ДЗ по алгоритмам № 3. Кривенко Андрей М3107

October 23, 2021

1.
```
int binSearchModified (int* arr, int l, int r, int x) {
  int mid = (l + r) / 2;

  if (l == r) {
    return mid;
  }

  if (arr[mid] == x) {
    return mid + 1;
  } else if (arr[mid] < x) {
    return binSearchModified(arr, mid + 1, r, x);
  } else {
    return binSearchModified(arr, l, mid, x);
  }
}

int algo (int* arr, int n, int x) {
  int* tmp = new int[n];
  tmp[0] = arr[0];
  for (int i = 1; i < n; ++i) {
    tmp[i] = tmp[i - 1] + arr[i];
  }

  return binSearchModified(tmp, 0, n, x);
}
```

2. Требуется найти разделяющий два массива элемент такой, что $a_{i-1} < a_i \wedge a_{i+1} < a_i$. После выполняем сортировку в 2 двух массивах по разным правилам: для возрастания и для убывания.

```
int findSeparator (int* arr, int l, int r) {
  int mid = (l + r) / 2;
  if (l >= r) {
    return -1;
  }

  if (arr[mid - 1] < arr[mid] && arr[mid + 1] < arr[mid]){
    return mid;
  }

  if (arr[mid - 1]  < arr[mid]) {
    return findSeparator(arr, mid, r);
  } else {
    return findSeparator(arr, l, mid);
  }
}
```

```
int algo (int* arr, int n, int x) {
  int sep = findSeparator(arr, 0, n);

  if (x == arr[sep]) {
    return sep;
  }

  if (x <= arr[sep - 1]) {
  int l = 0, r = sep;
  int mid = (l + r) / 2;
  while (l < r - 1) {
    mid = (l + r) / 2;
    if (arr[mid] == x) {
      return mid;
    } else if (arr[mid] < x) {
      l = mid;
    } else {
      r = mid;
    }
  }
  }

  if (x <= arr[sep + 1]) {
    int l = sep, r = n;
    int mid = (l + r) / 2;
    while (l < r - 1) {
      mid = (l + r) / 2;
      if (arr[mid] == x) {
        return mid;
      } else if (arr[mid] > x) {
        l = mid;
      } else {
        r = mid;
      }
    }
  }
  return -1;
}
```

3. Нет, так как при циклическом сдвиге последнее число, т.е. наибольшее в массиве, станет первым и массив будет таким, что: на первой позиции наибольший элемент, а далее элементы расположены по возрастанию $\Longrightarrow$ нельзя применить бинарный поиск

4. Можно записывать в стек структуру с элементами: число, сумма чисел предыдущих элементов. Тогда для того, чтобы получить сумму чисел в стеке, необходмо будет вернуть сумму чисел из последнего элемента стэка.

5. Проще написать псевдокод, чем реализацию на c++ из-за ньюансов. Для решения задачи 5 мы можем сначала перевести выражение из инфиксной нотации в постфиксную, и считать выражение уже в ней. Таким образом, решается сразу же и шестая задача.
   Алгоритм перевода из инфиксной нотации в постфиксную, на вход которого подается выражение (токеном назовем любой char из выражения):
   Предварительно создаем массив вывода и стэк операторов.

   while *Can read token* **do**
   Read token A
   if *A is number* **then**
   Push A to output array

end if
if $A\ is\ operator\ o_1$ **then**
while $There\ is\ operator\ o_2\ on\ top\ of\ operator\ stack\ \wedge\ o_2\ has\ greater\ precedence\ than\ o_1$ **do**
$Pop\ o_2\ from\ operator\ stack\ and\ put\ to\ output\ array$
end while
$Push\ o_1\ to\ operator\ stack$
end if
if $A\ is\ '('$ **then**
$Push\ o_1\ to\ operator\ stack$
end if
if $A\ is\ ')'$ **then**
while $Operator\ op_1\ at\ top\ of\ operator\ stack\ is\ not\ '('$ **do**
$Pop\ operator\ from\ operator\ stack\ to\ output\ array$
$Pop\ '('\ from\ operator\ stack$
end while
while $There\ are\ tokens\ on\ operator\ stack$ **do**
$Pop\ operator\ from\ operator\ stack\ to\ outputarray$
end while
end if
end while

Алгоритм подсчета выражения в постфиксной нотации:
while $Can\ read\ token\ from\ output\ array$ **do**
$Read\ token\ A$
if $A\ is\ operand$ **then**
$Push\ A\ to\ stack$
end if
if $A\ is\ operator$ **then**
$Pop\ stack\ and\ get\ X_1$
$Pop\ stack\ and\ get\ X_2$
$Count\ X = X_1\ A\ X_2,\ where\ A\ is\ operator$
$Push\ X\ to\ stack$
end if
end while
//Remaining element of stack is final value

6. Решено в предыдущей задаче

7. Для того, чтобы развернуть односвязный список, требуется в каждом эл-те списка указатель на следующий элемент заменить на указатель на предыдущий элемент.

8. Пусть за одну итерацию первый указатель на ноду переходит к следующей ноде, а второй указатель не две ноды вперед. Тогда, если эти два указателя встретятся, то список имеет кольцевой цикл.

9. Можем использовать алгоритм из предыдущей задачи. В списке будет должно быть два цикла в двух разных направлениях.

10. Сравниваем первые элементы двух отсортированных связных списков. Найти наименьшую голову среду первых эл-тов. Все же прочие эл-ты списков будут больше этой голову. Теперь запустим рекурсивную функцию, приминающую как параметры следующую ноду наименьшей головы и другую голову и возвращающую следующий наименьший эл-т связного списка. Таким образом: currentelement.next = recursivefunc()