# Алгоритм перестановок

Функция compareMatrixes

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| for (int j = i; j < n; j++) | 1 присваивании  1 сравнение  1 инкремент | N  N(N+1)/2+1  N(N+1)/2 |
| if (mat1[i][j] != mat2[i][j]) | 1 сравнение | N(N+1)/2 |
| return false; | 1 ретурн | 1 |
| return true; | 1 ретурн | 1 Если верхний не пройдет |

Худший случай (матрицы равны):

F(n) = 2 + 2N + N + N(N+1) + 2 = N^2 + 4N + 4 ~ O(N^2)

Лучший случай (матрицы не равны):

F(n) = 6

Средлний случай (матрицы не равны, но это обнаруживается на середине матриц):

F(n) = (N^2)/2 + 2N + 2 ~ O(N^2)

Функция shuffleAlgo

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| vector<int> shuffles = vector<int>(n); | 1 присваивание | 1 |
| int\*\* shaffledMat = new int\* [n]; | 1 присваивание | 1 |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| shuffles[i] = i; | 1 присваивании | N |
| shaffledMat[i] = new int[n]; | 1 присваивании | N |
| do |  |  |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | A  (N + 1)\*A  N\*A |
| for (int j = 0; j < n; j++) | 1 присваивании  1 сравнение  1 инкремент | N\*A  N\*(N + 1)\*A  N\*N\*A |
| shaffledMat[i][j] = mat1[shuffles[i]][shuffles[j]]; | 1 присваивании | N^2\*A |
| if (compareMatrixes(shaffledMat, mat2, n)) | compareMatrixes  1 сравнение | A \* B |
| return true; | 1 | 1 если if сработает |
| while (next\_permutation(shuffles.begin(), shuffles.end())) | 1 сравнение  next\_permutation | C\*A |
| return false; | 1 | 1 |

F(n) = 4 + 4N + A(2N+2) + A(2N + 2N^2) + A\*N^2 + A\*B + C\*A + 1 = 3AN^2 + 4AN + 2A + AB + AC + 4N + 5 = A \* (3N^2 +4N + B + C + 2) + 4N + 5

Худший случай (графы не изоморфны):

A (кол-во повторений вайла) = N!

B (кол-во операций в compareMatrixes) = (N^2) / 2 + 2N + 2

C = N/2

F(n) = 3\* N!\*N^2 + 4\* N!\*N + 2\* N! + N!\*((N^2) / 2 + 2N + 2) + N!\*N/2 + 4N + 5=

= N! \* (3.5\*N^2 + 6.5N + 4) + 4N + 5 ~ O(N!)

Лучший случай (графы сразу изоморфны):

A = 1

B = N^2 + 4N + 4

C = 0

F(n) = 3N^2 + 4N + 2 + N^2 + 4N + 4 + 4N + 5 = 4N^2 + 12N + 11 ~ O(N^2)

Средний случай (графы изоморфны, но это обнаружится спустя 0.5N! перестановок):

A = (N!)/2

B = (N^2) / 2 + 2N + 2

C = N/2

F(n) = (N!)/2 \* (3N^2 + 4N + (N^2) / 2 + 2N + 2 + N/2 + 2) + 4N + 5 =

= 0.5N! \* (3.5N^2 + 6.5N + 4) + 4N + 5 ~ O(N!)

# Алгоритм вектора степеней

Функция compareDegreeVectors

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  (A + 1)  A |
| if (v1[i] != v2[i]) | 1 сравнение | A |
| return false; | 1 ретурн | 1 |
| return true; | 1 ретурн | 1 если первый не сработал |

F(n) = 1 + A + 1 + A + A + 1 = 3\*A + 3

Худший случай (вектора равны):

A = N

F(n) 3\*N + 3 ~ O(n)

Лучший случай (векторы сразу не равны)

A = 0

F(n) = 3 ~ O(1)

Лучший случай (векторы на середине не равны)

A = N/2

F(n) = 1.5N + 3 ~ O(n)

Функция getDegreeVector

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| vector<int> res(n); | 1 присваивании | 1 |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| for (int j = 0; j < n; j++) | 1 присваивании  1 сравнение  1 инкремент | N  (N+1)\*N  N^2 |
| res[i] += mat[i][j]; | 1 сложение | N^2 |
| return res; | 1 ретурн | 1 |

F(n) = 1 + 1 + N + 1 + N + N + N^2 + N + N^2 + N^2 + 1 = 3N^2 + 4N + 4

Худший случай = Лучший = Средний:

F(n) = 3N^2 + 4N + 4 ~ O(n)

Функция degreeVectorAlgo

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| vector<int> v1 = getDegreeVector(mat1, n); | 1 присваивании  1 вызов функции | 1  1\* getDegreeVector(n) |
| vector<int> v2 = getDegreeVector(mat2, n); | 1 присваивании  1 вызов функции | 1  1\* getDegreeVector(n) |
| sort(v1.begin(), v1.end()); | 1 сортировка | 1 \* sort |
| sort(v2.begin(), v2.end()); | 1 сортировка | 1 \* sort |
| return compareDegreeVectors(v1, v2, n); | 1 ретурн  1 вызов функции | 1  1\*A |

sort = NlogN в среднем

F(n) = 1 + 3N^2 + 4N + 4 + 1 + 3N^2 + 4N + 4 + 2 \* NlogN + 1 + A =

= 6N^2 + 8N + 10 + 2NlogN + A

Худший случай (векторы равны):

А = N

F(n) = 6N^2 + 9N + 10 + 2NlogN ~ O(N^2)

Лучший случай (векторы сразу не равны)

A = 0

F(n) = 6N^2 + 8N + 10 + 2NlogN ~ O(N^2)

Лучший случай (векторы на середине не равны)

A = N/2

F(n) = 6N^2 + 8.5N + 10 + 2NlogN ~ O(N^2)

# Алгоритм векторов степеней второго порядка

Функция compare

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| if (left.first != right.first) | 1 сравнение | 1 |
| return left.first < right.first; | 1 сравнение  1 ретурн | A  A |
| if (left.second.size() != right.second.size()) | 1 сравнение | 1 \* |A-1| |
| return left.second.size() < right.second.size(); | 1 сравнение  1 ретурн | |A-1| \* B  |A-1| \* B |
| return left.second < right.second; | 1 сравнение  1 ретурн | 1 \* |A-1| \* |B-1| \* C  1 |

F(n) = 1 + 2A + |A-1| + 2 \* |A-1| \* B + |A-1| \* |B-1| \* C + 1

Лучший случай (не равны по первому условию):

A = 1

B = 0

C = 0

F(n) = 3 ~ O(n)

Худший случай (все полностью равны)

A = 0

B = 0

C = N

F(n) = 1 + 1 + N + 1 = N + 3 ~ O(n)

Средний случай (векторы не равны начиная с серидины):

A = 0

B = 0

C = N/2

F(n) = N/2 + 3 ~ O(n)

Функция compareSecondOrderDegreeVectors

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| if (v1[i].first != v2[i].first) | 1 сравнение | N |
| return false; | 1 ретурн | 1 |
| if (v1[i].second.size() != v2[i].second.size()) | 1 сравнение | N |
| return false; | 1 ретурн | 1 |
| for (int j = 0; j < v1[i].second.size(); j++) | 1 присваивании  1 сравнение  1 инкремент | N  N^2 + N  N^2 |
| if (v1[i].second[j] != v2[i].second[j]) | 1 сравнение | N^2 |
| return false; | 1 ретурн | 1 |
| return true; | 1 ретурн | 1 |

F(n) = …

Худший случай (Оба вектора степеней второго порядка полностью равны):

F(n) = 1 + N + 1 + N + N + N + N + N^2 + N + N^2 + N^2 + 1 = 3N^2 + 6N + 3 ~ O(n^2)

Лучший случай (Оба вектора степеней второго порядка сразу не равны):

F(n) = 1 + 2 + 1 + 1 = 4 ~ O(1)

Средний случай:

F(n) = 1.5N^2 + 3N + 1.5 ~ O(n^2)

Функция getSecondOrderDegreeVector

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| vector<int> degreeVec = getDegreeVector(mat, n); | 1 присваивание  1 getDegreeVector | 1  1 |
| vector<pair<int, vector<int>>> res(n); | 1 присваивание | 1 |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| res[i].first = degreeVec[i]; | 1 присваивании | N |
| for (int j = 0; j < n; j++) | 1 присваивании  1 сравнение  1 инкремент | N  N^2 + N  N^2 |
| if (mat[i][j] == 1) | 1 сравнение | N^2 |
| res[i].second.push\_back(degreeVec[j]); | 1 push\_back | A |
| return res; | 1 ретурн | 1 |

getDegreeVector = 3N^2 + 4N + 4

push\_back = 1

F(n) = 1 + 3N^2 + 4N + 4 + 1 + 2N + 2 + N + N + N^2 + N + N^2 + N^2 + A + 1 =

= 6N^2 + 9N + A + 9

Худший случай (в матрице все 1 кроме одной диагонали):

A = N^2-N

F(n) = 6N^2 + 9N + N^2-N + 9 = 7N^2 + 8N + 9 ~ O(n^2)

Лучший случай (в матрице 1 ребро):

A = 2

F(n) = 6N^2 + 9N + 11 ~ O(n^2)

Средний случай:

A = 0.5N^2

F(n) = 6N^2 + 9N + 0.5N^2 + 9 = 6.5N^2 + 9N + 9 ~ O(n^2)

Функция secondOrderDegreeVectorAlgo

|  |  |  |
| --- | --- | --- |
| **Строка** | **Действия** | **Повторения** |
| vector<pair<int, vector<int>>> v1 = getSecondOrderDegreeVector(mat1, n); | 1 присваивание  1 вызов ф-ии | 1  1 |
| vector<pair<int, vector<int>>> v2 = getSecondOrderDegreeVector(mat2, n); | 1 присваивание  1 вызов ф-ии | 1  1 |
| for (int i = 0; i < n; i++) | 1 присваивании  1 сравнение  1 инкремент | 1  N+1  N |
| sort(v1[i].second.begin(), v1[i].second.end()); | 1 sort | N |
| sort(v2[i].second.begin(), v2[i].second.end()); | 1 sort | N |
| sort(begin(v1), end(v1), compare); | sort | 1 |
| sort(begin(v2), end(v2), compare); | 1 sort | 1 |
| return compareSecondOrderDegreeVectors(v1, v2, n); | 1 ретурн  1 вызов ф-ии | 1  1 |

F(n) = 2 + 2\*A + 2 + 2N + 2\*N\*MlogM + 2\*NlogN + B + 1 = 2N + 2A + 2NMlogM + 2NlogN + B + 5

Худший случай (2 полных графа):

A = 7N^2 + 8N + 9

M = N

B = 3N^2 + 6N + 3

F(n) = 2N + 14N^2 + 16N + 18 + 2N^2logN + 2NlogN + 3N^2 + 6N + 3 + 5 =

= 17N^2 + 2N^2logN + 2NlogN + 24N + 26 ~ O(N^2\*logN)

Лучший случай ():

A = 6N^2 + 9N + 11

M = 1

B = 4

F(n) = 2N + 26N^2 + 9N + 11 + 2N + 2NlogN + 4 + 5=

= 26N^2 + 13N + 2NlogN + 20 ~ O(n^2)

Средний случай():

A = 6.5N^2 + 9N + 9

M = N/2

B = 1.5N^2 + 3N + 1.5

F(n) = 2N + 13N^2 + 18N + 18 + N^2log(N/2) + 2NlogN + 1.5N^2 + 3N + 1.5 + 5 =

= 14.5N^2 + N^2log(N/2) + 2NlogN + 23N + 24.5 ~ O(N^2\*logN)

# Худшие и лучшие тесты

## Алгоритм Перестановок

Лучший: графы одинаковые

Худший: Графы вообще не изоморфны

Это все в экселе