Занятие 14.

Линейные задачи.

План занятия

- повторение
- линейные задачи
- метод двух указателей

Задачи о количестве и наилучшем способе

- Перебор $\mathcal{O}(2^n)$
- Динамическое программирование $\mathcal{O}(n)$
- Формула $\mathcal{O}(1)$

Принцип ДП --- использование предпосчета для вычисления новых значений

Линейные задачи

$f(x) = kx + b$ --- линейная функция

Аналогично, линейные задачи --- это задачи, требующие $\mathcal{O}(n)$ вычислений.

$$5n + 3 = \mathcal{O}(n)$$

$3n^2 + 2n = \mathcal{O}(n^2)$ --- квадратичная сложность

Пройденные линейные алгоритмы:

- одномерное динамическое программирование
- нахождение max/min
- сумма/произведение
- линейный поиск

Задачи

1) Подается $N - 1$ число на вход из $\{1, \ldots, N\}$. Все числа различны. Какого числа не хватает?

Решение

- Посчитать сумму в массиве S. $\frac{N(N-1)}{2} - S - x = 0$

Время - $\mathcal{O}(n)$, память - $\mathcal{O}(1)$

- Заводим массив **A** длины N и заполняем 0-ками. Проходим по массиву данных и ставим 1 в соответствующий элемент массива **A**

Время - $\mathcal{O}(n)$, память - $\mathcal{O}(n)$

In [3]:

```
a = list(map(int, input().split()))
boollist = [0 for i in range(len(a) + 1)]
for i in a:
    boollist[i - 1] = 1
print(boollist.index(0)+ 1)
```

```
1 3 4 5
2
```

Задачи

2) Подается массив длины N и число $k < N$. Требуется посчитать все суммы вида $a[i:i+k]$

Решение

- "Наивное" -- вложенные for.

Сложность $\mathcal{O}(nk)$ нелинейное, если $k = n/2$

In [5]:

```
# x = list(map(int, input().split()))
x = [2, 1, 0, -2, 5, 4]
# k = int(input())
k = 3
result=  [0]*(len(x) - k + 1)
for i in range(len(x) - k + 1):
    for j in range(k):
        result[i] += x[i+j]
print(result)
```

```
[3, -1, 3, 7]
```

- $a[m+1] = a[m] - x[m] - x[m+k]$, где **a** -- массив частичных сумм

In [6]:

```
n = len(x)
result = [0]*(n - k + 1)
for i in range(k):
    result[0] += x[i]
for i in range(1, n-k+1):
    result[i] = result[i-1] - x[i-1] + x[i + k -1]
print(result)
```

```
[3, -1, 3, 7]
```

Онлайн-задачи

Во время выполнения задачи поступают запросы на которые необходимо отвечать. Пример: 'i, j' -> sum(a[i:j+1])

Задачи

3) Подается массив длины N и Q запросов вида (i, j) Требуется вернуть на каждый запрос сумму $a[i : j + 1]$

Решение

- Для каждого запроса вызываем функцию sum(a[i, j+1])

Сложность $\mathcal{O}(N \cdot Q)$

- Научимся тратить на все запросы время $\mathcal{O}(Q) \Rightarrow$ отвечать на запрос за $\mathcal{O}(1)$. Следовательно нужен предподсчет данных!
- Для каждой пары (i, j) посчитаем сумму на этом отрезке и запишем в табличку
- Время на предподсчет $\mathcal{O}(N^2)$, время на запросы $\mathcal{O}(Q)$

Сложность $\mathcal{O}(N^2 + Q)$

- Вычислим массив **b**, такой что $b[k] = a[0] + \ldots + a[k]$ за $\mathcal{O}(N)$
- $a[i] + \ldots + a[j] = b[j] - b[i - 1]$ за $\mathcal{O}(1)$

Сложность $\mathcal{O}(N + Q)$

In [7]:

```
x = [2, 1, 0, -2, 5, 10, -14, 7]
n = len(x)
b = [0]*(n+1)
for i in range(1,n+1):
    b[i] = b[i-1] + x[i-1]
```

```
    b[i] = b[i-1] + x[i-1]
print(b)
```

```
[0, 2, 3, 3, 1, 6, 16, 2, 9]
```

In [8]:

```
for k in range(2):
    i, j = map(int, input().split())
    print(b[j+1] - b[i])
```

```
0 3
1
2 7
6
```

Метод двух указателей

Задачи

4) Подаётся два **упорядоченных** списка из целых чисел A, B. Требуется найти

$$(x, y) : x \in A, y \in B \text{ и } |x - y| = min$$

Решение

- перебрать все возможные пары чисел

Сложность $\mathcal{O}(a \cdot b)$

метод "двух указателей"

- инициализируем указатели на начале обоих массивов
- сравниваем значения, обновляем минимум (если необходимо)
- двигаем тот указатель, значение которого меньше
- продолжаем шаги до тех пор, пока одному из указателей некуда будет двигаться

Сложность $\mathcal{O}(a + b)$

Задачи

5) Подаётся два **упорядоченных** списка из целых чисел A, B. Требуется получить отсортированный список из значений массивов A и B.

Решение

- sorted(a + b)

Сложность $\mathcal{O}(n \cdot log(n))$

метод "двух указателей"

- инициализируем указатели на начале обоих массивов
- сравниваем значения, записываем наименьшее в итоговый массив
- двигаем тот указатель, значение которого меньше
- продолжаем шаги до тех пор, пока одному из указателей некуда будет двигаться Сложность $\mathcal{O}(a + b)$

In [9]:

```
def mergesort(a, b):
    i = j = 0
    result = [0]*(len(a) + len(b))
    while i < len(a) and j < len(b):
        if a[i] < b[j]:
            result[i+j] = a[i]
            i += 1
        else:
            result[i+j] = b[j]
            j += 1
    if i == len(a):
        result[i+j:] = b[j:]
    else:
        result[j+i:] = a[i:]
    return result

a = list(map(int, input().split()))
b = list(map(int, input().split()))
print(*mergesort(a, b))
```

```
-2 4 6
0 1 7 8
-2 0 1 4 6 7 8
```