

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1
по курсу «Алгоритмы и структуры данных»
Тема: Жадные алгоритмы. Динамическое программирование No2
Вариант 3

Выполнил:

Бунос М.В.

К3141

Проверила:

Артамонова В.Е.

Санкт-Петербург

2023 г.

Содержание отчета

Содержание отчета	2
Задачи	3
Задача №1. Максимальная стоимость добычи (0.5 балла)	3
Задача №2. Заправки (0.5 балла)	7
Задача №3. Максимальный доход от рекламы (0.5 балла)	10
Задача №4. Сбор подписей (0.5 балла)	13
Задача №5. Максимальное количество призов (0.5 балла)	16
Задача №6. Максимальная зарплата (0.5 балла)	19
Задача №7. Проблема сапожника (0.5 балла)	22
Задача №8. Расписание лекций (1 балл)	25
Задача №9. Распечатка (1 балл)	28
Задача №10. Яблоки (1 балл)	32
Задача №11. Максимальное количество золота (1 балл)	36
Задача №12. Последовательность (1 балл)	39
Задача №13. Сувениры (1.5 балла)	42
Задача №14. Максимальное значение арифметического выражения (2 балла)	45
Задача №15. Удаление скобок (2 балла)	48
Задача №16. Продавец (2 балла)	51
Задача №17. Ход конем (2.5 балла)	54
Задача №18. Кафе (2.5 балла)	57
Задача №19. Произведение матриц (3 балла)	61
Задача №20. Почти палиндром (3 балла)	64
Задача №21. Игра в дурака (3 балла)	68
Задача №22. Симпатичные узоры (4 балла)	72
Вывод	76

Задачи

Задача №1. Максимальная стоимость добычи (0.5 балла)

Вор находит гораздо больше добычи, чем может поместиться в его сумке. Помогите ему найти самую ценную комбинацию предметов, предполагая, что любая часть предмета добычи может быть помещена в его сумку.

Цель - реализовать алгоритм для задачи о дробном рюкзаке.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

struct Item {
    int weight;
    int value;
```

```

};

bool cmp(Item a, Item b) {
    return (double)a.value / a.weight > (double)b.value / b.weight;
}

double fractionalKnapsack(int W, vector<Item>& items) {
    sort(items.begin(), items.end(), cmp);

    double totalValue = 0.0;

    for (int i = 0; i < items.size(); i++) {
        if (W == 0) {
            return totalValue;
        }

        if (items[i].weight <= W) {
            totalValue += items[i].value;
            W -= items[i].weight;
        } else {
            totalValue += ((double)W / items[i].weight) * items[i].value;
            W = 0;
        }
    }

    return totalValue;
}

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n, W;
    fin >> n >> W;

    vector<Item> items(n);

    for (int i = 0; i < n; i++) {
        fin >> items[i].value >> items[i].weight;
    }

    fout << fractionalKnapsack(W, items) << endl;

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Решение использует жадный алгоритм для решения задачи о дробном рюкзаке. Структура Item используется для хранения веса и стоимости

каждого элемента. Функция `cmp` используется для сортировки предметов в порядке убывания стоимости на единицу веса.

Функция `FractionalKnapsack` принимает вес W и список предметов. Сначала он сортирует элементы с помощью функции `cmp`. Затем он перебирает предметы, добавляя каждый предмет в рюкзак, если он может поместиться полностью, или добавляя его дробную часть, если он не может поместиться полностью. Функция возвращает общую стоимость предметов, добавленных в рюкзак.

Результат работы кода на примерах из текста задачи:

```

input.txt x
1 3 50
2 60 20
3 100 50
4 120 30

output.txt x
1 180
2

input.txt x
1 1 10
2 500 30

output.txt x
1 166.667
2
  
```

Проверка задачи на (openedu, astpr и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.792969 MB
Пример из задачи	0.001000 sec	0.808594 MB
Пример из задачи	0.001000 sec	0.792969 MB
Верхняя граница	0.001000 sec	0.808594 MB

диапазона значений входных данных из текста задачи		
--	--	--

Вывод по задаче:

Самая банальная задача на жадные алгоритмы.

Задача №2. Заправки (0.5 балла)

Вы собираетесь поехать в другой город, расположенный в d км от вашего родного города. Ваш автомобиль может проехать не более m км на полном баке, и вы начинаете с полным баком. По пути есть заправочные станции на расстояниях $stop_1, stop_2, \dots, stop_n$ из вашего родного города. Какое минимальное количество заправок необходимо?

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
```

```

int d, m, n;
fin >> d >> m >> n;

vector<int> stops(n+2);
for (int i = 1; i <= n; i++) {
    fin >> stops[i];
}
stops[0] = 0;
stops[n+1] = d;

int num_refills = 0, current_refill = 0;
while (current_refill <= n) {
    int last_refill = current_refill;
    while (current_refill <= n && stops[current_refill+1] -
stops[last_refill] <= m) {
        current_refill++;
    }
    if (current_refill == last_refill) {
        fout << "-1" << endl;
        return 0;
    }
    if (current_refill <= n) {
        num_refills++;
    }
}

fout << num_refills << endl;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

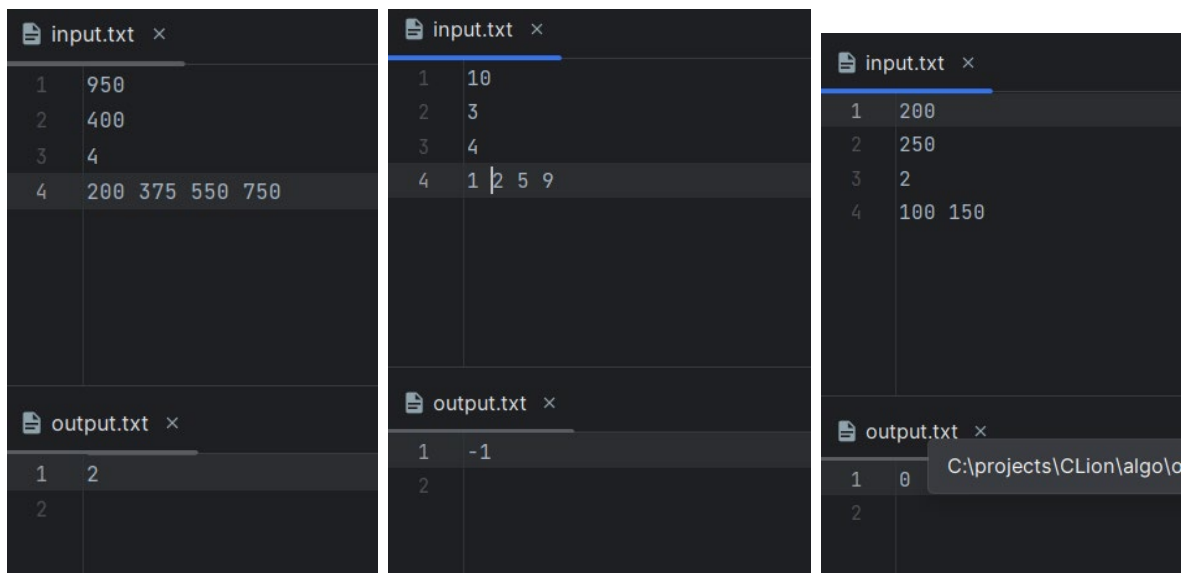
```

Текстовое объяснение решения:

Задача требует найти минимальное количество остановок для дозаправки, необходимое для проезда определенного расстояния d , учитывая максимальное расстояние m , которое можно преодолеть с полным баком, и список из n остановок для заправки по пути.

Мы можем решить эту проблему, перебирая остановки для заправки и отслеживая текущую и последнюю заправки. Мы можем начать с полного бака топлива, а затем проверить, сможем ли мы добраться до следующей заправочной станции без дозаправки. Если мы не можем, мы возвращаемся к последней заправочной остановке, до которой мы можем добраться с полным баком, и заправляем бак там. Мы повторяем этот процесс, пока не достигнем пункта назначения.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.808594 MB
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.792969 MB

Вывод по задаче:

Тоже довольно простая задача на жадные алгоритмы.

Задача №3. Максимальный доход от рекламы (0.5 балла)

У вас есть n объявлений для размещения на популярной интернет-странице. Для каждого объявления вы знаете, сколько рекламодатель готов платить за один клик по этому объявлению. Вы настроили n слотов на своей странице и оценили ожидаемое количество кликов в день для каждого слота. Теперь ваша цель -распределить рекламу по слотам, чтобы максимизировать общий доход.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();
```

```

// ---- code starts here ----

int n;
fin >> n;

vector<int> a(n);
for (int i = 0; i < n; i++) {
    fin >> a[i];
}

vector<int> b(n);
for (int i = 0; i < n; i++) {
    fin >> b[i];
}

sort(a.begin(), a.end(), greater<int>());
sort(b.begin(), b.end(), greater<int>());

long long ans = 0;
for (int i = 0; i < n; i++) {
    ans += (long long) a[i] * b[i];
}

fout << ans << endl;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

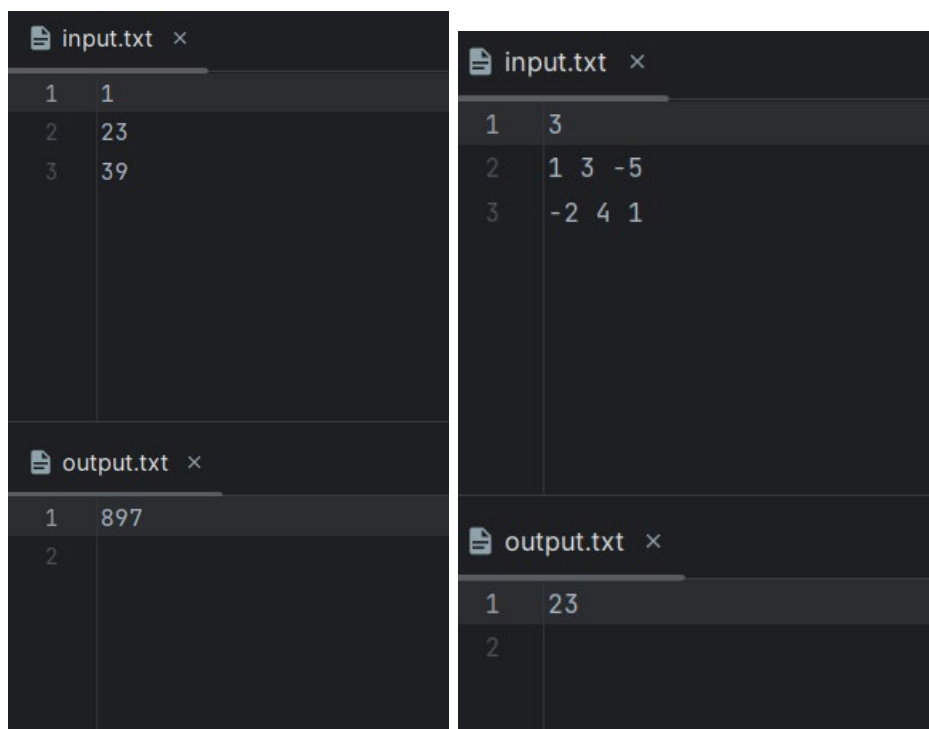
fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Это задача оптимизации, которую можно решить с помощью жадного подхода. Идея состоит в том, чтобы отсортировать массивы в порядке убывания, а затем умножить самый большой элемент из одного массива на самый большой элемент из другого массива, второй по величине на второй по величине и так далее. Сумма этих произведений будет максимально возможной выручкой.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.789062 MB

Вывод по задаче:

Аналогично простая задача на жадосы.

Задача №4. Сбор подписей (0.5 балла)

Вы несете ответственность за сбор подписей всех жильцов определенного здания. Для каждого жильца вы знаете период времени, когда он или она находится дома. Вы хотите собрать все подписи, посетив здание как можно меньше раз. Математическая модель этой задачи следующая. Вам дан набор отрезков на

прямой, и ваша цель - отметить как можно меньше точек на прямой так, чтобы каждый отрезок содержал хотя бы одну отмеченную точку.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();
```

```

// ---- code starts here -----

int n;
fin >> n;
vector<pair<int, int>> segments(n);
for (int i = 0; i < n; i++) {
    fin >> segments[i].first >> segments[i].second;
}

sort(segments.begin(), segments.end(), [](const auto& a, const auto& b)
{
    return a.second < b.second;
});

vector<int> points;
int rightmost = -1;

for (const auto& [left, right] : segments) {
    if (left > rightmost) {
        points.push_back(right);
        rightmost = right;
    }
}

fout << points.size() << endl;

for (int p: points) {
    fout << p << " ";
}

// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Чтобы решить эту проблему, мы можем использовать жадный алгос. Мы начинаем с сортировки сегментов по их правой конечной точке, а затем итерируем интервалы слева направо, отслеживая крайнюю правую точку, которую мы рассмотрели до сих пор. Всякий раз, когда мы сталкиваемся с интервалом, левая конечная точка которого больше, чем крайняя правая точка, которую мы рассмотрели, мы добавляем его правую конечную точку в наш набор точек и обновляем крайнюю правую точку, которую мы рассмотрели, чтобы она стала новой правой конечной точкой.

Результат работы кода на примерах из текста задачи:

input.txt ×

1 3
2 1 3
3 2 5
4 3 6

output.txt ×

1 1
2 3

input.txt ×

1 4
2 4 7
3 1 3
4 2 5
5 5 6

output.txt ×

1 2
2 3 6

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.789062 MB

Вывод по задаче:

Все как всегда – все просто.

Задача №5. Максимальное количество призов (0.5 балла)

Вы организуете веселый конкурс для детей. В качестве призового фонда у вас есть n конфет. Вы хотели бы использовать эти конфеты для раздачи k лучшим местам в конкурсе с естественным ограничением, заключающимся в том, что чем выше место, тем больше конфет. Чтобы осчастливить как можно больше детей, вам нужно найти наибольшее значение k , для которого это возможно.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();
```



```

// ---- code starts here ----

int n;
fin >> n;

int k = 0;
while (n > 0) {
    k++;
    n -= k;
}

if (n < 0) {
    k--;
    n += k + 1;
}

fout << k << endl;

vector<int> ans(k);
for (int i = 0; i < k - 1; i++) {
    ans[i] = i + 1;
}
ans[k - 1] = n + k;

for (int i = 0; i < k; i++) {
    fout << ans[i] << " ";
}

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

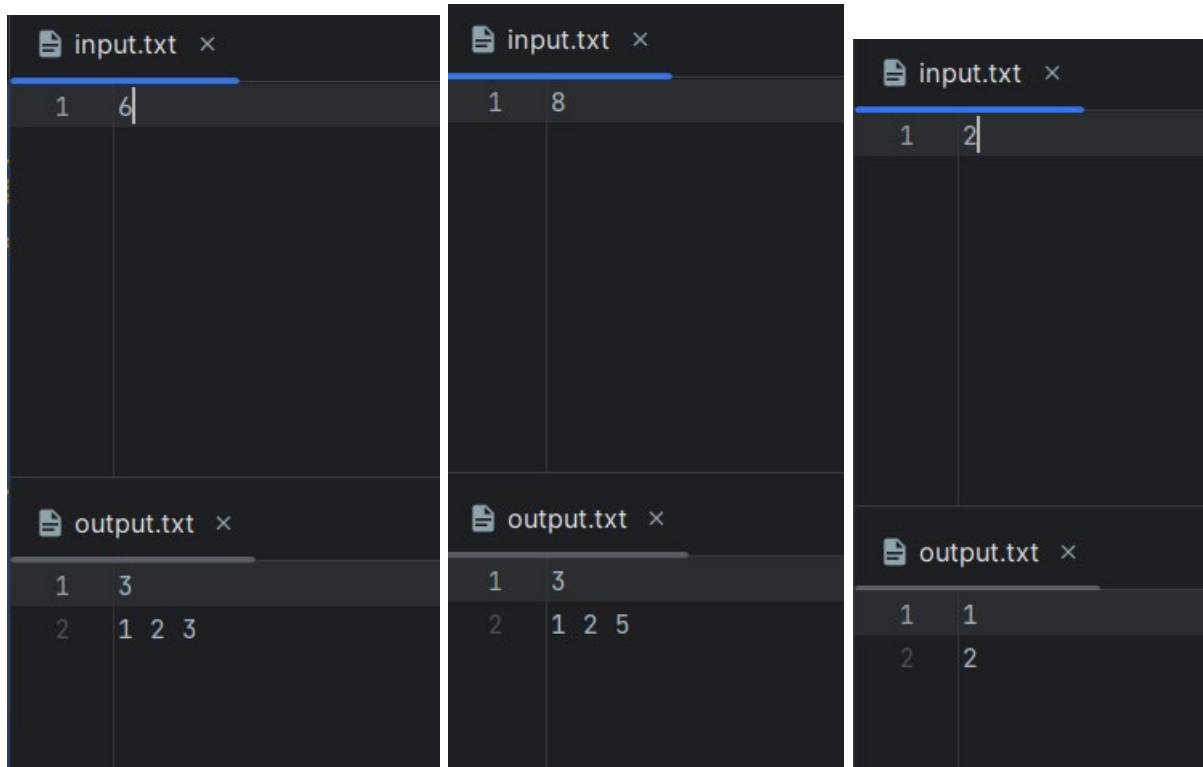
Текстовое объяснение решения:

Мы можем найти k , просто вычитая 1, 2, 3, ..., k из n до тех пор, пока мы больше не сможем этого делать. Количество раз, которое мы смогли вычесть, равно k , что является максимальным значением k , удовлетворяющим требованиям задачи.

Если мы перевычитаем, нам нужно прибавить последнее вычитаемое число к n и вычесть 1 из k . Получив значение k , мы можем просто построить k чисел, которые в сумме дают n , следующим образом: первые $k-1$ числа равны 1, 2, 3, ..., $k-1$, а последнее число равно $n + k$.

Затем мы выводим k , за которым следуют k чисел, которые мы только что построили. Этот алгоритм работает с временной сложностью $O(\sqrt{n})$, что очень эффективно даже при больших значениях n .

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.004000 sec	0.964844 MB

Вывод по задаче:

Аналогично очень простая задачка.

Задача №6. Максимальная зарплата (0.5 балла)

В качестве последнего вопроса успешного собеседования ваш начальник дает вам несколько листов бумаги с цифрами и просит составить из этих цифр наибольшее число. Полученное число будет вашей зарплатой, поэтому вы очень заинтересованы в максимизации этого числа. Как вы можете это сделать?

На лекциях мы рассмотрели следующий алгоритм составления наибольшего числа из заданных однозначных чисел.

К сожалению, этот алгоритм работает только в том случае, если вход состоит из однозначных чисел. Например, для ввода, состоящего из двух целых чисел 23 и 3 (23 не однозначное число!) возвращается 233, в то время как наибольшее число на самом деле равно 323. Другими словами, использование наибольшего числа из входных данных в качестве первого числа не является безопасным ходом.

Ваша цель в этой задаче – настроить описанный выше алгоритм так, чтобы он работал не только с однозначными числами, но и с произвольными положительными целыми числами.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}
```

```

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n;
    fin >> n;

    vector<int> nums(n);
    for (int i = 0; i < n; i++) {
        fin >> nums[i];
    }

    sort(nums.begin(), nums.end(), [](int a, int b) {
        return to_string(a) + to_string(b) > to_string(b) + to_string(a);
    });

    string ans = "";
    for (int i = 0; i < n; i++) {
        ans += to_string(nums[i]);
    }

    fout << ans << endl;

    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Чтобы реализовать это, мы можем создать пользовательскую функцию сравнения стр, которая принимает два целых числа a и b. Функция сначала преобразует a и b в строки, а затем сравнивает конкатенацию a и b с конкатенацией b и a. Если первое больше, то перед b следует поставить a, в противном случае b следует поставить перед a. Мы можем объединить отсортированные целые числа вместе, чтобы сформировать максимально возможное число, и вывести его в виде строки.

Результат работы кода на примерах из текста задачи:

input.txt ×

13

223 39 92

output.txt ×

⚠ This document contains very long

1923923

2

input.txt ×

12

221 2

output.txt ×

⚠ This document contains very long

1221

2

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.001000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.800781 MB

Вывод по задаче:

Просто написать компаратор...

Задача №7. Проблема сапожника (0.5 балла)

В некоей воинской части есть сапожник. Рабочий день сапожника длится K минут. Заведующий складом оценивает работу сапожника по количеству починенной обуви, независимо от того, насколько сложный ремонт требовался в каждом случае. Дано n сапог, нуждающихся в починке. Определите, какое максимальное количество из них сапожник сможет починить за один рабочий день.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();
```

```

// ---- code starts here ----

int K, n;
fin >> K >> n;

vector<int> t (n);
for (int i = 0; i < n; i++) {
    fin >> t[i];
}

sort(t.begin(), t.end());

int boots_repaired = 0;
int time_spent = 0;
for (int i = 0; i < n; i++) {
    if (time_spent + t[i] <= K) {
        time_spent += t[i];
        boots_repaired++;
    }
    else {
        break;
    }
}

fout << boots_repaired << endl;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Чтобы решить эту задачу, нам нужно найти максимальное количество ботинок, которые можно отремонтировать за K минут. Мы можем начать с сортировки ботинок по времени их ремонта. Затем мы можем приступить к ремонту ботинок один за другим, начиная с ботинка, на ремонт которого уходит наименьшее количество времени. Мы продолжаем чинить сапоги, пока не истечет время или пока не отремонтируем все сапоги.

Результат работы кода на примерах из текста задачи:

input.txt ×

```

1 10 3
2 6 2 8

```

output.txt ×

⚠ This document contains very

```

1 2
2

```

input.txt ×

```

1 3 2
2 10 20

```

output.txt ×

⚠ This document contains very

```

1 0
2

```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.781250 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB

Вывод по задаче:

Тоже довольно простая задача

Задача №8. Расписание лекций (1 балл)

Вы наверно знаете, что в ИТМО лекции читают одни из лучших преподаватели мира. К сожалению, лекционных аудиторий у нас не так уж и много, особенно на Биржевой, поэтому каждый преподаватель составил список лекций, которые он хочет прочитать студентам. Чтобы студенты, в начале февраля, увидели расписание лекций, необходимо его составить прямо сейчас. И без вас нам здесь не справиться. У нас есть список заявок от преподавателей на лекции для одной из аудиторий. Каждая заявка представлена в виде временного интервала $[s_i, f_i)$ - время начала и конца лекции. Лекция считается открытым интервалом, то есть какая-то лекция может начаться в момент окончания другой, без перерыва. Необходимо выбрать из этих заявок такое подмножество, чтобы суммарно выполнить максимальное количество заявок. Учтите, что одновременно в лекционной аудитории, конечно же, может читаться лишь одна лекция.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}
```

```

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here -----

    int n;
    fin >> n;

    vector<pair<int, int>> lectures(n);
    for (int i = 0; i < n; i++) {
        fin >> lectures[i].first >> lectures[i].second;
    }

    sort(lectures.begin(), lectures.end(), [](std::pair<int, int> a,
std::pair<int, int> b) {
        return a.second < b.second;
    });

    int end_time = 0;
    int count = 0;
    for (int i = 0; i < n; i++) {
        if (lectures[i].first >= end_time) {
            end_time = lectures[i].second;
            count++;
        }
    }

    fout << count;

    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

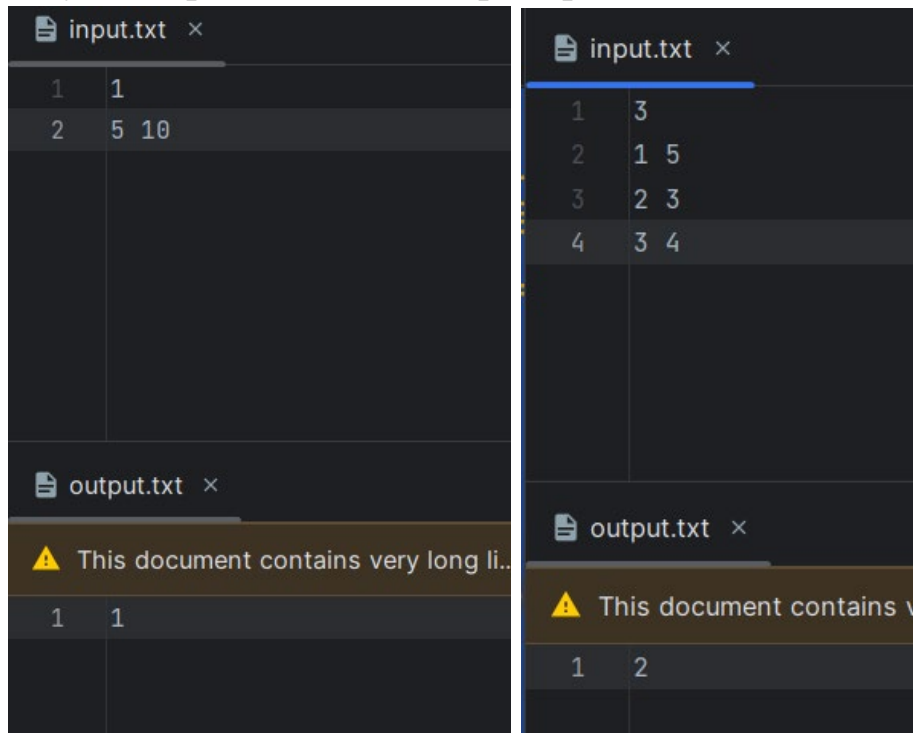
```

Текстовое объяснение решения:

В решении используется жадный алгоритм: лекции сортируются по времени окончания, а затем перебираются в этом порядке. Для каждой лекции проверяется, начинается ли она после окончания предыдущей выбранной лекции. Если это так, он выбирает эту лекцию и обновляет текущее время окончания до своего времени окончания. В конце он выводит количество выбранных лекций.

Временная сложность решения составляет $O(n \log n)$, в которой преобладает этап сортировки. Сложность пространства $O(n)$ для хранения списка лекций.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.781250 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.781250 MB

Вывод по задаче:

Аналогично простенькая задача на сортировку.

Задача №9. Распечатка (1 балл)

Диссертация дело сложное, особенно когда нужно ее печатать. При этом вам нужно распечатать не только текст самой диссертации, так и другие материалы (задание, рецензии, отзывы, афторефераты для защиты и т.п.). Вы оценили объём печати в N листов. Фирма, готовая размножить печатные материалы, предлагает следующие финансовые условия. Один лист она печатает за A1 рублей, 10 листов - за A2 рублей, 100 листов - за A3 рублей, 1000 листов - за A4 рублей, 10000 листов - за A5 рублей, 100000 листов - за A6 рублей, 1000000 листов - за A7 рублей. При этом не гарантируется, что один лист в более крупном заказе обойдется дешевле, чем в более мелком. И даже может оказаться, что для любой партии будет выгодно воспользоваться тарифом для одного листа. Печать конкретного заказа производится или путем комбинации нескольких тарифов, или путем заказа более крупной партии. Например, 980 листов можно распечатать, заказав печать 9 партий по 100 листов плюс 8 партий по 10 листов, сделав 98 заказов по 10 листов, 980 заказов по 1 листу или заказав печать 1000 (или даже 10000 и более) листов, если это окажется выгоднее. Требуется по заданному объему заказа в листах N определить минимальную сумму денег в рублях, которой будет достаточно для выполнения заказа.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
```

```

    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
<< " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
" sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here -----

    int n;
    vector <int> a (7);
    fin >> n;
    for (int i = 0; i < 7; i++) {
        fin >> a[i];
    }

    for(int i = 1; i < 7; ++ i) {
        a[i] = min(a[i], a[i - 1] * 10);
    }

    int cur = 0;
    long long ans = 0;
    int copyn = n;

    while(copyn) {
        ans += a[cur] * (copyn % 10);
        copyn /= 10;
        ++ cur;
    }

    int nowPower = 1;
    for(int i = 0; i < 7; ++ i) {
        if(nowPower >= n && a[i] < ans) {
            ans = a[i];
        }
        nowPower *= 10;
    }

    fout << ans << endl;

    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Сначала массив a обрабатывается с помощью динамического программирования. Для каждого i от 1 до 6 он вычисляет минимальную стоимость печати 10^i листов как минимум $a[i]$ и $a[i-1] * 10$. Это позволяет нам воспользоваться оптовыми скидками там, где они существуют, а также зафиксировать случай, когда печать большого количества листов на самом деле обходится дороже, чем их печать небольшими партиями.

Далее код перебирает цифры числа n справа налево. Для каждой цифры d к текущей сумме ans добавляется стоимость печати d листов. Чтобы вычислить стоимость, он умножает $a[i]$ на d , где i — индекс цифры. Это работает, потому что i -я цифра представляет собой количество 10^i листов, а мы уже вычислили минимальную стоимость печати такого количества листов в массиве a .

Наконец, код проверяет, не дешевле ли распечатать все задание сразу. Он делает это, перебирая цены в массиве и проверяя, не дешевле ли распечатать n листов по этой цене, чем распечатать каждую цифру отдельно. Он отслеживает самую дешевую цену в переменной ans .

Результат работы кода на примерах из текста задачи:

input.txt		input.txt	
1	980	1	980
2	1	2	1
3	9	3	10
4	90	4	100
5	900	5	1000
6	1000	6	900
7	10000	7	10000
8	10000	8	10000

output.txt		output.txt	
1	882	1	900
2		2	

Проверка задачи на (openedu, acmp и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница	0.000000 sec	0.792969 MB

диапазона значений входных данных из текста задачи		
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.789062 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.789062 MB

Вывод по задаче:

Легкая задача на очень простое ДП.

Задача №10. Яблоки (1 балл)

Алисе в стране чудес попались n волшебных яблок.

Про каждое яблоко известно, что после того, как его съешь, твой рост сначала уменьшится на a_i сантиметров, а потом увеличится на b_i сантиметров.

Алиса очень голодная и хочет съесть все n яблок, но боится, что в какой-то момент ее рост s станет равным нулю или еще меньше, и она пропадет совсем. Помогите ей узнать, можно ли съесть яблоки в таком порядке, чтобы в любой момент времени рост Алисы был больше нуля.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();
```



```

// ---- code starts here -----

int n, s;
fin >> n >> s;

vector<pair <pair <int, int> , int>> apples;
vector<pair <pair <int, int> , int>> left;
apples.reserve(n);
left.reserve(n);

for (int i = 0; i < n; ++i) {
    pair <int, int> a;
    fin >> a.first >> a.second;

    if(a.second > a.first)
        apples.push_back({a, i});
    else
        left.push_back({a, i});
}

sort(apples.begin(), apples.end());
sort(left.begin(), left.end(), [](const pair <pair <int, int> , int>
&a, const pair <pair <int, int> , int> &b) {
    return a.first.second - a.first.first < b.first.second -
b.first.first;
});

vector <int> answer;

for(auto &i : apples) {
    if(s >= i.first.first) {
        s += i.first.second - i.first.first;
        answer.push_back(i.second);
    } else {
        fout << "-1";
        return 0;
    }
}

for(auto &i : left) {
    if(s >= i.first.first) {
        s += i.first.second - i.first.first;
        answer.push_back(i.second);
    } else {
        fout << "-1";
        return 0;
    }
}

for(auto &i : answer)
    fout << i + 1 << " ";

// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();

```

```

    return 0;
}

```

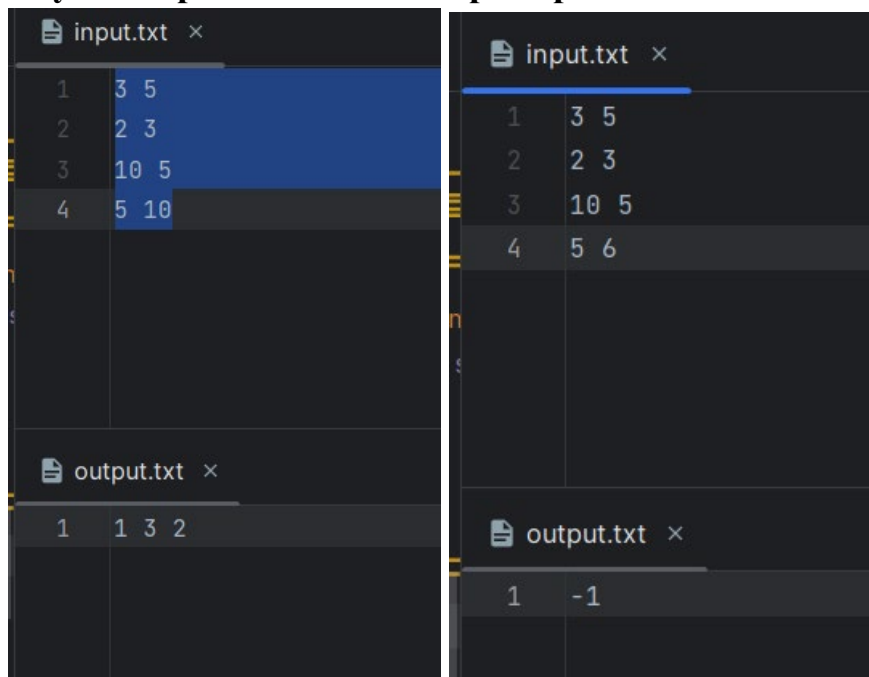
Текстовое объяснение решения:

Раскладываем яблоки в два разных вектора – вектор, где все яблоки в итоге дают положительный рост, сортированный по возрастанию уменьшения роста, и вектор, где все яблоки в итоге дают отрицательный рост, отсортированный в порядке возрастания уменьшения итогового роста.

Дальше проходимся по векторам и симулируем рост Алисы, смотрим чтобы не ушел в отрицательный.

Потом выводим ответ.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.804688 MB
Верхняя граница диапазона значений	0.001000 sec	0.789062 MB

ВХОДНЫХ ДАННЫХ ИЗ ТЕКСТА ЗАДАЧИ		
------------------------------------	--	--

Вывод по задаче:

Прикольный жадос.

Задача №11. Максимальное количество золота (1 балл)

Вам дается набор золотых слитков, и ваша цель - набрать как можно больше золота в свою сумку. Существует только одна копия каждого слитка, и для каждого слитка вы можете либо взять его, либо нет (т.е. вы не можете взять часть слитка).

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----

    int W, n;
    fin >> W >> n;
```

```

vector<int> weights(n);
for (int i = 0; i < n; i++) {
    fin >> weights[i];
}

sort(weights.begin(), weights.end());

vector<vector<int>> dp(n+1, vector<int>(W+1, 0));

for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= W; j++) {
        if (weights[i-1] > j) {
            dp[i][j] = dp[i-1][j];
        } else {
            dp[i][j] = max(dp[i-1][j], dp[i-1][j-weights[i-1]] +
weights[i-1]);
        }
    }
}

fout << dp[n][W] << endl;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Классическая задача о рюкзаке, используем массив $dp[n][w]$, где w – текущий вес, n – последний обработанный элемент, проходимся, пытаемся добавить элементы

Результат работы кода на примерах из текста задачи:

input.txt	
1	10 3
2	1 4 8
output.txt	
1	9
2	

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.785156 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.800781 MB

Вывод по задаче:

Классика, рюкзак.

Задача №12. Последовательность (1 балл)

Дана последовательность натуральных чисел a_1, a_2, \dots, a_n , и известно, что $a_i \leq i$ для любого $1 \leq i \leq n$. Требуется определить, можно ли разбить элементы последовательности на две части таким образом, что сумма элементов в каждой из частей будет равна половине суммы всех элементов последовательности.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
```

```

int n;
fin >> n;
vector<int> a(n);
for (int i = 0; i < n; i++) {
    fin >> a[i];
}
sort(a.begin(), a.end(), greater<int>());
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += a[i];
}
if (sum % 2 == 1) {
    fout << "-1";
    return 0;
}

vector<int> aswecan;
int sum1, sum2;
sum1 = sum2 = 0;

for(int i = 0; i < n; ++ i) {
    if(sum1 <= sum2) {
        sum1 += a[i];
        aswecan.push_back(a[i]);
    } else {
        sum2 += a[i];
    }
}

fout << aswecan.size() << '\n';
for(int i = 0; i < aswecan.size(); ++ i) {
    fout << aswecan[i] << ' ';
}

// ---- code ends here -----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Используем жадный алгоритм. Во-первых, входная последовательность сортируется в порядке невозрастания. Затем элементы добавляются к одной из двух частей, начиная с самого большого элемента, пока сумма первой части не станет больше или равна сумме второй части. Элементы, добавляемые к первой части, сохраняются в векторе с именем `aswecan`.

Если сумма последовательности нечетная, программа выводит -1, так как разделить последовательность на две части с одинаковыми суммами невозможно. В противном случае программа выводит размер вектора

aswecan, который соответствует количеству элементов в первой части последовательности, и самих элементов.

Результат работы кода на примерах из текста задачи:

```
input.txt x
1 3
2 1 2 3

output.txt x
1 1
2 3
```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.789062 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.789062 MB

Вывод по задаче:

Ух, увлекательный жадос, пока самый интересный из всех тасок.

Задача №13. Сувениры (1.5 балла)

Вы и двое ваших друзей только что вернулись домой после посещения разных стран. Теперь вы хотели бы поровну разделить все сувениры, которые все трое накопили.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n;
    fin >> n;
```

```

vector <int> a (n);
for(auto &i : a)
    fin >> i;

int sum = 0;
for(auto &i : a)
    sum += i;

if(sum % 3) {
    fout << 0;
    return 0;
}

vector <int> test (2 * sum / 3);

for(int i = 0; i < (1ll << n); i++) {
    int sum1 = 0;
    for (int j = 0; j < n; ++j) {
        if (i & (1ll << j))
            sum1 += a[j];
    }

    if (sum1 == sum / 3) {
        int cur = 0;
        for (int j = 0; j < n; ++j) {
            if (!(i & (1ll << j)))
                test[cur++] = a[j];
        }

        for (int j = 0; j < (1ll << cur); j++) {
            int sum2 = 0;
            for (int k = 0; k < cur; ++k) {
                if (j & (1ll << k))
                    sum2 += test[k];
            }

            if (sum2 == sum / 3) {
                fout << 1;
                printTimeUse();
                printMemoryUse();
                return 0;
            }
        }
    }
}

fout << 0;

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Используем двоичный перебор для разделения массива на две части, сумма $\text{sum}/3$ и сумма $2*\text{sum}/3$, если нашли то вторую часть разбиваем на две – $\text{sum}/3$ и $\text{sum}/3$.

Результат работы кода на примерах из текста задачи:

The screenshot shows four windows of a code editor. Each window has an 'input.txt' and an 'output.txt' file open. The input files contain the following data:

- Window 1: input.txt has [4], output.txt has [0].
- Window 2: input.txt has [1, 46], output.txt has [0].
- Window 3: input.txt has [11, 17, 59, 34, 57, 17, 23, 67, 1, 18, 2, 59], output.txt has [1].
- Window 4: input.txt has [13, 1, 2, 3, 4, 5, 7, 7, 8, 10, 12, 19], output.txt has [1].

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.060000 sec	37.367188 MB

Вывод по задаче:

Прикольная реализация прикольной задачи, с кайфом.

Задача №14. Максимальное значение арифметического выражения (2 балла)

Найдите максимальное значение арифметического выражения, указав порядок применения его арифметических операций с помощью дополнительных скобок.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int f(int i, int j, const string &s, bool isMax = true) {

    if (i == j) {
        return s[i] - '0';
    }

    if(isMax) {
        int ans = -1e9;
```

```

        for (int k = i; k < j; ++k) {
            if (s[k] == '+')
                ans = max(ans, f(i, k - 1, s) + f(k + 1, j, s));
            else if (s[k] == '-')
                ans = max(ans, f(i, k - 1, s) - f(k + 1, j, s, 0));
            else if (s[k] == '*')
                ans = max(ans, f(i, k - 1, s) * f(k + 1, j, s));
        }

        return ans;
    } else {
        int ans = 1e9;

        for (int k = i; k < j; ++k) {
            if (s[k] == '+')
                ans = min(ans, f(i, k - 1, s, 0) + f(k + 1, j, s, 0));
            else if (s[k] == '-')
                ans = min(ans, f(i, k - 1, s, 0) - f(k + 1, j, s, 1));
            else if (s[k] == '*')
                ans = min(ans, f(i, k - 1, s, 0) * f(k + 1, j, s, 0));
        }

        return ans;
    }
}

int main() {
    getFirstTime();

    // ---- code starts here -----

    string s;
    fin >> s;

    fout << f(0, s.size() - 1, s);

    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

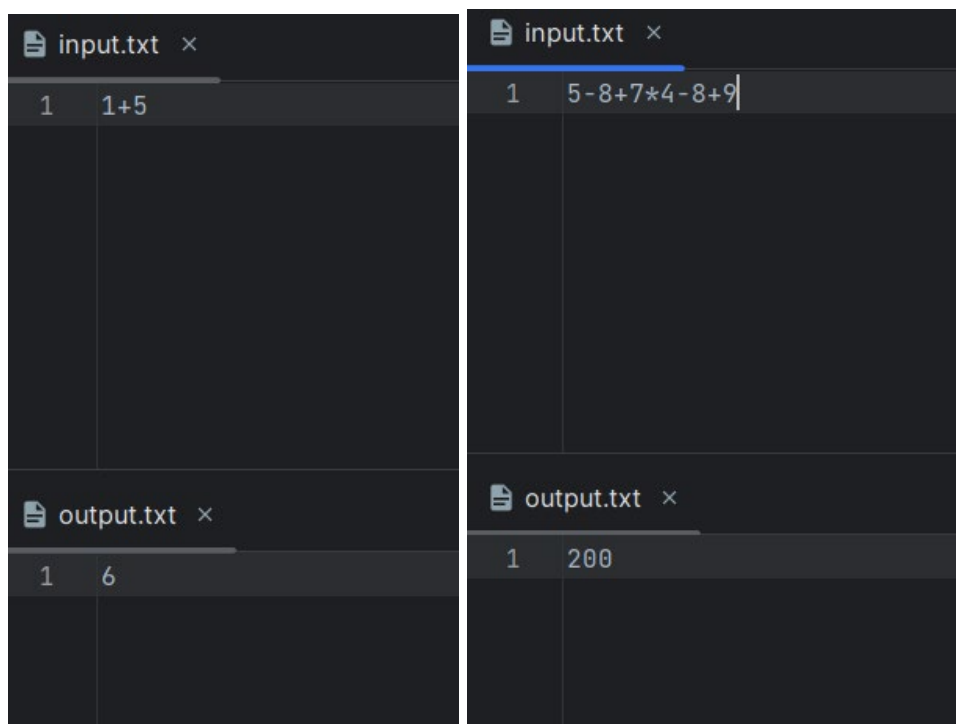
    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Напишем ДП в функции, которая будет принимать промежуток на которому мы ищем максимальный или минимальный ответ, а также саму строку. Переходы довольно очевидные, таким образом мы найдем наибольшее значение во всей строке.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.131000 sec	0.800781 MB

Вывод по задаче:

Хорошая задача на ДП.

Задача №15. Удаление скобок (2 балла)

Дана строка, составленная из круглых, квадратных и фигурных скобок. Определите, какое наименьшее количество символов необходимо удалить из этой строки, чтобы оставшиеся символы образовывали правильную скобочную последовательность.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int f(int l, int r, vector<vector<int>> &dp, vector <vector<string>>
&correct, string &s) {
    if(l > r) {
        return 0;
    }

    if(l == r)
```



```

        return 1;

    if(dp[l][r] != -1) {
        return dp[l][r];
    }

    dp[l][r] = INT_MAX;

    for(int k = l; k <= r; ++ k) {
        if(f(l, k, dp, correct, s) + f(k + 1, r, dp, correct, s) <
dp[l][r]) {
            dp[l][r] = f(l, k, dp, correct, s) + f(k + 1, r, dp, correct,
s);
            correct[l][r] = correct[l][k] + correct[k + 1][r];
        }
    }

    if(f(l + 1, r, dp, correct, s) + 1 < dp[l][r]) {
        dp[l][r] = f(l + 1, r, dp, correct, s) + 1;
        correct[l][r] = correct[l + 1][r];
    }

    if(f(l, r - 1, dp, correct, s) + 1 < dp[l][r]) {
        dp[l][r] = f(l, r - 1, dp, correct, s) + 1;
        correct[l][r] = correct[l][r - 1];
    }

    if((s[l] == '(' && s[r] == ')') || (s[l] == '[' && s[r] == ']') ||
(s[l] == '{' && s[r] == '}')) {
        if(f(l + 1, r - 1, dp, correct, s) < dp[l][r]) {
            dp[l][r] = f(l + 1, r - 1, dp, correct, s);
            correct[l][r] = s[l];
            correct[l][r] += s[r];
            if(l + 1 <= r - 1)
                correct[l][r] = s[l] + correct[l + 1][r - 1] + s[r];
        }
    }

    return dp[l][r];
}

int main() {
    getFirstTime();

    // ---- code starts here ----

    string s;
    fin >> s;

    vector<vector<int>> dp(s.size(), vector<int>(s.size(), -1));
    vector<vector<string>> correct(s.size(), vector<string>(s.size(), ""));

    f(0, s.size() - 1, dp, correct, s);
    fout << correct[0][s.size() - 1];
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
}

```

```

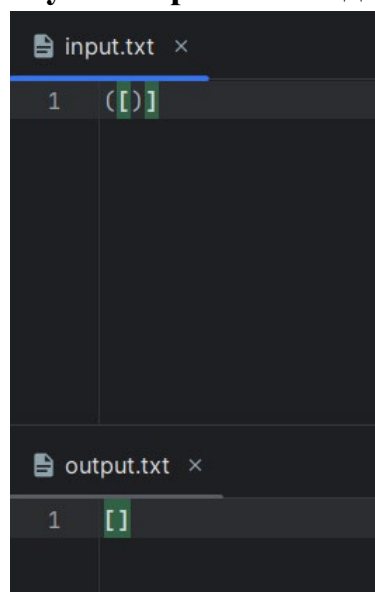
    return 0;
}

```

Текстовое объяснение решения:

Напишем ДП в функции, которая будет принимать промежуток на которому мы ищем минимальное количество скобок, которое необходимо удалить. Переходы будут как сумма строк внутри, как удаление первого или последнего элемента и, если первый и последний символы равны, то просто как минимум внутри этих скобок.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.796875 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.156000 sec	0.796875 MB

Вывод по задаче:

Баянистая задача на ДП.

Задача №16. Продавец (2 балла)

Продавец техники хочет объехать n городов, посетив каждый из них ровно один раз. Помогите ему найти кратчайший путь.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n;
    fin >> n;

    vector <vector <int>> a(n, vector <int>(n));
    for(int i = 0; i < n; ++ i)
```

```

        for(int j = 0; j < n; ++ j)
            fin >> a[i][j];

vector <vector <int> > dp ((1ll << n), vector <int> (n, 1e9));
vector <vector <int> > parent ((1ll << n), vector <int> (n, -1));
for(int i = 0; i < n; ++ i)
    dp[(1 << i)][i] = 0;

for(int i = 0; i < (1ll << n); ++ i) {
    for(int j = 0; j < n; ++ j) {
        if(!(i & (1ll << j)))
            continue;

        for(int k = 0; k < n; ++ k) {
            if((i & (1 << k)))
                continue;

            dp[i | (1 << k)][k] = min(dp[i | (1 << k)][k], dp[i][j] +
a[j][k]);
            if(dp[i | (1 << k)][k] == dp[i][j] + a[j][k])
                parent[i | (1 << k)][k] = j;
        }
    }

    int minimum = 1e9, lastparent = -1, curmask = (1 << n) - 1;
    vector <int> ans;
    for(int i = 0; i < n; ++ i) {
        if(dp[curmask][i] < minimum) {
            minimum = dp[curmask][i];
            lastparent = i;
        }
    }

    while(lastparent != -1) {
        ans.push_back(lastparent);
        int newmask = curmask ^ (1 << lastparent);
        lastparent = parent[curmask][lastparent];
        curmask = newmask;
    }

    fout << minimum << '\n';
    for(int i = ans.size() - 1; i >= 0; -- i)
        fout << " " << ans[i] + 1;
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Основная идея решения состоит в том, чтобы использовать динамическое программирование и подход TSP для поиска кратчайшего пути, который посещает каждый город ровно один раз. Пусть $dp[mask][i]$ — кратчайший путь, который проходит через города, представленные бинарной маской

mask, и заканчивается в городе i . В базовом случае $dp[1][i] = 0$, что означает, что кратчайший путь, заканчивающийся в городе I в самом начале равен 0.

Результат работы кода на примерах из текста задачи:

```

input.txt ×
1 5
2 0 183 163 173 181
3 183 0 165 172 171
4 163 165 0 189 302
5 173 172 189 0 167
6 181 171 302 167 0

output.txt ×
1 666
2 4 5 2 3 1

```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.252000 sec	24.367188 MB

Вывод по задаче:

Легчайшая задача на перебор по маске с ДП.

Задача №17. Ход конем (2.5 балла)

Шахматная ассоциация решила оснастить всех своих сотрудников такими телефонными номерами, которые бы набирались на кнопочном телефоне ходом коня. Например, ходом коня набирается телефон 340-49-27. При этом телефонный номер не может начинаться ни с цифры 0, ни с цифры 8.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----
```

```

const int mod = 1e9;

int n;
fin >> n;

vector<int64_t> dp (10, 1);
dp[8] = dp[0] = 0;

for(int i = 1; i < n; ++ i) {
    vector<int64_t> new_dp (10, 0);
    new_dp[0] = (dp[4] + dp[6]) % mod;
    new_dp[1] = (dp[6] + dp[8]) % mod;
    new_dp[2] = (dp[7] + dp[9]) % mod;
    new_dp[3] = (dp[4] + dp[8]) % mod;
    new_dp[4] = (dp[0] + dp[3] + dp[9]) % mod;
    new_dp[6] = (dp[0] + dp[1] + dp[7]) % mod;
    new_dp[7] = (dp[2] + dp[6]) % mod;
    new_dp[8] = (dp[1] + dp[3]) % mod;
    new_dp[9] = (dp[2] + dp[4]) % mod;

    dp = new_dp;
}

int64_t ans = 0;
for(int i = 0; i < 10; ++ i)
    ans = (ans + dp[i]) % mod;

fout << ans << '\n';

// ---- code ends here ----

printTimeUse();
printMemoryUse();

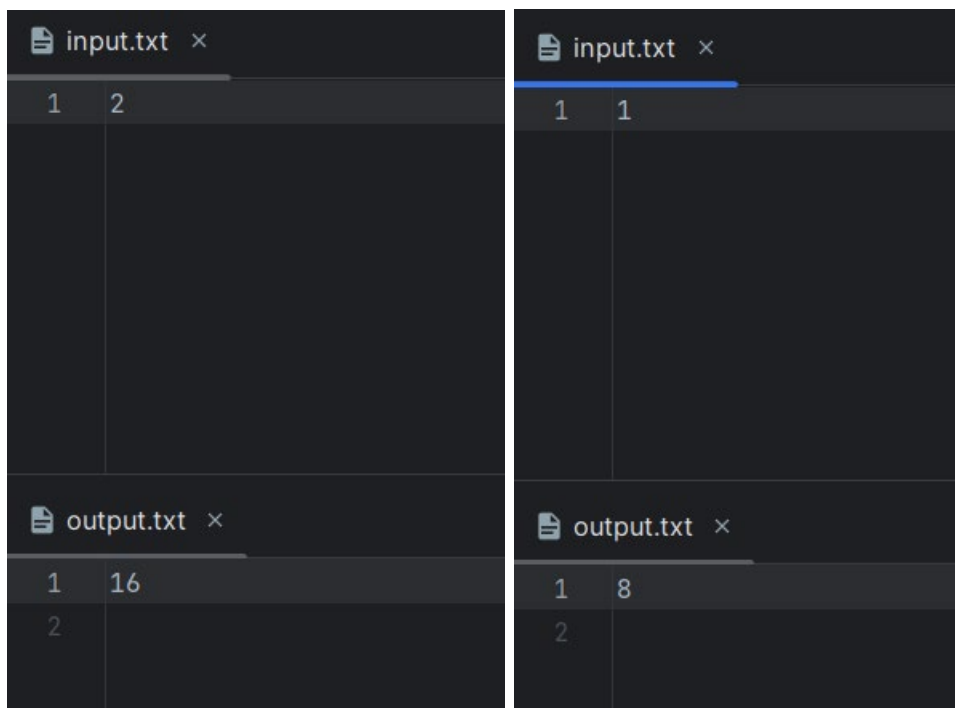
fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Заведем массив `dp[i]`, в котором будем хранить количество номеров, заканчивающихся цифрой `i`, сделаем переходы и посчитаем по модулю.

Результат работы кода на примерах из текста задачи:



Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.001000 sec	0.789062 MB

Вывод по задаче:

Очень простая задача на ДП

Задача №18. Кафе (2.5 балла)

Около университета недавно открылось новое кафе, в котором действует следующая система скидок: при каждой покупке более чем на 100 рублей покупатель получает купон, дающий право на один бесплатный обед (при покупке на сумму 100 рублей и меньше такой купон покупатель не получает). Однажды вам на глаза попался прейскурант на ближайшие n дней. Внимательно его изучив, вы решили, что будете обедать в этом кафе все n дней, причем каждый день вы будете покупать в кафе ровно один обед. Однако стипендия у вас небольшая, и поэтому вы хотите по максимуму использовать предоставляемую систему скидок так, чтобы ваши суммарные затраты были минимальны. Требуется найти минимально возможную суммарную стоимость обедов и номера дней, в которые вам следует воспользоваться купонами.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
```

```

    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n;
    int ans, k1, k2;

    fin >> n;

    vector<int> a(n + 1);
    vector<vector<int>> > b(n + 1, vector<int>(n + 1, -1));
    vector<vector<int>> > p(n + 1, vector<int>(n + 1, -1));
    vector<int> r(n + 1);

    for (int i = 1; i <= n; i++) {
        fin >> a[i];
    }

    b[0][0] = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j <= i; j++) {
            if (b[i][j] != -1) {
                if (j > 0) {
                    if (b[i + 1][j - 1] > b[i][j] || b[i + 1][j - 1] == -1)
{
                        b[i + 1][j - 1] = b[i][j];
                        p[i + 1][j - 1] = j;
                    }
                }

                int k = (a[i + 1] > 100) ? 1 : 0;

                if (b[i + 1][j + k] > b[i][j] + a[i + 1] || b[i + 1][j + k]
== -1) {
                    b[i + 1][j + k] = b[i][j] + a[i + 1];
                    p[i + 1][j + k] = j;
                }
            }
        }
    }

    ans = 1e9;
    for (int i = 0; i <= n; i++) {
        if (b[n][i] != -1 && b[n][i] <= ans) {
            ans = b[n][i];
            k1 = i;
        }
    }

    int j = k1;
    k2 = 0;
    for (int i = n; i >= 1; i--) {
        if (j + 1 == p[i][j]) {

```

```

        k2++;
        r[k2] = i;
    }

    j = p[i][j];
}

fout << ans << '\n' << k1 << " " << k2 << endl;
for (int i = k2; i >= 1; i--) {
    fout << r[i] << endl;
}

// ---- code ends here ----

printTimeUse();
printMemoryUse();

fin.close();
fout.close();
return 0;
}

```

Текстовое объяснение решения:

Заведем массив *b* для ДП, в котором будем осуществлять переходы, а также массив *p* – массив предков, *r* – массив дней, где мы едим на халяву.

input.txt	
1	5
2	110
3	40
4	120
5	110
6	60

output.txt	
2	0 2
3	3
4	5
5	

input.txt	
1	3
2	110
3	110
4	110

output.txt	
1	220
2	1 1
3	2
4	

Проверка задачи на (openedu, астр и тд при наличии в задаче):

№1 - №2								
ID	Дата	Автор	Задача	Язык	Результат	Тест	Время	Память
18819717	15.02.2023 14:52:34	Матвей Бунос	0247	C++	Accepted		0,03	516 Кб

	Время выполнения	Затраты памяти
Нижняя граница	0.000000 sec	0.800781 MB

диапазона значений входных данных из текста задачи		
Пример из задачи	0.000000 sec	0.792969 MB
Пример из задачи	0.000000 sec	0.785156 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.003000 sec	0.800781 MB

Вывод по задаче:

Вот это уже хорошая задачка

Задача №19. Произведение матриц (3 балла)

В произведении последовательности матриц полностью расставлены скобки, если выполняется один из следующих пунктов:

- Произведение состоит из одной матрицы.
- Оно является заключенным в скобки произведением двух произведений с полностью расставленными скобками.

Полная расстановка скобок называется оптимальной, если количество операций, требуемых для вычисления произведения, минимально.

Требуется найти оптимальную расстановку скобок в произведении последовательности матриц.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----
```

```

void print_sequence(vector<vector<int>>& split, int i, int j) {
    if (i == j) {
        fout << "A";
    } else {
        fout << "(";
        print_sequence(split, i, split[i][j]);
        print_sequence(split, split[i][j]+1, j);
        fout << ")";
    }
}

int main() {
    getFirstTime();

    // ---- code starts here ----

    int n;
    fin >> n;

    int tmp;
    vector<int> dim(n + 1);
    vector<vector<int>> split(n+1, vector<int>(n+1, 0));

    for (int i = 0; i < n; i++) {
        fin >> dim[i] >> tmp;
    }

    dim[n] = tmp;

    vector<vector<int>> dp(n + 1, vector<int>(n + 1, 0));

    for (int len = 2; len <= n; len++) {
        for (int i = 1; i <= n - len + 1; i++) {
            int j = i + len - 1;
            dp[i][j] = INT_MAX;
            for (int k = i; k < j; k++) {
                int cost = dp[i][k] + dp[k+1][j] + dim[i-1]*dim[k]*dim[j];
                if (cost < dp[i][j]) {
                    dp[i][j] = cost;
                    split[i][j] = k;
                }
            }
        }
    }

    fout << dp[1][n] << endl;
    print_sequence(split, 1, n);

    // ---- code ends here ----

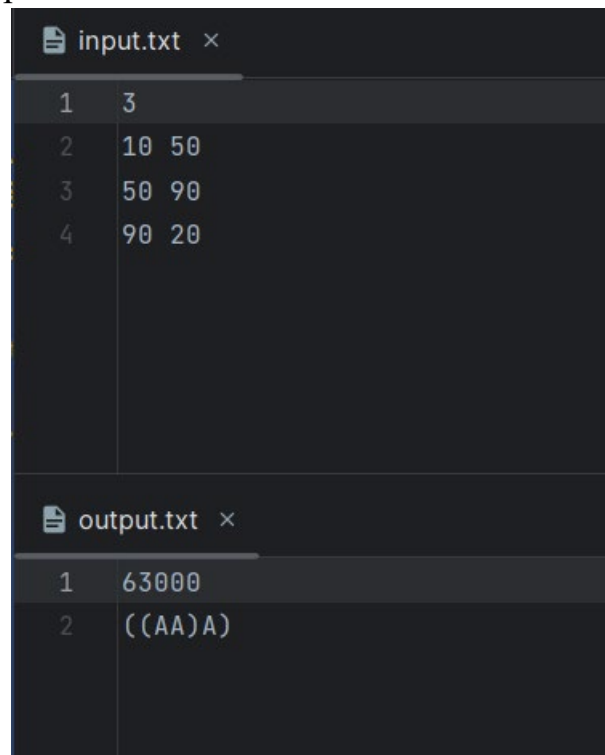
    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Заведем массив `dp[i][j]`, в котором будем хранить минимальное количество операций, чтобы перемножить матрицы с индексами от i до j , после чего просто будем использовать простейший переход, чтобы найти оптимальную. Также будем хранить массив `split`, который и будет определять последовательность.



```

input.txt x
1 3
2 10 50
3 50 90
4 90 20

output.txt x
1 63000
2 ((AA)A)
  
```

Проверка задачи на (openedu, астр и тд при наличии в задаче):

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.792969 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.005000 sec	0.816406 MB

Вывод по задаче:

Классика

Задача №20. Почти палиндром (3 балла)

Слово называется палиндромом, если его первая буква совпадает с последней, вторая – с предпоследней и т.д. Например: «abba», «madam», «х».

Для заданного числа K слово называется почти палиндромом, если в нем можно изменить не более K любых букв так, чтобы получился палиндром. Например, при $K = 2$ слова «reactor», «kolobok», «madam» являются почти палиндромами (подчеркнуты буквы, заменой которых можно получить палиндром).

Подсловом данного слова являются все слова, получающиеся путем вычеркивания из данного нескольких (возможно, одной или нуля) первых букв и нескольких последних. Например, подсловами слова «cat» являются слова «с», «а», «t», «са», «at» и само слово «cat» (а «ct» подсловом слова «cat» не является).

Требуется для данного числа K определить, сколько подслов данного слова S являются почти палиндромами.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
```



```

        start = clock();
    }

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

const int MAXN = 5005;

char s[MAXN];

int main() {
    getFirstTime();

    // ---- code starts here -----
    int n;
    fin >> n;

    int K;
    fin >> K >> s;
    int ans = 0;

    for (int i = 0; i < n; i++) {
        int l = i, r = i;
        int diff = 0;
        while (l >= 0 && r < n && diff <= K) {
            if (s[l] != s[r]) diff++;
            if (diff > K) break;
            ans++;
            l--, r++;
        }
        l = i, r = i + 1, diff = 0;

        while (l >= 0 && r < n && diff <= K) {
            if (s[l] != s[r]) diff++;
            if (diff > K) break;
            ans++;
            l--, r++;
        }
    }

    fout << ans << endl;
    // ---- code ends here -----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

В решении используется алгоритм, основанный на расширении вокруг центров. Для каждого центра палиндрома мы расширяем его в обе стороны,

чтобы найти самый длинный палиндром с этим центром. Считаем все найденные палиндромы и добавляем их к окончательному ответу.

Чтобы реализовать этот алгоритм, мы перебираем все возможные центры палиндромов в строке. Для каждого центра мы расширяем его, чтобы найти палиндромы с нечетной и четной длиной.

input.txt ×

13 3

2aaa

output.txt ×

16

2

input.txt ×

15 1

2abcde

output.txt ×

112

2

Проверка задачи на (openedu, астр и тд при наличии в задаче):

курсы] [олимпиады]

Учащийся: Матвей Бунос Выход

ИСХОДНИК РЕШЕНИЯ №18847274 ЗАДАЧИ №268

Вернуться к задаче] [Редактировать решение]

```

1  #include <iostream>
2  #include <cstring>
3
4  using namespace std;
5
6  const int MAXN = 5005;
7
8  char s[MAXN];
9
10 int main() {
11     int n;
12     cin >> n;
13     int K;
14     cin >> K >> s;
15     int ans = 0;
16     for (int i = 0; i < n; i++) {
17         int l = i, r = i;
18         int diff = 0;
19         while (l >= 0 && r < n && diff <= K) {
20             if (s[l] != s[r]) diff++;
21             if (diff > K) break;
22             ans++;
23             l--, r++;
24         }
25         l = i, r = i+1, diff = 0;
26         while (l >= 0 && r < n && diff <= K) {
27             if (s[l] != s[r]) diff++;
28             if (diff > K) break;
29             ans++;
30             l--, r++;
31         }
32     }
33     cout << ans << endl;
34     return 0;
35 }
```

Размер кода: 369

Тест	Результат	Время	Память
1	Accepted	0,03	424 Кб
2	Accepted	0,015	420 Кб
3	Accepted	0,03	424 Кб
4	Accepted	0,015	424 Кб
5	Accepted	0,015	428 Кб
6	Accepted	0,015	420 Кб
7	Accepted	0,015	428 Кб
8	Accepted	0,015	420 Кб
9	Accepted	0,015	416 Кб
10	Accepted	0,015	420 Кб
11	Accepted	0,062	424 Кб
12	Accepted	0,062	420 Кб
13	Accepted	0,03	416 Кб
14	Accepted	0,03	424 Кб
15	Accepted	0,03	424 Кб
16	Accepted	0,015	420 Кб
17	Accepted	0,03	424 Кб
18	Accepted	0,015	420 Кб
19	Accepted	0,015	424 Кб
20	Accepted	0,03	424 Кб

66

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.800781 MB
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.789062 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.062000 sec	0.816406 MB

Вывод по задаче:

Хорошая задача, жаль, что не на ДП.

Задача №21. Игра в дурака (3 балла)

Петя очень любит программировать. Недавно он решил реализовать популярную карточную игру «Дурак». Но у Пети пока маловато опыта, ему срочно нужна Ваша помощь.

Как известно, в «Дурака» играют колодой из 36 карт. В Петиней программе каждая карта представляется в виде строки из двух символов, где первый символ означает ранг ('6', '7', '8', '9', 'T', 'J', 'Q', 'K', 'A') карты, а второй символ означает масть ('S', 'C', 'D', 'H'). Ранги перечислены в порядке возрастания старшинства.

Пете необходимо решить следующую задачу: сможет ли игрок, обладая набором из N карт, отбить M карт, которыми под него сделан ход? Для того чтобы отбиться, игроку нужно покрыть каждую из карт, которыми под него сделан ход, картой из своей колоды. Карту можно покрыть либо старшей картой той же масти, либо картой козырной масти. Если кроющаяся карта сама является козырной, то её можно покрыть только старшим козырем. Одной картой можно покрыть только одну карту.

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}

void getFirstTime() {
```

```

    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int ReadRank() {
    const string kRanks = "6789TJQKA";
    char rank;
    fin >> rank;
    return static_cast<int>(kRanks.find(rank));
}

int ReadSuit() {
    const string kSuits = "SCDH";
    char suit;
    fin >> suit;
    return static_cast<int>(kSuits.find(suit));
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    int num_cards, num_tricks;
    fin >> num_cards >> num_tricks;
    int trump_suit = ReadSuit();
    vector<vector<bool>> available_ranks(9, vector<bool>(4, false));

    for (int i = 0; i < num_cards; ++i) {
        int rank = ReadRank();
        int suit = ReadSuit();
        available_ranks[rank][suit] = true;
    }

    for (int i = 0; i < num_tricks; ++i) {
        int rank = ReadRank();
        int suit = ReadSuit();
        bool found = false;

        for (int r = rank + 1; r <= 8; ++r) {
            if (available_ranks[r][suit]) {
                available_ranks[r][suit] = false;
                found = true;
                break;
            }
        }

        if (!found && suit != trump_suit) {
            for (int r = 0; r <= 8; ++r) {
                if (available_ranks[r][trump_suit]) {
                    available_ranks[r][trump_suit] = false;
                    found = true;
                    break;
                }
            }
        }
    }
}

```

```

    }

    if (!found) {
        fout << "NO";
        return 0;
    }

    fout << "YES";
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

    fin.close();
    fout.close();
    return 0;
}

```

Текстовое объяснение решения:

Мы смотрим карты, которые нужно побить, и проверяем, может ли игрок побить каждую карту. Для каждой карты программа проверяет, есть ли более старшая карта той же масти, которую может сыграть игрок. Если есть, программа помечает эту карту как недоступную и переходит к следующей карте. Если старшей карты той же масти нет, программа проверяет, является ли козырная масть мастью карты. Если его нет, программа проверяет, есть ли козырь, которым может играть игрок. Если есть, программа помечает эту карту как недоступную и переходит к следующей карте. Если карты, которую можно сыграть, нет, программа выводит «НЕТ» и завершает работу.

input.txt	
1	6 2 C
2	KD KC AD 7C AH 9C
3	6D 6C

output.txt	
1	YES

input.txt	
1	4 1 D
2	9S KC AH 7D
3	8D

output.txt	
1	NO

Проверка задачи на (openedu, астр и тд при наличии в задаче):

← ↻ 📄 астр.ru

Школа программиста

🔍 🛡️ 📄 ⬇

```

34 for (int i = 0; i < num_cards; ++i) {
35     int rank = ReadRank();
36     int suit = ReadSuit();
37     available_ranks[rank][suit] = true;
38 }
39
40 for (int i = 0; i < num_tricks; ++i) {
41     int rank = ReadRank();
42     int suit = ReadSuit();
43     bool found = false;
44
45     for (int r = rank + 1; r <= 8; ++r) {
46         if (available_ranks[r][suit]) {
47             found = true;
48             break;
49         }
50     }
51
52     if (!found && suit != trump_suit) {
53         for (int r = 0; r <= 8; ++r) {
54             if (available_ranks[r][trump_suit]) {
55                 available_ranks[r][suit] = false;
56                 found = true;
57                 break;
58             }
59         }
60     }
61
62     if (!found) {
63         cout << "NO";
64         return 0;
65     }
66 }
67
68 cout << "YES";
69 // ---- code ends here ----
70 return 0;
71 }
72

```

Размер кода: 1074

Посылки решений:

ID	Дата	Язык	Результат	Тест	Время	Память
18847295	19.02.2023 4:22:03	C++	Accepted		0.03	416 Кб

24	Accepted	0,015	404 Кб
25	Accepted	0,015	408 Кб
26	Accepted	0,015	404 Кб
27	Accepted	0,015	404 Кб
28	Accepted	0,015	408 Кб
29	Accepted	0,015	400 Кб
30	Accepted	0,03	404 Кб
31	Accepted	0,015	412 Кб
32	Accepted	0,015	408 Кб
33	Accepted	0,015	404 Кб
34	Accepted	0,015	408 Кб
35	Accepted	0,015	400 Кб
36	Accepted	0,015	400 Кб
37	Accepted	0,015	408 Кб
38	Accepted	0,03	408 Кб
39	Accepted	0,015	404 Кб
40	Accepted	0,015	400 Кб
41	Accepted	0,015	404 Кб
42	Accepted	0,015	408 Кб
43	Accepted	0,015	400 Кб
44	Accepted	0,015	404 Кб
45	Accepted	0,015	412 Кб
46	Accepted	0,015	400 Кб
47	Accepted	0,015	400 Кб
48	Accepted	0,015	416 Кб
49	Accepted	0,015	404 Кб
50	Accepted	0,015	408 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.000000 sec	0.789062 МВ
Пример из задачи	0.000000 sec	0.789062 МВ
Пример из задачи	0.000000 sec	0.800781 МВ
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.789062 МВ

Вывод по задаче:

Тоже хорошая задача, на реализацию чисто

Задача №22. Симпатичные узоры (4 балла)

Компания BrokenTiles планирует заняться выкладыванием во дворах у состоятельных клиентов узор из черных и белых плиток, каждая из которых имеет размер 1×1 метр. Известно, что дворы всех состоятельных людей имеют наиболее модную на сегодня форму прямоугольника $M \times N$ метров.

Однако при составлении финансового плана у директора этой организации появилось целых две серьезных проблемы: во первых, каждый новый клиент очевидно захочет, чтобы узор, выложенный у него во дворе, отличался от узоров всех остальных клиентов этой фирмы, а во вторых, этот узор должен быть симпатичным. Как показало исследование, узор является симпатичным, если в нем нигде не встречается квадрата 2×2 метра, полностью покрытого плитками одного цвета.

Для составления финансового плана директору необходимо узнать, сколько клиентов он сможет обслужить, прежде чем симпатичные узоры данного размера закончатся. Помогите ему!

Листинг кода:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <vector>
#include <math.h>
#include "windows.h"
#include "psapi.h"
#include <time.h>
#include <stdlib.h>
#include <algorithm>
#include <type_traits>

using namespace std;

//-----

ifstream fin("input.txt");
ofstream fout("output.txt");

clock_t start;

void printMemoryUse() {
    PROCESS_MEMORY_COUNTERS_EX pmc;
    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS_MEMORY_COUNTERS *)
    &pmc, sizeof(pmc));
    SIZE_T virtualMemUsedByMe = pmc.PrivateUsage;

    cerr << fixed << setprecision(6);
    cerr << "Memory used: " << double(virtualMemUsedByMe) / (1024. * 1024)
    << " MB\n";
}
```



```

void getFirstTime() {
    start = clock();
}

void printTimeUse() {
    cerr << fixed << setprecision(6);
    cerr << "Time used: " << (double) (clock() - start) / CLOCKS_PER_SEC <<
    " sec\n";
}

//-----

int isMaskGood(int curMask, int prevMask, int n) {
    for (int i = 1; i < n; ++ i) {
        int white = (((1 << (i - 1)) & (curMask)) > 0) +
                    (((1 << (i)) & (curMask)) > 0) +
                    (((1 << (i - 1)) & (prevMask)) > 0) +
                    (((1 << (i)) & (prevMask)) > 0);

        if (white == 4 || white == 0)
            return false;
    }
    return true;
}

int main() {
    getFirstTime();

    // ---- code starts here ----
    int n, m;
    fin >> n >> m;
    if (n > m)
        swap(n, m);

    int full = (1 << n);
    vector<vector<int64_t> > dp(32, vector<int64_t>(full, 0));
    for (int mask = 0; mask < full; mask++)
        dp[1][mask] = 1;

    for (int i = 2; i <= m; ++ i) {
        for (int curMask = 0; curMask < full; curMask++) {
            for (int prevMask = 0; prevMask < full; prevMask++) {
                if (isMaskGood(curMask, prevMask, n)) {
                    dp[i][curMask] += dp[i - 1][prevMask];
                }
            }
        }
    }

    int64_t ans = 0;
    for (int mask = 0; mask < full; mask++)
        ans += dp[m][mask];

    fout << ans;
    // ---- code ends here ----

    printTimeUse();
    printMemoryUse();

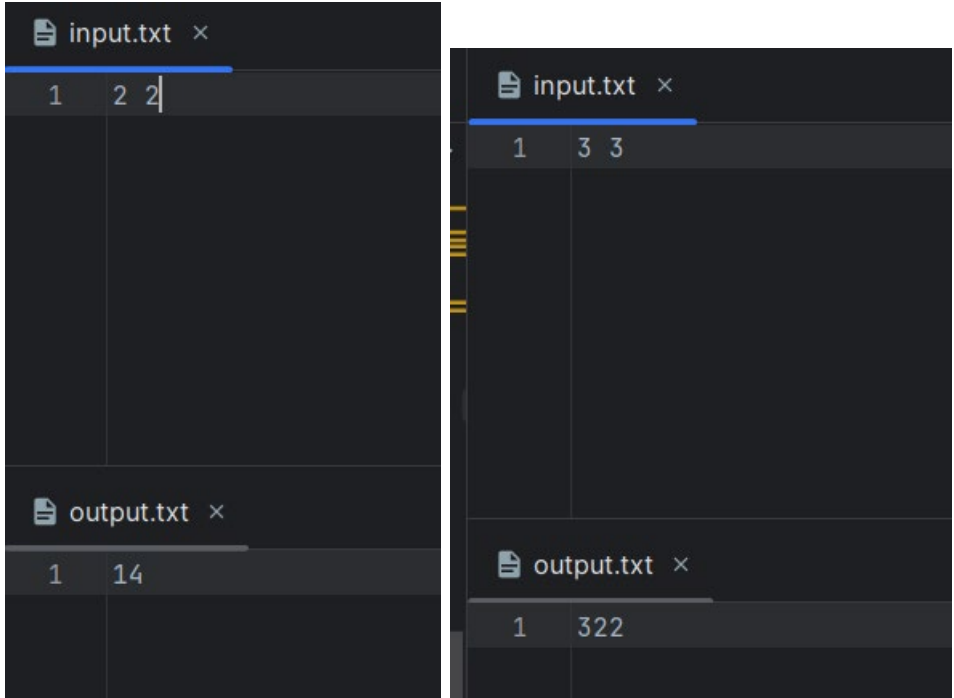
    fin.close();
}

```

```
fout.close();
return 0;
}
```

Текстовое объяснение решения:

Используем ДП по маскам, перебираем каждую строку и маски для предыдущих, если получается собрать по данному условию (не существует квадрата 2x2 из одного цвета), значит добавляем в ответ.



Проверка задачи на (openedu, астр и тд при наличии в задаче):

Вернуться к задаче [Редактировать решение]

```
1 #include <iostream>
2 #include <fstream>
3 #include <iomanip>
4 #include <vector>
5 #include <math.h>
6 #include <time.h>
7 #include <stdlib.h>
8 #include <algorithm>
9 #include <type_traits>
10
11 using namespace std;
12
13 int isMaskGood(int curMask, int prevMask, int n) {
14     for (int i = 1; i < n; ++ i) {
15         int white = (((1 << (i - 1)) & (curMask)) > 0) +
16                     (((1 << i) & (curMask)) > 0) +
17                     (((1 << (i - 1)) & (prevMask)) > 0) +
18                     (((1 << i) & (prevMask)) > 0);
19
20         if (white == 4 || white == 0)
21             return false;
22     }
23     return true;
24 }
25
26 int main() {
27     // ---- code starts here ----
28     int n, m;
29     cin >> n >> m;
30     if (n > m)
31         swap(n, m);
32
33     int full = (1 << n);
34     vector<vector<int64_t> > dp(32, vector<int64_t>(full, 0));
35     for (int mask = 0; mask < full; mask++)
36         dp[1][mask] = 1;
37
38     for (int i = 2; i <= m; ++ i) {
39         for (int curMask = 0; curMask < full; curMask++) {
```

Тест	Результат	Время	Память
1	Accepted	0,015	412 Кб
2	Accepted	0,015	408 Кб
3	Accepted	0,015	404 Кб
4	Accepted	0,015	412 Кб
5	Accepted	0,015	400 Кб
6	Accepted	0,015	404 Кб
7	Accepted	0,015	408 Кб
8	Accepted	0,015	408 Кб
9	Accepted	0,015	420 Кб
10	Accepted	0,015	416 Кб

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений	0.000000 sec	0.800781 MB

ВХОДНЫХ ДАННЫХ ИЗ ТЕКСТА ЗАДАЧИ		
Пример из задачи	0.000000 sec	0.789062 MB
Пример из задачи	0.000000 sec	0.800781 MB
Верхняя граница диапазона значений входных данных из текста задачи	0.015000 sec	0.804688 MB

Вывод по задаче:

Очень хорошая задача на ДП по маскам.

Вывод

Замечательная лабораторная работа, мне понравилось писать 22 задачи.