МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания № 8.2

Тема:

«ОТДЕЛЬНЫЕ ВОПРОСЫ АЛГОРИТМИЗАЦИИ»
«Реализация алгоритмов на основе сокращения числа переборов»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Жаворонкова А.А.

Группа: ИКБО-43-23

Москва - 2024

СОДЕРЖАНИЕ

1 ЦЕЛЬ

Освоить алгоритмы на основе сокращения числа переборов.

2 ЗАДАНИЕ

2.1 Постановка задачи

Разработать алгоритм решения задачи с применением метода, указанного в варианте и реализовать программу.

Оценить количество переборов при решении задачи стратегией «в лоб» - грубой силы. Сравнить с числом переборов при применении метода.

Сделать выводы о проделанной работе, основанные на полученных результатах.

Оформить отчет в соответствии с требованиями документирования разработки ПО: Постановка задачи, Описание алгоритмов и подхода к решению, Код, результаты тестирования, Вывод.

Индивидуальный вариант:

Пронумеровать позиции в матрице размером 5*5 следующим образом: если номер i ($1 \leq i \leq 25$) соответствует позиции (x,y), то номер i+1 может соответствовать позиции с координатами (z,w), вычисляемыми по одному из следующих правил:

1) $(z,w)=(x\pm3,y)$

2) $(z,w)=(x,y\pm3)$

3) $(z,w)=(x\pm2,y\pm2)$

1) Написать программу, которая последовательно нумерует позиции матрицы при заданных координатах позиции, в которой содержится номер 1.

2) Вычислить число всех возможных расстановок номеров для всех начальных позиций, расположенных под главной диагональю. Метод: метод ветвей и границ

Метод: метод ветвей и границ

2.2 Математическая модель решения

Инициализируем матрицу 5×5 и устанавливает все элементы в 0.

Начинает поиск расстановок, запуская алгоритм нумерации для каждой стартовой позиции под главной диагональю, то есть для всех позиций (x,y), где x>y.

Алгоритм использует метод ветвей и границ для поиска всех возможных расстановок, соблюдая правила перехода между позициями.

Каждое найденное решение выводится, и в конце работы выводится общее количество найденных расстановок.

Правила переходов

Используются следующие правила для переходов от текущей позиции (x,y) к следующей:

(z, w) = (x ± 3, y) — смещение на 3 клетки по горизонтали,

(z, w) = (x, y ± 3) — смещение на 3 клетки по вертикали,

(z, w) = (x ± 2, y ± 2) — диагональное смещение на 2 клетки.

Пример работы алгоритма

Рассмотрим выполнение программы на одной из стартовых позиций, например, (2,0).

Стартовая позиция (2, 0):

Матрица изначально заполнена нулями.

Устанавливаем matrix[2][0] = 1 как начальный номер и запускаем рекурсивную функцию для нумерации, начиная со следующего номера 2.

Из позиции (2, 0) проверяются все возможные переходы, используя правила перехода и массивы dx и dy.

Возможные позиции для перехода:

(5, 0) — за пределами матрицы (недопустимо).

(-1, 0) — за пределами матрицы (недопустимо).

(2, 3) — допустимая позиция.

(2, -3) — за пределами матрицы (недопустимо).

(4, 2) — допустимая позиция.

(0, 2) — допустимая позиция.

(4, -2) — за пределами матрицы (недопустимо).

(0, -2) — за пределами матрицы (недопустимо).

Следовательно, для размещения номера 2 возможны позиции (2, 3), (4, 2), и (0, 2).

Алгоритм продолжает рекурсивно искать допустимые позиции для следующего номера, увеличивая num и заполняя матрицу согласно правилам переходов.

Если удается найти полную последовательность от 1 до 25, программа выводит найденную расстановку.

Если программа достигает состояния, в котором невозможно разместить следующий номер, происходит откат: текущий номер удаляется, и продолжается поиск других путей.

Например, если для номера 4 все возможные переходы уже заняты или недоступны, программа возвращается к номеру 3 и пробует другую позицию для него.

Когда все возможные стартовые позиции под главной диагональю матрицы проверены, программа завершает работу и выводит общее количество найденных расстановок.

2.3 Код программы с комментариями

Продемонстрируем код программы на листинге 1, 2, 3, 4.

Код будет написан на языке программирования C++.

```
1. #include <iostream>
2. #include <vector>
3. #include <iomanip>
4.
5. using namespace std;
6.
7. // Создание матрицы
8. const int N = 5;
9. int matrix[N][N] = {0};
10.  int countSolutions = 0; // Счётчик расстановок
11.
12.  // Смещения
13.  int dx[] = {3, -3, 0, 0, 2, -2, 2, -2};
14.  int dy[] = {0, 0, 3, -3, 2, -2, -2, 2};
```

Листинг 1 - Код программы

```
15.  bool isValid(int x, int y) { // Проверка координат
16.      return (x >= 0 && x < N && y >= 0 && y < N && matrix[x][y]
  == 0);
17.  }
18.
19.  void printMatrix() { // Вывод матрицы
20.      for (int i = 0; i < N; ++i) {
21.          for (int j = 0; j < N; ++j) {
22.              cout << setw(3) << matrix[i][j] << " ";
23.          }
24.          cout << endl;
25.      }
26.      cout << "____________________________________" << endl;
27.  }
28.
29.  // Рекурсия для нумерации позиций
30.  void branchAndBound(int x, int y, int num) {
31.      if (num > N * N) { // Все позиции пронумерованы
32.          countSolutions++;
33.          cout << "Расстановка #" << countSolutions << ":" <<
  endl;
34.          printMatrix(); // Вывод матрицы
35.          return;
36.      }
37.      for (int i = 0; i < 8; ++i) {
38.          int nx = x + dx[i];
39.          int ny = y + dy[i];
40.
41.          if (isValid(nx, ny)) { // Если переход возможен
42.              matrix[nx][ny] = num; // Устанавливаем номер
43.              branchAndBound(nx, ny, num + 1); // Рекурсивный
  вызов для следующего номера
44.              matrix[nx][ny] = 0; //Возврат к прошлому состоянию
45.          }
46.      }
47.  }
48.
49.  int main() {
50.      for (int x = 1; x < N; ++x) {
51.          for (int y = 0; y < x; ++y) {
52.              matrix[x][y] = 1;
53.              branchAndBound(x, y, 2);
54.              matrix[x][y] = 0; // Сброс позиции
55.          }
56.      }
57.      cout << "Общее количество возможных расстановок: " <<
  countSolutions << endl;
58.      return 0;
59.  }
```

Листинг 2 - Продолжение кода программы

2.4 Результаты тестирования

Стоит задача протестировать программу. Результаты тестирования будут продемонстрированы на рисунке 1.

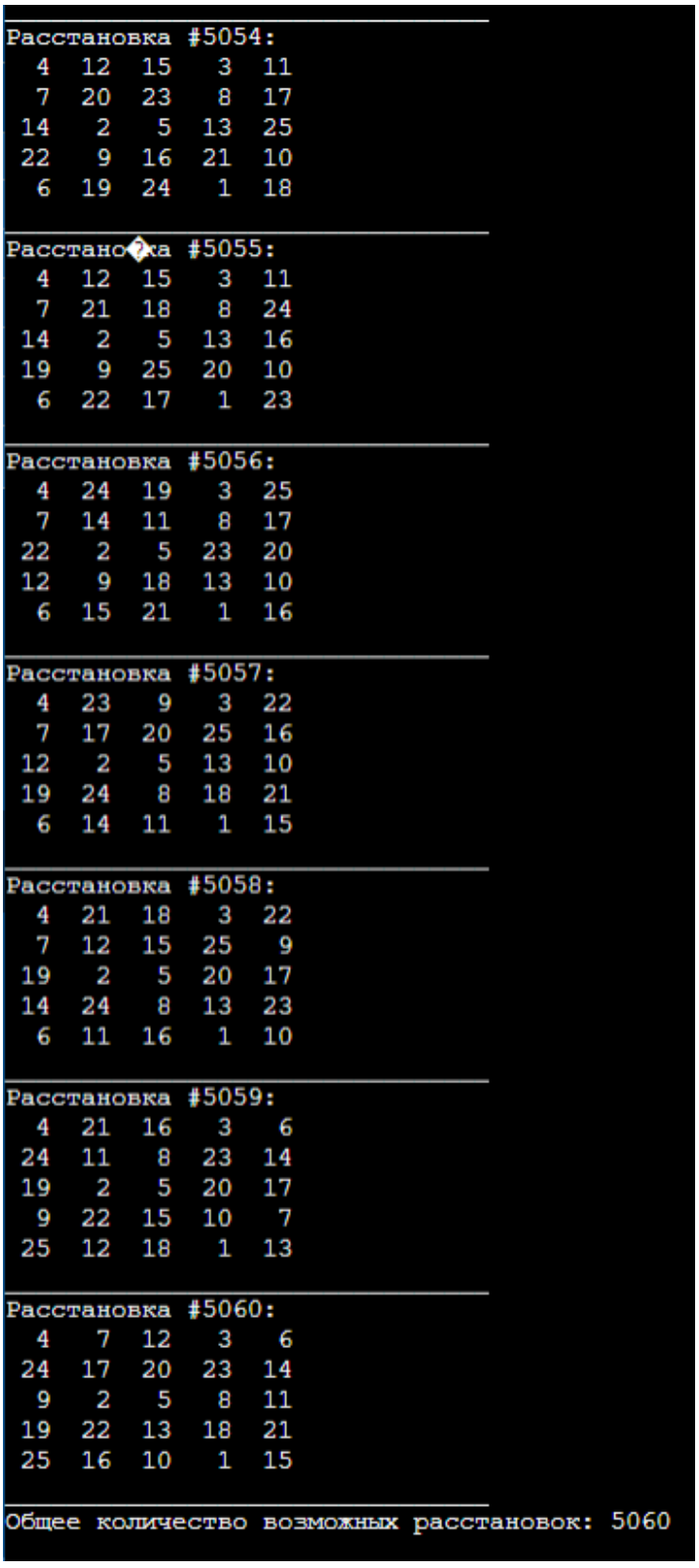

Рисунок 2 - Вывод на экран при выполнении программы

3 ВЫВОДЫ

Метод ветвей и границ значительно снижает количество переборов, так как исключает проверку заведомо неподходящих областей, улучшая производительность