

WYDZIAŁ
MATEMATYKI
I FIZYKI STOSOWANEJ
POLITECHNIKI RZESZOWSKIEJ

Przedmiot: Algorytmy i Struktury Danych

Damian Kobyliński, nr. albumu: 173154

Algorytm przeszukiwania maksymalnego iloczynu dwóch
liczb w różnego rodzaju tabelach jednowymiarowych
wykorzystując język C++

Opiekun pracy: dr inż. Mariusz Borkowski prof. PRz

Rzeszów

Spis treści

1	Wstęp	4
2	Opis problemu	4
3	Opis podstaw teoretycznych zagadnienia	4
4	Opis szczegółów implementacji problemu	6
4.1	Schemat blokowy algorytmu	6
4.2	Pseudokod algorytmu	7
4.3	Przykład działania programu	8
4.4	Złożoność obliczeniowa	8
5	Podsumowanie	10
6	Appendix	10

1 Wstęp

Otrzymane zadanie projektowe ma na celu wyznaczenie wszystkich możliwych par liczb całkowitych występujących we wszelakich tablicach jednowymiarowych (mających różną ilość elementów, zbiór elementów może obejmować cały zbiór liczb całkowitych, jak i tablica może być nie uporządkowana).

2 Opis problemu

Zadanie projektowe opiera się na utworzeniu algorytmu przeszukiwania wszystkich iloczynów dwóch liczb całkowitych, którego wartość jest największa, w tablicy jednowymiarowej (nazywanej później także tablicą główną). W tablicy może istnieć więcej niż jedna para liczb, których iloczyn dawał tak samo największą wartość iloczynu, w obrębie tablicy.

3 Opis podstaw teoretycznych zagadnienia

Osobistym podejściem do rozwiązania problemu jest utworzenie dwóch nowych tablic jednowymiarowych - jedna będzie przechowywała parę największych cyfr z tablic oraz wynik jego iloczynu, druga natomiast będzie przechowywała analogicznie parę najmniejszych cyfr jak i ich iloczyn. Obydwie tablice są 3 elementowe, do których w czasie ich inicjowania jako pierwsze dwa elementy umieszczamy dwa pierwsze elementy z tablicy wszystkich elementów. Następnym zadaniem jest przechodzenie przez wszystkie elementy tablicy i sprawdzanie następujących warunków:

1. Pierwsza pętla mająca odszukać największe elementy tablicy

- Jeżeli dany element pętli jest równy pierwszemu elementowi tablicy zbierającej parę największych cyfr i nie jest to pierwszy element tablicy - wprowadź jego wartość w pierwszym miejscu tablicy zbierającej największe cyfry.
- Jeżeli dany element pętli jest większy od zerowego elementu tablicy zbierającej parę największych cyfr - przenieść wartość zerowego elementu tablicy zbierającej największe cyfry na pierwszy, wprowadź element z pętli w zerowym miejscu tablicy zbierającej największe cyfry.
- Jeżeli dany element pętli jest mniejszy od zerowego elementu tablicy zbierającej parę największych cyfr, jednocześnie większy od pierwszego elementu danej tablicy - wprowadź jego wartość w pierwszym miejscu tablicy zbierającej największe cyfry.

2. Druga pętla mająca odszukać najmniejsze elementy tablicy

- Jeżeli dany element pętli jest równy pierwszemu elementowi tablicy zbierającej parę najmniejszych cyfr i nie jest to pierwszy element tablicy - wprowadź jego wartość w pierwszym miejscu tablicy zbierającej najmniejsze cyfry.
- Jeżeli dany element pętli jest mniejszy od zerowego elementu tablicy zbierającej parę najmniejszych cyfr - przenieść wartość zerowego elementu tablicy zbierającej najmniejsze cyfry na pierwszy, wprowadź element z pętli w zerowym miejscu tablicy zbierającej najmniejsze cyfry.
- Jeżeli dany element pętli jest większy od zerowego elementu tablicy zbierającej parę najmniejszych cyfr, jednocześnie mniejszy od pierwszego elementu danej tablicy - wprowadź jego wartość w pierwszym miejscu tablicy zbierającej najmniejsze cyfry.

Po wykonaniu dwóch pętli liczymy 2 element, każdej z naszej dodatkowych dwóch tablic będący iloczynem zerowego oraz pierwszego elementu danej tablicy. Po wykonaniu iloczynu porównujemy wartości tych elementów między dwoma tablicami:

- Jeżeli iloczyn jest większy w elemencie drugim tablicy zawierającej największe cyfry - wypisujemy parę liczb (zerowy i pierwszy element tablicy) z tablicy zawierającej największe cyfry z tablicy głównej.
- Jeżeli iloczyn jest większy w elemencie drugim tablicy zawierającej najmniejsze cyfry - wypisujemy parę liczb (zerowy i pierwszy element tablicy) z tablicy zawierającej najmniejsze cyfry z tablicy głównej.
- Jeżeli iloczyny drugich elementów, każdej z tablicy są sobie równe - wypisz parę liczb (zerowy i pierwszy element tablicy) z obydwu tablic.

4 Opis szczegółów implementacji problemu

Podstawą działania zadania jest utworzenie tablicy jednowymiarowej, która posiada co najmniej 2 elementy, aby można było policzyć iloczyn dwóch liczb. W celu obliczenia wielkości tablicy wykorzystuję funkcję `sizeof()` zapisując w zmiennej o typie całkowitym wielkość pamięci jaką zajmuje tablica, podzieloną przez wielkość pamięci jaką zajmuje typ danych całkowity `int`.

Następnym krokiem jest iteracja po dwóch następujących po sobie pętlach, sprawdzając warunki, o których działanie przedstawiłem w Rozdziale 3 sprawozdania - Opis podstaw teoretycznych zagadnienia - w liście numerowanej treść urzytku danej pętli, natomiast załączonej do każdej z nich listy wypunktowanej przedstawiłem warunki, które muszą pozostać spełnione.

4.1 Schemat blokowy algorytmu

Z powodu wielkości obrazu ze schematem blokowym algorytmu, obraz jego znajduje się `images/schemat_blokowy.png` w repozytorium projektu na stronie GitHub.

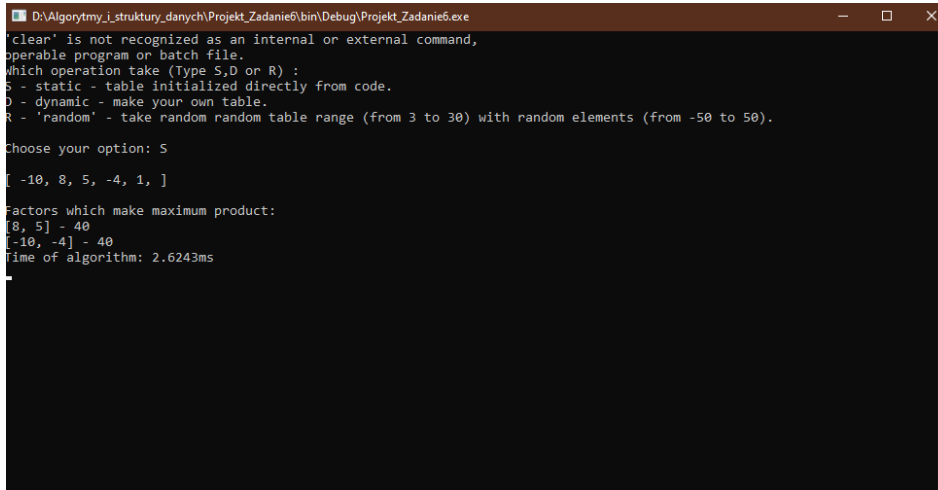
4.2 Pseudokod algorytmu

K1 START
K2 Wczytaj tablica[0], ... , tablica[rozmiar]
K3 rozmiar_tablicy = rozmiar
K4 tablica_maksymalnych_par = tablica[0],tablica[1],0
K5 tablica_minimalnych_par = tablica[0],tablica[1],0
K6 i = 0
K7 Jeżeli i < rozmiar_tablicy wykonuj
K7.1 Jeżeli tablica[i] == tablica_maksymalnych_par[0] oraz i != 0
K7.2 tablica_maksymalnych_par[1] = tablica[i]
K7.3 Jeżeli tablica[i] > tablica_maksymalnych_par[0]
K7.4 tablica_maksymalnych_par[1] = tablica_maksymalnych_par[0]
K7.5 tablica_maksymalnych_par[0] = tablica[i]
K7.6 Jeżeli tablica[i] < tablica_maksymalnych_par[0] oraz tablica[i] > tablica_maksymalnych_par[1]
K7.7 tablica_maksymalnych_par[1] = tablica[i]
K7.8 i = i + 1
K7.9 Powrót do K7
K8 i = 0
K9 Jeżeli i < rozmiar_tablicy wykonuj
K9.1 Jeżeli tablica[i] == tablica_minimalnych_par[0] oraz i != 0
K9.2 tablica_minimalnych_par[1] = tablica[i]
K9.3 Jeżeli tablica[i] < tablica_minimalnych_par[0]
K9.4 tablica_minimalnych_par[1] = tablica_minimalnych_par[0]
K9.5 tablica_minimalnych_par[0] = tablica[i]
K9.6 Jeżeli tablica[i] > tablica_minimalnych_par[0] oraz tablica[i] < tablica_minimalnych_par[1]
K9.7 tablica_minimalnych_par[1] = tablica[i]
K10 tablica_maksymalnych_par[2] = tablica_maksymalnych_par[0] * tablica_maksymalnych_par[1]
K11 tablica_minimalnych_par[2] = tablica_minimalnych_par[0] * tablica_minimalnych_par[1]
K12 STOP

4.3 Przykład działania programu

Na przedstawionym rysunku 1. można zobaczyć wynik działania algorytmu w przypadku, kiedy ustalimy sobie statyczną tablicę jednowymiarową. Obraz można także zobaczyć elektronicznie, znajduje się on w repozytorium projektu, w miejscu:

images/Przykład_działania_kodu.png

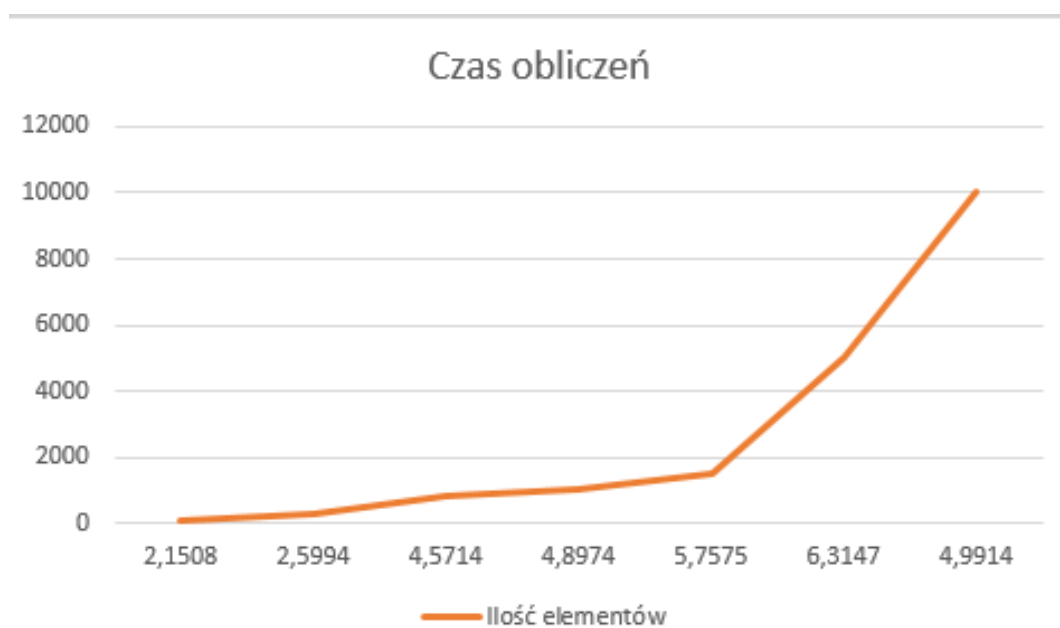


```
D:\Algorytmy_i_struktury_danych\Projekt_Zadanie6\bin\Debug\Projekt_Zadanie6.exe
'clear' is not recognized as an internal or external command,
operable program or batch file.
Which operation take (Type S,D or R) :
S - static - table initialized directly from code.
D - dynamic - make your own table.
R - 'random' - take random random table range (from 3 to 30) with random elements (from -50 to 50).
Choose your option: S
[-10, 8, 5, -4, 1, ]
Factors which make maximum product:
[8, 5] - 40
[-10, -4] - 40
Time of algorithm: 2.6243ms
```

4.4 Złożoność obliczeniowa

Z racji występowania po sobie dwóch pętli typu `for` w algorytmie, złożoność obliczeniowa programu wynosi $O(n)$.

Prezentacja czasu obliczeń z wykorzystaniem tablicy z wartościami pseudorandomowymi, od wartości -20 do 20. Wszystkie zrzuty prezentujące wyniki obliczeń mieszczą się w folderze `images/Ilosc_iteracji`.



Rysunek 1: Rysunek 1. Czas obliczeń algorytmu

5 Podsumowanie

Algorytm działa w pełni poprawnie. Rozwiązuje zadany problem przy czym posiada bardzo małą złożoność obliczeniową, czego wynikiem jest jego szybkie działanie.

6 Appendix

Kod algorytmu

```
1 #include <iostream>
2 #include <cstdio>
3 #include <ctime>
4 #include <cstdlib>
5 #include <chrono>
6 #include <cmath>
7
8 using namespace std;
9
10 //Printing which table from our additional table is greater or printing
    both if there are equal
11 void DefineMultiple(int highest_arr[3], int lowest_arr[3]){
12     if(highest_arr[2] == lowest_arr[2]){
13         cout<<"["<<highest_arr[0]<<"", "<<highest_arr[1]<<"] - "<<
            highest_arr[2]<<endl;
14         cout<<"["<<lowest_arr[0]<<"", "<<lowest_arr[1]<<"] - "<<
            lowest_arr[2]<<endl;
15     }else{
16         if(highest_arr[2] > lowest_arr[2]){
17             cout<<"["<<highest_arr[0]<<"", "<<highest_arr[1]<<"] - "<<
                highest_arr[2]<<endl;
18         }else{
19             cout<<"["<<lowest_arr[0]<<"", "<<lowest_arr[1]<<"] - "<<
                lowest_arr[2]<<endl;
20         }
21     }
22 }
23
24 //Presenting Array function
25 void showArray( int arr[], int leangth_of_the_array){
26     cout<<endl<<"[";
27     for(int i=0;i<leangth_of_the_array;i++){
28         cout<<arr[i]<<"", ";
29     }
30     cout<<"]"<<endl<<endl;
31 }
32
33 //Whole function with working algorithm and reference to DefineMultiple
    function
34 void Algorithm( int arr[], int leangth_of_the_array, int highest_arr
    [3], int lowest_arr[3])
35 {
36
37     for(int i=0; i<leangth_of_the_array;i++)
38     {
39         if(arr[i] == highest_arr[0] && i != 0){
40             highest_arr[1] = arr[i];
41         }
42         else if(arr[i] > highest_arr[0]){
```

```

43         highest_arr[1] = highest_arr[0];
44         highest_arr[0] = arr[i];
45     }
46     else if(arr[i] < highest_arr[0] && arr[i] > highest_arr[1]){
47         highest_arr[1] = arr[i];
48     }
49 }
50 for(int i=0;i<length_of_the_array;i++){
51     if(arr[i] == lowest_arr[0] && i != 0){
52         lowest_arr[1] = arr[i];
53     }
54     else if(arr[i] < lowest_arr[0]){
55         lowest_arr[1] = lowest_arr[0];
56         lowest_arr[0] = arr[i];
57     }
58     else if(arr[i] > lowest_arr[0] && arr[i] < lowest_arr[1]){
59         lowest_arr[1] = arr[i];
60     }
61 }
62
63 highest_arr[2] = highest_arr[0]*highest_arr[1];
64 lowest_arr[2] = lowest_arr[0]*lowest_arr[1];
65
66 cout<<"Factors which make maximum product:"<<endl;
67
68 DefineMultiple(highest_arr, lowest_arr);
69 }
70 //Menu - Listing of available function and table options
71 void StartingText(){
72     cout<<"Which operation take (Type S,D or R) : "<<endl;
73     cout<<"S - static - table initialized directly from code."<<endl;
74     cout<<"D - dynamic - make your own table."<<endl;
75     cout<<"R - 'random' - take random random table range (from 3 to 30)
76     with random elements (from -50 to 50)."<<endl;
77     cout<<endl;
78     cout<<"Choose your option: ";
79 }
80 char whichOperationToTake;
81
82 int main()
83 {
84     srand (( unsigned ) time ( NULL ) ) ;
85     while(true){
86         //clear console view depending on your OS
87         #if __linux__
88             system("clear");
89         #elif _WIN32
90             system("cls");
91         #else
92             system("clear");
93         #endif
94         StartingText();
95         cin.get(whichOperationToTake);
96         //Depending on your which option you took, the future action
97         will take place
98         switch(whichOperationToTake)
99         {
100             case 'S':{

```

```

100         int arr_static[5] = { -10, 8, 5, -4, 1};
101         int size_of_our_array = sizeof(arr_static)/sizeof(int);
102         if(size_of_our_array >= 2){
103
104             int highest_arr[3] = {arr_static[0],arr_static
105 [1],0};
106             int lowest_arr[3] = {arr_static[0],arr_static
107 [1],0};
108
109             showArray(arr_static, size_of_our_array);
110
111             auto start = chrono::steady_clock::now();
112             Algorithm(arr_static, size_of_our_array,
113 highest_arr, lowest_arr);
114             auto stop = chrono::steady_clock::now();
115
116             chrono::duration<double> time_of_algorithm = stop-
117 start;
118
119             cout<<"Time of algorithm: "<<time_of_algorithm.
120 count()*1000<<"ms"<<endl;
121         }
122         break;
123     }
124     case 'D':{
125         int elementsInDynamicArray;
126         cout<<"How many elements: ";
127         cin>>elementsInDynamicArray;
128
129         if(elementsInDynamicArray >= 2){
130
131             int *dynamicArray = new int[elementsInDynamicArray
132 ];
133
134             for(int i=0;i<elementsInDynamicArray;i++){
135                 cout<<i<<": ";
136                 cin>>dynamicArray[i];
137             }
138
139             int highest_arr[3] = {dynamicArray[0],dynamicArray
140 [1],0};
141             int lowest_arr[3] = {dynamicArray[0],dynamicArray
142 [1],0};
143
144             showArray(dynamicArray, elementsInDynamicArray);
145
146             auto start = chrono::steady_clock::now();
147             Algorithm(dynamicArray, elementsInDynamicArray,
148 highest_arr, lowest_arr);
149             auto stop = chrono::steady_clock::now();
150
151             chrono::duration<double> time_of_algorithm = stop-
152 start;
153
154             cout<<"Time of algorithm: "<<time_of_algorithm.
155 count()*1000<<"ms"<<endl;

```

```

148         delete dynamicArray;
149
150     }
151
152     break;
153 }
154 case 'R':{
155     //int randomRange = 10000; Using for algorithm time
calculation
156     int randomRange = 3 + (rand() % 28);
157     int *randomArray = new int[randomRange];
158     for(int i=0;i<randomRange;i++){
159         randomArray[i] = -20 + (rand() % 41);
160     }
161
162     int highest_arr[3] = {randomArray[0],randomArray[1],0};
163     int lowest_arr[3] = {randomArray[0],randomArray[1],0};
164
165     showArray(randomArray, randomRange);
166
167     auto start = chrono::steady_clock::now();
168     Algorithm(randomArray, randomRange, highest_arr,
lowest_arr);
169     auto stop = chrono::steady_clock::now();
170
171     chrono::duration<double> time_of_algorithm = stop-start
;
172
173     cout<<"Time of algorithm: "<<time_of_algorithm.count()
*1000<<"ms"<<endl;
174     delete randomArray;
175     break;
176 }
177 default:{
178     cout<<endl<<"Action with this letter is not created."<<
endl;
179     break;
180 }
181 }
182     getchar();
183     getchar();
184 }
185     return 0;
186 }

```