

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Кафедра «Вычислительная техника»

ОТЧЁТ

По лабораторной работе №7

По курсу «Логика и основы алгоритмизации в инженерных задачах»

На тему «Обход графа в глубину»

Выполнили

студенты

группы

23ВВВ4:

Святов И.Ю.

Епинин Д.В.

Приняли:

Юрова О.В.

Деев М.В.

Пенза 2024

Общие сведения:

Обход графа – одна из наиболее распространенных операций с графами. Задачей обхода является прохождение всех вершин в графе. Обходы применяются для поиска информации, хранящейся в узлах графа, нахождения связей между вершинами или группами вершин и т.д.

Одним из способов обхода графов является поиск в глубину. Идея такого обхода состоит в том, чтобы начав обход из какой-либо вершины всегда переходить по первой встречающейся в процессе обхода связи в следующую вершину, пока существует такая возможность. Как только в процессе обхода исчерпаются возможности прохода, необходимо вернуться на один шаг назад и найти следующий вариант продвижения. Таким образом, итерационно выполняя описанные операции, будут пройдены все доступные для прохождения вершины. Чтобы не заходить повторно в уже пройденные вершины, необходимо их пометить как пройденные.

Таким образом, можно предложить следующую рекурсивную реализацию алгоритма обхода в глубину.

Задание:

1. Сгенерируйте (используя генератор случайных чисел) матрицу смежности для неориентированного графа G . Выведите матрицу на экран.

Ответ:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <string.h>
#include <locale.h>

int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));
    const int G = 5; // Размер графа
    int M[G][G];      // Матрица смежности

    // Инициализация матрицы смежности
    for (int i = 0; i < G; i++) {
        for (int j = 0; j < G; j++) {
            if (i == j) {
                M[i][j] = 0; // Нет петли
            }
            else {
                M[i][j] = 0; // Изначально нет ребер
            }
        }
    }

    // Генерация случайных рёбер для неориентированного графа
    for (int i = 0; i < G; i++) {
        for (int j = i + 1; j < G; j++) {
            M[i][j] = M[j][i] = rand() % 2; // Случайное ребро между i и j
        }
    }

    // Вывод матрицы смежности на экран
    printf("Матрица смежности:\n");
```

```

    for (int i = 0; i < G; i++) {
        for (int j = 0; j < G; j++) {
            printf("%d ", M[i][j]);
        }
        printf("\n");
    }
    return 0;
}

```

2. Для сгенерированного графа осуществите процедуру обхода в глубину, реализованную в соответствии с приведенным выше описанием.

Ответ:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <string.h>
#include <locale.h>
const int G = 5;
int M[G][G];
int visited[G];
// Функция для выполнения обхода в глубину
void DFS(int v) {
    visited[v] = 1; // Помечаем текущую вершину как посещённую
    printf("%d ", v);

    // Обходим всех соседей текущей вершины
    for (int i = 0; i < G; i++) {
        // Если есть ребро и вершина не посещена
        if (M[v][i] == 1 && !visited[i]) {
            DFS(i); // Рекурсивно вызываем DFS для соседней вершины
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));

    // Инициализация матрицы смежности
    for (int i = 0; i < G; i++) {
        for (int j = 0; j < G; j++) {
            if (i == j) {
                M[i][j] = 0; // Нет петли
            }
            else {
                M[i][j] = 0; // Изначально нет рёбер
            }
        }
    }

    // Генерация случайных рёбер для неориентированного графа
    for (int i = 0; i < G; i++) {
        for (int j = i + 1; j < G; j++) {
            M[i][j] = M[j][i] = rand() % 2; // Случайное ребро между i и j
        }
    }

    // Вывод матрицы смежности на экран
    printf("Матрица смежности:\n");
    for (int i = 0; i < G; i++) {

```

```

        for (int j = 0; j < G; j++) {
            printf("%d ", M[i][j]);
        }
        printf("\n");
    }

    // Инициализация массива посещённых вершин
    for (int i = 0; i < G; i++) {
        visited[i] = 0;
    }

    printf("Обход в глубину (DFS):\n");
    DFS(0); // Начинаем с вершины 0

    return 0;
}

```

3. *Реализуйте процедуру обхода в глубину для графа, представленного списками смежности.

Ответ:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <string.h>
#include <locale.h>
const int G = 5;
typedef struct Node {
    int vertex;
    struct Node* next;
} Node;
Node* adjList[G]; // Массив указателей для хранения списков смежности
int visited[G]; // Массив для отслеживания посещённых вершин

// Функция для добавления ребра в список смежности
void addEdge(int src, int dest) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->vertex = dest;
    newNode->next = adjList[src]; // Добавляем узел в начало списка
    adjList[src] = newNode;

    // Поскольку граф неориентированный, добавляем рёбра в обе стороны
    newNode = (Node*)malloc(sizeof(Node));
    newNode->vertex = src; // добавляем обратное ребро
    newNode->next = adjList[dest]; // Добавляем узел в начало списка
    adjList[dest] = newNode;
}

// Функция для выполнения обхода в глубину
void DFS(int v) {
    visited[v] = 1; // Помечаем текущую вершину как посещённую
    printf("%d ", v); // Выводим текущую вершину

    // Обходим всех соседей текущей вершины
    Node* temp = adjList[v];

```

```

    while (temp) {
        if (!visited[temp->vertex]) {
            DFS(temp->vertex); // Рекурсивно вызываем DFS для соседней вершины
        }
        temp = temp->next;
    }
}
// Функция для освобождения памяти, выделенной под список смежности
void freeGraph() {
    for (int i = 0; i < G; i++) {
        Node* temp = adjList[i];
        while (temp) {
            Node* toFree = temp;
            temp = temp->next;
            free(toFree); // Освобождаем память для узла
        }
    }
}
int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));

    // Инициализация списков смежности
    for (int i = 0; i < G; i++) {
        adjList[i] = NULL; // Изначально пусто
        visited[i] = 0; // Все вершины не посещены
    }

    // Генерация случайных рёбер для неориентированного графа
    for (int i = 0; i < G; i++) {
        for (int j = i + 1; j < G; j++) {
            if (rand() % 2) { // С вероятностью 0.5 добавляем ребро
                addEdge(i, j); // Добавляем ребро в граф
            }
        }
    }

    // Вывод списков смежности на экран
    printf("Списки смежности:\n");
    for (int i = 0; i < G; i++) {
        printf("%d:", i);
        Node* temp = adjList[i];
        while (temp) {
            printf(" %d", temp->vertex);
            temp = temp->next;
        }
        printf("\n");
    }

    // Инициализация массива посещённых вершин
    for (int i = 0; i < G; i++) {
        visited[i] = 0;
    }

    // Запуск обхода в глубину начиная с первой вершины (0)
    printf("Обход в глубину (DFS):\n");
    DFS(0); // Начинаем с вершины 0
    // Освобождаем выделенную память
    freeGraph();
}

```

```

    return 0;
}

```

Задание 2*

1. Для матричной формы представления графов выполните преобразование рекурсивной реализации обхода графа к не рекурсивной.

Ответ:

```

#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <string.h>
#include <locale.h>

const int G = 5; // Размер графа
int M[G][G]; // Матрица смежности
int visited[G]; // Массив для отслеживания посещённых вершин
// Функция для выхода из DFS (не рекурсивная реализация)
void DFS_Iterative(int start) {
    int stack[G]; // Стек для итеративного DFS
    int top = -1; // Указатель на верхушку стека

    // Добавляем начальную вершину в стек и отмечаем её как посещённую
    stack[++top] = start;
    visited[start] = 1;
    printf("%d ", start); // Выводим начальную вершину

    while (top != -1) { // Пока стек не пуст
        int v = stack[top]; // Берём вершину с вершины стека
        int found = 0; // Флаг для проверки не посещённых соседей

        for (int i = 0; i < G; i++) {
            // Ищем непосещённого соседа
            if (M[v][i] == 1 && !visited[i]) {
                stack[++top] = i; // Добавляем соседа в стек
                visited[i] = 1; // Отмечаем соседа как посещённого
                printf("%d ", i); // Выводим соседа
                found = 1; // Отмечаем, что нашли непосещённого
                break; // Прерываем цикл, чтобы начать обход с нового узла
            }
        }

        if (!found) {
            top--; // Если не нашли соседей, убираем вершину из стека
        }
    }
}

int main() {
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));
    // Инициализация матрицы смежности
    for (int i = 0; i < G; i++) {
        for (int j = 0; j < G; j++) {
            if (i == j) {
                M[i][j] = 0; // Нет петли
            }
        }
    }
}

```

```

    }
    else {
        M[i][j] = 0; // Изначально нет рёбер
    }
}
}
// Генерация случайных рёбер для неориентированного графа
for (int i = 0; i < G; i++) {
    for (int j = i + 1; j < G; j++) {
        M[i][j] = M[j][i] = rand() % 2; // Случайное ребро между i и j
    }
}
// Вывод матрицы смежности на экран
printf("Матрица смежности:\n");
for (int i = 0; i < G; i++) {
    for (int j = 0; j < G; j++) {
        printf("%d ", M[i][j]);
    }
    printf("\n");
}
// Инициализация массива посещённых вершин
for (int i = 0; i < G; i++) {
    visited[i] = 0;
}
// Запуск итеративного обхода в глубину начиная с первой вершины (0)
printf("Итеративный обход в глубину (DFS):\n");
DFS_Iterative(0); // Начинаем с вершины 0

return 0;
}

```

Задание: проверить за сколько выполняется код

Ответ:

Матрица смежности:

The screenshot shows a Visual Studio console window with the following output:

```

6 299 298 300 302 303 301 304 305 309 306 315 307 308 311 313 310 312 314 322 316 319 320 317 324 318 321 323 325 330 32
8 326 327 329 331 332 334 338 333 335 336 337 339 340 341 342 345 344 343 346 347 349 348 350 351 355 353 356 352 357 35
4 360 361 358 362 359 363 368 365 366 367 364 370 374 371 369 373 378 375 376 372 377 379 380 382 381 383 385 386 387 38
4 391 395 389 388 390 392 394 396 397 404 393 399 398 400 403 401 407 405 402 406 411 408 409 410 412 413 416 414 417 41
5 418 420 421 422 419 423 424 427 425 429 426 430 432 433 428 431 434 435 437 436 438 439 442 440 441 443 451 447 448 44
5 446 452 453 449 455 444 456 450 454 457 458 461 459 464 463 460 465 462 466 471 467 468 469 470 473 472 474 475 478 47
9 481 476 480 477 482 484 485 486 487 489 490 494 483 488 492 491 498 497 493 495 496 500 501 499 503 502 504 505 511 50
6 507 510 508 512 509 516 514 513 515 517 518 520 521 522 519 524 523 525 526 529 527 530 528 533 531 534 532 535 537 53
6 538 540 541 543 542 545 544 548 546 539 547 550 554 551 549 555 552 553 557 556 560 559 561 558 562 564 563 566 565 56
7 572 569 573 570 571 568 581 574 577 575 576 578 579 583 582 580 584 586 587 588 592 589 585 590 593 595 596 591 594 59
7 598 600 602 603 599 604 606 605 601 608 609 607 611 610 612 617 614 615 616 618 613 619 624 621 620 625 622 623 626 62
9 627 628 631 630 634 632 639 635 637 636 638 633 640 644 641 642 647 643 648 649 646 652 645 653 650 651 658 656 654 65
5 659 664 657 660 661 666 665 662 667 663 671 672 668 669 674 670 677 673 676 675 678 680 679 681 683 684 685 686 682 68
7 689 688 690 691 694 692 693 698 699 695 696 697 701 704 702 700 703 705 713 706 707 708 710 709 720 711 712 715 714 71
6 717 719 721 723 718 722 724 726 725 729 731 730 728 727 732 733 734 736 735 739 740 738 737 741 744 742 748 743 746 75
0 751 749 745 747 753 756 752 754 755 762 757 758 759 760 761 765 763 769 764 766 767 771 768 772 773 770 774 778 775 78
3 776 779 777 781 782 786 780 784 785 788 787 789 797 790 791 793 792 794 795 796 798 799 802 800 801 803 804 805 806 81
3 807 808 809 810 816 811 817 819 815 814 818 820 812 822 823 825 826 827 829 821 824 828 831 830 833 832 834 835 836 83
7 838 841 842 843 839 844 845 849 840 846 848 847 851 853 852 850 854 856 855 857 859 858 863 860 861 864 865 866 867 86
8 873 862 870 876 872 869 871 874 875 880 881 877 882 879 878 884 888 886 883 887 889 893 890 885 897 892 894 891 899 89
8 895 900 903 901 896 904 902 905 910 906 909 911 908 913 907 915 914 912 918 920 921 919 916 924 922 923 928 917 925 92
7 930 931 926 935 929 932 933 934 936 937 939 941 942 938 946 945 940 943 944 949 948 952 947 958 951 955 953 950 954 95
6 957 960 959 961 962 963 966 965 967 969 968 970 964 971 972 973 974 976 977 975 982 979 978 983 980 981 984 985 986 98
7 988 993 989 991 990 992 994 995 996 997 999 998
Время: 0,178000 секунд
C:\Users\dimav\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 10812) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Списки смежности:

```
Консоль отладки Microsoft Visual Studio

1 759 757 755 758 756 754 752 751 749 753 746 747 745 750 744 748 741 742 739 743 740 737 736 735 738 733 730 734 731 73
2 728 724 729 727 725 718 726 720 723 722 719 716 721 717 715 712 714 713 710 708 711 707 706 703 705 704 701 697 709 69
6 700 695 699 702 686 694 698 692 691 690 688 693 689 684 687 683 682 685 681 680 678 677 679 672 676 674 673 675 671 66
8 670 666 667 662 669 664 665 663 661 656 657 660 659 653 658 655 654 649 652 650 651 648 647 646 644 645 640 643 642 63
9 641 638 636 637 635 634 633 630 632 631 625 628 629 626 627 623 624 622 619 621 620 618 616 614 613 611 609 612 617 61
0 606 607 605 615 608 602 603 601 604 600 599 595 597 596 598 594 593 589 590 592 591 586 588 587 583 578 584 582 585 58
1 575 576 580 579 574 573 570 577 572 569 567 571 568 566 561 564 563 565 560 559 562 558 557 555 553 556 554 552 551 54
9 550 548 547 543 546 545 541 544 542 539 538 536 534 537 540 533 535 530 531 532 528 529 527 526 525 523 524 522 521 52
0 519 518 515 517 516 514 513 512 511 510 509 507 506 508 505 504 500 503 501 502 499 495 497 496 498 492 494 490 491 48
9 493 487 486 485 488 484 483 482 480 481 479 478 475 477 473 474 476 469 470 472 468 467 471 466 465 463 460 464 458 46
2 459 461 457 456 455 454 450 449 451 453 448 452 447 444 446 445 442 443 441 440 439 438 435 437 434 436 432 430 431 43
3 428 429 427 415 426 424 425 423 422 421 419 413 420 416 418 411 417 414 410 409 412 407 408 401 405 406 404 403 402 39
4 400 398 397 399 396 395 389 393 391 392 388 390 384 386 387 378 385 383 381 375 372 379 382 377 380 374 371 376 368 36
9 373 370 367 366 365 363 360 364 362 356 358 361 357 352 353 354 359 355 349 351 347 348 350 343 344 346 342 345 341 33
8 337 336 335 340 331 339 334 329 327 333 332 330 325 328 324 323 321 322 326 319 317 315 318 316 320 312 313 311 314 31
0 309 308 303 306 305 304 300 302 307 299 298 294 301 296 295 293 292 291 290 297 289 287 288 284 286 285 283 280 279 28
2 276 281 278 277 275 271 274 272 270 273 269 268 265 267 266 263 264 262 260 261 258 259 256 257 253 255 254 252 251 24
9 250 246 248 247 242 245 244 243 241 239 238 240 237 234 236 235 231 230 229 233 232 224 227 222 226 223 228 221 225 22
0 219 215 211 206 218 217 216 214 210 213 209 212 203 208 205 204 199 201 207 202 200 197 198 192 194 196 190 195 193 18
8 191 189 187 186 183 185 181 184 180 179 182 178 177 176 173 174 168 175 172 170 171 161 167 166 169 164 155 165 163 15
4 162 160 159 156 158 147 157 153 146 152 150 149 145 151 148 144 143 142 141 137 139 134 138 140 136 135 130 133 132 12
9 131 127 128 124 123 126 125 119 120 118 117 115 122 116 121 114 110 111 113 106 112 109 108 107 103 105 104 100 102 10
1 99 96 98 97 95 93 94 92 91 89 86 85 90 88 87 81 84 83 82 77 79 80 73 78 74 76 72 75 71 69 70 63 66 65 67 61 68 64 57 6
2 58 60 56 59 54 53 55 47 51 50 52 49 48 46 45 43 38 44 41 39 36 40 42 35 34 37 33 32 29 31 30 26 25 28 27 21 22 24 20 1
9 23 18 17 16 14 10 15 12 11 7 8 9 4 6 5 1 3 2
Время: 0,278000 секунд

C:\Users\dimav\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (процесс 15224) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рекурсивная реализация для матрицы смежности:

```
Консоль отладки Microsoft Visual Studio

1 299 300 303 298 301 304 302 307 308 305 312 311 306 315 309 313 316 310 318 314 320 322 323 319 327 325 317 324 321 32
06 328 329 330 332 331 334 335 337 338 340 336 343 339 333 341 342 344 345 346 348 350 349 347 352 353 351 355 356 354 35
17 360 358 359 361 362 364 365 368 363 371 366 369 373 374 370 376 367 372 375 377 378 379 380 381 383 382 384 385 386 38
07 388 390 391 389 392 396 398 395 400 393 394 405 397 399 401 402 404 403 406 407 409 410 408 412 413 411 414 415 417 42
02 419 420 421 416 423 424 427 425 418 428 426 430 431 429 433 436 434 432 435 437 438 439 440 444 441 443 442 446 448 44
15 449 447 450 451 452 453 454 455 458 456 459 457 461 463 460 464 465 468 466 467 469 470 462 471 473 472 475 474 476 47
08 480 482 484 477 481 483 485 488 486 489 490 479 491 494 495 487 493 496 499 492 497 498 501 502 500 505 504 508 507 50
09 514 503 506 511 512 515 513 510 516 518 517 519 521 526 522 524 520 525 523 527 529 528 530 532 531 535 533 534 537 53
06 538 540 539 541 542 545 543 546 544 547 549 548 550 552 551 553 554 559 555 557 563 556 558 560 562 561 565 564 567 56
16 569 571 568 570 572 573 574 575 576 579 578 577 580 581 583 582 584 585 586 590 587 588 592 595 591 589 607 593 594 59
16 600 597 601 602 598 599 603 605 604 606 615 609 610 608 611 613 614 617 623 622 612 618 616 620 619 621 624 626 628 63
12 625 627 630 629 634 635 631 636 637 638 633 640 641 643 647 639 656 642 644 645 646 648 649 650 653 654 651 652 658 65
15 659 660 662 663 661 657 664 666 667 665 669 668 673 670 676 671 674 677 672 678 679 680 684 681 682 683 687 675 686 68
05 688 690 691 692 689 694 696 697 693 695 700 698 699 701 703 702 705 706 707 708 711 709 710 704 712 713 714 717 716 71
15 719 718 721 724 722 727 720 723 729 725 731 726 730 728 732 733 735 736 734 747 738 741 739 743 737 740 742 745 744 74
06 750 748 749 751 752 753 754 755 756 758 760 757 759 761 763 765 762 764 766 769 767 768 770 771 773 777 778 772 774 78
01 775 780 779 783 776 782 784 785 786 792 787 790 788 791 795 789 794 793 799 796 797 800 798 802 805 808 807 809 803 80
1 804 806 811 810 812 816 813 814 818 817 815 821 825 820 819 822 823 827 824 826 829 830 835 833 831 832 838 828 834 83
16 839 837 841 842 840 843 844 846 848 845 849 850 851 847 854 853 852 855 857 859 856 862 858 861 863 865 860 864 867 86
16 869 868 870 871 878 872 873 874 877 879 875 876 881 880 884 883 885 886 882 890 888 892 887 889 891 894 895 896 893 89
07 898 899 900 901 903 908 904 905 902 909 907 913 914 906 910 911 915 918 912 917 916 919 922 920 921 925 924 923 934 92
06 929 927 930 928 931 932 933 935 936 938 940 937 942 943 939 941 945 946 944 947 948 949 950 953 951 955 954 952 956 95
17 960 958 964 959 963 962 961 971 966 969 965 967 968 970 972 973 975 974 979 976 981 982 977 980 978 984 983 985 987 98
09 992 988 990 986 993 994 995 991 996 997 998 999
Время: 0,200000 секунд

C:\Users\dimav\source\repos\ConsoleApplication2\Debug\ConsoleApplication2.exe (процесс 14596) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...
```

Вывод: в ходе выполнения лабораторной работы были успешно сгенерированы случайные графы в матричной и списочной форме. Также были реализованы процедуры обхода графа в глубину как в рекурсивной, так и в не рекурсивной формах. Работы демонстрируют различные подходы к представлению графов и алгоритмам обхода.