**1. Design document**

**Introduction:**

In this program we will implement the algorithm kSmall by a C++ function. We will assign the first values of the array as the pivot and create another function that will carry out the partition of the array. The partition function not only rearranges the array elements but also calculates the pivot’s index when the partition is done. If the first element is used as the pivot, S1 is empty. In this case, no swap is needed. Eventually, the array will have no change. In the end, the program depending on the user inputs gives us Kth smallest number of a list of numbers.

**Data structures:**

The program only uses one data structure, an integer (array of integers) called anArray. This array holds as many integers as the user defines. This array is then passed to partition which returns the pivot index and the kSmall function which returns the kth smallest integer in the array.

**Functions:**

There are only two functions in this program. The first function is the partition function which takes in the array and indexes of the first and last element of the array. Then based on logic provided in the function, it outputs the the index of the pivot

The second function is kSmall function which also takes in the array and indexes of the first and last element of the array and calls the partition functions and passes these values. It then performs the logic provided to find the kth smallest element in the array and returns the result.

**The main program:**

The main program at first asks the user to array size and then asks the user to enter the list based on the size using for loop and stores the inputs. It then asks the user for an integer value for for the kth smallest element. The program then calls the kSmall whilst passing the inputs, kSmall function calls the partition function and returns the result.

**2. Code list**

/\*\*This program implements the kSmall algorithm and partition function to find the kth smallest element in an array.

Mohammad Aryan Khan

py9242yg

09/27/2023

Instructor: Jie Meichsner

\*/

#include <iostream>

using namespace std;

/\*Partition function to rearrange elements in the array and calculate the pivot's index

\* @param list\_of\_elements pointer to the array

\* @param first\_indice first index of the array

\* @param last\_indice last index of the array

\* @pre while the left\_side indice is less than or equal to to the right\_side indice.

\* @post right\_side index of pivot

\*/

int partition(int \*list\_of\_elements, int first\_indice, int last\_indice) {

int pivot=list\_of\_elements[first\_indice];//pivot is the first element of the array

int left\_side=first\_indice + 1;

int right\_side=last\_indice;

while (left\_side<=right\_side) {

while (list\_of\_elements[left\_side] < pivot && left\_side <= last\_indice) {

left\_side++;

}

while (list\_of\_elements[right\_side] > pivot && right\_side >= first\_indice) {

right\_side--;

}

if (left\_side<=right\_side) {

int temp=list\_of\_elements[left\_side]; //swaps the elements

list\_of\_elements[left\_side]=list\_of\_elements[right\_side];

list\_of\_elements[right\_side]=temp;

left\_side++;

right\_side--;

}

}

list\_of\_elements[first\_indice]=list\_of\_elements[right\_side];

list\_of\_elements[right\_side]=pivot;

return right\_side;

}

/\*kSmall function to find the kth smallest element in the array

\* @param list\_of\_elements pointer to the array

\* @param first\_indice first index of the array

\* @param last\_indice last index of the array

\* @param k index to find the kth smallest element

\* @pre none

\* @post kth smallest element in the array

\*/

int kSmall(int \*list\_of\_elements, int first\_indice, int last\_indice, int k) {

int pivotIndex=partition(list\_of\_elements, first\_indice, last\_indice);

if (pivotIndex==k) {

return list\_of\_elements[k];

}

if (k<pivotIndex) {

return kSmall(list\_of\_elements, first\_indice, pivotIndex - 1, k);

}

else {

return kSmall(list\_of\_elements, pivotIndex + 1, last\_indice, k);

}

}

int main() {

int size;

cout<<"Enter the number of elements in the array(must be an integer!): ";

cin>>size;

int \*arr=new int[size];// dynamic array generated using new operator

cout<<"List the elements in the following way for example: 1 3 5 6 4 8 9\n";

cout<<"Enter the elements of the array(integers only!): ";

for (int i=0; i<size; i++) {

cin>>arr[i];

}

int k;

cout<<"Enter the index of the kth smallest element to find(integer only!): ";

cin>>k;

k=k-1;

int result=kSmall(arr, 0, size - 1, k);

cout<<"The " <<k + 1<<" smallest element is: "<<result<<endl;

delete[] arr;

return 0;

}

**3. User document**

The program we will implement the algorithm kSmall by a C++ function. The program assigns the first value of the array as the pivot and the function called partition will carry out the partition of the array. The partition function not only rearranges the array elements but also calculates the pivot’s index when the partition is done. If the first element is used as the pivot, S1 is empty. In this case, no swap is needed. Eventually, the array will have no change. In the end, the program depending on the user inputs gives us Kth smallest number from a list of numbers.

The project’s name is project3.cpp. It is located at the following directory on CentOS:

/home/STCLOUDSTATE/py9242yg/CSCI301/project3

To compile the program simply enter:

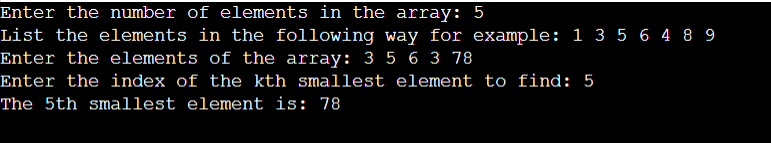
g++ project3.cpp

To run the program, enter:

./a.out

Then respond to the program’s prompts for inputs as specified. The program will continue to prompt the user for specific until the user decides to terminate the program.

For example,



**4. Test data plan**

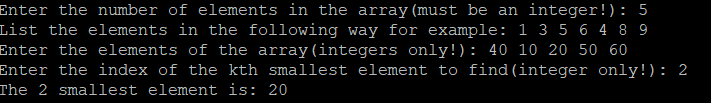
* Valid input:

Example of valid input values listing the same number of elements as the declared size of the array as shown above.

Testing the algorithm by 3 cases:

For simplicities sake I set the size to 5.

1st case: kth small is in s1

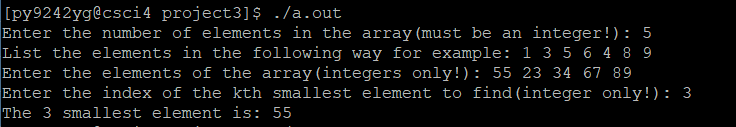


Here 40 is the pivot and as we can our kth smallest element 20 is in s1.

If we arrange the list of elements it would look like this: 10 20 40 50 60

40 is the pivot and our kth element 20 is in s1.

2nd case: kth small is the pivot

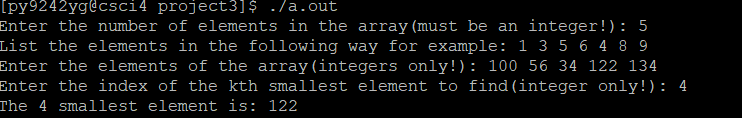


Here 55 is the pivot and also the kth smallest element.

If we arrange the list of elements it would look like this: 23 34 55 67 89

55 is the pivot and our kth element 55 which is the pivot.

3rd case: kth small is in s2



Here 100 is the pivot and as we can see our kth smallest element 122 is in s2.

If we arrange the list of elements it would look like this: 34 56 100 122 134

100 is the pivot and our kth element 122 which is in s2.

* Boundary input:

For input values the list of arrays could written like this:

Enter the elements of the array:1

2

3

4

And so, on

* Invalid input:

Any input that is not a positive integer will end up showing an error where the program basically terminates.

**5.Summary**

In this programming assignment we learned how to use recursive methods to solve a problem like finding the kth smallest element in an array. Recursion techniques basically involve making the problem at hand smaller than the original whilst making sure the problem remains the same. This program contains two functions, one is partition which looks through the array to find the index of the pivot and the kSmall function which calls the partition function and then computes the kth smallest element in the array.