Algorithms and Data Structures

Master's ITMO, 2024

Theoretical tasks

Plan:

Неделя 1. Асимптотические оценки алгоритмов

```python
# функция min_search для поиска минимума в массиве arr
def min_search(arr: int[]):
    ans: int = MAX_INT
    for i: int = 0...len(arr)-1:
        if arr[i] < ans:
            ans = arr[i]
    return ans
```

Listing 1: поиск минимума в массиве

Конспект

- ▷ что такое алгоритм?
  - ⋆ это конечная последовательность действий для абстрактного вычислителя
  - ⋆ алгоритм получает на вход данные `input` и после завершения работы выдает данные `output`
- ▷ что такое RAM-модель?
  - ⋆ грубо говоря, это очень упрощенная модель компьютера
  - ⋆ есть память с произвольным доступом — RAM, Random Access Memory
  - ⋆ RAM представляет собой последовательный набор ячеек, где у каждой ячейки есть свой адрес
  - ⋆ за одно действие, можем обратится к одной ячейке памяти по ее адресу, для выполнения операции чтения/записи
  - ⋆ за одно действие, можем выполнить любую арифметическую/логическую операцию с двумя ячейками
- ▷ чем отличается реальный компьютер?
  - ⋆ про язык ассемблера можно почитать [здесь]
  - ⋆ арифметические инструкции (например `ADD`) работают примерно за один такт процессора, в то время как инструкции работы с оперативной памятью (например `MOV`) работают за несколько сотен тактов
  - ⋆ для оптимизации процессор *кеширует* данные: при обращении в оперативную память к ячейке с адресом i, процессор заодно выгружает из оперативной памяти несколько соседних с i-й ячеек — это называется *кеш-линией* — и запоминает их в *кеш-память*, которая находится прямо в процессоре и очень быстрая; теперь, если мы захотим обратится в оперативную память к ячейке с адресом $i + 1$, процессор не пойдет в оперативную память, а возьмет данные из *кеш*-а
- ▷ как оценить производительность алгоритма?
  - ⋆ засекать время плохо — у разных компьютеров разная производительность
  - ⋆ введем функцию $T(n)$ — количество абстрактных действий, которые делает наш алгоритм в RAM-модели, если на вход алгоритму подаются входные данные `input` и размер этих входных данных равен n

- ⋆ определим понятия *сложность алгоритма* и *время работы алгоритма* как функцию $T(n)$
- ⋆ на лекции мы посчитали, что время работы алгоритма [1] поиска минимума в массиве примерно равно $T(n) = 8 \cdot n + 3$ абстрактных операций

Def. Так как нас интересует только *асимптотика* функции $T(n)$ (порядок роста функции $T(n)$ относительно размера n входных данных), мы сказали, что нам интересен только наибольший член этой функции, а так же неинтересны константы. Чтобы формально и коротко это записать, мы ввели следующие обозначения:

- ⋆ $T(n) = \mathcal{O}(f(n))$
- ⋆ $T(n) = \Omega(g(n))$
- ⋆ $T(n) = \Theta(h(n))$

В примерах выше, функции $f(n)$, $g(n)$ — это соответственно *оценка сверху* и *оценка снизу* для нашей функции $T(n)$, в то время как $h(n)$ — это *точная оценка*.

Def. Напомним используемые далее определения:

- ⋆ $f(n) = \mathcal{O}(g(n)) \equiv \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \leqslant c \cdot g(n)$
- ⋆ $f(n) = \Omega(g(n)) \equiv \exists n_0, c > 0 : \forall n \geqslant n_0 : f(n) \geqslant c \cdot g(n)$
- ⋆ $f(n) = \Theta(g(n)) \equiv \exists n_0, c_1, c_2 > 0 : \forall n \geqslant n_0 : c_1 \cdot g(n) \leqslant f(n) \leqslant c_2 \cdot g(n)$

Практика

☐ Для каждой из рассматриваемых далее функций $f(n)$ найдите наиболее компактно записываемую $g(n)$, такую что $f(n) = \Theta(g(n))$.

1. $f(n) = 7n^2 - 7(n-3)^2$
2. $f(n) = 5n + 2\sqrt[3]{n}$
3. $f(n) = 10(n+1)^2 + 3(n-2)$
4. $f(n) = \log(\sqrt{n}) + \sqrt{\log n}$
5. $f(n) = n \cdot 3^{n+1} + n^{10}$
6. $f(n) = \frac{10n^2+2}{7n-1}$
7. $f(n) = \log(2n \log n)$

☐ Докажите следующие соотношения по определению (подберите константы c и n_0).

8. $\log n = \Omega(20)$
9. $2^n = \mathcal{O}(3^n)$
10. $n(n-8) = \Omega(n^2)$
11. $3n + 2\sqrt{n} = \mathcal{O}(n \log n)$
12. $n! = \Omega(5^n)$

☐ **Операции с нотацией $\mathcal{O}$-большое**. Докажите, что:

13. если $f(n) = \mathcal{O}(h(n))$ и $g(n) = \mathcal{O}(h(n))$, то верно что $f(n) + g(n) = \mathcal{O}(h(n))$
14. $f(n) + g(n) = \mathcal{O}(\max(f(n), g(n)))$

☐ Для следующих пар функций $f(n)$ и $g(n)$ покажите, верно ли, что $f(n) = \mathcal{O}(g(n))$.

15. $f(n) = \log n$
    $g(n) = \sqrt{n}$
16. $f(n) = n!$
    $g(n) = 2^n$
17. $f(n) = n \log n$
    $g(n) = \log(n!)$

☐ Время работы некоторого алгоритма задано следующим рекуррентным соотношением. Найдите Θ-асимптотику времени работы этого алгоритма, *построив дерево рекурсивных вызовов*.

18. $T(n) = 2 \cdot T(n-1) + 1$
19. $T(n) = T(\frac{n}{2}) + n$
20. $T(n) = 3 \cdot T(\frac{n}{2}) + n$
21. $T(n) = T(\frac{n}{3}) + \log n$
22. $T(n) = T(n-3) + n^3$

☐ Время работы некоторого алгоритма задано следующим рекуррентным соотношением. Построив *дерево рекурсивных вызовов*, докажите, что:

23. если $T(n) = 2T\left(\frac{n}{2}\right) + n$, то $T(n) = \mathcal{O}(n \log n)$
24. если $T(n) = 2T\left(\frac{n}{2}\right) + 1$, то $T(n) = \mathcal{O}(n)$

☐ Гармонический ряд:

25. Докажите, что $\sum\limits_{t=1}^{n} \frac{1}{t} = \Omega(\log n)$.

☐ Для каждой из приведенных ниже программ найдите и аргументируйте точную $\mathcal{O}$-асимптотику времени ее работы.

26.
```
for i = 0..n:
    for j = 0..i:
        for k = 0..j:
            print(i, j, k)
```

27.
```
def f(n):
    if n < 2:
        return 1
    return f(n - 1) + 5 * f(n - 1)
```

28.
```
def f(n):
    if n < 2:
        return 1
    return 3 * f(n / 2)
```

29.
```
for i = 0..n:
    j = i
    while j > 0:
        j = j / 2
```

Неделя 2. Элементарные структуры данных. Амортизационный анализ.

Конспект

▷ какие элементарные структуры данных бывают?

⋆ массив, связный список, стек, очередь

▷ массив (фиксированного размера n):

⋆ последовательно храним набор элементов в памяти

⋆ можем обратиться к i-му элементу за $\mathcal{O}(1)$

⋆ можем добавить/удалить элемент на позиции сразу после i-го элемента за $\mathcal{O}(n)$ (придется сдвинуть оставшиеся элементы массива влево)

▷ стек (на массиве):

⋆ девиз: последним вошел — первым вышел (`LIFO`, last-in first-out)

⋆ храним указатель `H` (`HEAD`) на верхний элемент стека

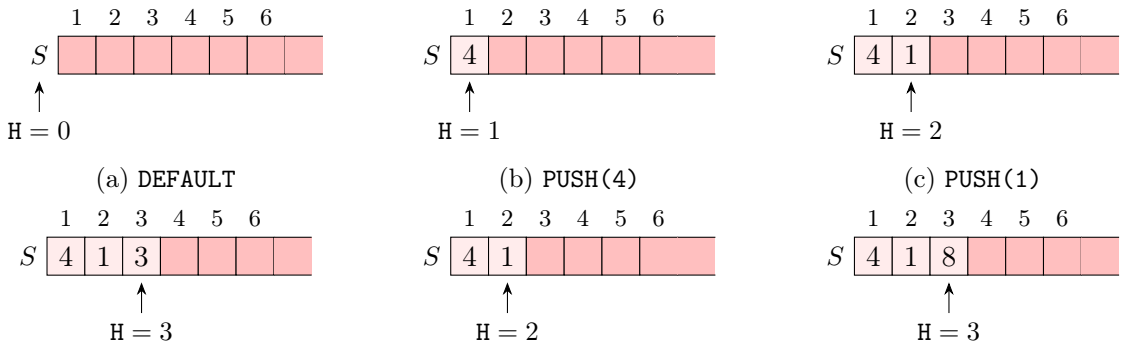


Рис. 1: Операции со стеком

▷ очередь (на массиве):

⋆ девиз: первым вошел — первым вышел (`FIFO`, first-in first-out)

⋆ храним указатели `H` и `T` (`HEAD` и `TAIL`) на начало и конец очереди соответственно

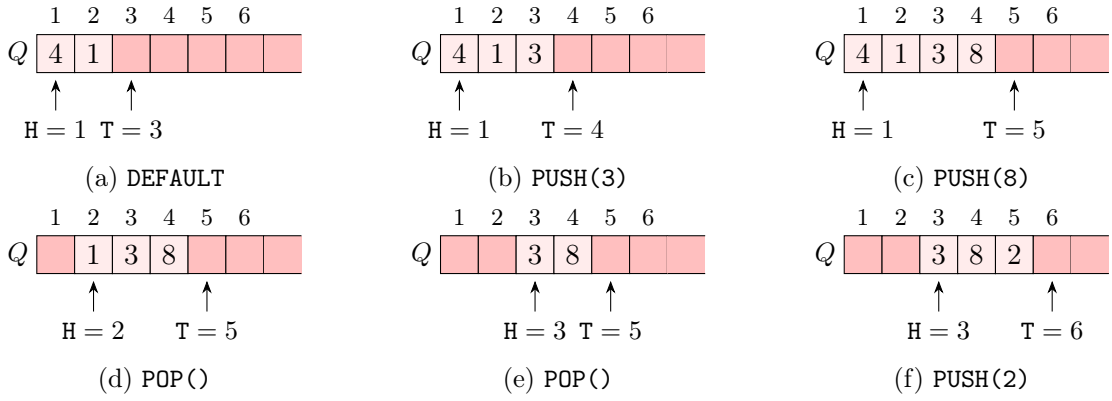
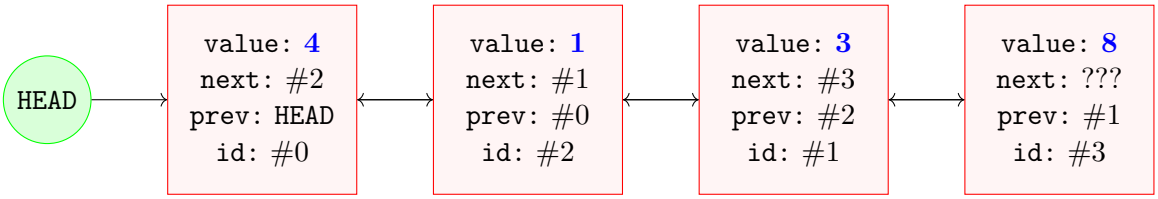


Рис. 2: Операции с очередью

▷ связный список:

- ⋆ объекты разбросаны по разным местам в памяти
- ⋆ для каждого объекта храним:
  - ⋆ значение (`value`)
  - ⋆ идентификатор следующего за ним объекта (`next`)
  - ⋆ идентификатор предыдущего перед ним объекта (`prev`)
  - ⋆ уникальный идентификатор объекта (`id`)
- ⋆ храним ссылку `HEAD` на первый фиктивный элемент связного списка
- ⋆ можем обратиться к i-му элементу за $\mathcal{O}(n)$, так как нужно будет пройти все элементы начиная с `HEAD` итеративно
- ⋆ можем добавить/удалить элемент на позиции сразу после i-го элемента за $\mathcal{O}(1)$ (при условии что нам дали ссылку на i-ый элемент), так как нужно будет всего лишь поменять несколько ссылок



Практика

30. Придумайте модификацию стека, позволяющую за $\mathcal{O}(1)$ времени отвечать на запрос «вернуть сумму всех элементов в стеке».

31. Придумайте модификацию стека, позволяющую за $\mathcal{O}(1)$ времени отвечать на запрос «вернуть минимум среди всех элементов в стеке».

32. Придумайте модификацию очереди, позволяющую за $\mathcal{O}(1)$ времени отвечать на запрос «вернуть сумму всех элементов в очереди».

33. Придумайте модификацию очереди, позволяющую за $\mathcal{O}(1)$ времени отвечать на запрос «вернуть минимум среди всех элементов в очереди».

34. Придумайте модификацию очереди (при помощи нескольких стеков), которая поддерживает добавление и удаление элементов из обоих концов, то есть поддерживает следующие операции: `push_back()`, `push_front()`, `pop_back()`, `pop_front()`.

Def. Скобочная последовательность называется *правильной*, если она может быть получена из некоторого арифметического выражения удалением всех не-скобочных символов.

Например: `()[()]`, `[[[]{}]]` и `(()[]{})` — правильные скобочные последовательности, в то время как: `([)]`, `([{}]` и `()]` — неправильные.

35. Дана скобочная последовательность длины n из одного типа скобок: `()`. Определите, является ли она правильной. Время $\mathcal{O}(n)$.

36. Дана скобочная последовательность длины n из трех типов скобок: `()` `[]` и `{}`. Определите, является ли она правильной. Время $\mathcal{O}(n)$.

37. Дано арифметическое выражение в [*постфиксной записи*] длины n. Найдите результат вычисления этого выражения за $\mathcal{O}(n)$.

38. Дано арифметическое выражение в инфиксной записи со скобками (привычное нам арифметическое выражение). Найдите результат вычисления этого выражения за $\mathcal{O}(n)$.

39. Дан массив чисел длины n. Найдите на нем отрезок с максимальной суммой за $\mathcal{O}(n)$ времени.

40. Дан массив из целых чисел. Для каждого элемента найдите ближайший элемент слева, меньший его. Время $\mathcal{O}(n)$.

41. Как развернуть односвязный список за время $\mathcal{O}(n)$ с $\mathcal{O}(1)$ дополнительной памяти?

42. Дан набор из n элементов, в каждом есть ссылка на какой-то другой. Проверьте, правда ли эти элементы образуют один большой кольцевой список (менять ссылки нельзя). Время $\mathcal{O}(n)$, память $\mathcal{O}(1)$.

43. Дан набор из n элементов, в каждом есть ссылка на какой-то другой. Пусть гарантируется, что в структуре есть ровно один цикл. Найдите его длину. Время $\mathcal{O}(n)$, память $\mathcal{O}(1)$.

44. Дан набор из n элементов, в каждом есть ссылка на какой-то другой. Проверьте, правда ли эти элементы образуют один большой линейный список (начальный элемент неизвестен, менять ссылки нельзя). Время $\mathcal{O}(n)$, память $\mathcal{O}(1)$.

45. Дан набор из n элементов, в каждом есть ссылка на следующий и предыдущий. Проверьте, правда ли эти элементы образуют несколько кольцевых списков (менять ссылки нельзя). Время $\mathcal{O}(n)$, память $\mathcal{O}(1)$.

46. Дан набор из n элементов, в каждом есть ссылка на следующий и предыдущий. Проверьте, правда ли эти элементы образуют несколько связных списков и, если да, сконкатенируйте эти списки в один большой (в любом порядке). Время $\mathcal{O}(n)$, память $O(1)$.

47. Слить два отсортированных односвязных списка в один за время $\mathcal{O}(n)$ с $\mathcal{O}(1)$ дополнительной памяти.

48. Отсортировать связный список за время $\mathcal{O}(n \log n)$ с $\mathcal{O}(1)$ дополнительной памяти.