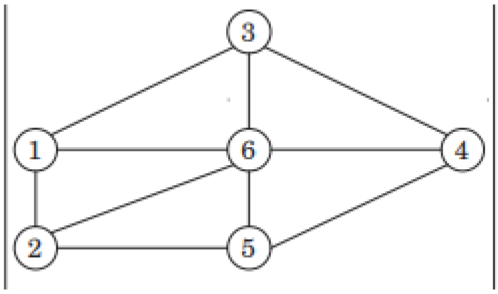
Цель работы:

- Получение практических навыков оценки надежности вычислительных сетей.

Вариант задания:

Вывод формулы вероятности существования пути в случайном графе, как функции от p.

Вариант 23



$x_i=2$
$x_j=3$

Рис. 1 - Топология случайного графа. Необходимо вычислить вероятность пути 2,3

Упрощение графа:

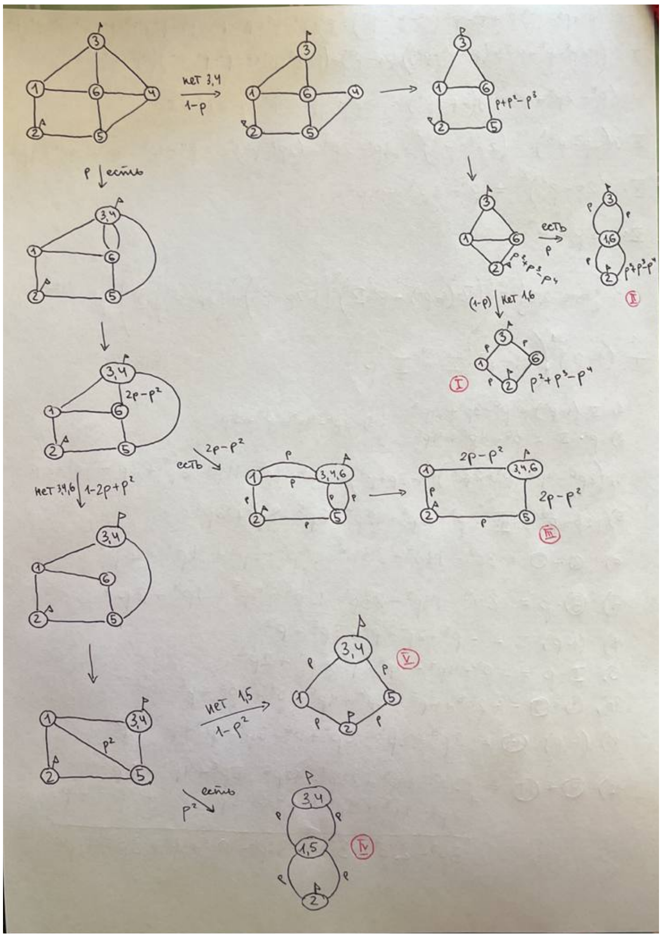


Рис. 2 - Упрощение графа

Результирующая формула для вероятности пути в общем виде выглядит следующим образом:

$$\mathrm{I}:\ (p^2+p^3-p^4)\cdot p + p^2 - p^3\cdot(p^2+p^3-p^4) = p^3+p^4-p^5+p^2-p^5-p^6+p^7 = p^7-p^6-2p^5+p^4+p^3+p^2$$

$$\mathrm{II}:\ \left(p^2+p^3-p^4+p-p\cdot(p^2+p^3-p^4)\right)\cdot(2p-p^2) = (p^2+p^3-p^4+p-p^3-p^4+p^5)\cdot(2p-p^2) =$$

$$= (p^5-2p^4+p^2+p)(2p-p^2) = -p^7+4p^6-4p^5-p^4+p^3+2p^2$$

$$\mathrm{III}:\ 2\cdot(2p^2-p^3) - (2p^2-p^3)^2 = 4p^2-2p^3-(2p^2-p^3)^2 = -p^6+4p^5-4p^4-2p^3+4p^2$$

$$\mathrm{IV}:\ (2p-p^2)^2 = p^4-4p^3+4p^2$$

$$\mathrm{V}:\ 2p^2-p^4$$

$$Pr\{\text{путь } 2,3\} = \left(\left(\mathrm{V}\overset{1}{\cdot}(1-p^2)\overset{3}{+}p^2\overset{2}{\cdot}\mathrm{IV}\right)\overset{4}{\cdot}(1-2p+p^2)\overset{6}{+}(2p-p^2)\overset{5}{\cdot}\mathrm{III}\right)\overset{7}{\cdot}p\ \overset{12}{+}$$

$$\overset{12}{+}(1-p)\overset{11}{\cdot}\left((1-p)\overset{8}{\cdot}\mathrm{I}\overset{10}{+}\mathrm{II}\overset{9}{\cdot}p\right)$$

1) $\mathrm{V}\cdot(1-p^2) = p^6-3p^4+2p^2$

2) $p^2\cdot\mathrm{IV} = p^6-4p^5+4p^5$

$\Big\}$ 3) $2p^6-4p^5+p^4+2p^2$

4) $(2p^6-4p^5+p^4+2p^2)(1-2p+p^2) = 2p^8-8p^7+11p^6-6p^5+3p^4-4p^3+2p^2$

5) $(2p-p^2)\cdot\mathrm{III} = p^8-6p^7+12p^6-6p^5-8p^4+8p^3$

6) $4+5 = 3p^8-14p^7+23p^6-12p^5-5p^4+4p^3+2p^2$

7) $6\cdot p = 3p^9-14p^8+23p^7-12p^6-5p^5+4p^4+2p^3$

8) $(1-p)\cdot\mathrm{I} = -p^8+2p^7+p^6-3p^5+p^2$

9) $\mathrm{I}\cdot p = -p^8+4p^7-4p^6-p^5+p^4+2p^3$

10) $8+9 = -2p^8+6p^7-3p^6-4p^5+p^4+2p^3+p^2$

11) $(1-p)\cdot 10 = 2p^9-8p^8+9p^7+p^6-5p^5-p^4+p^3+p^2$

12) $7+11 =$

$$\begin{array}{r} 3p^9-14p^8+23p^7-12p^6-5p^5+4p^4+2p^3 \\ +\ 2p^9-8p^8+9p^7+p^6-5p^5-p^4+p^3+p^2 \\ \hline 5p^9-22p^8+32p^7-11p^6-10p^5+3p^4+3p^3+p^2 \end{array}$$

Рис. 3 - Результирующая формула для вероятности пути в общем виде

Описание программы вычисления вероятности существования пути в случайном графе с использованием алгоритма полного перебора.

Задача программы состоит в том, чтобы обойти все подграфы и если в этом подграфе существует путь, то учесть вероятность его появления в общую вероятность.

В программе граф задаётся в виде списка смежности. Перебор осуществляется по всем вероятностям существования рёбер от 0 до 1 с шагом 0.1 и всем подграфам, количество которых 2^L, где L – количество ребер. Для нахождения пути в подграфе используется алгоритм подобный DFS.

Результаты работы программы:

```
p = 0
probability: 0
probability PRACT: 0

p = 0.1
probability: 0.013192
probability PRACT: 0.013192

p = 0.2
probability: 0.0652518
probability PRACT: 0.0652518

p = 0.3
probability: 0.168634
probability PRACT: 0.168634

p = 0.4
probability: 0.320666
probability PRACT: 0.320666
```

```
p = 0.5
probability: 0.501953
probability PRACT: 0.501953

p = 0.6
probability: 0.682652
probability PRACT: 0.682652

p = 0.7
probability: 0.83331
probability PRACT: 0.83331

p = 0.8
probability: 0.935404
probability PRACT: 0.935404

p = 0.9
probability: 0.986874
probability PRACT: 0.986874

p = 1
probability: 1
probability PRACT: 1
```

Рис. 4 - Результат работы программы.

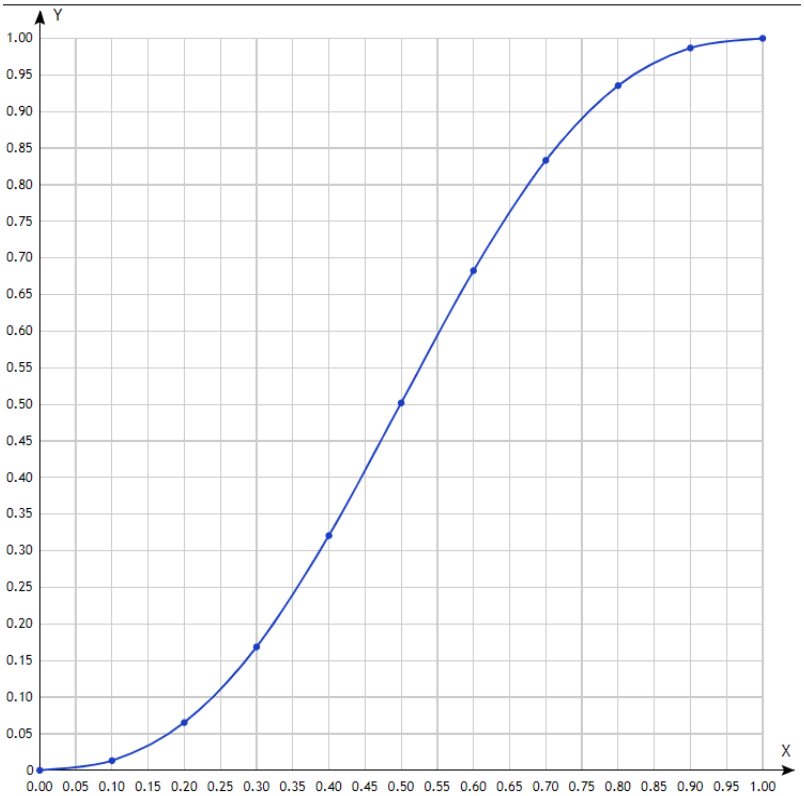


Рис. 5 - Зависимость вероятности существования пути от вероятности существования рёбер при полном переборе всех подграфов

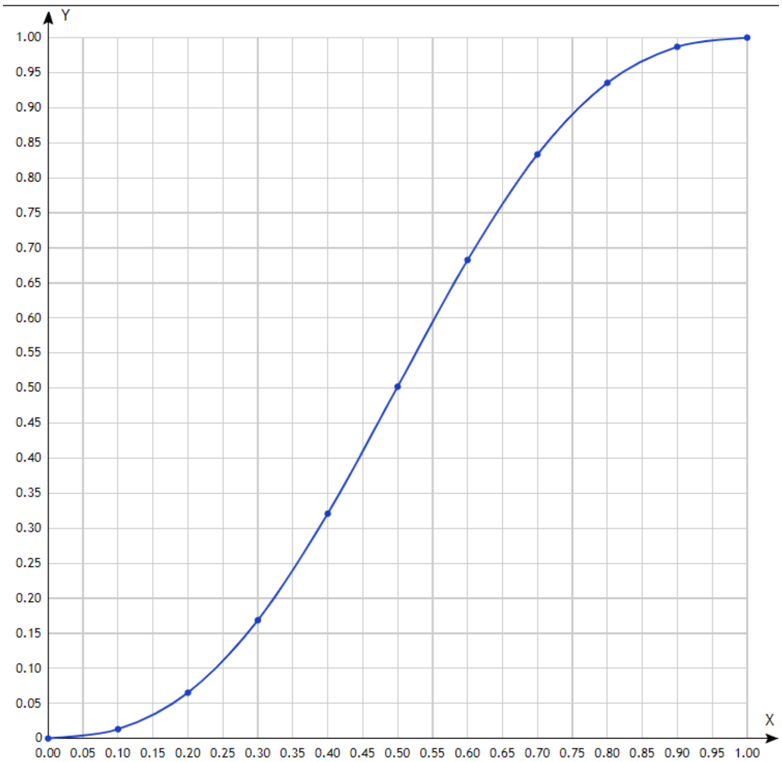


Рис. 6 Зависимость вероятности существования пути от вероятности существования рёбер при подстановке значений в результирующую формулу

Вывод:

- При возрастании вероятности существования рёбер возрастает вероятность существования пути в обоих случаях (При полном переборе и путём упрощения графа)
- Показатели вероятностей при полном переборе и упрощении графа совпадают.

Листинг:

```
using namespace std;
#include <iostream>
#include <math.h>

int bin_code[1000] = { 0 };
int matrix[11][11] = { 0,0 };
int x;
int y;
double sum[11] = {};
double p[11] = { 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 };
float sumPR[11] = {};
```

```
int N = 0;
int ogranic = 0;

int visit[10] = { 0 };

void test(int i) {
	for (int j = 0; j < ogranic; j++) {
		if (matrix[i][j] == 1 && visit[j] == 0) {
			visit[j] = 1;
			test(j);
		}
	}
}

int main() {

	cout << "Enter:" << endl << "N: ";
	cin >> N;

	cout << "x: "; cin >> x;
	x--;
	cout << "y: "; cin >> y;
	y--;

	int flag = 0; //ограничитель
	int Perenos = 1;

	int matrix_dano[10][10] = {
		{0,1,1,0,0,1},
		{0,0,0,0,1,0},
		{0,0,0,1,0,1},
		{0,0,0,0,1,1},
		{0,0,0,0,0,1},
		{0,0,0,0,0,0},
	};

	long bin_code_dano[1000] = { 0 };

	int t = 1;
	int k = 0;

	for (int i = 0; i < N; i++) {
		for (int j = 0 + t; j < N; j++) {
			bin_code_dano[k] = matrix_dano[i][j];
			if (bin_code_dano[k] == 1) {
				ogranic++;
			}
			k++;
		}
		t++;
	}
```

```
//перебор
while (flag != pow(2, ogranic) - 1){

	for (int i = 0; i < N; i++) {
		visit[i] = 0;
	}

	Perenos = 1;
	int col_1 = 0;

	for (long i = ogranic-1; i >= 0; i--) {
		if (Perenos == 1) {
			if (bin_code[i] == 1) {
				bin_code[i] = 0;
			}
			else {
				bin_code[i] = 1;
				Perenos = 0;
			}
		}
	}

	int st = 0;
	for (int i = 0; i < N; i++) {
		for (int j = 0; j < N; j++) {
			if (matrix_dano[i][j] == 1) {
				matrix[i][j] = bin_code[st];
				matrix[j][i] = bin_code[st];
				st++;
			}
		}
	}

	visit[x] = 1;
	test(x);//обычный DFS

	double pr[11] = {1,1,1,1,1,1,1,1,1,1,1};

	if (visit[y] == 1) {

		for (int i = 0; i < ogranic; i++) {
			if (bin_code[i] == 0) {
				for (int i = 0; i < 11; i++) {
					pr[i] = pr[i] * (1.0 - p[i]);
				}
			}
			else {
				for (int i = 0; i < 11; i++) {
					pr[i] = pr[i] * p[i];
				}
```

```
                    }
                }
                for (int i = 0; i < 11; i++) {
                    sum[i] = sum[i] + pr[i];
                }
            }
            else {}
            flag++;
        }
        for (int i = 0; i < 11; i++) {
            sumPR[i] = 5 * pow(p[i],9) - 22 * pow(p[i],8) + 32 * pow(p[i],7) - 11
* pow(p[i],6) - 10 * pow(p[i],5) + 3 * pow(p[i],4) + 3 * pow(p[i],3) + pow(p[i],2);
        }
        for (int i = 0; i < 11; i++) {
            cout << "p = " <<p[i] << endl << "probability: " << sum[i] << endl;
            cout << "probability PRACT: " << sumPR[i] << endl << endl;
        }
    }
```