Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 4
"Задачи 1401, 1604, 1726"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1401, 1604 и 1726.

2 Задача 1401

1401. Игроки

Ограничение времени: 2.0 секунды
Ограничение памяти: 64 МБ

Известно, что господин Чичиков зарабатывал свой капитал и таким способом: он спорил со всякими недотёпами, что сможет доказать, что квадратную доску размера 512 × 512 нельзя замостить следующими фигурами:

```
X    XX    X    XX
XX   X    XX     X
```

и всегда выигрывал. Однако один из недотёп оказался не так уж глуп, и сказал, что сможет замостить такими фигурами доску размера 512 × 512 без правой верхней клетки. Чичиков, не подумав, ляпнул, что он вообще может любую доску размера $2^n \times 2^n$ без одной произвольной клетки замостить такими фигурами. Слово за слово, они поспорили. Чичиков чувствует, что сам он не докажет свою правоту. Помогите же ему!

Исходные данные

В первой строке записано целое число n ($1 \le n \le 9$). Во второй строке через пробел даны два целых числа x, y: координаты «выколотой» клетки доски ($1 \le x, y \le 2^n$), x — номер строки, y — номер столбца. Левый верхний угол доски имеет координаты (1, 1).

Результат

Ваша программа должна выдать 2^n строчек по 2^n чисел в каждой строке. На месте выбитой клетки должно стоять число 0. На месте остальных клеток должны стоять числа от 1 до $(2^{2n} - 1) / 3$ — номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, выведите «−1».

Пример

исходные данные	результат
2 1 1	0 1 3 3 1 1 4 3 2 4 4 5 2 2 5 5

Автор задачи: Алексей Самсонов
Источник задачи: XII командный чемпионат школьников Свердловской области по программированию (15 октября 2005 года)

Рис. 1. Условие задачи 1401.

2.1 Краткое описание алгоритма

1. Входные данные: в первой строке записано целое число n ($1 \le n \le 9$). Во второй строке через пробел даны два целых числа x, y: координаты "выколотой"клетки доски ($1 \le x, y \le 2^n$), x – номер строки, y – номер столбца. Левый верхний угол доски имеет координаты (1, 1).

2. Основу алгоритма составляет **рекурсия**. Нетрудно заметить, что квадрат 2 на 2 имеет можно замостить требуемым образом.
3. По методу математической индукции докажем, что можно замостить требуемым образом квадрат размером 2^n на 2^n. Пусть для квадратов 2^n на 2^n задача решена, тогда покажем, что для квадрата 2^{n+1} на 2^{n+1} решение также существует.
4. Разобьем большой квадрат на 4 квадрата размера 2^n на 2^n, в одном из них содержится выколотая клетка, значит, мы можем его замостить, по предположению индукции. Вырежем треугольник из центра большого исходного квадрата и получим 3 малых квадрата, у каждого из которых 1 выколотая клетка, опять же по предположению индукции задача для них решена, значит, решена и задача для всего большого квадрата.
5. Выходные данные: программа должна выдать 2^n строчек по 2^n чисел в каждой строке. На месте выбитой клетки должно стоять число 0. На месте остальных клеток должны стоять числа от 1 до $(2^{2n} - 1)/3$ – номер фигуры, закрывающей данную клетку. Разумеется, одинаковые номера должны образовывать фигуры. Если же такую доску нельзя покрыть фигурами, вывести -1.

2.2 Листинг

Листинг 1. Исходный код для 1401

```
#include <iostream>
#include <cmath>


int result_points[512][512];
int current_number = 3;


void creator(int n, int x, int y,  int empty_x, int empty_y) {
    // base square
    if (n == 2) {
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                if (x + i != empty_x || y + j != empty_y) {
                    result_points[x + i][y + j] =
                        current_number / 3;
                    ++current_number;
                }
```

```
18              }
19          }
20          return;
21      }
22      // work with middle triangle
23      for (int i = 0; i < 2; ++i) {
24          for (int j = 0; j < 2; ++j) {
25              int x_1 = x + i * n / 2;
26              int x_2 = x + i * n / 2 + n / 2;
27              int y_1 = y + j * n / 2;
28              int y_2 =  y + j * n / 2 + n / 2;
29
30              if (x_1 > empty_x || x_2 <= empty_x ||
31              y_1 > empty_y || y_2 <= empty_y) {
32                  result_points[x + n / 2 - 1 + i][y + n / 2 - 1
                        + j] = current_number / 3;
33                  ++current_number;
34              }
35          }
36      }
37
38      for (int i = 0; i < 2; ++i) {
39          for (int j = 0; j < 2; ++j) {
40              int x_1 = x + i * n / 2;
41              int x_2 = x + i * n / 2 + n / 2;
42              int y_1 = y + j * n / 2;
43              int y_2 = y + j * n / 2 + n / 2;
44
45              // parts with empty point and without
46              x_1 <= empty_x && x_2 > empty_x &&
47              y_1 <= empty_y && y_2 > empty_y ?
48              creator(n / 2, x_1, y_1, empty_x, empty_y) :
49              creator(n / 2, x_1, y_1, x + n / 2 - 1 + i, y + n
                    / 2 - 1 + j);
50          }
51      }
52  }
53
54  int main() {
55      int n, empty_x, empty_y;
56      std::cin >> n >> empty_x >> empty_y;
57      n = pow(2, n);
58
```

```
59      creator(n, 0, 0, empty_x - 1, empty_y - 1);
60
61      for (int i = 0; i < n; ++i) {
62          for (int j = 0; j < n; ++j) {
63              std::cout << result_points[i][j] << " ";
64          }
65          std::cout << std::endl;
66      }
67      return 0;
68  }
```

2.3 Результат

Рис. 2. Результат отправки задачи 1401.

3 Задача 1604

1604. В Стране Дураков

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Главный бульдог-полицейский Страны Дураков решил ввести ограничение скоростного режима на автомобильной трассе, ведущей от Поля Чудес к пруду Черепахи Тортиллы. Для этого он заказал у Папы Карло n знаков ограничения скорости. Папа Карло слабо разбирался в дорожном движении и поэтому изготовил знаки с разными ограничениями на скорость: 49 км/ч, 34 км/ч, 42 км/ч, и т.д. Всего получилось k различных ограничений: n_1 знаков с одним ограничением, n_2 знаков со вторым ограничением, и т.д. ($n_1 + \ldots + n_k = n$)

Бульдог-полицейский ничуть не расстроился, получив такие знаки, напротив, он решил извлечь из этого экономическую выгоду. Дело в том, что по Правилам дорожного движения Страны Дураков ограничение на скорость действует вплоть до следующего знака. Если на знаке написано число 60, это означает, что участок от данного знака до следующего нужно проехать ровно со скоростью 60 километров в час — не больше и не меньше. Бульдог распорядился расставить знаки так, чтобы обогатившимся на Поле Чудес автолюбителям во время своего движения по трассе приходилось как можно больше раз менять скорость. Для этого нужно расставить имеющиеся знаки в правильном порядке. Если Вы поможете бульдогу это сделать, то он готов будет поделиться с Вами частью своих доходов.

Исходные данные

В первой строке дано число k — количество различных типов знаков с ограничением скорости ($1 \le k \le 10000$). Во второй строке через пробел перечислены целые положительные числа n_1, …, n_k. Сумма всех n_i не превосходит 10000.

Результат

Выведите n целых чисел в пределах от 1 до k — порядок, в котором нужно расставить по трассе имеющиеся знаки. Вне зависимости от того, какой знак стоит первым, считается, что, проезжая его, водитель меняет скорость, так как до этого ограничения не действовали. Если задача имеет несколько решений, выведите любое.

Пример

исходные данные	результат
2 2 2	1 2 1 2

Автор задачи: Александр Ипатов (идея — Александр Торопов)
Источник задачи: Девятое открытое личное первенство УрГУ (1 марта 2008)

Рис. 3. Условие задачи 1604.

3.1 Краткое описание алгоритма

1. Входные данные: в первой строке дано число k – количество различных типов знаков с ограничением скорости ($1 \le k \le 10000$). Во второй строке через пробел перечислены целые положительные числа $n_1, \ldots, n_k$. Сумма всех n_i не превосходит 10000.

2. Сначала применим к исходному массиву знаков сортировку по количеству.
3. Если наибольшее количество знаков больше половины суммарного количества всех знаков, расставим через одного оставшиеся знаки, начиная с первой позиции, так мы добьемся максимального количества смен, на остальные места расставим знаки наибольшего по количеству знака.
4. В противном случае расставим по порядку через одного все знаки, таким образом, каждый раз будет происходить смена, потому что количество каждого знака меньше половины от суммарного.
5. Выходные данные: вывести n целых чисел в пределах от 1 до k – порядок, в котором нужно расставить по трассе имеющиеся знаки. Вне зависимости от того, какой знак стоит первым, считается, что, проезжая его, водитель меняет скорость, так как до этого ограничения не действовали. Если задача имеет несколько решений, вывести любое.

3.2 Листинг

Листинг 2. Исходный код для 1604

```
#include <iostream>


std::pair<int, int> signs[10000];

bool compare(std::pair<int, int> s_1, std::pair<int, int> s_2) {
    return s_1.second > s_2.second;
}

//quicksort
void sorter(int left, int right) {
    int i = left;
    int j = right;

    std::pair<char, int> cur = signs[(left + right) / 2];

    while (i <= j) {
        while (compare(signs[i], cur)) {
            ++i;
        }
        while (compare(cur, signs[j])) {
            --j;
```

```
23          }
24
25          if (i <= j) {
26              std::swap(signs[i++], signs[j--]);
27          }
28      }
29
30      if (i < right) sorter(i, right);
31
32      if (j > left) sorter(left, j);
33
34  }
35
36  int main() {
37      int k;
38      std::cin >> k;
39
40      int sum = 0;
41
42      for (int i = 0; i < k; ++i) {
43          std::cin >> signs[i].second;
44          signs[i].first = i + 1;
45          sum += signs[i].second;
46      }
47
48      sorter(0, k - 1);
49      int res[sum];
50
51      bool check = false;
52      if (signs[0].second > (sum + 1) / 2) {
53          int i = 1;
54          int j = k - 1;
55
56          while (i < sum) {
57              res[i] = signs[j].first;
58              --signs[j].second;
59              signs[j].second == 0 ? --j : j;
60              i += 2;
61
62              if (!check && i > sum - 1) {
63                  check = true;
64                  i = 0;
65              }
```

```
66          }
67      } else {
68          int i = 0;
69          int j = 0;
70
71          while (i < sum) {
72              res[i] = signs[j].first;
73              --signs[j].second;
74              signs[j].second == 0 ? ++j : j;
75              i += 2;
76
77              if (!check && i > sum - 1) {
78                  check = true;
79                  i = 1;
80              }
81          }
82      }
83
84      for (int i = 0; i < sum; ++i) {
85          std::cout << res[i] << " ";
86      }
87      return 0;
88  }
```

3.3 Результат

Рис. 4. Результат отправки задачи 1604.

4 Задача 1726

1726. Кто ходит в гости…

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Программный комитет школьных соревнований по программированию, проходящих в УрГУ — многочисленная, весёлая и дружная команда. Дружная настолько, что общения в университете им явно не хватает, поэтому они часто ходят друг к другу в гости. Все ребята в программном комитете очень спортивные и ходят только пешком.

Однажды хранитель традиций олимпиадного движения УрГУ подумал, что на пешие прогулки от дома к дому члены программного комитета тратят слишком много времени, которое могли бы вместо этого потратить на придумывание и подготовку задач. Чтобы доказать это, он решил посчитать, какое расстояние в среднем преодолевают члены комитета, когда ходят друг к другу в гости. Хранитель традиций достал карту Екатеринбурга, нашёл на ней дома всех членов программного комитета и выписал их координаты. Но координат оказалось так много, что хранитель не смог справиться с этой задачей самостоятельно и попросил вас помочь ему.

Город Екатеринбург представляет собой прямоугольник со сторонами, ориентированными по сторонам света. Все улицы города идут строго с запада на восток или с севера на юг, проходя через весь город от края до края. Дома всех членов программного комитета расположены строго на пересечении каких-то двух перпендикулярных улиц. Известно, что все члены комитета ходят только по улицам, поскольку идти по тротуару гораздо приятнее, чем по дворовым тропинкам. И, конечно, при переходе от дома к дому они всегда выбирают кратчайший путь. Программный комитет очень дружный, и все его члены ходят в гости ко всем одинаково часто.

Исходные данные

Первая строка содержит целое число n — количество членов программного комитета ($2 \le n \le 10^5$). В i-й из следующих n строк через пробел записаны целые числа x_i, y_i — координаты дома i-го члена программного комитета ($1 \le x_i, y_i \le 10^6$).

Результат

Выведите среднее расстояние, которое проходит член программного комитета от своего дома до дома своего товарища, округлённое вниз до целых.

Пример

исходные данные	результат
3 10 10 20 20 10 20	13

Автор задачи: Денис Дублённых
Источник задачи: Уральская региональная командная олимпиада по программированию 2009

Рис. 5. Условие задачи 1726.

4.1 Краткое описание алгоритма

1. Входные данные: первая строка содержит целое число n – количество членов программного комитета ($2 \le n \le 10^5$). В i-й из следующих n строк через пробел записаны целые числа x_i, y_i – координаты дома i-го члена программного комитета ($1 \le x_i, y_i \le 10^6$).
2. Рассчитаем расстояние как манхэттенское, то есть отдельно будем счи-

тать по двум координатм. Получим среднее значение, посчитав сумму расстояний между всеми домами и разделив ее на количество путей.
3. С помощью алгоритма быстрой сортировки отсортируем все координаты. Вычислим расстояние между каждой парой соседних домов r по одной координате
4. Далее посчитаем, какое число человек проходит по этому пути проходит: i человек справа и $n - i$ слева, суммарный путь, пройденный получим по формуле $r \cdot i \cdot (n - i)$.
5. Выходные данные: выведите среднее расстояние, которое проходит член программного комитета от своего дома до своего товарища, округленное вниз до целых.

4.2 Листинг

Листинг 3. Исходный код для 1726

```
#include <iostream>

unsigned long long x[100000];
unsigned long long y[100000];

//quicksort for x
void sorter_x(int left, int right) {
    int i = left;
    int j = right;

    unsigned long long medium = x[(left + right) / 2];

    while (i <= j) {
        while (x[i] < medium) {
            ++i;
        }
        while (medium < x[j]) {
            --j;
        }

        if (i <= j) {
            std::swap(x[i++], x[j--]);
        }
    }

    if (i < right) sorter_x(i, right);
```

```
27
28      if (j > left) sorter_x(left, j);
29
30  }
31
32  //quicksort for y
33  void sorter_y(int left, int right) {
34      int i = left;
35      int j = right;
36
37      unsigned long long medium = y[(left + right) / 2];
38
39      while (i <= j) {
40          while (y[i] < medium) {
41              ++i;
42          }
43          while (medium < y[j]) {
44              --j;
45          }
46
47          if (i <= j) {
48              std::swap(y[i++], y[j--]);
49          }
50      }
51
52      if (i < right) sorter_y(i, right);
53
54      if (j > left) sorter_y(left, j);
55  }
56
57  int main() {
58      long long n;
59      std::cin >> n;
60      unsigned long long sum = 0l;
61
62      for (long long i = 0; i < n; ++i) {
63          std::cin >> x[i] >> y[i];
64      }
65
66      sorter_x(0, (int) n - 1);
67      sorter_y(0, (int) n - 1);
68
69      for (int i = 0; i < n; ++i) {
```

```
70          sum += (x[i] - x[i - 1]) * i * (n - i);
71          sum += (y[i] - y[i - 1]) * i * (n - i);
72      }
73      sum = 2l * sum / (n * (n - 1));
74      std::cout << sum;
75      return 0;
76  }
```

4.3 Результат

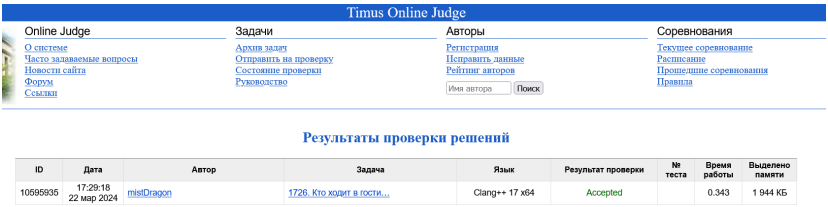

Рис. 6. Результат отправки задачи 1726.

5 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1401, 1604 и 1726. Все эти задачи объединяет необходимость применения рекурсии, в двух последних рекурсия реализуется внутри алгоритма **quicksort**.