­­САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №5

по курсу «Алгоритмы и структуры данных»

Тема: Жадные алгоритмы. Динамическое программирование №2

Вариант №22

Выполнил:

Скирляк Я. Ю.

К3139

Проверила:

Артамонова В.Е.

Санкт-Петербург

2022 г.

# Содержание отчета

[Содержание отчета 2](#_Toc128686156)

[Задачи по варианту 3](#_Toc128686157)

[Задача №1. Максимальная стоимость добычи (0.5 балла) 3](#_Toc128686158)

[Задача №7. Проблема сапожника (0.5 балла) 7](#_Toc128686159)

[Задача №9. Распечатка (1 балл) 10](#_Toc128686160)

[Задача №17. Ход конем (2.5 балла) 15](#_Toc128686161)

[Задача №21. Игра в дурака (3 балла) 20](#_Toc128686162)

[Дополнительные задачи 23](#_Toc128686163)

[Задача №2. Заправки (0.5 балла) 23](#_Toc128686164)

[Задача №3. Максимальный доход от рекламы (0.5 балла) 26](#_Toc128686165)

[Задача №4. Сбор подписей (0.5 балла) 29](#_Toc128686166)

[Задача №5. Максимальное количество призов (0.5 балла) 34](#_Toc128686167)

[Задача №6. Максимальная зарплата (0.5 балла) 38](#_Toc128686168)

[Задача №8. Расписание лекций (1 балл) 42](#_Toc128686169)

[Задача №10. Яблоки (1 балл) 47](#_Toc128686170)

[Задача №11. Максимальное количество золота (1 балл) 51](#_Toc128686171)

[Задача №12. Последовательность (1 балл) 55](#_Toc128686172)

# Задачи по варианту

## Задача №1. Максимальная стоимость добычи (0.5 балла)

***Текст задачи***

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.

• Формат ввода / входного файла (input.txt). В первой строке входных дан-

ных задано целое число n - количество предметов, и W - вместимость сумки.

Следующие n строк определяют значения веса и стоимости предметов. В

i-ой строке содержатся целые числа pi и wi – стоимость и вес i-го предмета,

соответственно.

• Ограничения на входные данные. 1 ≤ n ≤ 10^3, 0 ≤ W ≤ 2 · 10^6,

0 ≤ pi ≤ 2 · 10^6, 0 ≤ wi ≤ 2 · 10^6 для всех 1 ≤ i ≤ n. Все числа - целые.

• Формат вывода / выходного файла (output.txt). Выведите максимальное значение стоимости долей предметов, которые помещаются в сумку. Абсолютная погрешность между ответом вашей программы и оптимальным

значением должно быть не более 10−3

. Для этого выведите свой ответ как

минимум с четырьмя знаками после запятой (иначе ваш ответ, хотя и будет

рассчитан правильно, может оказаться неверным из-за проблем с округлением).

• Ограничение по времени. 2 сек.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <iomanip>

#include <mach/task.h>

#include <mach/mach\_init.h>

using namespace std;

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getMemoryUsage() {

struct task\_basic\_info t\_info;

mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

&t\_info\_count))

{

printf("Error!");

}

else

{

printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

}

}

void getFirstTime() {

start = clock();

}

void printTimeUse() {

printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

int main() {

getFirstTime();

int n, W;

fin >> n >> W;

int\* p = new int[n];

int\* w = new int[n];

double\* pw = new double[n];

for (int i = 0; i < n; i++) {

fin >> p[i] >> w[i];

}

for (int i = 0; i < n; i++) {

pw[i] = (double)p[i] / w[i];

}

int maxIndex = 0;

double maxRatio = 0;

double sum = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (pw[j] > maxRatio) {

maxRatio = pw[j];

maxIndex = j;

}

}

if (w[maxIndex] < W) {

sum += p[maxIndex];

W -= w[maxIndex];

}

else {

sum += (double)W \* pw[maxIndex];

W = 0;

}

pw[maxIndex] = 0;

maxRatio = 0;

}

fout << sum;

printTimeUse();

getMemoryUsage();

return 0;

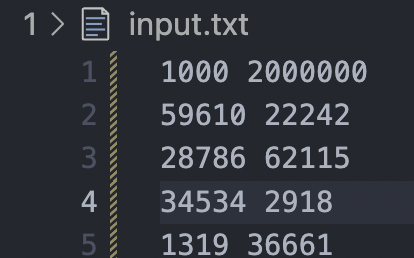
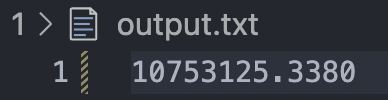
}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значение из файла
5. Создаем массив, в который мы занесем соотношение цены и веса
6. Дальше мы будем искать максимальное соотношение цены и веса и «класть» в рюкзак

***Результат работы кода на максимальных значениях:(скрины input output файлов)***

Input: Output:

|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи | 0.0000330s | 1.265625 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0037680 | 1.281250 |

**Вывод по задаче:** Удалось решить эту задачу

## Задача №7. Проблема сапожника (0.5 балла)

***Текст задачи***

• Постановка задачи. 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.

• Формат ввода / входного файла (input.txt). В первой строке вводятся

натуральные числа K и n. Затем во второй строке идет n натуральных

чисел t1, ...tn - количество минут, которые требуются, чтобы починить i-й

сапог.

• Ограничения на входные данные. 1 ≤ K ≤ 1000, 1 ≤ n ≤ 500, 1 ≤ ti ≤

100 для всех 1 ≤ i ≤ n

• Формат вывода / выходного файла (output.txt). Выведите одно число –

максимальное количество сапог, которые можно починить за один рабочий

день.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <time.h>

#include <algorithm>

#include <vector>

#include <iomanip>

#include <sys/time.h>

#include <mach/task.h>

#include <mach/mach\_init.h>

using namespace std;

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getMemoryUsage() {

struct task\_basic\_info t\_info;

mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

&t\_info\_count))

{

printf("Error!");

}

else

{

printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

}

}

void getFirstTime() {

start = clock();

}

void printTimeUse() {

printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

int main() {

getFirstTime();

int K, n;

fin >> K >> n;

vector<int> shoes;

for (int i = 0; i < n; i++) {

int k;

fin >> k;

shoes.push\_back(k);

}

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

int sum = 0;

int count = 0;

for (int i = 0; i < n; i++) {

if (sum + shoes[i] < K) {

sum += shoes[i];

count += 1;

}

}

fout << count;

printTimeUse();

getMemoryUsage();

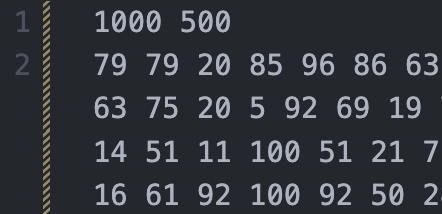
}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значение из файла
5. Получаем количество рабочих часов и количество обуви
6. В вектор записываем все значения из второй строки и сортируем этот вектор
7. Пока мы можем суммировать значения – суммируем

***Результат работы кода на максимальных значениях:(скрины input output файлов)***

Input: Output:

|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000310s | 1.265625 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0000910s | 1.265625 |

***Вывод по задаче:*** Удалось решить эту задачу.

## Задача №9. Распечатка (1 балл)

***Текст задачи***

• Постановка задачи. 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.

• Формат ввода / входного файла (input.txt). На вход программе сначала по-

дается число N – количество листов в заказе. В следующих 7 строках ввода

находятся натуральные числа A1, A2, A3, A4, A5, A6, A7 соответственно.

• Ограничения на входные данные. 1 ≤ N ≤ 2 × 10^9, 1 ≤ Ai ≤ 10^6

• Формат вывода / выходного файла (output.txt). Выведите одно число –

минимальную сумму денег в рублях, которая нужна для выполнения заказа.

Гарантируется, что правильный ответ не будет превышать 2 × 10^9.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <vector>

#include <cmath>

#include <algorithm>

using namespace std;

// ---------- macos ----------

#include <mach/task.h>

#include <mach/mach\_init.h>

void printMemoryUsage() {

struct task\_basic\_info t\_info;

mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

&t\_info\_count))

{

printf("Error!");

}

else

{

printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

}

}

// ---------- windows ----------

// #include "windows.h"

// #include "psapi.h"

// void printMemoryUse() {

// PROCESS\_MEMORY\_COUNTERS\_EX pmc;

// GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

// SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

// cout << fixed << setprecision(6);

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

// }

// ---------- linux ----------

// #include <sys/resource.h>

// #include <sys/time.h>

// void printMemoryUsage() {

// struct rusage usage;

// getrusage(RUSAGE\_SELF, &usage);

// cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

start = clock();

}

void printTimeUse() {

printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

// ---------- code ----------

int main() {

getFirstTime();

int minCost = pow(10, 10);

int N;

fin >> N;

int result = 0;

vector<int> prices(7);

for (int i = 0; i < 7; i++) {

fin >> prices[i];

}

for (int i = 6; i >= 0; i--) {

int N2 = N;

if (N != 0) {

int k = N / pow(10, i);

if (k > 0) {

result += k \* prices[i];

N -= k \* pow(10, i);

}

if (k == 0) {

N2 -= pow(10, i);

if (N2 <= 0) {

minCost = min(minCost, prices[i]);

}

}

} else {

break;

}

}

fout << min(result, minCost);

fin.close();

fout.close();

printTimeUse();

printMemoryUsage();

return 0;

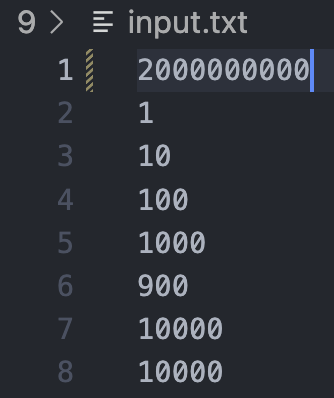
}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значение из файла
5. Получаем все цены, за количество бумаги
6. Вычисляем сколько какой бумаги надо купить, а заодно проверяем нельзя ли купить больше бумаги, но дешевле
7. В ответ выводим минимально возможные затраты

***Результат работы кода на максимальных значениях:(скрины input output файлов)***

Input: Output:

|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000770s | 1.250000 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0001870 | 1.312500 |

***Вывод по задаче:*** Удалось решить эту задачу.

## Задача №17. Ход конем (2.5 балла)

***Текст задачи***

• Постановка задачи. 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.

1 2 3

4 5 6

7 8 9

. 0 .

Напишите программу, определяющую количество телефонных номеров длины N, набираемых ходом коня. Поскольку таких номеров может быть очень много, выведите ответ по модулю 10^9.

• Формат ввода / входного файла (input.txt). Во входном файле записано

одно целое число N.

• Ограничения на входные данные. 1 ≤ N ≤ 1000.

• Формат вывода / выходного файла (output.txt). Выведите в выходной

файл искомое количество телефонных номеров по модулю 10^9.

• Ограничение по времени. 1 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <time.h>

#include <algorithm>

#include <vector>

#include <iomanip>

#include <math.h>

using namespace std;

// ---------- macos ----------

#include <mach/task.h>

#include <mach/mach\_init.h>

void printMemoryUsage() {

struct task\_basic\_info t\_info;

mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

&t\_info\_count))

{

printf("Error!");

}

else

{

printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

}

}

// ---------- windows ----------

// #include "windows.h"

// #include "psapi.h"

// void printMemoryUse() {

// PROCESS\_MEMORY\_COUNTERS\_EX pmc;

// GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

// SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

// cout << fixed << setprecision(6);

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

// }

// ---------- linux ----------

// #include <sys/resource.h>

// #include <sys/time.h>

// void printMemoryUsage() {

// struct rusage usage;

// getrusage(RUSAGE\_SELF, &usage);

// cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

start = clock();

}

void printTimeUse() {

printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

// ---------- code ----------

int phone(int n) {

vector<vector<long long> > dp(10, vector<long long>(n + 1));

int mod = pow(10, 9);

for (int i = 0; i < 10; i++) {

dp[i][1] = 1;

}

for (int j = 2; j <= n; j++) {

for (int i = 0; i < 10; i++) {

switch (i) {

case 0:

dp[0][j] = (dp[4][j - 1] + dp[6][j - 1]) % mod;

break;

case 1:

dp[1][j] = (dp[6][j - 1] + dp[8][j - 1]) % mod;

break;

case 2:

dp[2][j] = (dp[9][j - 1] + dp[7][j - 1]) % mod;

break;

case 3:

dp[3][j] = (dp[8][j - 1] + dp[4][j - 1]) % mod;

break;

case 4:

dp[4][j] = (dp[0][j - 1] + dp[3][j - 1] + dp[9][j - 1]) % mod;

break;

case 6:

dp[6][j] = (dp[0][j - 1] + dp[1][j - 1] + dp[7][j - 1]) % mod;

break;

case 7:

dp[7][j] = (dp[6][j - 1] + dp[2][j - 1]) % mod;

break;

case 8:

dp[8][j] = (dp[1][j - 1] + dp[3][j - 1]) % mod;

break;

case 9:

dp[9][j] = (dp[2][j - 1] + dp[4][j - 1]) % mod;

break;

}

}

}

int sum = 0;

for (int i = 1; i < 10; i++) {

if (i != 8 && i != 0) {

sum = (sum + dp[i][n]) % mod;

}

}

return sum;

}

int main() {

getFirstTime();

int n;

fin >> n;

fout << phone(n) << endl;

printTimeUse();

printMemoryUsage();

return 0;

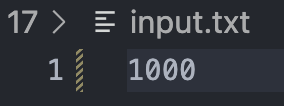
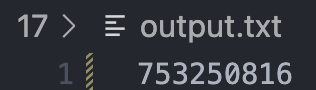
}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значение из файла
5. Создаем двумерный массив и в каждый элемент передаем значение 1, так как номер с длинной 1 можно набрать один раз
6. Далее мы вместо j подставляем количество ходов, которые надо сделать, а вместо i
7. В ответ выводим минимально возможные затраты

***Результат работы кода на максимальных значениях:(скрины input output файлов)***

Input: Output:

|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0001060s | 1.390625 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0002240s | 1.375000 |

***Вывод по задаче:*** Удалось решить эту задачу.

## Задача №21. Игра в дурака (3 балла)

***Текст задачи***

• Постановка задачи. 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 Петиной 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.

• Формат входного файла (input.txt). В первой строке входного файла находятся два натуральных числа N и M, а также символ R, означающий козырную масть. Во второй строке перечислены N карт, находящихся на

руках у игрока. В третьей строке перечислены M карт, которые необходимо

отбить. Все карты отделены друг от друга одним пробелом.

• Ограничения на входные данные. N ≤ 35, M ≤ 4, M ≤ N.

• Формат выходного файла (output.txt). В выходной файл выведите «YES»

в случае, если отбиться можно, либо «NO», если нельзя.

• Ограничение по времени. 1 сек.

• Ограничение по памяти. 16 мб.

***Листинг кода***

from itertools import groupby

from typing import Dict, List, Tuple

cards\_priority = {

"6": 0,

"7": 1,

"8": 2,

"9": 3,

"T": 4,

"J": 5,

"Q": 6,

"K": 7,

"A": 8,

}

def beat(deck: List[Tuple[str, str]], card: Tuple[str, str]) -> Tuple[str, str]:

return min((c for c in deck if cards\_priority[c[0]] > cards\_priority[card[0]]), default=None)

def main(R, cards, sorted\_deck) -> None:

for card in cards:

val, suit = card

if suit != R:

res = beat(sorted\_deck[suit], card)

if res is not None:

sorted\_deck[suit].remove(res)

continue

res = beat(sorted\_deck[R], card) if suit == R else sorted\_deck[R][0] if sorted\_deck[R] else None

if res is not None:

sorted\_deck[R].remove(res)

continue

return("NO")

quit()

return("YES")

if \_\_name\_\_ == '\_\_main\_\_':

with open('input.txt') as f, open('output.txt', 'w') as out:

N, M, R = f.readline().split()

handss = f.readline()

cardss = f.readline()

hands = sorted([(i[0], i[1]) for i in handss.split()], key=lambda x: (x[1] != R, cards\_priority[x[0]]))

cards = sorted([(i[0], i[1]) for i in cardss.split()], key=lambda x: (x[1] != R, cards\_priority[x[0]]))

sorted\_deck = {key: list(items) for key, items in groupby(hands, lambda x: x[1])}

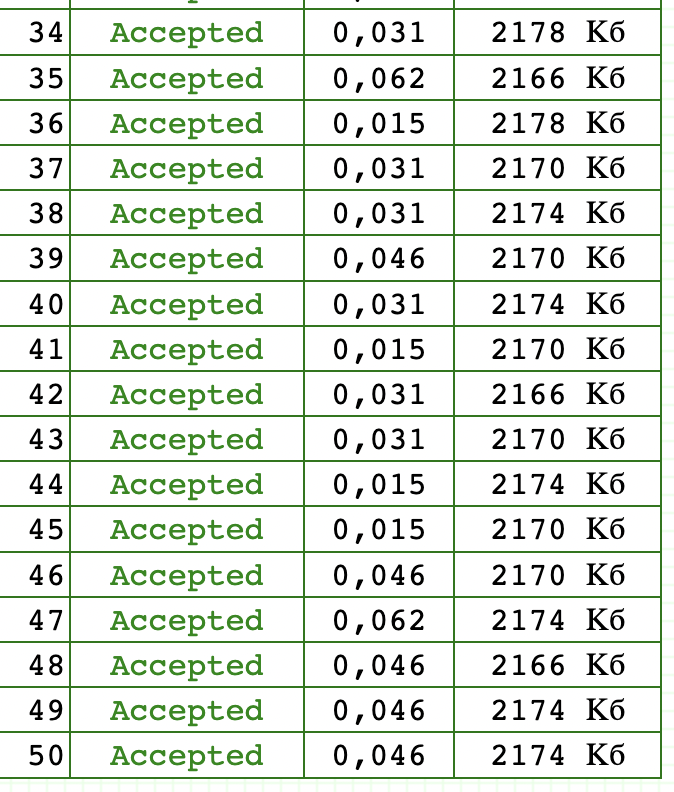
res = main(R, cards, sorted\_deck)

out.write(res)

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Добавляем каждую карту в определенный массив и сразу же сортируем: сначала по масти, чтобы найти козыри, а затем по значение карты
6. Далее запускаем цикл, который проверяет является ли карта козырной, ее значение и затем запускает функцию beat(), которая ищет наименьшую карту, которой можно отбить и удаляет эту карту
7. Если все карты отбиты, то возвращаем «YES», а иначе «NO»

***Результат работы кода на максимальных значениях:(скрины input output файлов)***



***Вывод по задаче:*** Со слезами на глазах удалось решить эту задачу.

# 

# Дополнительные задачи

## Задача №2. Заправки (0.5 балла)

***Текст задачи***

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 stop1, stop2, ..., stopn из вашего родного города. Какое минимальное количество заправок необходимо?

• Формат ввода / входного файла (input.txt). В первой строке содержится

d - целое число. Во второй строке - целое число m. В третьей находится

количество заправок на пути - n. И, наконец, в последней строке - целые

числа через пробел - остановки stop1, stop2, ..., stopn.

• Ограничения на входные данные. 1 ≤ d ≤ 10^5, 1 ≤ m ≤ 400, 1 ≤ n ≤ 300, 1 < stop1 < stop2 < ... < stopn < d

• Формат вывода / выходного файла (output.txt). Предполагая, что расстояние между городами составляет d км, автомобиль может проехать не более m км на полном баке, а заправки есть на расстояниях stop1, stop2, ..., stopn по пути, выведите минимально необходимое количество заправок. Предположим, что машина начинает ехать с полным баком. Если до места назначения добраться невозможно, выведите −1.

• Ограничение по времени. 2 сек.

***Листинг кода***

int main() {

getFirstTime();

int d, m, n;

fin >> d >> m >> n;

vector<int> stops(n);

for (int i = 0; i < n; i++) {

fin >> stops[i];

}

stops.push\_back(d);

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

int currentStop = 0;

int currentFuel = m;

int result = 0;

for (int i = 0; i < stops.size(); i++) {

if (stops[i] - currentStop > currentFuel) {

if (stops[i] - stops[i-1] > m) {

result = -1;

break;

}

currentStop = stops[i-1];

currentFuel = m;

result++;

}

currentFuel -= stops[i] - currentStop;

currentStop = stops[i];

}

fout << result;

printTimeUse();

printMemoryUsage();

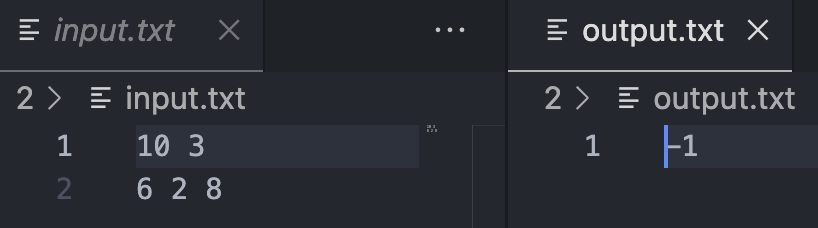
return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Добавляем каждую остановку, а также конечную остановку в вектор и сортируем этот вектор
6. Далее запускаем цикл, который проверяет хватит ли у нас топлива, чтобы добраться до остановки. Если нет, то проверяем хватит ли, если мы заправимся на предыдущей остановке
7. Если мы можем добраться до остановки, то вычитаем из топлива пройденное расстояние и назначаем новую остановку.

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0001130s | 1.343750 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0001240s | 0.0001240s |

***Вывод по задаче:*** Удалось реализовать функции нахождение высоты дерева по заданным параметрам.

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

***Текст задачи***

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.

• Постановка задачи. Даны две последовательности a1, a2, ..., an (ai - при-

быль за клик по i-му объявлению) и b1, b2, ..., bn (bi - среднее количество кликов в день i-го слота), нужно разбить их на n пар (ai, bj ) так, чтобы сумма их произведений была максимальной.

• Формат ввода / входного файла (input.txt). В первой строке содержится

целое число n, во второй - последовательность целых чисел a1, a2, ..., an, в

третьей - последовательность целых чисел b1, b2, ..., bn.

• Ограничения на входные данные. 1 ≤ n ≤ 103, −105 ≤ ai, bi ≤ 105, для всех 1 ≤ i ≤ n.

• Формат вывода / выходного файла (output.txt). Выведите максимальное

значение Pni=1 aici, где c1, c2, ..., cn является перестановкой b1, b2, ..., bn.

• Ограничение по времени. 2 сек.

***Листинг кода***

int main() {

getFirstTime();

int n;

fin >> n;

vector<int> a(n);

vector<int> b(n);

for (int i = 0; i < n; i++) {

int ai;

fin >> ai;

a[i] = ai;

}

for (int i = 0; i < n; i++) {

int bi;

fin >> bi;

b[i] = bi;

}

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

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

int sum = 0;

for (int i = 0; i < n; i++) {

sum += a[i] \* b[i];

}

fout << sum << endl;

printTimeUse();

printMemoryUsage();

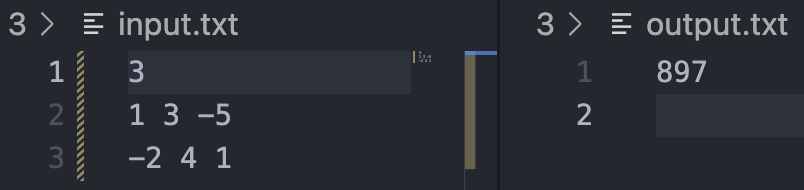
return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Записываем в векторы значения строк и сортируем их
6. Перемножаем все элементы с одинаковыми индексами, суммируем и выводим ответ

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0001030s | 1.312500 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0007490s | 1.281250 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №4. Сбор подписей (0.5 балла)

***Текст задачи***

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.

• Постановка задачи.Дан набор из n отрезков [a0, b0], [a1, b1], ..., [an−1, bn−1] с координатами на прямой, найдите минимальное количество m точек такое, чтобы каждый отрезок содержал хотя бы одну точку. То есть найдите набор целых чисел X минимального размера такой, чтобы для любого отрезка [ai, bi] существовала точка x ∈ X такая, что ai ≤ x ≤ bi.

• Формат ввода / входного файла (input.txt). Первая строка входных данных содержит количество отрезков n. Каждая из следующих n строк содержит два целых числа ai и bi (через пробел), определяющие координаты концов i-го отрезка.

• Ограничения на входные данные. 1 ≤ n ≤ 102, 0 ≤ ai, bi ≤ 109 - целые

для всех 1 ≤ i ≤ n.

• Формат вывода / выходного файла (output.txt). Выведите минимальное

количество m точек в первой строке и целочисленные координаты этих

m точек (через пробел) во второй строке. Вывести точки можно в любом

порядке. Если таких наборов точек несколько, можно вывести любой набор.

(Нетрудно видеть, что всегда существует множество точек минимального

размера, для которых все координаты точек - целые числа.)

• Ограничение по времени. 2 сек.

***Листинг кода***

#include <iostream>

#include <vector>

#include <algorithm>

#include <fstream>

#include <time.h>

#include <iomanip>

#include "Windows.h"

#include "psapi.h"

using namespace std;

void printMemoryUse()

{

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

    SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

    cout << fixed << setprecision(6);

    cout << "Virtual memory used by me: " << double(virtualMemUsedByMe) / (1024. \* 1024) << " bytes\n";

}

// ---------- macos ----------

// #include <mach/task.h>

// #include <mach/mach\_init.h>

// void printMemoryUsage() {

//  struct task\_basic\_info t\_info;

//  mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

//  if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

//      TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

//      &t\_info\_count))

//  {

//      printf("Error!");

//  }

//  else

//  {

//      printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

//  }

// }

// ---------- windows ----------

// #include "Windows.h"

// #include "psapi.h"

// void printMemoryUse()

// {

//     PROCESS\_MEMORY\_COUNTERS\_EX pmc;

//     GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

//     SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

//     cout << fixed << setprecision(6);

//     cout << "Virtual memory used by me: " << virtualMemUsedByMe << " bytes\n";

// }

// // ---------- linux ----------

// // #include <sys/resource.h>

// // #include <sys/time.h>

// // void printMemoryUsage() {

// //    struct rusage usage;

// //     getrusage(RUSAGE\_SELF, &usage);

// //     cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// // }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

    start = clock();

}

void printTimeUse() {

    printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

// ---------- code ----------

vector<int> solve(int n, vector<pair<int, int> > segments) {

    sort(segments.begin(), segments.end(), [](const auto& x, const auto& y) {

        return x.second < y.second;

    });

    vector<int> points;

    int rightmost = -1;

    for (const auto& segment : segments) {

        if (segment.first > rightmost) {

            rightmost = segment.second;

            points.push\_back(rightmost);

        }

    }

    return points;

}

int main() {

    getFirstTime();

    int n;

    fin >> n;

    vector<pair<int, int> > segments(n);

    for (int i = 0; i < n; i++) {

        int a, b;

        fin >> a >> b;

        segments[i] = make\_pair(a, b);

    }

    vector<int> res = solve(n, segments);

    fout << res.size() << "\n";

    for (int point : res) {

        fout << point << " ";

    }

    fout << "\n";

    printTimeUse();

    printMemoryUse();

    return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Сортируем все вектора по правому значению по возрастанию
6. Далее находим точку, которая находится правее предыдущей и записываем ее в файл

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***

***Изображение выглядит как текст

Автоматически созданное описание***

|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи | 0.0000000s | 0.792969 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0000000s | 0.792969 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №5. Максимальное количество призов (0.5 балла)

***Текст задачи***

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.

• Постановка задачи. Необходимо представить заданное натуральное число n в виде суммы как можно большего числа попарно различных натуральных чисел. То есть найти максимальное k такое, что n можно записать как a1 + a2 + ... + ak, где a1, ..., ak - натуральные числа и ai ̸= aj для всех 1 ≤ i < j ≤ k.

• Формат ввода / входного файла (input.txt). Входные данные состоят из

одного целого числа n.

• Ограничения на входные данные. 1 ≤ n ≤ 109.

• Формат вывода / выходного файла (output.txt). В первой строке выведите

максимальное число k такое, что n можно представить в виде суммы k

попарно различных натуральных чисел. Во второй строке выведите эти k

попарно различных натуральных чисел, которые в сумме дают n (если таких представлений много, выведите любое из них).

• Ограничение по времени. 2 сек.

***Листинг кода***

#include <iostream>

#include <vector>

#include <fstream>

#include <iomanip>

#include <algorithm>

// ---------- macos ----------

// #include <mach/task.h>

// #include <mach/mach\_init.h>

// void printMemoryUsage() {

//  struct task\_basic\_info t\_info;

//  mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

//  if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

//      TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

//      &t\_info\_count))

//  {

//      printf("Error!");

//  }

//  else

//  {

//      printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

//  }

// }

// ---------- windows ----------

#include "Windows.h"

#include "psapi.h"

using namespace std;

void printMemoryUsage()

{

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

    SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

    cout << fixed << setprecision(6);

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

}

// // ---------- linux ----------

// // #include <sys/resource.h>

// // #include <sys/time.h>

// // void printMemoryUsage() {

// //    struct rusage usage;

// //     getrusage(RUSAGE\_SELF, &usage);

// //     cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// // }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

    start = clock();

}

void printTimeUse() {

    printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

int main() {

    getFirstTime();

    int n;

    fin >> n;

    vector<int> result;

    int current\_sum = 0;

    if (current\_sum != n) {

        for (int i = 1; ; i++) {

            if (current\_sum + i <= n) {

                current\_sum += i;

                result.push\_back(i);

            }

            else if (current\_sum + i > n){

                result.back() += n - current\_sum;

                break;

            }

        }

    }

    fout << result.size() << endl;

    for (auto x : result) {

        fout << x << " ";

    }

    printTimeUse();

    printMemoryUsage();

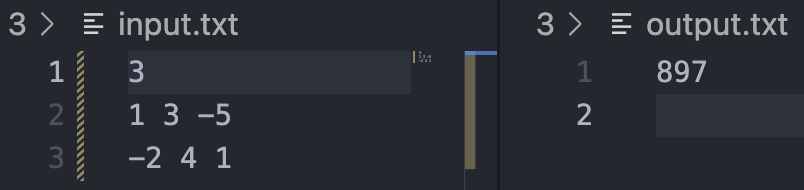
    return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Создаем переменную current\_sum с которой будем сравнивать результат
6. запускаем цикл в котором прибавляем каждый раз новое число

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0001030s | 1.312500 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0007490s | 1.281250 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №6. Максимальная зарплата (0.5 балла)

***Текст задачи***

В качестве последнего вопроса успешного собеседования ваш начальник дает вам несколько листков 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 наибольшегочисла 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.

• Постановка задачи. Составить наибольшее число из набора целых чисел.

• Формат ввода / входного файла (input.txt). Первая строка входных данных

содержит целое число n. Во второй строке даны целые числа a1, a2, ..., an.

• Ограничения на входные данные. 1 ≤ n ≤ 10^2, 1 ≤ ai ≤ 10^3 для всех 1 ≤ i ≤ n.

• Формат вывода / выходного файла (output.txt). Выведите наибольшее

число, которое можно составить из a1, a2, ..., an.

• Ограничение по времени. 2 сек.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include <time.h>

#include <string>

#include <vector>

// ---------- macos ----------

// #include <mach/task.h>

// #include <mach/mach\_init.h>

// void printMemoryUsage() {

//  struct task\_basic\_info t\_info;

//  mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

//  if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

//      TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

//      &t\_info\_count))

//  {

//      printf("Error!");

//  }

//  else

//  {

//      printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

//  }

// }

// ---------- windows ----------

#include "Windows.h"

#include "psapi.h"

using namespace std;

void printMemoryUsage()

{

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

    SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

    cout << fixed << setprecision(6);

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

}

// // ---------- linux ----------

// // #include <sys/resource.h>

// // #include <sys/time.h>

// // void printMemoryUsage() {

// //    struct rusage usage;

// //     getrusage(RUSAGE\_SELF, &usage);

// //     cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// // }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

    start = clock();

}

void printTimeUse() {

    printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

int main() {

    getFirstTime();

    vector<int> nums;

    int n, a;

    fin >> n;

    for (int i = 0; i < n; i ++){

        int a;

        fin >> a;

        nums.push\_back(a);

    }

    for (int i = 0; i < n; i++) {

        int maxi = 0;

        for (auto num : nums) {

            if (num > maxi) {

                maxi = num;

            }

        }

        fout << to\_string(maxi);

        nums.erase(find(nums.begin(), nums.end(), maxi));

        maxi = 0;

    }

    printTimeUse();

    printMemoryUsage();

    return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. В вектор записываем все числа и запускаем цикл
6. Каждый раз ищем наибольшее число в векторе и приписываем его к строке, а затем удаляем это число из вектора

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0001030s | 1.312500 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0007490s | 1.281250 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №8. Расписание лекций (1 балл)

***Текст задачи***

Постановка задачи. 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 [ si, fi ) - 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.

• Формат ввода / входного файла (input.txt). В первой строке вводится

натуральное число N - общее количество заявок на лекции. Затем вводится

N строк с описаниями заявок - по два числа в каждом si и fi для каждой

лекции i. Гарантируется, что si < fi. Время начала и окончания лекции -

натуральные числа, не превышают 1440 (в минутах с начала суток).

• Ограничения на входные данные. 1 ≤ N ≤ 1000, 1 ≤ si, fi ≤ 1440

• Формат вывода / выходного файла (output.txt). Выведите одно число

– максимальное количество заявок на проведение лекций, которые можно

выполнить.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <time.h>

#include <algorithm>

#include <vector>

#include <iomanip>

#include <sys/time.h>

// ---------- macos ----------

// #include <mach/task.h>

// #include <mach/mach\_init.h>

// using namespace std;

// void printMemoryUsage() {

//  struct task\_basic\_info t\_info;

//  mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

//  if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

//      TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

//      &t\_info\_count))

//  {

//      printf("Error!");

//  }

//  else

//  {

//      printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

//  }

// }

// ---------- windows ----------

#include "windows.h"

#include "psapi.h"

using namespace std;

void printMemoryUsage() {

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

    SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

    cout << fixed << setprecision(6);

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

}

// ---------- linux ----------

// #include <sys/resource.h>

// #include <sys/time.h>

// using namespace std;

// void printMemoryUsage() {

//    struct rusage usage;

//    getrusage(RUSAGE\_SELF, &usage);

//    cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

    start = clock();

}

void printTimeUse() {

    printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

// --------------- code ---------------

int maxLectures(vector<int>& begins, vector<int>& ends, int lecturesAmount) {

    int lastLect = 0, minTimeLectListen = 0, count = 0;

    while (minTimeLectListen < 1440) {

        minTimeLectListen = 1440;

        for (int i = 0; i < lecturesAmount; i++) {

            if (begins[i] >= lastLect && ends[i] < minTimeLectListen) {

                minTimeLectListen = ends[i];

            }

        }

        if (minTimeLectListen < 1440) {

            lastLect = minTimeLectListen;

            count++;

        }

    }

    return count;

}

int main() {

    getFirstTime();

    vector<pair<int, int> > segments;

    int n;

    fin >> n;

    vector<int> begins;

    vector<int> ends;

    for (int i = 0; i < n; i++) {

        int s, f;

        fin >> s >> f;

        begins.push\_back(s);

        ends.push\_back(f);

    }

    fout << maxLectures(begins, ends, n);

    printTimeUse();

    printMemoryUsage();

    return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Записываем в отдельные массивы время начала и окончания лекции и передаем их в функцию для подсчета количества лекций
6. Если начало следующей лекции больше или равно окончанию следующей лекции, а также окончание лекции раньше чем наступает раньше предыдущей лекции, то сохраняем эту лекцию как кротчайшую.
7. Повышаем счетчик проведенных лекций

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000000s | 0.804688 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0020000s | 0.824219 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №10. Яблоки (1 балл)

***Текст задачи***

Постановка задачи. Алисе в стране чудес попались n волшебных яблок.

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

• Формат ввода / входного файла (input.txt). В первой строке вводятся на-

туральные числа n и s – число яблок и начальный рост Алисы. В следующих n строках вводятся пары натуральных чисел ai, bi через пробел.

• Ограничения на входные данные. 1 ≤ n ≤ 1000, 1 ≤ s ≤ 1000, 1 ≤ ai, bi ≤ 1000.

• Формат вывода / выходного файла (output.txt). Если яблоки съесть нель-

зя, выведите число -1. Иначе выведите n чисел – номера яблок, в том по-

рядке, в котором их нужно есть.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <vector>

#include <algorithm>

#include <fstream>

// ---------- macos ----------

// #include <mach/task.h>

// #include <mach/mach\_init.h>

// void printMemoryUsage() {

//  struct task\_basic\_info t\_info;

//  mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

//  if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

//      TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

//      &t\_info\_count))

//  {

//      printf("Error!");

//  }

//  else

//  {

//      printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

//  }

// }

// ---------- windows ----------

#include "Windows.h"

#include "psapi.h"

#include <iomanip>

using namespace std;

void printMemoryUsage()

{

    PROCESS\_MEMORY\_COUNTERS\_EX pmc;

    GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

    SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

    cout << fixed << setprecision(6);

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

}

// // ---------- linux ----------

// // #include <sys/resource.h>

// // #include <sys/time.h>

// // void printMemoryUsage() {

// //    struct rusage usage;

// //     getrusage(RUSAGE\_SELF, &usage);

// //     cout << "Memory used: " << usage.ru\_maxrss / 1024.0 << " MB\n";

// // }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

    start = clock();

}

void printTimeUse() {

    printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

struct Apple {

    int a, b, id;

};

bool compare\_apples(const Apple& x, const Apple& y) {

    return x.b > y.b;

}

int main() {

    getFirstTime();

    int n, s;

    fin >> n >> s;

    vector<Apple> apples(n);

    for (int i = 0; i < n; i++) {

        fin >> apples[i].a >> apples[i].b;

        apples[i].b -= apples[i].a;

        apples[i].id = i + 1;

    }

    sort(apples.begin(), apples.end(), compare\_apples);

    vector<int> order;

    while (!apples.empty()) {

        bool possible = false;

        for (int i = 0; i < apples.size(); i++) {

            if (s - apples[i].a > 0) {

                s += apples[i].b;

                order.push\_back(apples[i].id);

                apples.erase(apples.begin() + i);

                possible = true;

                break;

            }

        }

        if (!possible) {

            fout << -1 << endl;

            return 0;

        }

    }

    for (int i = 0; i < order.size(); i++) {

        fout << order[i] << " ";

    }

    fout << endl;

    printTimeUse();

    printMemoryUsage();

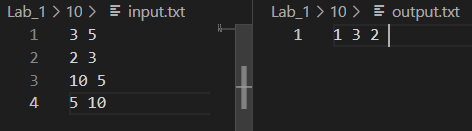
    return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Записываем в отдельные массивы время начала и окончания лекции и передаем их в функцию для подсчета количества лекций
6. Если начало следующей лекции больше или равно окончанию следующей лекции, а также окончание лекции раньше чем наступает раньше предыдущей лекции, то сохраняем эту лекцию как кротчайшую.
7. Повышаем счетчик проведенных лекций

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000000s | 0.792969 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0020000s | 0.824219 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №11. Максимальное количество золота (1 балл)

***Текст задачи***

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).

• Постановка задачи. Даны n золотых слитков, найдите максимальный вес

золота, который поместится в сумку вместимостью W.

• Формат ввода / входного файла (input.txt). Первая строка входных данных содержит вместимость W сумки и количество n золотых слитков. В следующей строке записано n целых чисел w0, w1, ..., wn−1, определяющие вес золотых слитков.

• Ограничения на входные данные. 1 ≤ W ≤ 10^4, 1 ≤ n ≤ 300, 0 ≤ w0, ..., wn−1 ≤ 10^5

• Формат вывода / выходного файла (output.txt). Выведите максимальный

вес золота, который поместится в сумку вместимости W.

• Ограничение по времени. 5 сек.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <math.h>

#include <algorithm>

#include <vector>

// ---------- macos ----------

#include <mach/task.h>

#include <mach/mach\_init.h>

using namespace std;

void printMemoryUsage() {

struct task\_basic\_info t\_info;

mach\_msg\_type\_number\_t t\_info\_count = TASK\_BASIC\_INFO\_COUNT;

if (KERN\_SUCCESS != task\_info(mach\_task\_self(),

TASK\_BASIC\_INFO, (task\_info\_t)&t\_info,

&t\_info\_count))

{

printf("Error!");

}

else

{

printf("Memory used: %f MB\n", t\_info.resident\_size / 1024.0 / 1024.0);

}

}

// ---------- windows ----------

// #include "windows.h"

// #include "psapi.h"

// using namespace std;

// void printMemoryUsage() {

// PROCESS\_MEMORY\_COUNTERS\_EX pmc;

// GetProcessMemoryInfo(GetCurrentProcess(), (PROCESS\_MEMORY\_COUNTERS\*)&pmc, sizeof(pmc));

// SIZE\_T virtualMemUsedByMe = pmc.PrivateUsage;

// cout << fixed << setprecision(6);

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

// }

// ---------- linux ----------

// #include <sys/resource.h>

// #include <sys/time.h>

// void printMemoryUsage() {

// struct rusage usage;

// getrusage(RUSAGE\_SELF, &usage);

// cout << "Memory used: " << usage.ru\_maxrss / 1024.0 / 1024.0 << " MB\n";

// }

ifstream fin("input.txt");

ofstream fout("output.txt");

clock\_t start;

void getFirstTime() {

start = clock();

}

void printTimeUse() {

printf("Time taken: %.7fs\n", (double)(clock() - start)/CLOCKS\_PER\_SEC);

}

int main() {

getFirstTime();

int W, n;

fin >> W >> n;

// чтение весов слитков

vector<int> w(n);

for (int i = 0; i < n; i++) {

fin >> w[i];

}

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

// создание двумерного динамического массива

int \*\*dp = new int\*[n];

for (int i = 0; i <= n; i++) {

dp[i] = new int[W];

}

// заполнение первой строки нулями

for (int w = 0; w <= W; w++) {

dp[0][w] = 0;

}

// заполнение таблицы

for (int i = 1; i <= n; i++) {

for (int j = 0; j <= W; j++) {

dp[i][j] = dp[i-1][j]; // заполняем из предыдущей строки

if (w[i-1] <= j) { // если слиток помещается в рюкзак

dp[i][j] = max(dp[i-1][j-w[i-1]] + w[i-1], dp[i-1][j]);

}

cout << dp[i][j] << " ";

}

cout << endl;

}

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

for (int i = 0; i <= n; i++) {

delete[] dp[i];

}

delete[] dp;

printTimeUse();

printMemoryUsage();

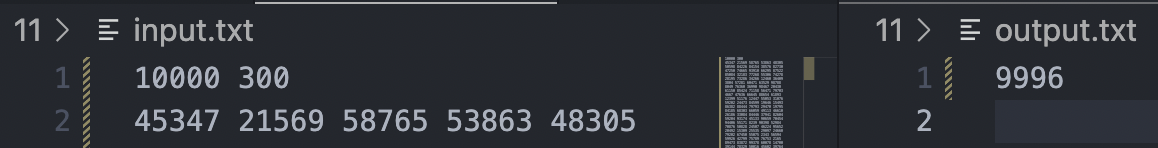
return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Считываем веса слитков и создаем двумерный массив, первая строка которого будет заполнена нулями
6. Заполнаем двумерный массив весами слитков

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000410s | 1.281250 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0119810s | 15.468750 |

***Вывод по задаче:*** Смог решить эту задачу

## Задача №12. Последовательность (1 балл)

***Текст задачи***

Постановка задачи. Дана последовательность натуральных чисел a1, a2, ..., an, и известно, что ai ≤ i для любого 1 ≤ i ≤ 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.

• Формат ввода / входного файла (input.txt). В первой строке входного

файла находится одно целое число n. Во второй строке находится n целых

чисел a1, a2, ..., an.

• Ограничения на входные данные. 1 ≤ n ≤ 40000, 1 ≤ ai ≤ i.

• Формат вывода / выходного файла (output.txt). В первую строку выходного файла выведите количество элементов последовательности в любой из получившихся двух частей, а во вторую строку через пробел номера этих элементов. Если построить такое разбиение невозможно, выведите -1.

• Ограничение по времени. 2 сек.

• Ограничение по памяти. 256 мб.

***Листинг кода***

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

ifstream fin("input.txt");

ofstream fout("output.txt");

int main() {

int n;

fin >> n;

vector<int> a(n);

int sum = 0;

for (int i = 0; i < n; i++) {

fin >> a[i];

sum += a[i];

}

if (sum % 2 != 0) {

fout << -1 << endl;

return 0;

}

vector<vector<int> > dp(n + 1, vector<int>(sum / 2 + 1, 0));

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= sum / 2; j++) {

if (j < a[i - 1]) {

dp[i][j] = dp[i - 1][j];

} else {

dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - a[i - 1]] + a[i - 1]);

}

}

}

if (dp[n][sum / 2] != sum / 2) {

fout << -1 << endl;

return 0;

}

vector<int> ans;

int i = n, j = sum / 2;

while (i > 0 && j > 0) {

if (dp[i][j] == dp[i - 1][j]) {

i--;

} else {

ans.push\_back(i);

j -= a[i - 1];

i--;

}

}

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

for (int i = 0; i < ans.size(); i++) {

fout << ans[i] << " ";

}

fout << endl;

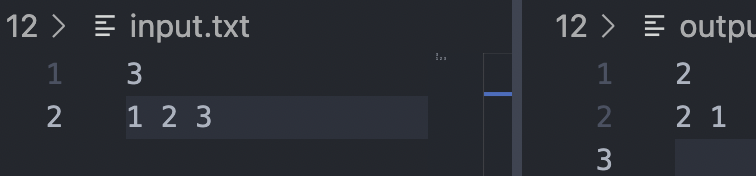
return 0;

}

***Текстовое объяснение решения***

1. Подключаем все необходимые библиотеки
2. Открываем нужные файлы
3. Прописываем функции для замера памяти и времени
4. В main начинаем отсчет времени и считываем значения из файла
5. Считываем веса слитков и создаем двумерный массив, первая строка которого будет заполнена нулями
6. Заполнаем двумерный массив весами слитков

***Результат работы кода на примерах из текста задачи:(скрины input output файлов)***



|  |  |  |
| --- | --- | --- |
|  | Время выполнения (seconds) | Затраты памяти (MiB) |
| Пример из задачи 1 | 0.0000360s | 1.281250 |
| Верхняя граница диапазона значений входных данных из текста задачи | 0.0119810s | 15.468750 |

***Вывод по задаче:*** Смог решить эту задачу

В ходе данной лабораторной работы удалось на практике познакомиться с со структурами данных: очередь с приоритетом и пирамида. Реализовать необходимые классы для работы с такими структурами данных, а также применить их в решений практических задач