ՀԱՅԱՍՏԱՆԻ ԱԶԳԱՅԻՆ ՊՈԼԻՏԵԽՆԻԿԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ԴԻՍԿՐԵՏ ՄԱԹԵՄԱՏԻԿԱ

ԿՈՒՐՍԱՅԻՆ ՆԱԽԱԳԻԾ

Առաջադրանքը տրվեց՝ 02.02.2022

Կուրսայինի պաշտպանություն՝ 02.06.2022

Թեմա՝ «Գրաֆի կողային կապակցվածություն»

Դասախոս՝ Գարեգին Սարգսյան

Ուսանող՝ Էլեն Տոնոյան

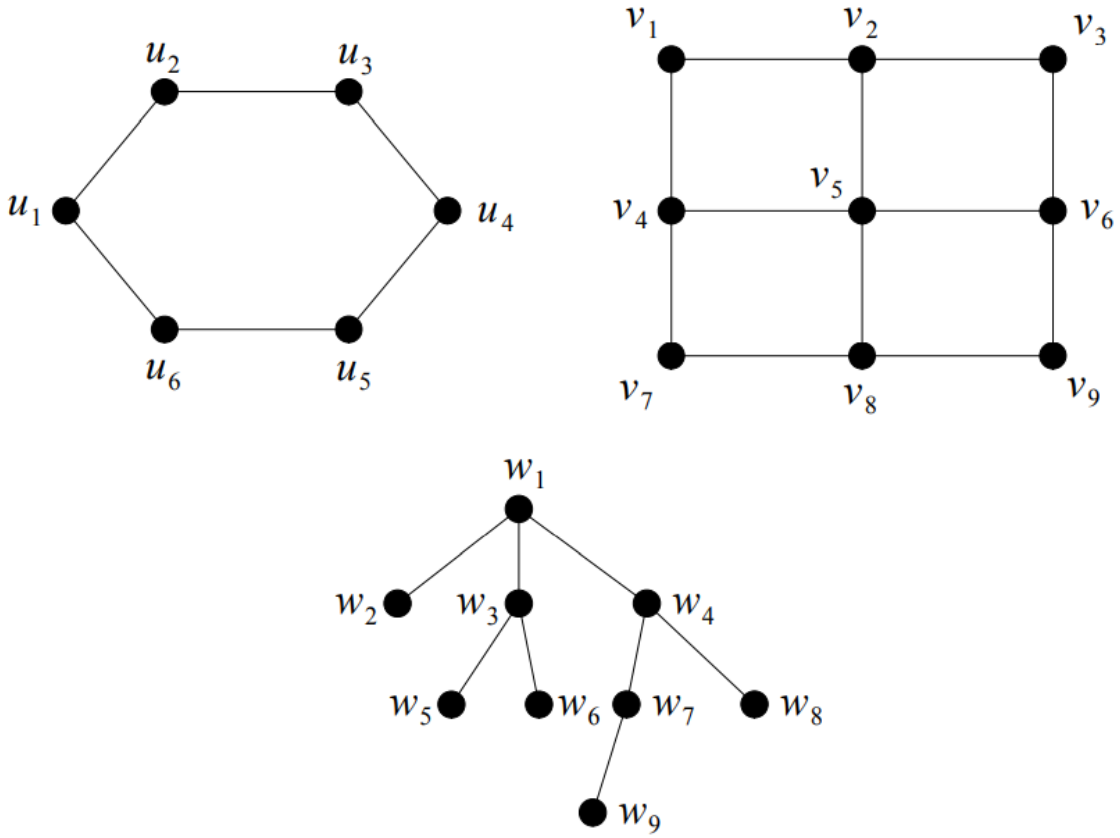
ԲՈՎԱՆԴԱԿՈՒԹՅՈՒՆ

Ներածություն **3**
Խնդիր **5**
Լուծում **6**
Կարգերի ալգորիթմ 7
Ծրագրի ներկայացում **8**
Արդյունք **9**
Հղում **10**
Օգտագործված գրականության ցանկ **11**

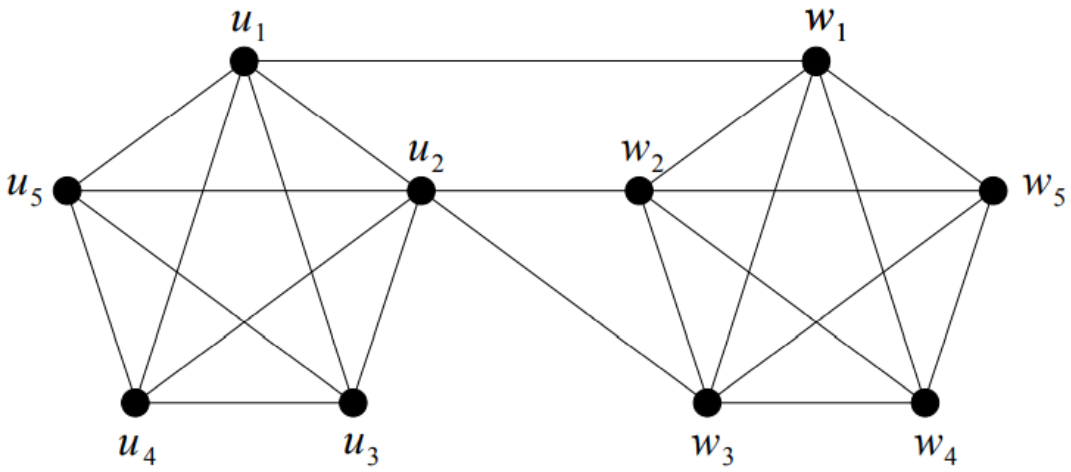
Ներածություն

Դիցուք G = (V, E)-ն գրաֆ է:

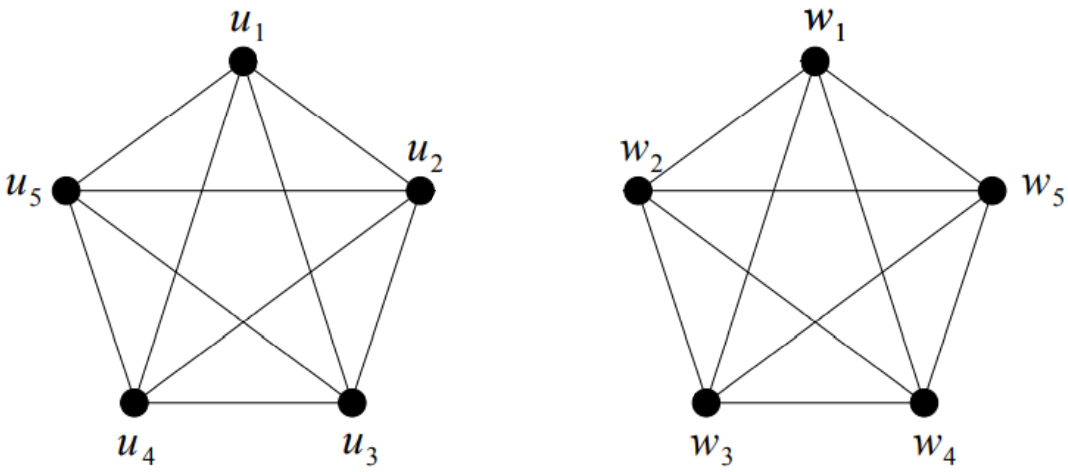
G գրաֆը կանվանենք կապակցված, եթե նրա ցանկացած երկու u, v գագաթների համար G գրաֆում գոյություն ունի (u, v) ճանապարհ:



G գրաֆի կողային կապակցվածություն կոչվում է կողերի նվազագույն քանակը, որոնց հեռացման արդյունքում առաջանում է ոչ կապակցված գրաֆ:



Վերոնշյալ կապակցված գրաֆից հեռացնելով u_1w_1, u_2w_2, u_2w_3, գագաթները, կստանանք ոչ կապակցված գրաֆ:



Քանի որ կողերի նվազագույն քանակը, որից հետո ստացանք ոչ կապակցված գրաֆ 3 է, կարող ենք ասել որ տրված գրաֆի կողային կապակցվածության թիվը 3 է:

Խնդիր

Տրված G(V, E) վերջավոր գրաֆի կողային կապակցվածության թվի որոշման ալգորիթմի մշակում և ծրագրային իրացում:

Լուծում

G գրաֆի կողային կապակցվածություն կոչվում է կողերի նվազագույն քանակը, որոնց հեռացման արդյունքում առաջանում է ոչ կապակցված գրաֆ: Ընդ որում, G գրաֆը կանվանենք կապակցված, եթե նրա ցանկացած երկու u և v գագաթների համար G գրաֆում գոյություն ունի (u, v) ճանապարհ:

Եթե գրաֆը կապակցված չէ, ապա կողային կապակցվածության թիվը հավասար է 0-ի:

Կողային կապակցվածության թվի որոշման տարբեր ալգորիթմներից ընտրել եմ Կարգերի մեթոդը:

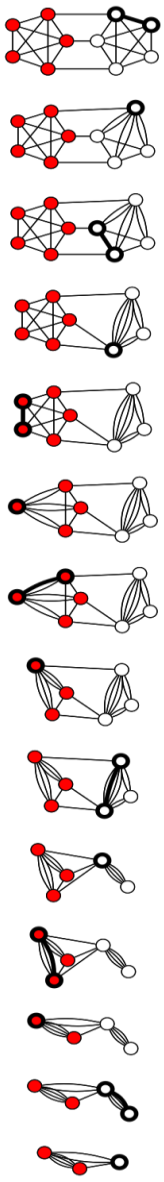
Մինչ բուն մեթոդին անցնելը, պետք է հասկանանք, թե ինչ է գրաֆի կրճատումը, որը Կարգերի ալգորիթմի հիմքն է:

Գրաֆի կրճատում

G(V, E) գրաֆի համար տրված է e = (v, w) ∈ E կողը: e-ն կանվանենք կրճատում, եթե ստեղծենք նոր գրաֆ, որը չի պարունակում e կողը: Կրճատման կատարումը՝

1. v և w գագաթները փոխարինենք նոր y գագաթով:
2. Բոլոր (v, z) ∈ E կողերը փոխարինենք նոր (y, z) կողերով: Բոլոր (w, z) ∈ E կողերը փոխարինեք (y, z)-ով:
3. Ջնջենք բոլոր (v, w) ∈ E կողերը:

Կարգերի ալգորիթմ



Ալգորիթմի իրականացում

Կարգերի ալգորիթմը պատահական սկզբունքով ընտրում է երկու գագաթ և դրանք միավորում մեկ գագաթում: Ամեն ցիկլի ժամանակ մեկ գագաթ կրճատվում է և վերջում մնում են երկու գերգագաթներ կամ, այսպես կոչված, կլաստերներ, որոնց մինջև կապերի քանակն էլ բնորոշում է կապակցվածության թիվը:

Ալգորիթմը կարող ենք ներկայացնել որպես քայլերի հաջորդականություն՝

Քանի դեռ կան երկու գագաթից ավելի գագաթներ

- Ընտրել երկու տարբեր գագաթներ,
- Միավորել դրանք մի գագաթում, թարմացնելով կրճատված գրաֆը:
- Հեռացնել ինքնահոսքերը(երբ կողմը միացնում է միևնույն գագաթը)

Վերադարձնել մնացած երկու գագաթները:

Ալգորիթմի բարդություն

Կարգերի ալգորիթմի մեկ գործարկման բարդությունը $O(n^2)$ է: Քանի որ ալգորիթմը հիմնված է պատահականությամբ որևէ թիվ ընտրելու վրա, ապա ծրագիրը պետք է գործարկվի մի քանի անգամ լավագույն արդյունքի համար: Այդ դեպքում ալգորիթմի բարդությունը կլինի $O(n^2 \log_2 n)$:

Ծրագրի ներկայացում

Ալգորիթմի իրականացումը բաղկացած է 2 քայլից

1. Ֆայլից գրաֆի ընթերցում և հարևանության մատրիցի տեսքով ներկայացում:
2. Կարգերի մեթոդի իրագործում:

Առաջին քայլն իրականացվում է convert_to_connectivity_matrix ֆունկցիայի միջոցով:

```
std::string graph_file()
{
    std::string temp;
    std::ifstream input("C:\\Users\\elent\\Desktop\\graph_input.txt");
    std::stringstream sstr;
    while (input >> sstr.rdbuf());
    temp = sstr.str();
    return temp;
}

std::tuple<matrix, int> convert_to_connectivity_matrix(std::string graph)
{
    //E = {{1,2}, {1,3},{1,4},{1,5}, {2,3},{2,4},{2,5}, {3,4},{3,5}, {4,5}, {3,6}, {6,7}, {6,8}, {6,9}, {6,10}, {7,8},{7,9},{7,10},{8,9},{8,10},{9,10}}
    std::vector<int> vec(graph.size());
    int count_of_brackets{};
    int i = 0;
    int size = 1;
    int max{};
    while (i <= graph.size())
    {
        if (graph[i] == '{') count_of_brackets++;
        if (graph[i] == '}') count_of_brackets--;
        if (isdigit(graph[i])) { ... }
        else i++;
    }
    if (count_of_brackets != 0) { ... }
    size--;
    matrix mat(max + 1, std::vector<int>(max + 1, 0));
    i = 1;
    while (i <= size)
    {
        mat[vec[i]][vec[i + 1]] = 1;
        mat[vec[i + 1]][vec[i]] = 1;
        i += 2;
    }
    return std::make_tuple(mat, max);
}
```

Երկրորդ քայլով իրագործվում է Կարգերի մեթոդը:

```
void kargers_algorithm(Graph& g)
{
    g.remove_self_loops();
    int u = 0, v = 0;
    while (g.count_vertices() > 2)
    {
        u = 0;
        v = 0;
        do
        {
            u = (rand() % g.get_size());
            v = (rand() % g.get_size());
        } while (g.get(u, v) == 0);
        g.merge_vertices(u, v);
        g.remove_self_loops();
    }

    return;
}
```

Իրագործումը կատարվում է ըստ Կարգերի ալգորիթմի(տես էջ 7):

Ֆունկցիայի ներսում օգտագործված ֆունկցիաները ներկայացված են Graph կլասում:

Վերջնական արդյունքը տպվում է մատրիցի տեսքով, որտեղ 1-երը հարևան գագաթներն են: Վերջին տողում նշվում է կապակցվածության թիվը:

```
class Graph
{
public:
    int vertices;
    matrix edges;
    Graph();

    Graph(matrix, int);
    void set(int, int, int);
    int get(int, int);

    void set_size(int);
    int get_size();
    int count_vertices();
    int count_edges();

    Graph& remove_self_loops();

    Graph& merge_vertices(int, int);
};
```

Արդյունք

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0	0	0
3	1	1	0	1	1	1	0	0	0	0
4	1	1	1	0	1	0	0	0	0	0
5	1	1	1	1	0	0	0	0	0	0
6	0	0	1	0	0	0	1	1	1	1
7	0	0	0	0	0	1	0	1	1	1
8	0	0	0	0	0	1	1	0	1	1
9	0	0	0	0	0	1	1	1	0	1
10	0	0	0	0	0	1	1	1	1	0

Vertex connectivity of the given graph is 1

Հղում

GitHub

Օգտագործված գրականության ցանկ

1. Պ.Ա. Պետրոսյան, Վ.Վ. Մկրտչյան, Ռ.Ռ. Քամալյան - Գրաֆների տեսություն
2. https://courses.cs.washington.edu/courses/cse521/16sp/521-lecture-1.pdf
3. https://en.wikipedia.org/wiki/Karger%27s_algorithm
4. Karger, David (1993). "Global Min-cuts in RNC and Other Ramifications of a Simple Mincut Algorithm"
5. http://www.columbia.edu/~cs2035/courses/ieor6614.S09/Contraction.pdf