Assignment2

Writer : 20171657 우성원

HW1 Problem

Given a string and a pattern , find all starting indices that matches a pattern inside a string.

HW1 Pseudocode

```
char s[31];
char p[31];
int failure[31];
bool success_idx[30];

void kmp_algorithm(char * string, char * pattern, int * failure, bool* success_idx){
    while (s_index < len_of_string) and (p_index < len_of_pattern){
      i = s_index, j = p_index
      if string[i] == pattern[j] {
        if end of pattern?
          then FIND MATCH
          success_idx[i-j] = True
          set i to  i-j + 1 and set j to 0
        else
          then increase 1 to s_index and p_index
      }
      else
        if ( j != 0 ) then j = failure[j-1] + 1
        else then NO MATCH , set i = i + 1, set j to 0
    }
}

void initialize_failure_function(char * pattern, int * failure){

    failure[0] = -1;
    for(int j=1; j< len_of_pattern ; j++){

        int i = failure[j-1];

        Set i = failure[i] until pattern[j] equals to pattern[i+1] or i < 0 (while loop)

        case 1 : failure[j] = i+1 if pattern[j] == pattern[i+1]
        csse 2 : failure[j] = -1 if i < 0
    }

}

main function:
  initialize_failure_function();
  kmp_algorithm();
```

HW1 Code Description

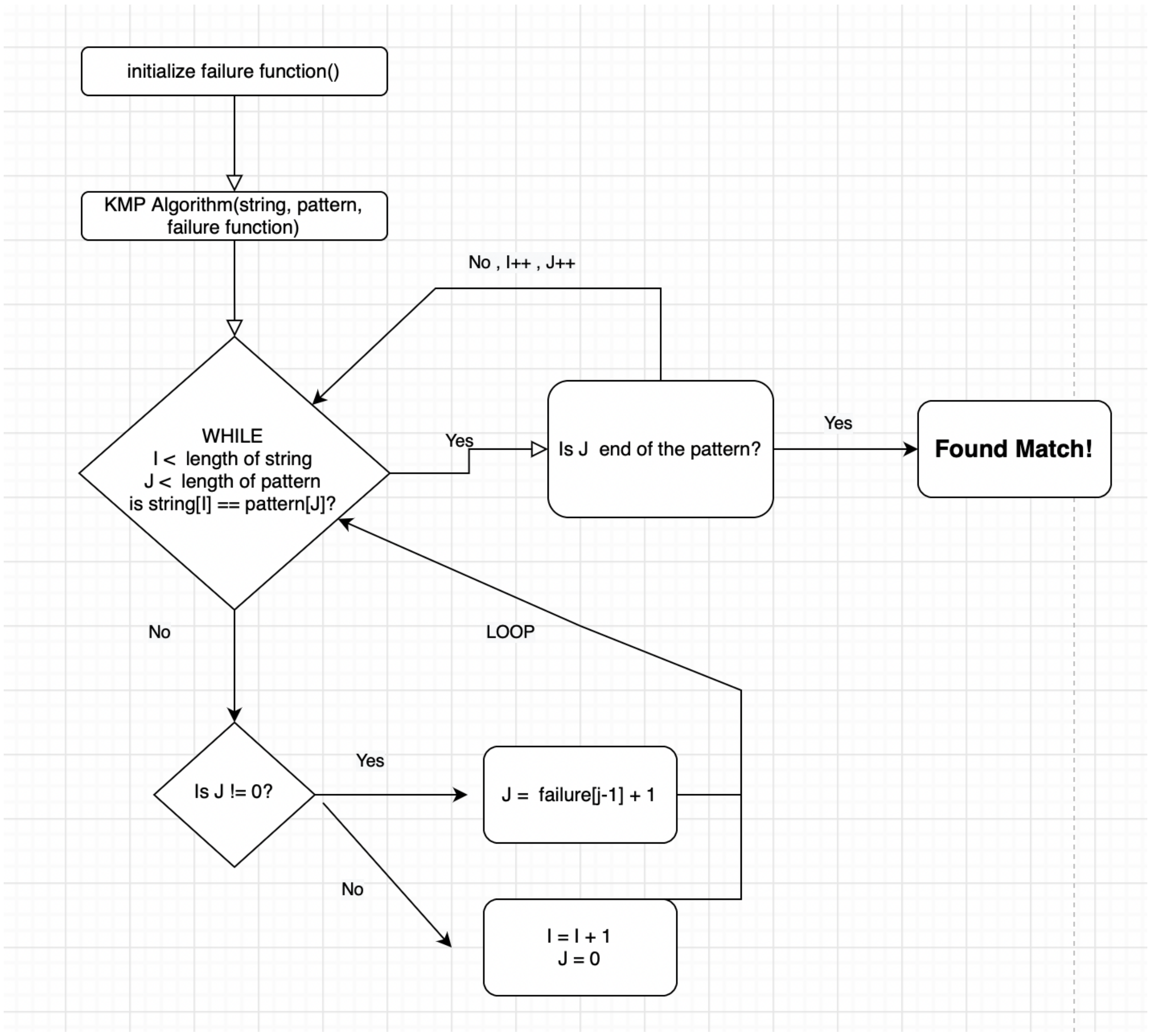
1. First, initialize failure function

- If index = 0, then failure[0] = -1
- For each index in Pattern Array 'j'
    - Initialize i = failure[j-1], then do while loop until pattern[j] equals to pattern[i+1] or i < 0
    - When breaks out of while loop ,
        - Case 1 : If pattern[j] is equal to pattern[i+1], then set failure[j] to pattern[i+1]
        - Case 2 : If not (i.e i < 0) , then failure[j] = -1

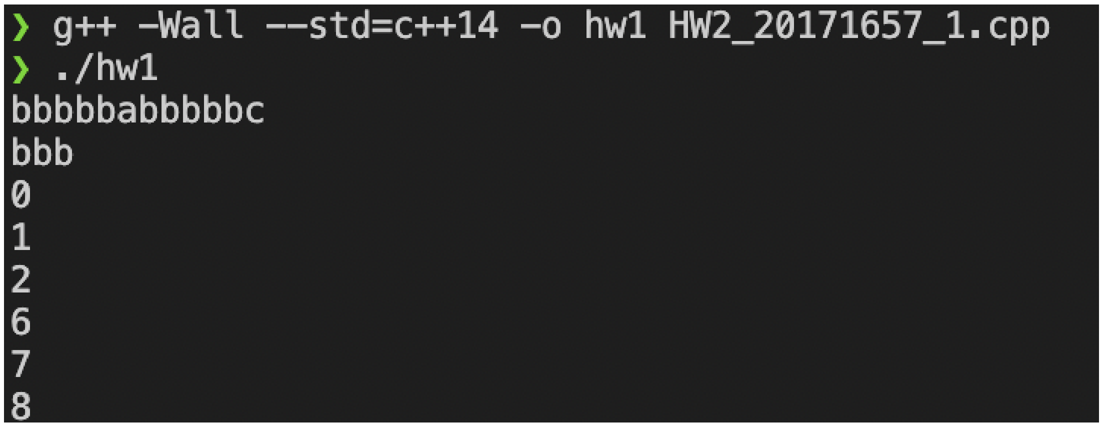
2. Second, do KMP algorithm

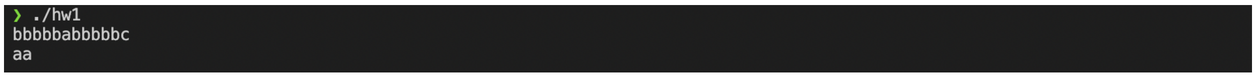
- Do While loop until **index I** is smaller than length of String and **index J** is smaller than length of pattern.
    - Inside a loop, compare String[I] and Pattern[J], if it is same value and J is the end of the index, then it is a Found Match.
    - If J is not end of the index, then increase I and J by 1.
    - If String[I] and Pattern[J] is a mismatch
        - Case 1 : If J ≠0, then set J = failure[J-1] + 1
        - Case 2 : Else, then set J = 0, I = I + 1

HW1 FlowChart


initialize failure function()
KMP Algorithm(string, pattern, failure function)
No , I++ , J++
WHILE I < length of string J < length of pattern is string[I] == pattern[J]?
Yes
Is J end of the pattern?
Yes
Found Match!
No
LOOP
Is J != 0?
Yes
J = failure[j-1] + 1
No
I = I + 1 J = 0

HW1 Test Cases


```
❯ g++ -Wall --std=c++14 -o hw1 HW2_20171657_1.cpp
❯ ./hw1
bbbbbabbbbbc
bbb
0
1
2
6
7
8
```


```
❯ ./hw1
bbbbbabbbbbc
aa
```

HW2 Problem

Given an Array, check if the elements in the array are consecutive values.

Condition : Time complexity : O(N)

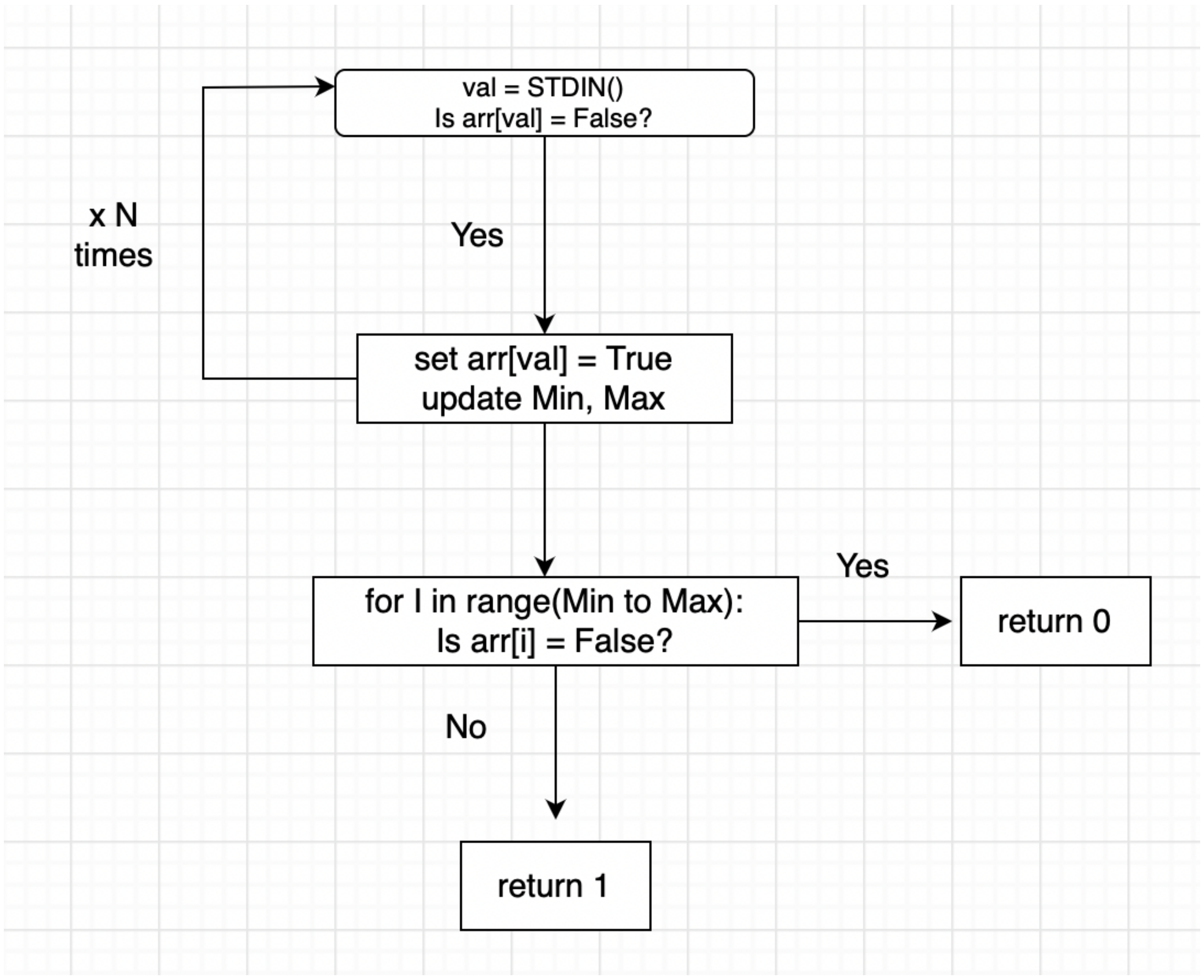
HW2 Pseudocode

```
bool arr[101]; // initially all false
int Min, Max;
N = stdin();

while( number of N ){
  val = stdin();

  if arr[val] == false?
    then arr[val] = true
  else
    set flag = 1

  if max < val
    then max = val
  if min > val
    then min = val
}

for (i = min ; i< max ; i++){
  if arr[i] == false?
    // then numbers are not consecutive
    return 0;
}
// numbers are consecutive
  return 1;
```

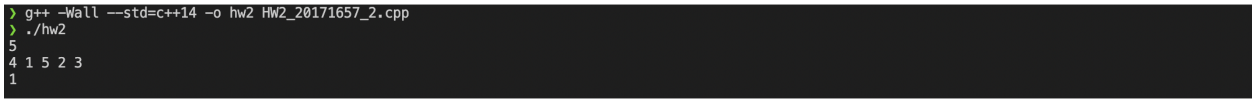
HW2 Code Description

1. First , make a boolean array and initialize **false**.
2. From stdin value, make array[value] to **true** . Update, Min, Max values at the same time.
3. After all stdin, from Min to Max, if there is a false value in array[], then the numbers are not consecutive.

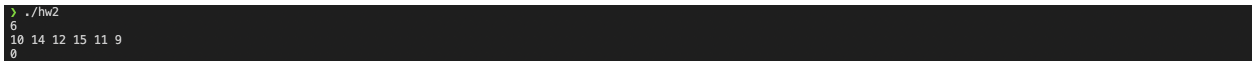
HW2 Flowchart



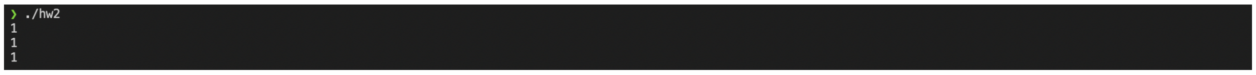
HW2 Test Cases



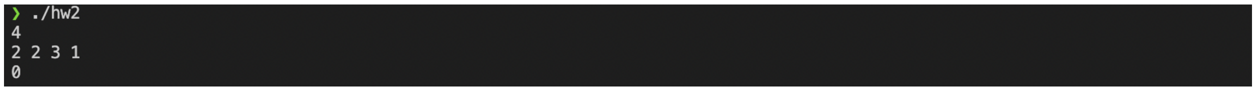
```
❯ g++ -Wall --std=c++14 -o hw2 HW2_20171657_2.cpp
❯ ./hw2
5
4 1 5 2 3
1
```



```
❯ ./hw2
6
10 14 12 15 11 9
0
```



```
❯ ./hw2
1
1
1
```



```
❯ ./hw2
4
2 2 3 1
0
```

HW3 Problem

Read data from student.txt , make a program that sorts student name according to lexical order.

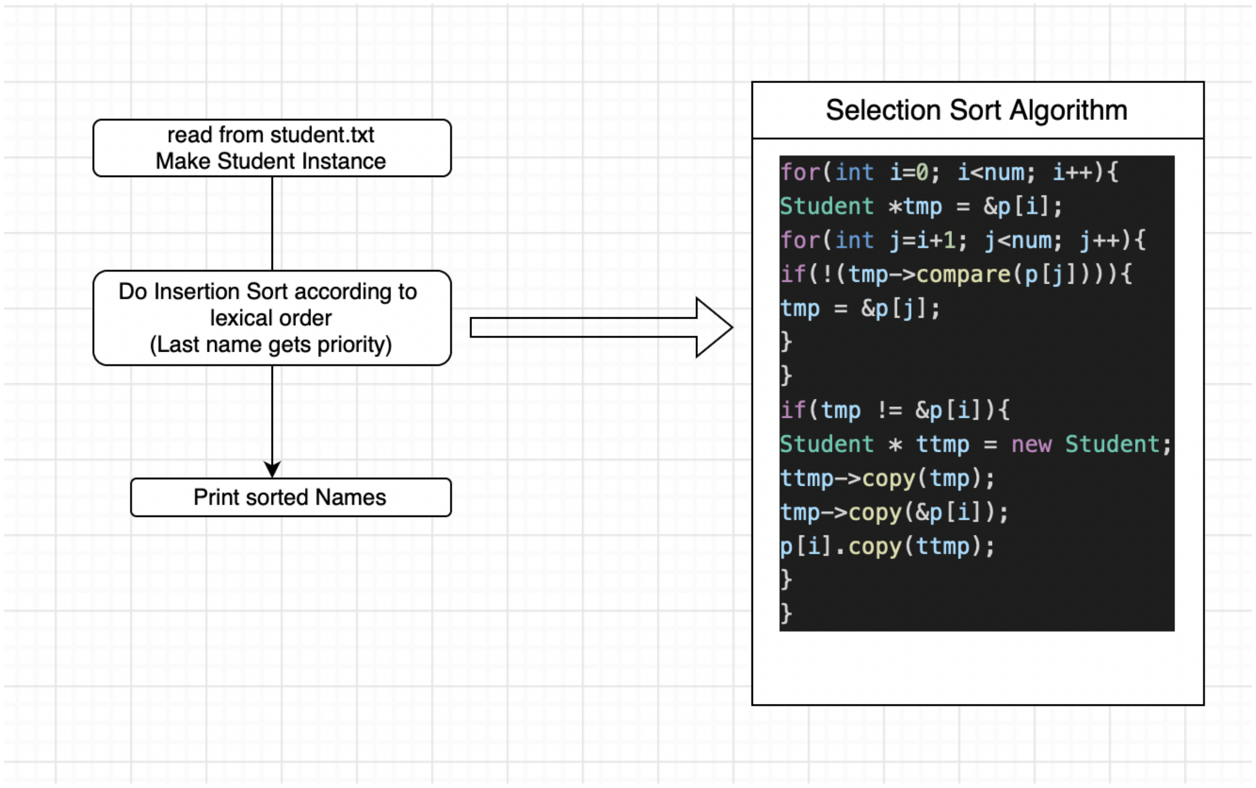
HW3 Pseudocode

```
class Student{
private:
    char last_name[50];
    char first_name[50];
public:
  void init_str(char * name)
   // initialize string to ""
  void replace_strcpy(char *str1, char * str2)
   // copying str2 to str1
  int replace_strcmp(char *str1, char *str2)
   // compare str1 and str2, If same then return 0, If str1 > str2 then return 1 , else return -1
  int replace_strlen(char * str)
   // returns length of str
  void allocate(char * n_l, char * n_f)
   // allocate private variable 'last_name' and 'first_name' from n_l , n_f
  bool compare(Student s)
   // compare Student1 and Student2 's last_name and first_name .
   // If Student1 name > Student2 name then return false else return true
  void print()
   // stdout last_name and first_name
  void copy(Student *tmp)
   // Copy name from Student tmp
}
function selection sort(){
  for i <= 1 to n-1
  min = i
  for j = i+1 to n
    if A[j] < A[min] then
      min = j
  end for
  if min != i then interchange A[i] and A[min]
}
main func(){
  1. Read from student.txt
  2. Initialize Student Instance
  3. Do insertion sort
  4. Stdout sorted names
}
```

HW3 Code Description

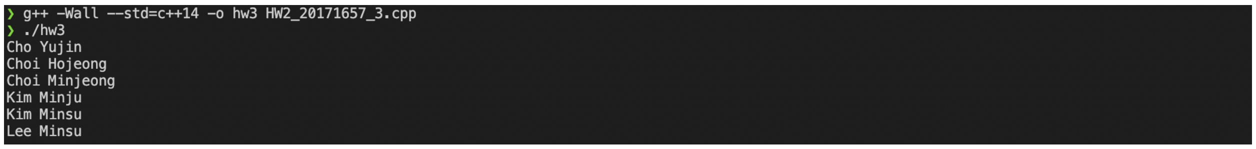
1. First, read data from student.txt
2. Second, Create a 'student instance' as many as the number read from ‘student.txt’.
3. Then do an selection sort according to lexical order. (Last Name gets priority than First Name, Based on ASCII value, but big alphabet gets priority then small alphabet if same alphabet [ex: A > a])
4. Print the sorted names.

HW3 Flowchart



```
for(int i=0; i<num; i++){
Student *tmp = &p[i];
for(int j=i+1; j<num; j++){
if(!(tmp->compare(p[j]))){
tmp = &p[j];
}
}
if(tmp != &p[i]){
Student * ttmp = new Student;
ttmp->copy(tmp);
tmp->copy(&p[i]);
p[i].copy(ttmp);
}
}
```

HW3 Test Cases



```
❯ g++ -Wall --std=c++14 -o hw3 HW2_20171657_3.cpp
❯ ./hw3
Cho Yujin
Choi Hojeong
Choi Minjeong
Kim Minju
Kim Minsu
Lee Minsu
```