AI Masters Algorithms **Name:** *Денис Грачев*

Homework 6

23 октября 2022 г.

Task 1

Обозначим очереди как a, b. Будем поддерживать следующий инвариант:

- После каждой операции b - пустая.

- Элементы в a находятся в том же порядке что мы их добавляли.

Тогда операции push(x) и pop(x) устроены следующим образом.

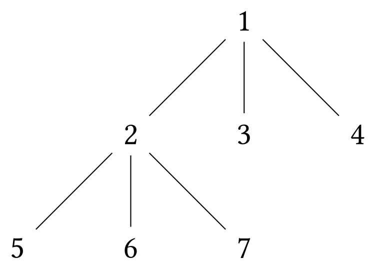
```
push(x) {
    a.add(x)
}


pop() {
    while not a.empty() {
        x = a.extract()
        if not a.empty() {
            b.add(x)
        }
    }
    while not b.empty() {
        a.add(b.extract())
    }
    return x
}
```

То есть при добавлении элемента, мы добавляем его в конец a и инвариант сохраняется. При доставании элемента мы перекладываем все кроме последнего элемента в b и затем обратно в том же порядке в a, то есть инвариант сохраняется и возвращаем мы последний добавленный элемент. Сложность операции push $O(1)$, сложность операции pop $O(n)$.

Task 2

Аналогично пронумеруем все вершины. Докажем тогда, что вершина с номером x имеет детей с номерами $3x-1, 3x, 3x+1$. Соотвественно родитель вершины x имеет номер $\lfloor\frac{x+1}{3}\rfloor$.



Пусть это верно для слоя h. Тогда самый средний лист этого слоя имеет индекс 3^{h-1}. Всего листов на этом слое 3^{h-1}, соотвественно самый левый лист имеет индекс $l_h = 3^{h-1} - \frac{3^{h-1}-1}{2}$, самый правый лист имеет индекс $3^{h-1} + \frac{3^{h-1}-1}{2}$.

Тогда самый левый индекс слоя $h+1$ имеет номер

$$\begin{aligned}
3^{h-1} + \frac{3^{h-1}-1}{2} + 1 &= \frac{2 * 3^{h-1} + 3^{h-1} + 1}{2} \\
&= \frac{3^h + 1}{2} \\
&= \frac{2 * 3^h - 3^h + 3 - 2}{2} \\
&= 3^h - \frac{3^h - 3}{2} - 1 \\
&= \left(3^{h-1} - \frac{3^{h-1}-1}{2}\right) 3 - 1 \\
&= l_h * 3 - 1
\end{aligned}$$

Следовательно для самого левого листа слоя h и его левого ребенка формула работает. Очевидно что для остальных его детей формула так же работает, то есть они имеют индексы $3l_h - 1, 3l_h, 3l_h + 1$. Тогда формула работает и для следующей вершины слоя h и тд, так как она имеет индекс $l_h + 1$, а ее дети $3l_h + 2, 3l_h + 3, 3l_h + 4 = 3(l_h + 1) - 1, 3(l_h + 1), 3(l_h + 1) + 1$.

База индукции видна на картинке.

Task 3

y - предок x

Пусть y не предок x. Тогда путь соединяющий x и y проходит через корень (обозначим корень r). Так как дерево бинарное, то они x и y по разным сторонам от r.
Таким образом

- либо $x < r < y$. Не может быть, так как y следующий по возрастанию от x.
- либо $x > r > y$. Не может быть, так как $x < y$.

Следовательно y - предок x.

y самый нижний, чей левый дочерний узел так же является предком x или самим x.

Так как y - предок x и $y > x$, то x находится в левом от y поддереве. Следовательно левый ребенок x является либо предком x либо сам является x.
Пусть z обладает тем же свойством но находится ниже y. Тогда т.к. y - предок x и z предок x, но ниже, то y - предок z и по доказанному ранее z находится в левом поддерве y. Тогда $x < z < y$, противоречье. Следовательно y самый низкий элемент обладающий таким свойством.

Task 4

a предок b либо b предок a

Рассмотрим вершину a. Докажем, что либо a предком b, либо b является предком a. Аналогично предыдущей задаче, если они находятся по разные стороны от корня, то либо $a < r < b$, либо $a > r > b$, оба варианта противоречивы.

a предок b

Пусть a предок b. Так как $a < b$, то b находится в правом поддереве a. Обозначим левого и правого ребенка b как l и r. Тогда $a < l < b < r$. Противоречье.

b предок a

Пусть b предок a. Так как $a < b$, то a находится в левом поддереве b. Пусть у a есть правый ребенок r, тогда $a < r < b$. Противоречье, следовательно правого ребенка у нее нет.

Для c

Аналогично для c

- либо c предок b, либо b предок c
- c не может быть предком b
- у c не может быть левого ребенка

Task 5

1

$s(n, k, 1)$. Так как $t = 1$, следовательно мы должны за 1 вопрос уметь отвечать принадлежит ли множеству A элемент k.

Будем хранить n битов, каждый из которых отвечает на вопрос лежит ли в множестве A элемент под номер i.

Тогда построения этой строки состоит из проставленяи 1 в соответсвующие ячейки.

Докажем что необходимо n битов.

Действительно представим себе n различных ситуаций где мы проверяем лежит ли число i в множестве A.

Все эти вопросы должны проверять разные биты строки w (пусть одним битом мы можем сказать что то про 2 числа, тогда 2 значениями мы закодировали 4 ситуации $a, b \in A; a, b \notin A; a \in A, b \notin A; b \in A, a \notin A$, противоречье). Следовательно нам необходимо хотя n бит.

2

Task 6

Докажем, что если отсортировать клиентов по времени обслуживаения, то суммарное время ожидания будет минимальным.

Рассмотрим инверсию $a_i, a_j : a_i > a_j, i < j$. Тогда если поменять местами a_i, a_j то суммарное время обсуживания уменьшится, так как для всех людей от 1 до $i - 1$ и j до n места время ожидания не изменится (это легко видеть, так как набор слагаемых времени их ожидания не изменился), а для людей от i до $j - 1$ места время ожидания уменьшится на $a_i - a_j$, так как одно из слагаемых a_j заменилось на a_i.

Таким образом если массива есть массива есть инверсия, то перестановку можно улучшить. Единственный порядок у которого нет инверсий - отсортированный.

Следовательно отсортированная очередь будет иметь наименьшее суммарное время ожидания.

Сложность алгоритма $O(nlog(n))$.