Zadanie programistyczne nr 2

# Kod programu

Język – C++.

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

#include <vector>

#include <map>

/\*\*

 \* @brief Metoda odpowiedzialna za utworzenie n-elementowej listy, sluzacej potem do utworzenia losowej permutacji.

 \*

 \* @param n Ilosc elementow w liscie.

 \* @return int\* Lista postaci [n] - przechowujaca n kolejnych liczb naturalnych, zaczynajac od 1.

 \*/

int\* createPermutationBase(const int n) {

    int\* base = new int[n];

    for(int i=0; i < n; i++) {

        base[i] = i+1;

    }

    return base;

}

/\*\*

 \* @brief Metoda odpowiedzialna za zamienienie miejscami wartosci listy.

 \*

 \* @param i Indeks pierwszej wartosci do zamienienia miejscami.

 \* @param j Indeks drugiej wartosci do zamienienia miejscami.

 \* @param list Lista, na ktorej wykonywana jest zamiana indeksow dwoch wartosci.

 \*/

void swapIndexes(int i, int j, int\* list) {

    int tmp = list[i];

    list[i] = list[j];

    list[j] = tmp;

}

/\*\*

 \* @brief Metoda generujaca losowa permutacje.

 \*

 \* @param n Ilosc elementow permutacji.

 \* @return int\* Tablica przechowujaca losowa permutacje.

 \*/

int\* randomPermutation(const int n) {

    int\* result = createPermutationBase(n);

    int j;

    for(int i = n-1; i > 0; i--) {

        j = std::rand() % (i + 1);

        swapIndexes(i, j, result);

    }

    return result;

}

/\*\*

 \* @brief Metoda odpowiedzialna za znalezienie indeksu, pod ktorym znajduje sie kolejny cykl w permutacji.

 \*

 \* @param checking\_base Lista okreslajaca czy dany indeks byl sprawdzany czy nie.

 \* @param size Ilosc elementow w permutacji.

 \* @return int Indeks, pod ktorym znajduje sie kolejny cykl w permutacji.

 \*/

int findUncheckedIndex(const int\* checking\_base, int size) {

    for(int i=0; i < size; i++) {

        if(checking\_base[i] != 0)

            return i;

    }

    return -1;

}

/\*\*

 \* @brief Procedura rozkladajaca permutacje na cykle.

 \*

 \* @param permutation Losowo wygenerowana permutacja.

 \* @param size Ilosc elementow permutacji.

 \* @return std::vector<std::vector<int>> Dwuwymiarowa tablica przechowujaca cykle permutacji.

 \*/

std::vector<std::vector<int>> getCycles(const int\* permutation, int size) {

    std::vector<std::vector<int>> cycles;

    std::vector<int> cycle;

    if(size == 0)

        return cycles;

    int\* checking\_base = createPermutationBase(size);

    int first\_element;

    int index = 0;

    int count = 0;

    do {

        first\_element = index + 1;

        checking\_base[index] = 0;

        int next\_element = first\_element;

        do {

            cycle.push\_back(next\_element);

            next\_element = permutation[next\_element - 1];

            checking\_base[next\_element - 1] = 0;

        } while(next\_element != first\_element);

        cycles.push\_back(cycle);

        cycle.clear();

        index = findUncheckedIndex(checking\_base, size);

    } while(index != -1);

    return cycles;

}

/\*\*

 \* @brief Test poprawnosci - znalezienie wartosci oczekiwanej liczby cykli permutacji.

 \*

 \*/

void meanNumberOfCyclesTest() {

    std::ofstream file("test.txt");

    int n = 100;

    int k = 1000;

    file << "n;E(X)\n";

    for(int i = 1; i <= 100; i++) {

        double count\_cycles = 0;

        for(int j=0; j < k; j++) {

            int\* permutation = randomPermutation(i);

            std::vector<std::vector<int>> cycles = getCycles(permutation, i);

            count\_cycles += (double)cycles.size();

        }

        double mean\_number\_of\_cycles = count\_cycles / (double)k;

        file << i << ";" << mean\_number\_of\_cycles << "\n";

    }

    file.close();

}

/\*\*

 \* @brief Zmienna losowa - metoda ta zwraca dlugosc najdluzszego cycklu w permutacji.

 \*

 \* @param permutation Losowo wygenerowana permutacja.

 \* @param size Ilosc elementow permutacji.

 \* @return int Dlugosc najdluzszego cyklu w permutacji.

 \*/

int longestCycle(int\* permutation, int size) {

    std::vector<std::vector<int>> cycles = getCycles(permutation, size);

    int max = 0;

    for(std::vector<int> cycle : cycles) {

        if(cycle.size() > max)

            max = cycle.size();

    }

    return max;

}

/\*\*

 \* @brief Glowne zadanie - znalezienie wartosci oczekiwanej najwiekszej dlugosci cycklu w rozkladzie permutacji na cykle.

 \*

 \*/

void mainTask() {

    std::ofstream file("main\_task.txt");

    int n = 100;

    int k = 1000;

    file << "n;M(X)\n";

    for(int i=1; i <= n; i++) {

        double probability;

        int longest\_cycle = 0;

        double sum = 0;

        std::map<int, int> cycle\_length\_and\_count;

        // generujemy k permutacji i na ich podstawie liczymy wartosc oczekiwana dla danej ilosci elementow permutacji

        for(int j=0; j < k; j++) {

            int\* permutation = randomPermutation(i);

            if(longestCycle(permutation, i) > longest\_cycle) {

                longest\_cycle = longestCycle(permutation, i);

                cycle\_length\_and\_count[longest\_cycle] = 1;

            } else {

                cycle\_length\_and\_count[longestCycle(permutation, i)] += 1;

            }

        }

        for(auto const & [cycle\_length, count] : cycle\_length\_and\_count) {

            probability = (double)count / 1000.0;

            sum += probability \* cycle\_length;

        }

        file << i << ";" << sum << "\n";

    }

    file.close();

}

int main(int argc, char const \*argv[])

{

    std::srand(std::time(NULL));

    meanNumberOfCyclesTest();

    mainTask();

    return 0;

}

# Test poprawności

|  |  |
| --- | --- |
| n | E(X) |
| 1 | 1 |
| 2 | 1,501 |
| 3 | 1,846 |
| 4 | 2,06 |
| 5 | 2,252 |
| 6 | 2,401 |
| 7 | 2,645 |
| 8 | 2,745 |
| 9 | 2,799 |
| 10 | 2,984 |
| 11 | 3,055 |
| 12 | 3,103 |
| 13 | 3,234 |
| 14 | 3,281 |
| 15 | 3,303 |
| 16 | 3,375 |
| 17 | 3,497 |
| 18 | 3,536 |
| 19 | 3,554 |
| 20 | 3,633 |
| 21 | 3,618 |
| 22 | 3,725 |
| 23 | 3,733 |
| 24 | 3,829 |
| 25 | 3,814 |
| 26 | 3,875 |
| 27 | 3,906 |
| 28 | 3,916 |
| 29 | 3,95 |
| 30 | 3,981 |
| 31 | 4,095 |
| 32 | 4,106 |
| 33 | 4,066 |
| 34 | 4,059 |
| 35 | 4,116 |
| 36 | 4,282 |
| 37 | 4,205 |
| 38 | 4,186 |
| 39 | 4,207 |
| 40 | 4,285 |
| 41 | 4,393 |
| 42 | 4,267 |
| 43 | 4,381 |
| 44 | 4,429 |
| 45 | 4,299 |
| 46 | 4,38 |
| 47 | 4,452 |
| 48 | 4,504 |
| 49 | 4,443 |
| 50 | 4,484 |
| 51 | 4,55 |
| 52 | 4,561 |
| 53 | 4,508 |
| 54 | 4,555 |
| 55 | 4,564 |
| 56 | 4,623 |
| 57 | 4,602 |
| 58 | 4,706 |
| 59 | 4,58 |
| 60 | 4,565 |
| 61 | 4,624 |
| 62 | 4,708 |
| 63 | 4,685 |
| 64 | 4,636 |
| 65 | 4,848 |
| 66 | 4,682 |
| 67 | 4,767 |
| 68 | 4,778 |
| 69 | 4,827 |
| 70 | 4,805 |
| 71 | 4,787 |
| 72 | 4,828 |
| 73 | 4,887 |
| 74 | 4,85 |
| 75 | 4,846 |
| 76 | 4,935 |
| 77 | 4,888 |
| 78 | 4,866 |
| 79 | 4,963 |
| 80 | 4,975 |
| 81 | 4,881 |
| 82 | 4,981 |
| 83 | 4,917 |
| 84 | 5,053 |
| 85 | 4,99 |
| 86 | 4,971 |
| 87 | 5,063 |
| 88 | 5,058 |
| 89 | 5,071 |
| 90 | 5,129 |
| 91 | 5,07 |
| 92 | 5,133 |
| 93 | 5,138 |
| 94 | 5,034 |
| 95 | 5,145 |
| 96 | 5,146 |
| 97 | 5,176 |
| 98 | 5,116 |
| 99 | 5,214 |
| 100 | 5,28 |

Dla porównania, wykres 100 pierwszych liczb harmonicznych.

# Główne zadanie

### Metoda rozwiązania

Zadanie zostało przeze mnie rozwiązane poprzez wygenerowanie dla danego n 1000 permutacji, a następnie zmapowanie długości najdłuższego cyklu oraz ilości permutacji z takim cyklem.

Na podstawie tej mapy obliczone zostało prawdopodobieństwo wygenerowania permutacji z wybraną długością najdłuższego cyklu, a na koniec uzyskana została wartość oczekiwana.

### Wyniki głównego zadania

|  |  |
| --- | --- |
| n | M(X) |
| 1 | 1 |
| 2 | 1,492 |
| 3 | 2,191 |
| 4 | 2,824 |
| 5 | 3,454 |
| 6 | 4,05 |
| 7 | 4,662 |
| 8 | 5,286 |
| 9 | 5,911 |
| 10 | 6,604 |
| 11 | 7,111 |
| 12 | 7,909 |
| 13 | 8,447 |
| 14 | 8,998 |
| 15 | 9,646 |
| 16 | 10,224 |
| 17 | 10,973 |
| 18 | 11,454 |
| 19 | 12,256 |
| 20 | 12,798 |
| 21 | 13,178 |
| 22 | 14,062 |
| 23 | 14,57 |
| 24 | 15,455 |
| 25 | 15,927 |
| 26 | 16,783 |
| 27 | 17,351 |
| 28 | 17,677 |
| 29 | 18,679 |
| 30 | 19,073 |
| 31 | 19,345 |
| 32 | 20,29 |
| 33 | 20,091 |
| 34 | 21,68 |
| 35 | 22,493 |
| 36 | 22,827 |
| 37 | 23,331 |
| 38 | 24,592 |
| 39 | 24,562 |
| 40 | 25,084 |
| 41 | 26,243 |
| 42 | 26,445 |
| 43 | 27,107 |
| 44 | 27,576 |
| 45 | 28,828 |
| 46 | 29,154 |
| 47 | 29,413 |
| 48 | 30,208 |
| 49 | 31,411 |
| 50 | 31,532 |
| 51 | 32,214 |
| 52 | 32,407 |
| 53 | 33,438 |
| 54 | 33,853 |
| 55 | 34,784 |
| 56 | 35,754 |
| 57 | 35,942 |
| 58 | 36,29 |
| 59 | 37,284 |
| 60 | 38,611 |
| 61 | 38,83 |
| 62 | 38,616 |
| 63 | 40,011 |
| 64 | 40,549 |
| 65 | 41,153 |
| 66 | 41,304 |
| 67 | 43,089 |
| 68 | 42,348 |
| 69 | 43,387 |
| 70 | 44,147 |
| 71 | 43,706 |
| 72 | 44,427 |
| 73 | 46,319 |
| 74 | 46,816 |
| 75 | 47,481 |
| 76 | 48,19 |
| 77 | 48,712 |
| 78 | 49,381 |
| 79 | 49,348 |
| 80 | 49,734 |
| 81 | 51,11 |
| 82 | 51,759 |
| 83 | 51,99 |
| 84 | 52,991 |
| 85 | 52,645 |
| 86 | 54,462 |
| 87 | 55,646 |
| 88 | 55,247 |
| 89 | 55,869 |
| 90 | 57,266 |
| 91 | 56,242 |
| 92 | 57,214 |
| 93 | 59,264 |
| 94 | 60,094 |
| 95 | 59,497 |
| 96 | 60,388 |
| 97 | 59,254 |
| 98 | 61,502 |
| 99 | 62,301 |
| 100 | 62,855 |

### Wnioski

Zauważyć można, że wartość oczekiwana długości najdłuższego cyklu w losowo wygenerowanej permutacji rośnie liniowo, więc jej stosunek do ilości elementów w permutacji jest stały. Znajdźmy go:

|  |  |
| --- | --- |
| n | M(X)/n |
| 1 | 1 |
| 2 | 0,746 |
| 3 | 0,730333 |
| 4 | 0,706 |
| 5 | 0,6908 |
| 6 | 0,675 |
| 7 | 0,666 |
| 8 | 0,66075 |
| 9 | 0,656778 |
| 10 | 0,6604 |
| 11 | 0,646455 |
| 12 | 0,659083 |
| 13 | 0,649769 |
| 14 | 0,642714 |
| 15 | 0,643067 |
| 16 | 0,639 |
| 17 | 0,645471 |
| 18 | 0,636333 |
| 19 | 0,645053 |
| 20 | 0,6399 |
| 21 | 0,627524 |
| 22 | 0,639182 |
| 23 | 0,633478 |
| 24 | 0,643958 |
| 25 | 0,63708 |
| 26 | 0,6455 |
| 27 | 0,64263 |
| 28 | 0,631321 |
| 29 | 0,644103 |
| 30 | 0,635767 |
| 31 | 0,624032 |
| 32 | 0,634063 |
| 33 | 0,608818 |
| 34 | 0,637647 |
| 35 | 0,642657 |
| 36 | 0,634083 |
| 37 | 0,630568 |
| 38 | 0,647158 |
| 39 | 0,629795 |
| 40 | 0,6271 |
| 41 | 0,640073 |
| 42 | 0,629643 |
| 43 | 0,630395 |
| 44 | 0,626727 |
| 45 | 0,640622 |
| 46 | 0,633783 |
| 47 | 0,625809 |
| 48 | 0,629333 |
| 49 | 0,641041 |
| 50 | 0,63064 |
| 51 | 0,631647 |
| 52 | 0,623212 |
| 53 | 0,630906 |
| 54 | 0,626907 |
| 55 | 0,632436 |
| 56 | 0,638464 |
| 57 | 0,630561 |
| 58 | 0,62569 |
| 59 | 0,631932 |
| 60 | 0,643517 |
| 61 | 0,636557 |
| 62 | 0,622839 |
| 63 | 0,635095 |
| 64 | 0,633578 |
| 65 | 0,633123 |
| 66 | 0,625818 |
| 67 | 0,643119 |
| 68 | 0,622765 |
| 69 | 0,628797 |
| 70 | 0,630671 |
| 71 | 0,615577 |
| 72 | 0,617042 |
| 73 | 0,634507 |
| 74 | 0,632649 |
| 75 | 0,63308 |
| 76 | 0,634079 |
| 77 | 0,632623 |
| 78 | 0,63309 |
| 79 | 0,624658 |
| 80 | 0,621675 |
| 81 | 0,630988 |
| 82 | 0,631207 |
| 83 | 0,626386 |
| 84 | 0,630845 |
| 85 | 0,619353 |
| 86 | 0,633279 |
| 87 | 0,639609 |
| 88 | 0,627807 |
| 89 | 0,627742 |
| 90 | 0,636289 |
| 91 | 0,618044 |
| 92 | 0,621891 |
| 93 | 0,637247 |
| 94 | 0,639298 |
| 95 | 0,626284 |
| 96 | 0,629042 |
| 97 | 0,610866 |
| 98 | 0,627571 |
| 99 | 0,629303 |
| 100 | 0,62855 |