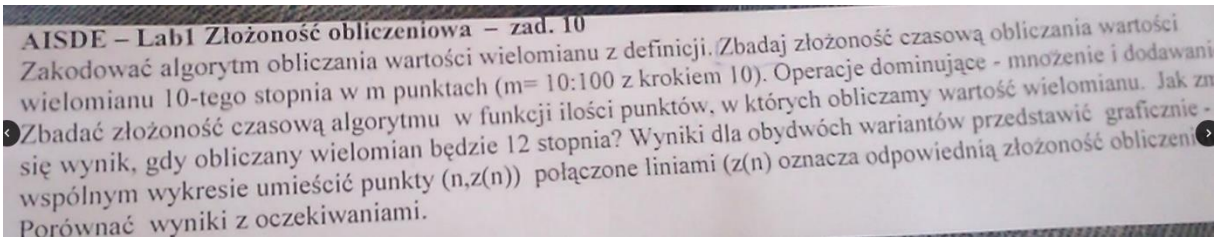
Piotr Dąbrowski - Opracowanie Algorytmów z Lab1 Aisde z prof. Ogrodzkim.

Algorytm 1 - zad.10


AISDE – Lab1 Złożoność obliczeniowa – zad. 10
Zakodować algorytm obliczania wartości wielomianu z definicji. Zbadaj złożoność czasową obliczania wartości wielomianu 10-tego stopnia w m punktach (m= 10:100 z krokiem 10). Operacje dominujące - mnożenie i dodawani Zbadać złożoność czasową algorytmu w funkcji ilości punktów, w których obliczamy wartość wielomianu. Jak zm się wynik, gdy obliczany wielomian będzie 12 stopnia? Wyniki dla obydwóch wariantów przedstawić graficznie - wspólnym wykresie umieścić punkty (n,z(n)) połączone liniami (z(n) oznacza odpowiednią złożoność obliczeni Porównać wyniki z oczekiwaniami.

Samo Poly z wykładu:

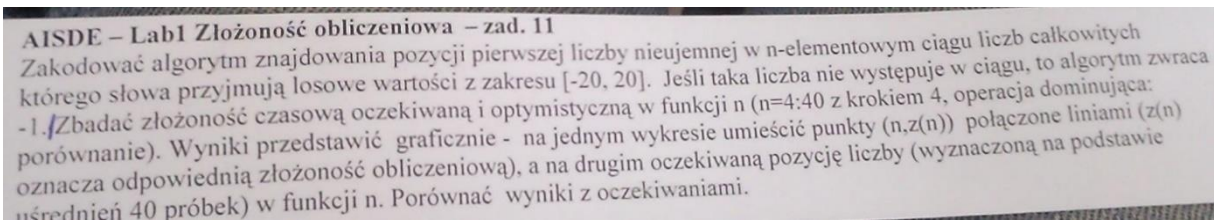
```
function s=poly(a,x)
s=a(1);n=length(a);
for i=2:n
    c=a(i);
    for j=1:i-1
        c=c*x;
    end
    s=s+c;
end
```

Kod ulepszonego algorytmu (lepsza poly!):

```
%obliczanie wartości wielomianu z def.
function sum=poly2(a,x)
sum=a(1);
n=length(a);value=length(x);sum=(1:value);
 for k=1:value %dzięki takiemu zabiegowi będziemy zwracać
               % macierz wyników przez co będzie nam nieco wygodniej
     for i=2:n
         part=a(i);
         for j=1:i-1
             part=part*x(k);
         end
         sum(k)=sum(k)+part;
     end
 end
end
```

Nie do końca rozumiem co oznacza magiczne „w m punktach” i skąd mamy wziąć poszczególne współczynniki a ,ale podam przykład jak uważam . Podam do wejścia dwie macierze ,dla przetestowania ,że algorytm działa skorzystałem z niewielkich liczb mianowicie a=(0.1:0.1:1) i x =(0.1:0.1:10) lecz na labach macierz x powinno się inicjować w sposób x=(10:10:100).

Algorytm 2 - zad.11


AISDE – Lab1 Złożoność obliczeniowa – zad. 11
Zakodować algorytm znajdowania pozycji pierwszej liczby nieujemnej w n-elementowym ciągu liczb całkowitych którego słowa przyjmują losowe wartości z zakresu [-20, 20]. Jeśli taka liczba nie występuje w ciągu, to algorytm zwraca -1. Zbadać złożoność czasową oczekiwaną i optymistyczną w funkcji n (n=4:40 z krokiem 4, operacja dominująca: porównanie). Wyniki przedstawić graficznie - na jednym wykresie umieścić punkty (n,z(n)) połączone liniami (z(n) oznacza odpowiednią złożoność obliczeniową), a na drugim oczekiwaną pozycję liczby (wyznaczoną na podstawie uśrednień 40 próbek) w funkcji n. Porównać wyniki z oczekiwaniami.

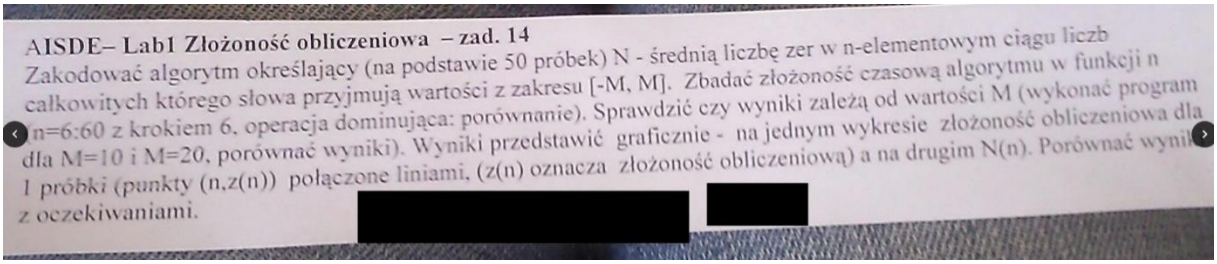
Kod samego algorytmu:

```
%%position.m
function s=position(a)
n=length(a);
s=-1;
for i=1:n
    if a(i)>-1
        s=i;
        break
    end
 end
end
```

Kod tworzenia n-elementowego ciągu z zakresu od [-20,20]:

A=randi([-20,20],1,n);

Algorytm 3 – zad.14



AISDE– Lab1 Złożoność obliczeniowa – zad. 14
Zakodować algorytm określający (na podstawie 50 próbek) N - średnią liczbę zer w n-elementowym ciągu liczb całkowitych którego słowa przyjmują wartości z zakresu [-M, M]. Zbadać złożoność czasową algorytmu w funkcji n dla n=6:60 z krokiem 6, operacja dominująca: porównanie). Sprawdzić czy wyniki zależą od wartości M (wykonać program dla M=10 i M=20, porównać wyniki). Wyniki przedstawić graficznie - na jednym wykresie złożoność obliczeniowa dla 1 próbki (punkty (n,z(n)) połączone liniami, (z(n) oznacza złożoność obliczeniową) a na drugim N(n). Porównać wyni z oczekiwaniami.

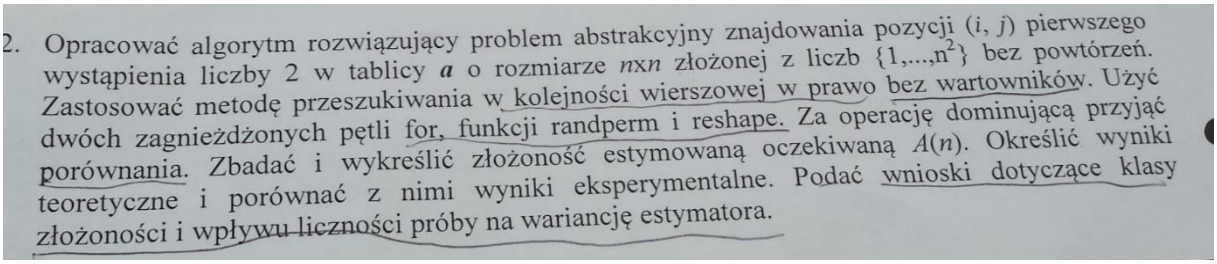
Kod samego algorytmu:

```
function value=average(a) %algorytm  wyliczający
n=length(a);x=0;          %średnią liczbę 0
for i=1:n
    c=a(i);
    if c==0 x=x+1;
    end
end
value=x/n;
end
```

Wzór :

```
A=randi([-M,M],1,n)
```

Algorytm 4



2. Opracować algorytm rozwiązujący problem abstrakcyjny znajdowania pozycji (i, j) pierwszego wystąpienia liczby 2 w tablicy $\boldsymbol{a}$ o rozmiarze nxn złożonej z liczb $\{1,...,n^2\}$ bez powtórzeń. Zastosować metodę przeszukiwania w kolejności wierszowej w prawo bez wartowników. Użyć dwóch zagnieżdżonych pętli for, funkcji randperm i reshape. Za operację dominującą przyjąć porównania. Zbadać i wykreślić złożoność estymowaną oczekiwaną $A(n)$. Określić wyniki teoretyczne i porównać z nimi wyniki eksperymentalne. Podać wnioski dotyczące klasy złożoności i wpływu liczności próby na wariancję estymatora.

Kod algorytmu zwracający pozycję (v1):

```
function pos=find2(a) %ten algorytm zwraca macierz
```

```
n=length(a);           %nie jest to jakiś wybitnie
m=size(a,1);           %skomplikowany algorytm
pos=(1,2);
for i=1:n
    for j=1:m
        if a(i,j)==2
            pos(1,1)=i;
            pos(1,2)=j;
            break
        end
    end
end
end
```

Kod algorytmu zwracający pozycję (v2):

```
function [x,y]=find2(a)    %ten algorytm jest krótszy
n=length(a);               %ale nieco w inny sposób
m=size(a,1);               %się z niego korzysta
for i=1:n
    for j=1:m
        if a(i,j)==2
            x=i;y=j;
            break
        end
    end
end
end
```

Co należy wpisać aby algorytm zadziałał – v1 wystarczy wpisać fin2(A) - A(macierz X na Y) i tyle natomiast algorytm v2 należy skorzystać z takiej linii kodu -> [x,y]=find3(A) i osobno uzyskujemy współrzędne oddzielnie. Aby wylosować macierz bez powtórzeń o zadanych parametrach należy skorzystać z mojego algorytmu bo nie ma takiej funkcji albo po prostu nie znalazłem takiej funkcji.

Mój kod funkcji potrzebnej do takiego losowania:

```
function sum=rndpermtrix(a,x)
randmatrix=randperm(a);value=1; %losujemy wektor wartości od 1 do
for i=1:x                       % a(sqrt(a)>=x) bo inaczej wywali error'a
    for j=1:x
        sum(i,j)=randmatrix(value);
        value=value+1;
    end
  end
end
```

Mam nadzieję ,że pomogłem.