Вводные слайды

Иван Сергеевич Казменко

Санкт-Петербургский Государственный Университет

Вторник, 16 февраля 2021 года

Содержание

Contents

Ссылка

Ссылка на материалы занятий:
`http://acm.math.spbu.ru/~gassa/dp-2021`

Предупреждение: всё далее — предварительная информация, она может поменяться!

Структура курса

Что мы будем делать?

- Слушать теорию
- Решать задачи за компьютером
- Рассказывать решения на разборе
- Делать свою задачу

Структура занятия

Примерная структура занятия:

- Разбираем друг другу задачи с предыдущих занятий
- Обсуждаем новую тему, теорию и реализацию
- Решаем задачи на новую тему между занятиями

Как получить зачёт

Как получить зачёт?

- Набрать баллы за решения (40% или 50% или 60%)
  - Обычные задачи: дедлайн в 23:59 перед следующим занятием
  - Задачи со звёздочкой: сложнее, но без дедлайна
  - После дедлайна: решение даёт половину баллов
- Сделать свою задачу

Проверка решений

Проверка решений

- Автоматическая проверяющая система: `http://acm.math.spbu.ru/tsweb`
- Логины и пароли будут разосланы на почту
- Submit: послать решение на проверку
- Monitor: кто что решил
- Clarifications: задать вопрос по формальному условию задачи
- Submissions: предыдущие попытки и их результаты (`NO` $\rightarrow$ `OK`, `WA`, `TL`, ...)
- Statements: условия задач
- List of all contests: предыдущие занятия

Чем пользоваться

Чем можно пользоваться?

- Документация по языку, конспекты, Wikipedia, Google, OEIS, ...
- Задавать вопросы преподавателю
- Вместе придумывать решения

Что нужно делать самим

- Писать код решения
- Делать свою задачу

Contents

Числа Фибоначчи

Определение: $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ при $n \geqslant 2$.
Задача: дано n, вычислите F_n.

Числа Фибоначчи

C++, рекурсия:

```cpp
#include <iostream>
using namespace std;




int fib (int n) {
    if (n < 2) return n;
    return fib (n - 1) + fib (n - 2);
}



int main () {
    int n;
    while (cin >> n)
        cout << fib (n) << endl;
    return 0;
}
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

C++, рекурсия:

```
1   #include <iostream>
2   using namespace std;
3
4
5
6
7   int fib (int n) {
8       if (n < 2) return n;
9       return fib (n - 1) + fib (n - 2);
10  }
11
12
13
14  int main () {
15      int n;
16      while (cin >> n)
17          cout << fib (n) << endl;
18      return 0;
19  }
```

Пример $n = 5$:

- $5 \to 4, 3$
- . $4 \to 3, 2$
- . . $3 \to 2, 1$
- . . . $2 \to 1, 0$
- 1
- 0
- . . . 1
- . . $2 \to 1, 0$
- . . . 1
- . . . 0
- . $3 \to 2, 1$
- . . $2 \to 1, 0$
- . . . 1
- . . . 0
- . . 1

Числа Фибоначчи

C++, рекурсия с запоминанием (мемоизация):

```cpp
#include <iostream>
using namespace std;

int f [99];

int fib (int n) {
    if (f[n] == 0) {
        if (n < 2) f[n] = n;
        else f[n] = fib (n - 1) + fib (n - 2);
    }
    return f[n];
}

int main () {
    int n;
    while (cin >> n)
        cout << fib (n) << endl;
    return 0;
}
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

C++, рекурсия с запоминанием (мемоизация):

```
1  #include <iostream>
2  using namespace std;
3
4  int f [99];
5
6  int fib (int n) {
7      if (f[n] == 0) {
8          if (n < 2) f[n] = n;
9          else f[n] = fib (n - 1) + fib (n - 2);
10     }
11     return f[n];
12 }
13
14 int main () {
15     int n;
16     while (cin >> n)
17         cout << fib (n) << endl;
18     return 0;
19 }
```

Пример $n = 5$:

- $5 \to 4, 3$
- . $4 \to 3, 2$
- . . $3 \to 2, 1$
- . . . $2 \to 1, 0$
- 1
- 0
- . . . 1 запомнено
- . . 2 запомнено
- . 3 запомнено

Числа Фибоначчи

C++, итеративное решение:

```cpp
#include <iostream>
using namespace std;

int main () {
    int f [99];
    f[0] = 0;
    f[1] = 1;
    for (int i = 2; i < 99 /*???*/; i++)
        f[i] = f[i - 1] + f[i - 2];
    int n;
    while (cin >> n)
        cout << f[n] << endl;
    return 0;
}
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

D, рекурсия:

```d
import std;


int fib (int n) {
    if (n < 2) return n;
    return fib (n - 1) + fib (n - 2);
}

void main () {
    foreach (line; stdin.byLine) {
        auto n = line.to !(int);
        writeln (fib (n));
    }
}
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

D, рекурсия с запоминанием (мемоизация):

```d
import std;

alias fib = memoize !(fibImpl);
int fibImpl (int n) {
    if (n < 2) return n;
    return fib (n - 1) + fib (n - 2);
}

void main () {
    foreach (line; stdin.byLine) {
        auto n = line.to !(int);
        writeln (fib (n));
    }
}
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

Python, рекурсия:

```
1  import sys
2
3
4  def fib (n):
5      if n < 2:
6          return n
7      return fib (n - 1) + fib (n - 2)
8
9  for line in sys.stdin:
10     n = int (line)
11     print (fib (n))
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

Числа Фибоначчи

Python, рекурсия с запоминанием (мемоизация):

```
1  import functools, sys
2
3  @functools.lru_cache (maxsize = None)
4  def fib (n):
5      if n < 2:
6          return n
7      return fib (n - 1) + fib (n - 2)
8
9  for line in sys.stdin:
10     n = int (line)
11     print (fib (n))
```

Ввод:

```
5
30
40
45
```

Вывод:

```
5
832040
102334155
1134903170
```

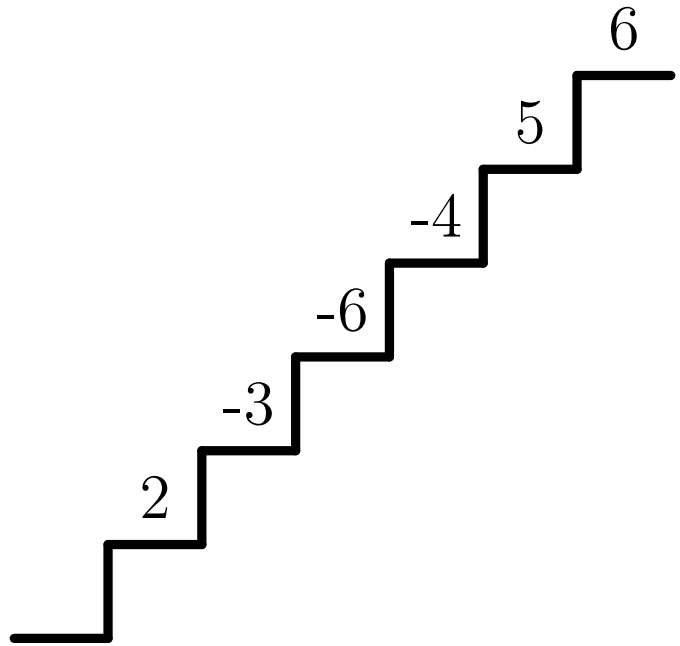
Лестница

- Есть лестница из n ступенек.
- Мы начинаем у подножия лестницы, на нулевой ступеньке.
- На ступеньках написаны числа $a_1, a_2, \ldots, a_n$.
- Мы должны оказаться на n-й ступеньке.
- За один шаг мы поднимаемся либо на одну, либо на две ступеньки.
- Наши баллы — сумма чисел, на которые мы шагнули.
- Как подняться по лестнице, чтобы максимизировать баллы?

Лестница

- Есть лестница из n ступенек.
- Мы начинаем у подножия лестницы, на нулевой ступеньке.
- На ступеньках написаны числа $a_1, a_2, \ldots, a_n$.
- Мы должны оказаться на n-й ступеньке.
- За один шаг мы поднимаемся либо на одну, либо на две ступеньки.
- Наши баллы — сумма чисел, на которые мы шагнули.
- Как подняться по лестнице, чтобы максимизировать баллы?

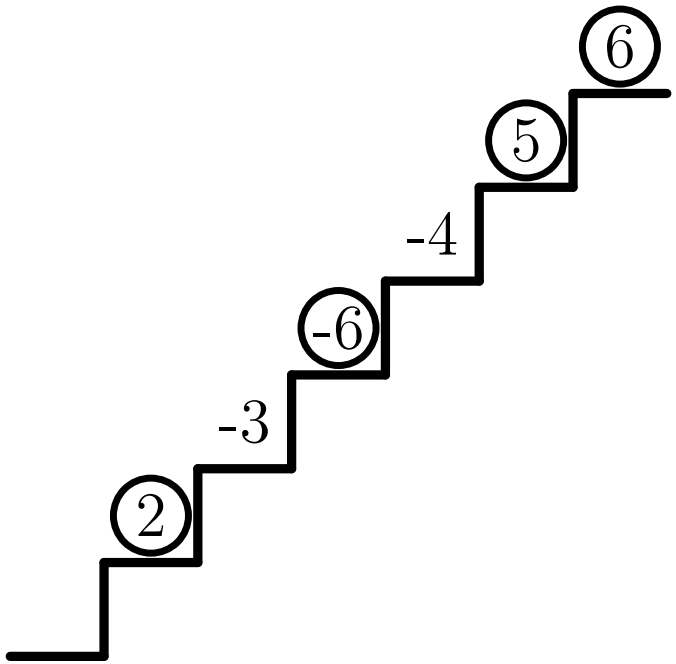
Пример 1:



Лестница

- Есть лестница из n ступенек.
- Мы начинаем у подножия лестницы, на нулевой ступеньке.
- На ступеньках написаны числа $a_1, a_2, \ldots, a_n$.
- Мы должны оказаться на n-й ступеньке.
- За один шаг мы поднимаемся либо на одну, либо на две ступеньки.
- Наши баллы — сумма чисел, на которые мы шагнули.
- Как подняться по лестнице, чтобы максимизировать баллы?

Пример 1, решение:

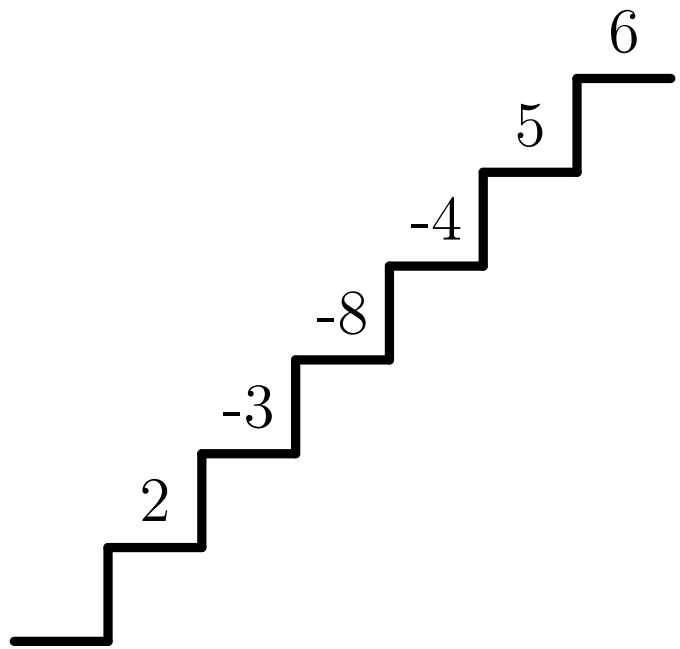


Ответ:
$2 + (-6) + 5 + 6 = 7$.

Лестница

- Есть лестница из n ступенек.
- Мы начинаем у подножия лестницы, на нулевой ступеньке.
- На ступеньках написаны числа $a_1, a_2, \ldots, a_n$.
- Мы должны оказаться на n-й ступеньке.
- За один шаг мы поднимаемся либо на одну, либо на две ступеньки.
- Наши баллы — сумма чисел, на которые мы шагнули.
- Как подняться по лестнице, чтобы максимизировать баллы?

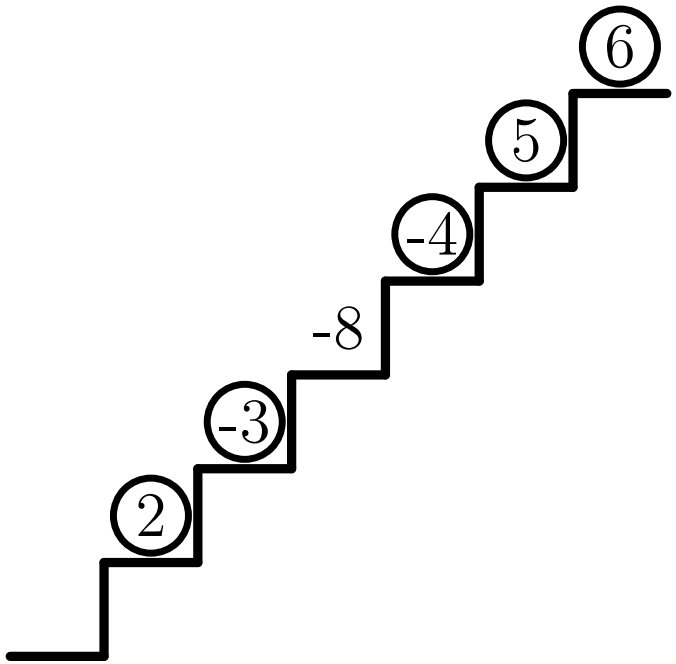
Пример 2:



Лестница

- Есть лестница из n ступенек.
- Мы начинаем у подножия лестницы, на нулевой ступеньке.
- На ступеньках написаны числа $a_1, a_2, \ldots, a_n$.
- Мы должны оказаться на n-й ступеньке.
- За один шаг мы поднимаемся либо на одну, либо на две ступеньки.
- Наши баллы — сумма чисел, на которые мы шагнули.
- Как подняться по лестнице, чтобы максимизировать баллы?

Пример 2, решение:



Ответ:
$2 + (-3) + (-4) + 5 + 6 = 6.$

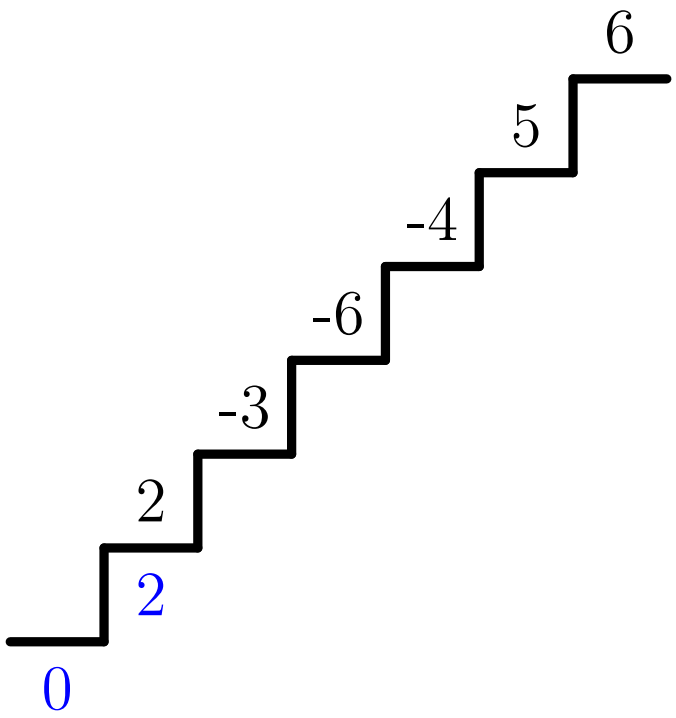
Лестница, решение

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Как подняться на k-ю ступеньку, чтобы максимизировать баллы?

Лестница, решение

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Как подняться на k-ю ступеньку, чтобы максимизировать баллы?
- Назовём это значение $f(k)$.
- База: $f(0) = 0$.
- База: $f(1) = a_1$.

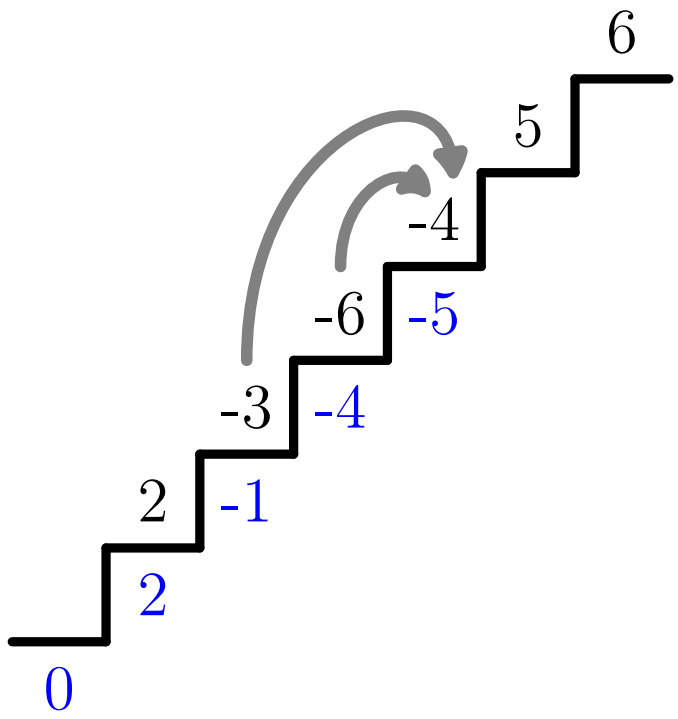
Пример 1:



Лестница, решение

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Как подняться на k-ю ступеньку, чтобы максимизировать баллы?
- Назовём это значение $f(k)$.
- База: $f(0) = 0$.
- База: $f(1) = a_1$.
- Пусть мы уже нашли $f(k')$ для всех $k' < k$.
- Как подняться на k-ю ступеньку?
- Мы пришли либо с $k-1$-й ступеньки, либо с $k-2$-й ступеньки.

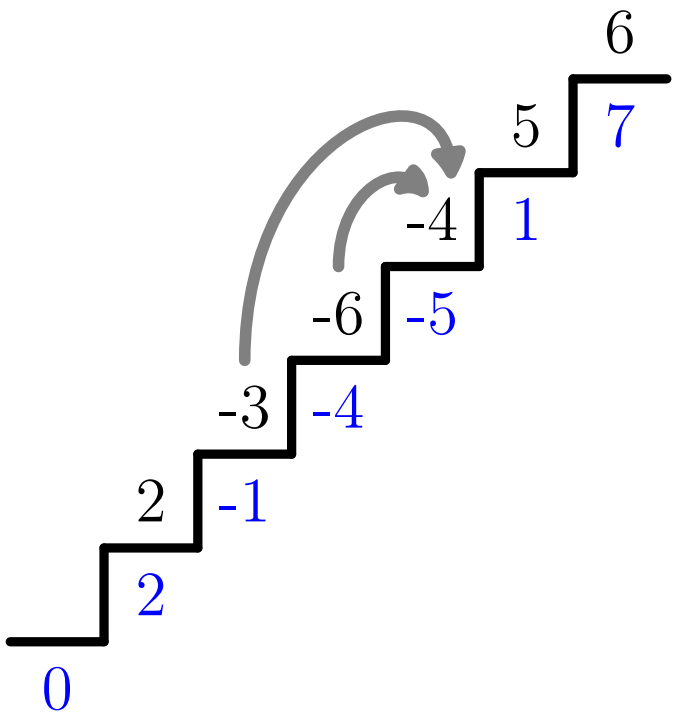
Пример 1, $k = 4$:



Лестница, решение

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Как подняться на k-ю ступеньку, чтобы максимизировать баллы?
- Назовём это значение $f(k)$.
- База: $f(0) = 0$.
- База: $f(1) = a_1$.
- Пусть мы уже нашли $f(k')$ для всех $k' < k$.
- Как подняться на k-ю ступеньку?
- Мы пришли либо с $k-1$-й ступеньки, либо с $k-2$-й ступеньки.

Пример 1, $k = 4$:



$f(k) = a_k + \max(f(k-1), f(k-2))$

Лестница, код

Рекурсия:

```
1   #include <algorithm>
2   #include <iostream>
3   using namespace std;
4
5   int a [10001];
6
7
8   int f (int n) {
9       if (n == 0) return 0;
10      if (n == 1) return a[1];
11
12      return a[n] + max (f (n - 1), f (n - 2));
13  }
14
15
16  int main () {
17      int n;  cin >> n;
18      for (int i = 1; i <= n; i++)  cin >> a[i];
19
20      cout << f (n) << endl;
21      return 0;
22  }
```

Лестница, код

Рекурсия с запоминанием (мемоизация):

```
1  #include <algorithm>
2  #include <iostream>
3  using namespace std;
4
5  int a [10001];
6  int mf [10001];
7
8  int f (int n) {
9      if (n == 0) return 0;
10     if (n == 1) return a[1];
11     if (mf[n] == INT_MAX)
12         mf[n] = a[n] + max (f (n - 1), f (n - 2));
13     return mf[n];
14 }
15
16 int main () {
17     int n;  cin >> n;
18     for (int i = 1; i <= n; i++)  cin >> a[i];
19     for (int i = 0; i <= n; i++)  mf[i] = INT_MAX;
20     cout << f (n) << endl;
21     return 0;
22 }
```

Лестница, код

Итеративное решение:

```
1  #include <algorithm>
2  #include <iostream>
3  using namespace std;
4
5  int a [10001];
6  int f [10001];
7
8  int main () {
9      int n;  cin >> n;
10     for (int i = 1; i <= n; i++)  cin >> a[i];
11     f[0] = 0;
12     f[1] = a[1];
13     for (int i = 2; i <= n; i++)
14         f[i] = a[i] + max (f[i - 1], f[i - 2]);
15     cout << f[n] << endl;
16     return 0;
17 }
```

Динамика и индукция

Динамическое программирование:

- Найдём $f(n)$.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже найдены для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$.
- Ответ: $f(n)$.

Математическая индукция:

- Докажем, что $f(n)$ посчитано верно.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже посчитаны верно для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$ тоже посчитано верно.
- Значит, $f(n)$ посчитано верно.

Динамика и индукция

Динамическое программирование:

- Найдём $f(n)$.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже найдены для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$.
- Ответ: $f(n)$.

Математическая индукция:

- Докажем, что $f(n)$ посчитано верно.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже посчитаны верно для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$ тоже посчитано верно.
- Значит, $f(n)$ посчитано верно.

Динамика и индукция

Динамическое программирование:

- Найдём $f(n)$.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже найдены для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$.
- Ответ: $f(n)$.

Математическая индукция:

- Докажем, что $f(n)$ посчитано верно.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже посчитаны верно для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$ тоже посчитано верно.
- Значит, $f(n)$ посчитано верно.

Динамика и индукция

Динамическое программирование:

- Найдём $f(n)$.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже найдены для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$.
- Ответ: $f(n)$.

Математическая индукция:

- Докажем, что $f(n)$ посчитано верно.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже посчитаны верно для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$ тоже посчитано верно.
- Значит, $f(n)$ посчитано верно.

Динамика и индукция

Динамическое программирование:

- Найдём $f(n)$.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже найдены для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$.
- Ответ: $f(n)$.

Математическая индукция:

- Докажем, что $f(n)$ посчитано верно.
- База: $f(0) = 0$, $f(1) = a_1$.
- Предположение: пусть $f(k')$ уже посчитаны верно для всех $k' < k$.
- Переход: тогда $f(k) = a_k + \max(f(k-1) + f(k-2))$ тоже посчитано верно.
- Значит, $f(n)$ посчитано верно.

Анти-лестница

- Есть последовательность из n чисел: $a_1, a_2, \ldots, a_n$.
- Мы хотим выбрать подпоследовательность.
- При этом нельзя выбирать два соседних элемента.
- Как максимизировать сумму выбранных чисел?

Анти-лестница

- Есть последовательность из n чисел: $a_1, a_2, \ldots, a_n$.
- Мы хотим выбрать подпоследовательность.
- При этом нельзя выбирать два соседних элемента.
- Как максимизировать сумму выбранных чисел?

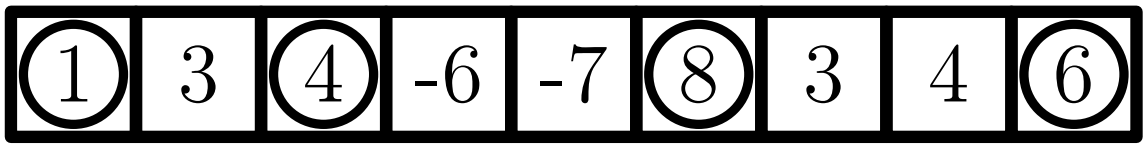
Пример 1:

1	3	4	-6	-7	8	3	4	6

Анти-лестница

- Есть последовательность из n чисел: $a_1, a_2, \ldots, a_n$.
- Мы хотим выбрать подпоследовательность.
- При этом нельзя выбирать два соседних элемента.
- Как максимизировать сумму выбранных чисел?

Пример 1, решение:

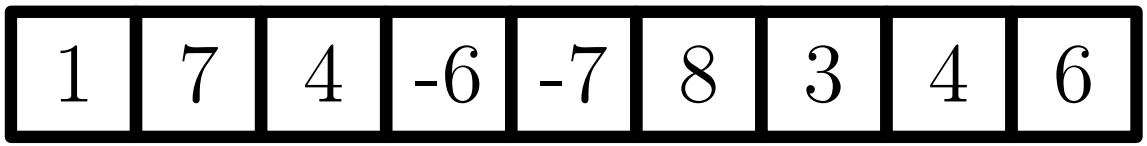


Ответ: $1 + 4 + 8 + 6 = 19$.

Анти-лестница

- Есть последовательность из n чисел: $a_1, a_2, \ldots, a_n$.
- Мы хотим выбрать подпоследовательность.
- При этом нельзя выбирать два соседних элемента.
- Как максимизировать сумму выбранных чисел?

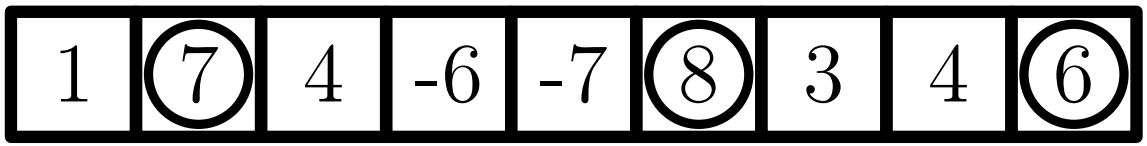
Пример 2:



Анти-лестница

- Есть последовательность из n чисел: $a_1, a_2, \ldots, a_n$.
- Мы хотим выбрать подпоследовательность.
- При этом нельзя выбирать два соседних элемента.
- Как максимизировать сумму выбранных чисел?

Пример 2, решение:

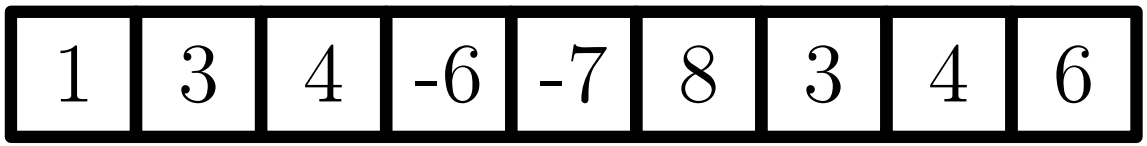


Ответ: $7 + 8 + 6 = 21$.

Анти-лестница, решение 1

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Пусть мы взяли k-е число последним.
- Как при этом максимизировать сумму?

Пример 1:



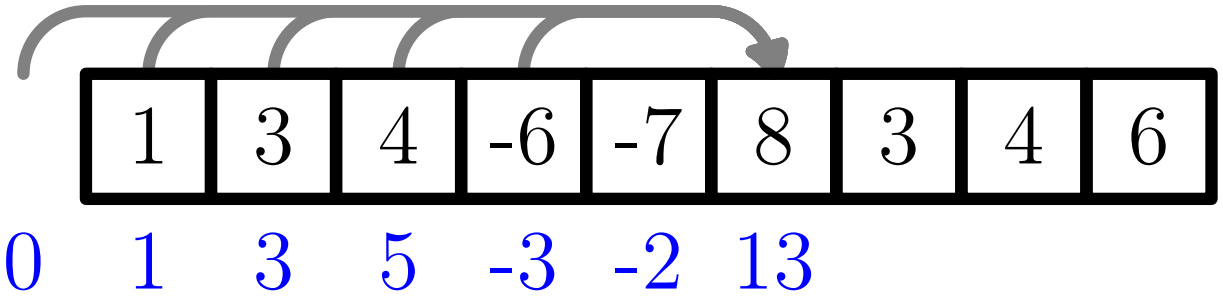
0 1

- Назовём это значение $f(k)$.
- База: пусть $f(0) = 0$.
- База: $f(1) = a_1$.

Анти-лестница, решение 1

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Пусть мы взяли k-е число последним.
- Как при этом максимизировать сумму?

Пример 1:

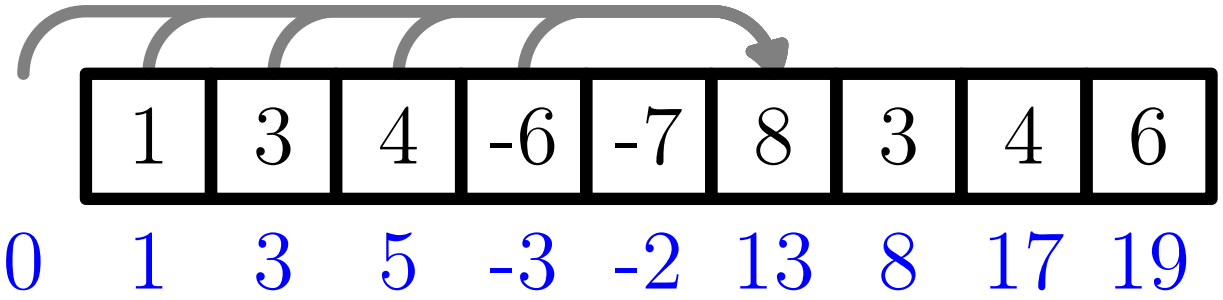


- Пусть мы уже нашли $f(k')$ для всех $k' < k$.
- Как посчитать $f(k)$?
- Предыдущим мы могли взять $k-2$-е число, или $k-3$-е, ..., или первое, или нулевое (то есть ничего не взять).

Анти-лестница, решение 1

- Давайте решать более общую задачу.
- Для каждого k от 0 до n:
- Пусть мы взяли k-е число последним.
- Как при этом максимизировать сумму?

Пример 1:

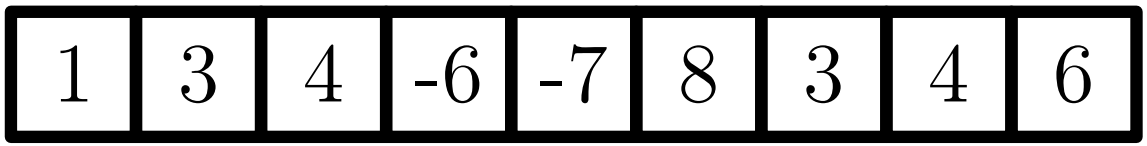


- Получается $f(k) = a_k + \max(f(k-2), f(k-3), \ldots, f(1), f(0))$.
- Ответ — это $\max(f(0), f(1), \ldots, f(n))$.
- Решение работает за $O(n^2)$.

Анти-лестница, решение 2

- Давайте решать другую более общую задачу.
- Для каждого k от 0 до n:
- Рассмотрим префикс последовательности: $a_1, a_2, \ldots, a_k$.
- Как при этом максимизировать сумму?

Пример 1:



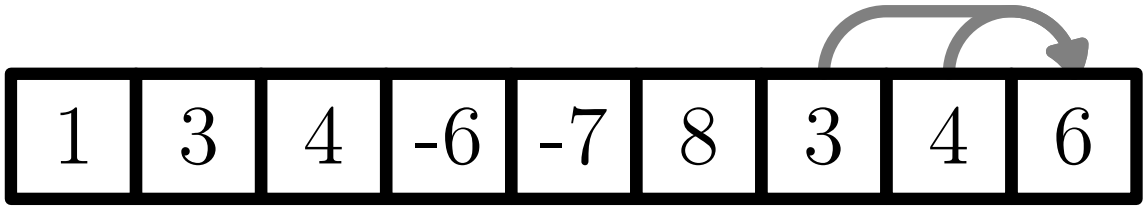
0 1

- Назовём это значение $g(k)$.
- База: $g(0) = 0$.
- База: $g(1) = \max(0, a_1)$.

Анти-лестница, решение 2

- Давайте решать другую более общую задачу.
- Для каждого k от 0 до n:
- Рассмотрим префикс последовательности: $a_1, a_2, \ldots, a_k$.
- Как при этом максимизировать сумму?

Пример 1:



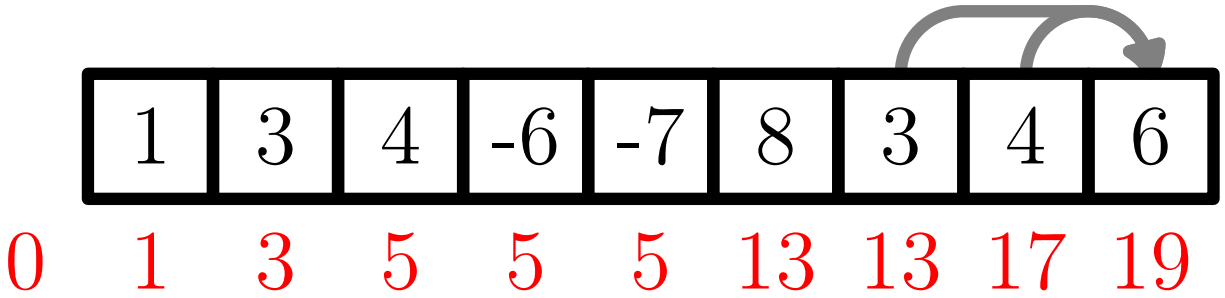
Как найти $g(k)$, если мы уже знаем $g(k')$ для всех $k' < k$?

- Мы можем не брать a_k, тогда возьмём ответ из $g(k-1)$.
- Мы можем взять a_k, тогда добавим ответ из $g(k-2)$.

Анти-лестница, решение 2

- Давайте решать другую более общую задачу.
- Для каждого k от 0 до n:
- Рассмотрим префикс последовательности: $a_1, a_2, \ldots, a_k$.
- Как при этом максимизировать сумму?

Пример 1:



- Получается $g(k) = \max(g(k-1), a_k + g(k-2))$.
- Ответ — это $g(n)$.
- Решение работает за $O(n)$.

Кузнечик — может или нет

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- Пример: $a = 2$, $b = 3$, $c = 5$.
- Может ли кузнечик добраться до клетки n?

Кузнечик — может или нет

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- Пример: $a = 2$, $b = 3$, $c = 5$.
- Может ли кузнечик добраться до клетки n?

Обобщим задачу: для каждой клетки k выясним, может ли кузнечик до неё добраться.

- База: $f(1) = \texttt{true}$.
- Пересчёт: $f(k) = f(k - a) \vee f(k - b) \vee f(k - c)$.

Кузнечик — может или нет

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- Пример: $a = 2$, $b = 3$, $c = 5$.
- Может ли кузнечик добраться до клетки n?

Псевдокод:

```
f[1] := true
for k := 2, ..., n:
  f[k] := false
  if k >= a and f[k - a]:  f[k] := true
  if k >= b and f[k - b]:  f[k] := true
  if k >= c and f[k - c]:  f[k] := true
```

Кузнечик — как можно быстрее

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Кузнечик — как можно быстрее

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Обобщим задачу: для каждой клетки k выясним, за какое минимальное число прыжков кузнечик может до неё добраться.

- База: $f(1) = 0$.
- Пересчёт: $f(k) = 1 + \min(f(k - a), f(k - b), f(k - c))$.

Кузнечик — как можно быстрее

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Псевдокод:

```
f[1] := 0
for k := 2, ..., n:
  f[k] := infinity
  if k >= a:  f[k] := min (f[k], 1 + f[k - a])
  if k >= b:  f[k] := min (f[k], 1 + f[k - b])
  if k >= c:  f[k] := min (f[k], 1 + f[k - c])
```

Кузнечик — ямы

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Кузнечик — ямы

- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Обобщим задачу: для каждой клетки k выясним, за какое минимальное число прыжков кузнечик может до неё добраться.

- База: $f(1) = 0$.
- Пересчёт: $f(k) = 1 + \min(f(k-a), f(k-b), f(k-c))$.
- Если $h_k =$ `true` (hole, яма в клетке k), то пусть $f(k) = \infty$.

Кузнечик — ямы

- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?

Псевдокод:

```
f[1] := 0
for k := 2, ..., n:
  f[k] := infinity
  if not h[k]:
    if k >= a:  f[k] := min (f[k], 1 + f[k - a])
    if k >= b:  f[k] := min (f[k], 1 + f[k - b])
    if k >= c:  f[k] := min (f[k], 1 + f[k - c])
```

Кузнечик — сок

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- В некоторых клетках — ямы, туда попадать нельзя.
- В каждой клетке есть неотрицательное количество сока.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Из всех таких маршрутов выберите тот, на котором кузнечик соберёт как можно больше сока.

Кузнечик — сок

- В некоторых клетках — ямы, туда попадать нельзя.
- В каждой клетке есть неотрицательное количество сока.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Из всех таких маршрутов выберите тот, на котором кузнечик соберёт как можно больше сока.

Обобщим задачу. Для каждой клетки k найдём пару: минимальное количество прыжков до неё, максимальное количество сока при таком количестве прыжков.

- База: $f(1) = \{0, j_1\}$; пусть j_k (juice) — количество сока в клетке k.
- Пересчёт: $f(k) = \{1, j_k\} + \text{best}(f(k-a), f(k-b), f(k-c))$.
- Если $h_k =$ `true` (hole, яма в клетке k), то пусть $f(k) = \{\infty, 0\}$.

Кузнечик — сок

- В некоторых клетках — ямы, туда попадать нельзя.
- В каждой клетке есть неотрицательное количество сока.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Из всех таких маршрутов выберите тот, на котором кузнечик соберёт как можно больше сока.

Псевдокод:

```
f[1] := {0, -j[1]}
for k := 2, ..., n:
  f[k] := {infinity, 0}
  if not h[k]:
    if k >= a:  f[k] := min (f[k], {1, -j[k]} + f[k - a])
    if k >= b:  f[k] := min (f[k], {1, -j[k]} + f[k - b])
    if k >= c:  f[k] := min (f[k], {1, -j[k]} + f[k - c])
```

Восстановление ответа

- Есть лента из клеток, по которой прыгает кузнечик.
- Он начинает в клетке 1.
- Из любой клетки x он может прыгнуть в клетки $x + a$, $x + b$ и $x + c$.
- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Как именно нужно прыгать кузнечику для этого?

Восстановление ответа

- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Как именно нужно прыгать кузнечику для этого?

Обобщим задачу: для каждой клетки k выясним, за какое минимальное число прыжков кузнечик может до неё добраться.

- База: $f(1) = 0$.
- Пересчёт: $f(k) = 1 + \min(f(k-a), f(k-b), f(k-c))$.

Восстановление ответа

- В некоторых клетках — ямы, туда попадать нельзя.
- За какое минимальное число прыжков кузнечик может добраться до клетки n?
- Как именно нужно прыгать кузнечику для этого?

Псевдокод:

```
f[1] := 0
for k := 2, ..., n:
  f[k] := infinity
  if not h[k]:
    if k >= a:  f[k] := min (f[k], 1 + f[k - a])
    if k >= b:  f[k] := min (f[k], 1 + f[k - b])
    if k >= c:  f[k] := min (f[k], 1 + f[k - c])
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

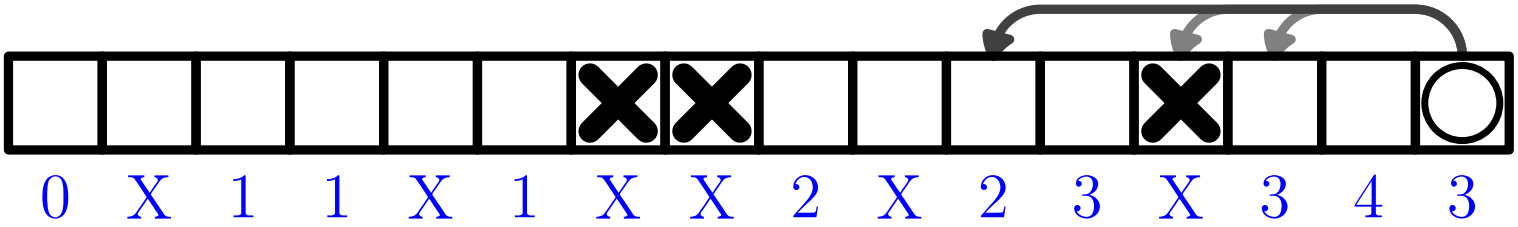
0 X 1 1 X 1 X X 2 X 2 3 X 3 4 3

Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

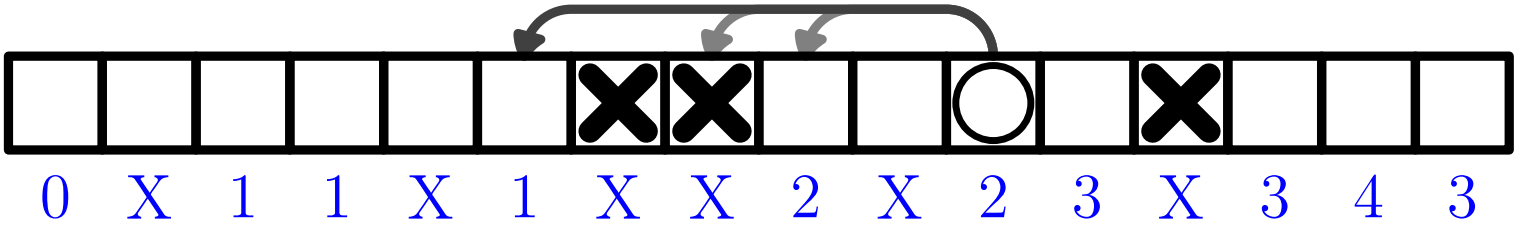


Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

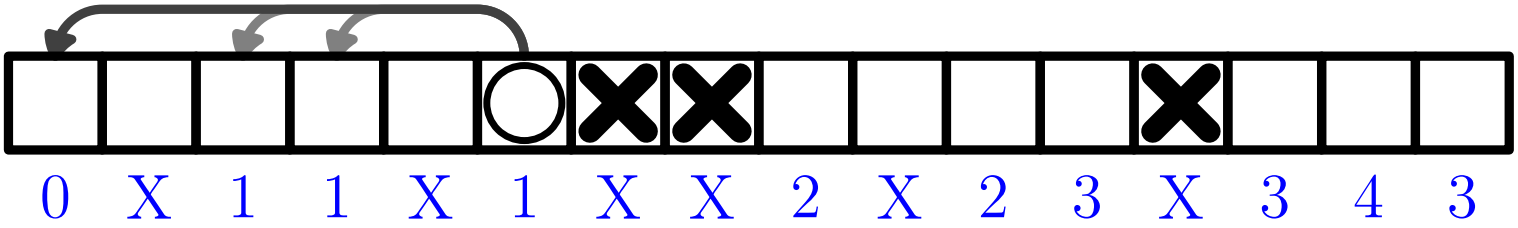


Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.



Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — обратные рёбра

Будем восстанавливать путь с конца: стоя в клетке x, смотреть, из какой клетки могло получиться значение $f(x)$.

0 X 1 1 X 1 X X 2 X 2 3 X 3 4 3

Псевдокод:

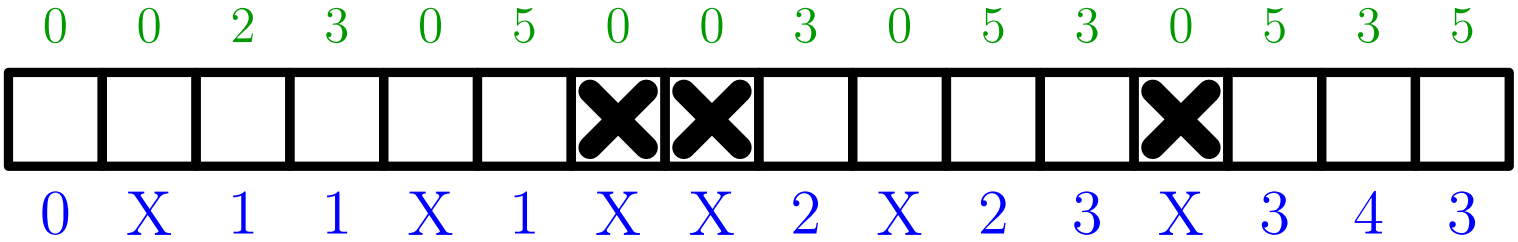
```
x := n
steps.add (x)
while x > 1:
  if      x >= a and f[x] = 1 + f[x - a]:  x -= a
  else if x >= b and f[x] = 1 + f[x - b]:  x -= b
  else if x >= c and f[x] = 1 + f[x - c]:  x -= c
  steps.add (x)
reverse (steps)
```

Восстановление ответа — хранение предков

Для каждой клетки k не только выясним, за какое минимальное число прыжков кузнечик может до неё добраться, но и запомним, каким прыжком он туда попал.

Восстановление ответа — хранение предков

Для каждой клетки k не только выясним, за какое минимальное число прыжков кузнечик может до неё добраться, но и запомним, каким прыжком он туда попал.

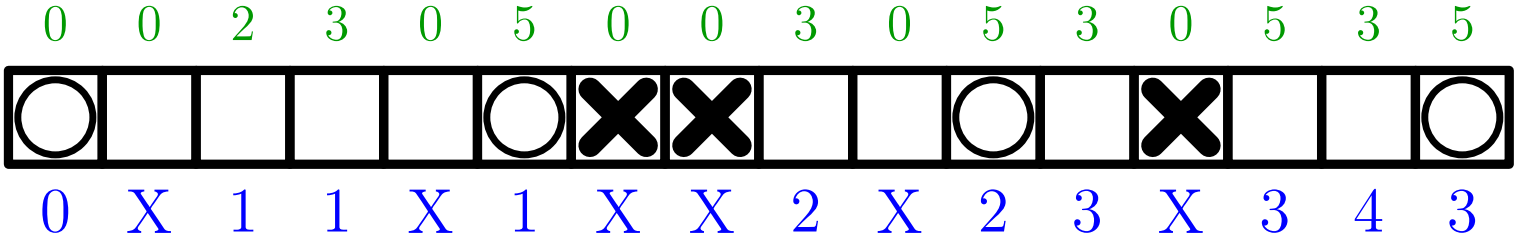


Псевдокод:

```
f[1] := 0
for k := 2, ..., n:
  f[k] := infinity
  if not h[k]:
    if k >= a and f[k] > 1 + f[k - a]:
      p[k] := a,  f[k] := 1 + f[k - a]
    if k >= b and f[k] > 1 + f[k - b]:
      p[k] := b,  f[k] := 1 + f[k - b]
    if k >= c and f[k] > 1 + f[k - c]:
      p[k] := c,  f[k] := 1 + f[k - c]
```

Восстановление ответа — хранение предков

Для каждой клетки k не только выясним, за какое минимальное число прыжков кузнечик может до неё добраться, но и запомним, каким прыжком он туда попал.



Будем восстанавливать путь с конца: стоя в клетке x, сдвигаться назад на величину прыжка p_x.
Псевдокод:

```
x := n
steps.add (x)
while x > 1:
  x -= p[x]
  steps.add (x)
reverse (steps)
```

Вопросы?

Вопросы?