorted array

Due: 11:59 am on Oct. 18, 2018

Proportion: 85%

- For this part, you have to revise your heapify program (which is your previous assignment) to an assembly function and write a C code to call the assembly function. Then find the k-th smallest element from an unsorted array (**sorting is not allowed in this assignment**).
- For example, give an array [15, 20, 10, 2, 7, 4, 8] and K = 3, your program should report 7.
- As for exception handling, you don't need to print any error message, simply use `return -1;`.
- We already prepared a template for you in the previous assignment, you can download it by the following command:

```
$ git clone http://gitlab.larc-nthu.net/ee3450_2018/pa1.git
```

- This is the content of our C code template. (pa1/pa1-2-heapify-function/main.c) Note that this is just a sample, the output is not the final output of your assignment.

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

void heapify_asm(int64_t nums[], int size);

void print_array(int64_t nums[], int len){
    for (int i = 0; i < len; ++i){
        printf("%lld ", nums[i]);
    }
    printf("\n");
```

```
}

int main(){

    int size = 0;
    int64_t *list = NULL;

    scanf("%d",&size);
    list = (int64_t*) malloc(sizeof(int64_t) * size);
    for(int i = 0; i < size; i++){
        scanf("%lld",&list[i]);
    }
    printf("Before heapify\n");
    print_array(list, size);
    heapify_asm(list, size);
    printf("After heapify\n");
    print_array(list, size);

    return 0;
}
```

- This sample code only presents how to use stdin and dynamic memory allocation to load data from a text file, you have to do some modifications on it.

- Using following commands to compile

```
$ cd pa1/pa1-2-heapify-function
$ make
```

- After your work, the expected output must looks like this.

```
$ make run < data.txt
```

```
7
$
```

- If you don't know how to call an assembly function in C code, please refer the lab1.

1.3 Part II Test Case

Due: 11:59 am on Oct. 18, 2018

Proportion: 15%

- Your test case should look like this

```
9
3
8 7 15 4 20 10 22 12 2
```

- Rule

1. The first line should be the number of integers. The maximum number is limited by the size of immediate field of RISC-V instructions (2^11-1). Make sure the number represents a valid complete binary tree.
2. The second line is the K value.
3. The third line includes positive or negative **32-bit integers**. The same integers can appear in the array.
4. The integers in the third line are separated with a space.
5. If you generate your test case file on Windows, please follow the guide to convert your file to a correct Unix text file.

1.4 Submission

- For example, if your student ID is **103061232**,
  1. Your part I assembly and C file name will be **hw2_103061232.S** and **hw2_103061232.c** respectively.
  2. Your test case file name will be **hw2_103061232.txt**
  3. Submit your home work via the link.

1.5 Hints

- Before writing the assembly code, we highly recommend you to write this program in high-level language first.