добавлено: 30 Oct 2008 23:24
редактировано: 20 Aug 2012 23:55

Модификация стека и очереди для нахождения минимума за O (1)

Здесь мы рассмотрим три задачи: модифицирование стека с добавлением нахождения наименьшего элемента за O (1), аналогичное модифицирование очереди, а также применение их к задаче нахождения минимума во всех подотрезках фиксированной длины данного массива за O (N).

Содержание [скрыть]

Модификация стека

Требуется добавить возможность нахождения минимума в стеке за O (1), сохранив такой же асимптотику добавления и удаления элементов из стека.

Для этого будем хранить в стеке не сами элементы, а пары: элемент и минимум в стеке, начиная с этого элемента и ниже. Иными словами, если представить стек как массив пар, то

```
stack[i].second = min { stack[j].first }
                  j = 0..i
```

Понятно, что тогда нахождение минимума во всём стеке будет заключаться просто во взятии значения stack.top().second.

Также очевидно, что при добавлении нового элемента в стек величина second будет равна min (stack.top().second, new_element). Удаление элемента из стека ничем не отличается от удаления из обычного стека, поскольку удаляемый элемент никак не мог повлиять на значения second для оставшихся элементов.

Реализация:

```
stack< pair<int,int> > st;
```

- Добавление элемента:

```
int minima = st.empty() ? new_element : min (new_element, st.top().second);
st.push (make_pair (new_element, minima));
```

- Извлечение элемента:

```
int result = st.top().first;
st.pop();
```

- Нахождение минимума:

```
minima = st.top().second;
```

Модификация очереди. Способ 1

Здесь рассмотрим простой способ модификации очереди, но имеющий тот недостаток, что модифицированная очередь реально может хранить не все элементы (т.е. при извлечении элемента из очереди нам надо будет знать значение элемента, который мы хотим извлечь). Ясно, что это весьма специфичная ситуация (обычно очередь нужна как раз для того, чтобы узнавать очередной элемент, а не наоборот), однако этот способ привлекателен своей простотой. Также этот метод применим к задаче о нахождении минимума в подотрезках (см. ниже).

Ключевая идея заключается в том, чтобы реально хранить в очереди не все элементы, а только нужные нам для определения минимума. А именно, пусть очередь представляет собой неубывающую последовательность чисел (т.е. в голове хранится наименьшее значение), причём, разумеется, не произвольную, а всегда содержащую минимум. Тогда минимум во всей очереди всегда будет являться первым её элементом. Перед добавлением нового элемента в очередь достаточно произвести "срезку": пока в хвосте очереди находится элемент, больший нового элемента, будем удалять этот элемент из очереди; затем добавим новый элемент в конец очереди. Тем самым мы, с одной стороны, не нарушим

порядка, а с другой стороны, не потеряем текущий элемент, если он на каком-либо последующем шаге окажется минимумом. Но при извлечении элемента из головы очереди его там, вообще говоря, может уже не оказаться - наша модифицированная очередь могла выкинуть этот элемент в процессе перестроения. Поэтому при удалении элемента нам надо знать значение извлекаемого элемента - если элемент с этим значением находится в голове очереди, то извлекаем его; иначе просто ничего не делаем.

Рассмотрим реализацию вышеописанных операций:

```
deque<int> q;
```

- Нахождение минимума:

```
current_minimum = q.front();
```

- Добавление элемента:

```
while (!q.empty() && q.back() > added_element)
        q.pop_back();
q.push_back (added_element);
```

- Извлечение элемента:

```
if (!q.empty() && q.front() == removed_element)
        q.pop_front();
```

Понятно, что в среднем время выполнения всех этих операций есть O (1).

Модификация очереди. Способ 2

Рассмотрим здесь другой способ модификации очереди для нахождения минимума за O (1), который несколько более сложен для реализации, однако лишён основного недостатка предыдущего метода: все элементы очереди реально сохраняются в ней, и, в частности, при извлечении элемента не требуется знать его значение.

Идея заключается в том, чтобы свести задачу к задаче на стеках, которая уже была нами решена. Научимся моделировать очередь с помощью двух стеков.

Заведём два стека: s1 и s2; разумеется, имеются в виду стеки, модифицированные для нахождения минимума за O (1). Добавлять новые элементы будет всегда в стек s1, а извлекать элементы - только из стека s2. При этом, если при попытке извлечения элемента из стека s2 он оказался пустым, просто перенесём все элементы из стека s1 в стек s2 (при этом элементы в стеке s2 получатся уже в обратном порядке, что нам и нужно для извлечения элементов; стек s1 же станет пустым). Наконец, нахождение минимума в очереди будет фактически заключаться в нахождении минимума из минимума в стеке s1 и минимума в стеке s2.

Тем самым, мы выполняем все операции по-прежнему за O (1) (по той простой причине, что каждый элемент в худшем случае 1 раз добавляется в стек s1, 1 раз переносится в стек s2 и 1 раз извлекается из стека s2).

Реализация:

```
stack< pair<int,int> > s1, s2;
```

- Нахождение минимума:

```
if (s1.empty() || s2.empty())
        current_minimum = s1.empty ? s2.top().second : s1.top().second;
else
        current_minimum = min (s1.top().second, s2.top().second);
```

- Добавление элемента:

```
int minima = s1.empty() ? new_element : min (new_element, s1.top().second);
s1.push (make_pair (new_element, minima));
```

- Извлечение элемента:

```
if (s2.empty())
        while (!s1.empty()) {
                int element = s1.top().first;
```

```
                s1.pop();
                int minima = s2.empty() ? element : min (element, s2.top().second);
                s2.push (make_pair (element, minima));
        }
result = s2.top().first;
s2.pop();
```

Задача нахождения минимума во всех подотрезках фиксированной длины данного массива

Пусть дан массив A длины N, и дано число M ≤ N. Требуется найти минимум в каждом подотрезке длины M данного массива, т.е. найти:

```
min A[i],    min A[i],    min A[i],    ...,    min A[i]
0≤i≤M-1      1≤i≤M        2≤i≤M+1              N-M≤i≤N-1
```

Решим эту задачу за линейное время, т.е. O (N).

Для этого достаточно завести очередь, модифицированную для нахождения минимума за O (1), что было рассмотрено нами выше, причём в данной задаче подойдёт любой из двух методов реализации такой очереди. Далее решение уже понятно: добавим в очередь первые M элементов массива, найдём в ней минимум и выведем его, затем добавим в очередь следующий элемент, и извлечём из неё первый элемент массива, снова выведем минимум, и т.д. Поскольку все операции с очередью выполняются в среднем за константное время, то и асимптотика всего алгоритма получится O (N).

Стоит заметить, что реализация модифицированной очереди первым методом проще, однако для неё, вероятно, потребуется хранить весь массив (поскольку на i-ом шаге потребуется знать i-ый и (i-M)-ый элементы массива). При реализации очереди вторым методом массив A хранить явно не понадобится - только узнавать очередной, i-ый элемент массива.