# Содержание



---

У вас не получается читать/выводить данные?
Воспользуйтесь примерами (c++) (python).

Обратите внимание, входные данные лежат в **стандартном потоке ввода** (он же stdin), вывести ответ нужно в **стандартный поток вывода** (он же stdout).

Обратите внимание на `GNU C++` компиляторы с суффиксом `inc`.
Подни можно пользоваться дополнительной библиотекой (`optimization.h`).
То есть, использовать быстрый ввод-вывод: **пример про числа и строки**.
И быструю аллокацию памяти (ускоряет vector-set-map-весь-STL): пример.

Для тех, кто хочет разобраться, как всё это работает.
Короткая версия быстрого ввода-вывода (тык) и короткая версия аллокатора (тык).

# Must have

## Задача 23A. Тест к рюкзаку [0.1, 256]

Задача о рюкзаке:

*Даны $n$ предметов с положительными* **целыми** *весами $w_i$ и стоимостями $cost_i$. И рюкзак размера $W$. Выберите, какие предметы положить в рюкзак, чтобы их суммарный вес не превосходил $W$, а суммарная стоимость была максимальна?*

Ваша задача: найти тест, на котором не работает следующая жадность.

*Переберём $i$ — какой один предмет точно нужно положить в рюкзак. Оставшиеся предметы ($n-1$ штук) отсортируем по убыванию удельной стоимости $\frac{cost_i}{w_i}$, а при равенстве удельной стоимости по возрастанию индекса (стабильная сортировка). Переберём оставшиеся предметы в таком порядке и будем пытаться жадно добавлять в рюкзак. Из полученных $n$ ответов выберем лучший.*

### Формат входных данных

На первой строке целые числа $n$ ($3 \leqslant n \leqslant 12$) — количество предметов, $W$ ($4 \leqslant W \leqslant 50$) — размер рюкзака. На второй и третьей строках целые числа $A$, $B$, $C$, $D$ ($1 \leqslant A < B \leqslant 20$, $1 \leqslant C < D \leqslant 20$) — ограничения на веса и стоимости. Вам нужно найти тест с данными $n$ и $W$, который удовлетворяет ограничениям $A \leqslant w_i \leqslant B$, $C \leqslant cost_i \leqslant D$.

Гарантируется, что тест с такими ограничениями существует.

### Формат выходных данных

Тест, на котором жадность даст неверный ответ.
В первой строке выведите $n$ и $W$. Следующие $n$ строк описывают предметы $w_i$ $cost_i$.

### Примеры

| stdin | stdout |
|---|---|
| 6 50<br>5 12<br>1 10 | 6 50<br>12 10<br>11 4<br>12 5<br>5 4<br>7 3<br>8 8 |

### Подсказка по решению

Пора научиться писать стресс-тесты. Суть в том, что вы много раз генерите случайный тест, пока ответы верного и неверного решения не начнут отличаться.

В `C++` мы можем использовать функцию `stable_sort`.

# Обязательные задачи

## Задача 23B. Самое дешевое ребро [0.1, 256]

Дано подвешенное дерево с корнем в первой вершине. Все ребра имеют веса (стоимости). Вам нужно ответить на $M$ запросов вида "найти у двух вершин минимум среди стоимостей ребер пути между ними".

### Формат входных данных

В первой строке файла записано количество вершин $n$ ($2 \leqslant n \leqslant 50\,000$). В следующих $n-1$ строках записаны отцы вершин $2..n$. Для вершины $i$ записана пара $x$ $y$. Число $x$ означает, что $x$ — предок вершины $i$, $y$ означает стоимость ребра. $x < i, |y| \leqslant 10^6$.

Далее число запросов $m$ ($0 \leqslant m \leqslant 50\,000$) и $m$ запросов вида $(x, y)$ — найти минимум на пути из $x$ в $y$ ($x \neq y$).

### Формат выходных данных

$m$ ответов на запросы.

### Пример

| stdin | stdout |
|---|---|
| 5<br>1 2<br>1 3<br>2 5<br>3 2<br>2<br>2 3<br>4 5 | 2<br>2 |

### Подсказка по решению

Эту задачу можно решать несколькими способами, например двоичные подъёмы; или перебирать рёбра в порядке убывания веса, а внутри DSU. Здесь эта задача дана, чтобы вы научились использовать centroid decomposition. В следующем контесте на центроиды будет уже несколько задач, тренируйтесь.

## Задача 23C. Задача про два станка ($F2 \,||\, C_{max}$) [0.1, 256]

Имеется множество из $n$ работ и два станка. Время выполнения $i$-й работы на первом станке равно $a_i$, время выполнения $i$-й работы на втором станке равно $b_i$. Каждую работу надо выполнить сначала на первом станке, потом на втором. И на первом, и на втором станке работы можно выполнять в произвольном порядке. Каждый станок в каждый момент времени может выполнять только одну работу.

Минимизируйте $C_{max}$ — время выполнения последней работы на втором станке.

### Формат входных данных

В первой строке дано одно целое число $n$ ($1 \leqslant n \leqslant 100\,000$) — количество работ.

В следующей строке $n$ целых чисел от 0 до $10^6$ – время выполнения работ на 1-м станке.

В следующей строке $n$ целых чисел от 0 до $10^6$ – время выполнения работ на 2-м станке.

### Формат выходных данных

В первой строке выведите единственное число — $C_{max}$.

Во второй строке выведите перестановку – порядок выполнения работ на первом станке.

В третьей строке выведите перестановку – порядок выполнения работ на втором станке.

### Примеры

| stdin | stdout |
|---|---|
| 3<br>1 2 3<br>5 5 5 | 16<br>1 3 2<br>1 2 3 |
| 2<br>3 2<br>1 3 | 6<br>2 1<br>2 1 |

### Подсказка по решению

Задача про 2 станка. На лекции (в конспекте) разобрано два решения.

## Задача 23D. Тесты к задаче про два станка [0.1, 256]

Имеется множество из $n$ работ и два станка. Время выполнения $i$-й работы на первом станке равно $a_i$, время выполнения $i$-й работы на втором станке равно $b_i$.

Каждую работу надо выполнить сначала на первом станке, потом на втором. На первом и на втором станках работы нужно выполнять в одинаковом порядке. Каждый станок в каждый момент времени может выполнять только одну работу.

Задача о двух станках – минимизировать время выполнения последней работы на втором станке.

Ваша задача – построить тесты к всевозможным жадностям.

### Формат входных данных

В первой строке дано одно целое число $n$ ($4 \leqslant n \leqslant 12$) — количество работ в тесте, который вам нужно построить.

Во второй строке $L, R$ ($1 \leqslant L \leqslant 10$, $L < R \leqslant 10^9$) — ограничения на числа $a_i, b_i$ ($L \leqslant a_i, b_i \leqslant R$).

В третьей строке написан тип жадности, против которой нужно построить тест.

Каждая жадность – запуск `stable_sort` с компаратором `less(i,j)`:

1 : `a[i] < a[j]`
2 : `b[i] < b[j]`
3 : `a[i] + b[i] < a[j] + b[j]`
4 : `a[i] - b[i] < a[j] - b[j]`
5 : `a[i] * b[i] < a[j] * b[j]`
6 : `(double)a[i] / b[i] < (double)a[j] / b[j]`
7 : `min(a[i], b[j]) < min(a[j], b[i])`
8 : `max(a[i], b[j]) < max(a[j], b[i])`
9 : `max(a[i], b[i]) < max(a[j], b[j])`

Далее написано число 0 или 1 — нужно ли развернуть массив после сортировки.

Гарантируется, что такой тест существует.

### Формат выходных данных

В первой строке $n$.

Во второй строке массив $a$.

В третьей строк массив $b$.

### Примеры

| stdin | stdout |
|---|---|
| 5<br>10 20<br>7<br>1 | 5<br>12 10 15 14 11<br>15 14 15 14 15 |
| 5<br>10 20<br>7<br>0 | 5<br>13 10 11 13 10<br>11 10 11 15 13 |

### Замечание

В первом примере вам нужно построить тест против следующей жадности:

```
stable_sort(p.begin(), p.end(), [](int i, int j) {
     return max(a[i], b[j]) < max(a[j], b[i]);
});
reverse(p.begin(), p.end());
```

### Подсказка по решению

Для $n = 12$ решение за $\mathcal{O}(n!)$ вряд ли зайдёт.

`stable_sort` $\neq$ `sort`, будьте внимательны!

При больших $R$ тест не найдётся, потому что не возникнут крайние случаи $a_i = b_i, a_i = a_j$. Это проявляется, например, на 84-м тесте. Вообще всегда, когда вы стрессите, на больших $n$ и $a_i$ тесты **НЕ** находятся, ищите тесты на маленьких $n$ и $a_i$.

# Для искателей острых ощущений

## Задача 23E. TSP [1, 256]

Даны $n$ случайных точек на плоскости. Расстояние между точками обычное Евклидово. Постройте кратчайший цикл коммивояжера.

### Формат входных данных

На первой строке число точек $n$ ($1 \leqslant n \leqslant 500$).
Далее сами точки. Координаты целые до $10^9$.

### Формат выходных данных

Выведите перестановку $n$ чисел от 1 до $n$.
Ваш ответ должен отличаться от оптимального не более чем на 15%.
Если хороших ответов несколько, выведите любой.

### Пример

| stdin | stdout |
|---|---|
| 4<br>0 0<br>1 1<br>0 1<br>1 0 | 2 4 1 3 |

### Подсказка по решению

Вы умеете строить хорошее приближение.
Вы умеете делать локальные оптимизация `reverse`-ом.
Вы знаете технику random-walk (запускай несколько раз одно и то же с разным random).

## Задача 23F. Гамильтонов путь конём [1, 256]

Дана доска размера $n \times n$ ($5 \leqslant n \leqslant 400$).
Известно, что такую доску можно обойти конём, посетив каждую клетку ровно один раз.

### Формат входных данных

Число $n$.

### Формат выходных данных

Выведите гамильтонов **путь** – $n \times n$ строк, каждая содержит очередную клетку.
Координаты нумеруются с нуля.
**Путь** можно строить из любой клетки. Если путей несколько, выведите любой.

### Примеры

| stdin | stdout |
|---|---|
| 5 | 4 0<br>3 2<br>4 4<br>2 3<br>0 4<br>1 2<br>0 0<br>2 1<br>4 2<br>3 4<br>1 3<br>0 1<br>2 0<br>4 1<br>3 3<br>1 4<br>0 2<br>1 0<br>3 1<br>4 3<br>2 4<br>0 3<br>1 1<br>3 0<br>2 2 |

### Подсказка по решению

Начинайте со случайной точки несколько раз, и всё получится.
Кроме минимизации остаточной степени, можно ещё стремиться ближе к краю.