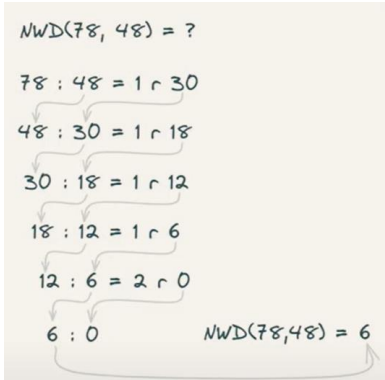
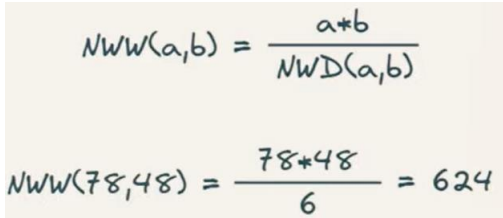


ALGORYTMY MATURALNE, KTÓRE TRZEBA ZNAĆ

NR.	Nazwa ALGORYTMU	Co robi?
1.	Algorytm Euklidesa	<p>1. To sposób na znalezienie NWD największego wspólnego dzielnika dwóch liczb.</p> <ul style="list-style-type: none"> Polega na: wykonywaniu kolejnych dzielen z resztą, gdzie za każdym razem zmieniamy liczby miejscami, aż jedna z nich będzie równa 0, a druga liczba, to obliczany największy wspólny dzielnik.  <p>2. NWW – Najmniejsza Wspólna Wielokrotność</p> <ul style="list-style-type: none"> NWW można obliczyć korzystając z NWD według wzoru, że: NWW dwóch liczb a i b, to $a * b$ przez $NWD(a, b)$ 
2.	Sprawdzanie pierwszości liczby	<p>Algorytm sprawdzania, czy liczba jest pierwsza, polega na upewnieniu się: czy liczba nie dzieli się przez żadną mniejszą od siebie liczbą poza 1.</p> <ul style="list-style-type: none"> Najpierw sprawdzamy, czy: liczba nie jest mniejsza od 2, bo 0 i 1 nie są liczbami pierwszymi. Potem sprawdzamy: podzielność tej liczby od dwóch do samej siebie minus 1. Jeśli znajdziemy liczbę, przez którą dana liczba dzieli się bez reszty, to nie jest ona pierwsza. Jeśli żadna liczba nie spełni tego warunku to oznacza, że liczba jest pierwsza

Sprawdzenie czy liczba 7 jest pierwsza

$7 < 2$ - Fałsz
 $7 \geq 2$ - Fałsz
 $7 \bmod 2 = 1 (\neq 0)$
 $7 \geq 3$ - Fałsz
 $7 \bmod 3 = 1 (\neq 0)$
 $7 \geq 4$ - Fałsz
 $7 \bmod 4 = 3 (\neq 0)$
 $7 \geq 5$ - Fałsz
 $7 \bmod 5 = 2 (\neq 0)$
 $7 \geq 6$ - Fałsz
 $7 \bmod 6 = 1 (\neq 0)$
 $7 \geq 7$ - Prawda
 Liczba jest pierwsza

3. Wyznaczanie n-tego wyrazu ciągu Fibonacciego

```

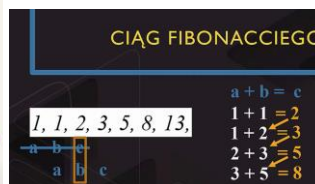
1 #rozwiązanie iteracyjne
2 def fib(n):
3     a, b = 1, 1
4     print("1,1,",end='')
5     for _ in range(n-2):
6         c = a + b
7         print(c,end=',')
8         a = b
9         b = c
10
11 #rozwiązanie rekurencyjne
12 def FIB(n):
13     if n==1 or n==2:
14         return 1
15     else:
16         return FIB(n-1)+FIB(n-2)
17
18 n = int(input("n="))
19 print('iteracyjnie ',end='')
20 fib(n)
21 print()
22 print('rekurencyjnie ',FIB(n))
  
```

Ciąg F. to – **ciąg liczb naturalnych, gdzie każdy kolejny wyraz jest sumą dwóch poprzednich**, zaczynając od tego, że **pierwszymi elementami ciągu są 0 i 1**.

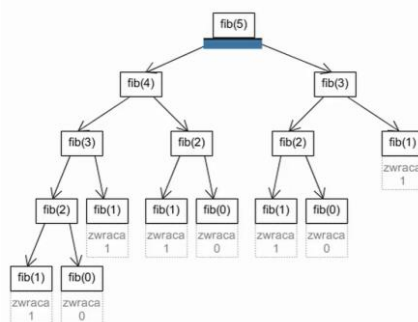
- Aby obliczyć n-ty wyraz ciągu np. drugi, trzeci, czwarty itd. **zaczynamy od dwóch pierwszych liczb ciągu** od 0 i 1 i obliczamy kolejne wyrazy **dodając do siebie dwie poprzednie liczby** np. jeśli $n = 2$, to sumą dwóch poprzednich jest 1, a jeśli $n = 3$, to wynikiem jest 2. Robimy to do momentu, aż uzyskamy n-ty wyraz, który zwracamy jako wynik.
- Na bazie tego algorytmu uczymy się **rekurencji**.

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19
0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584	4181

F0: 0
 F1: 1
 F2: 0 + 1 = 1
 F3: 1 + 1 = 2
 F4: 1 + 2 = 3
 F5: 2 + 3 = 5



REKURENCYJNIE ZWRACA OSTATNIE:



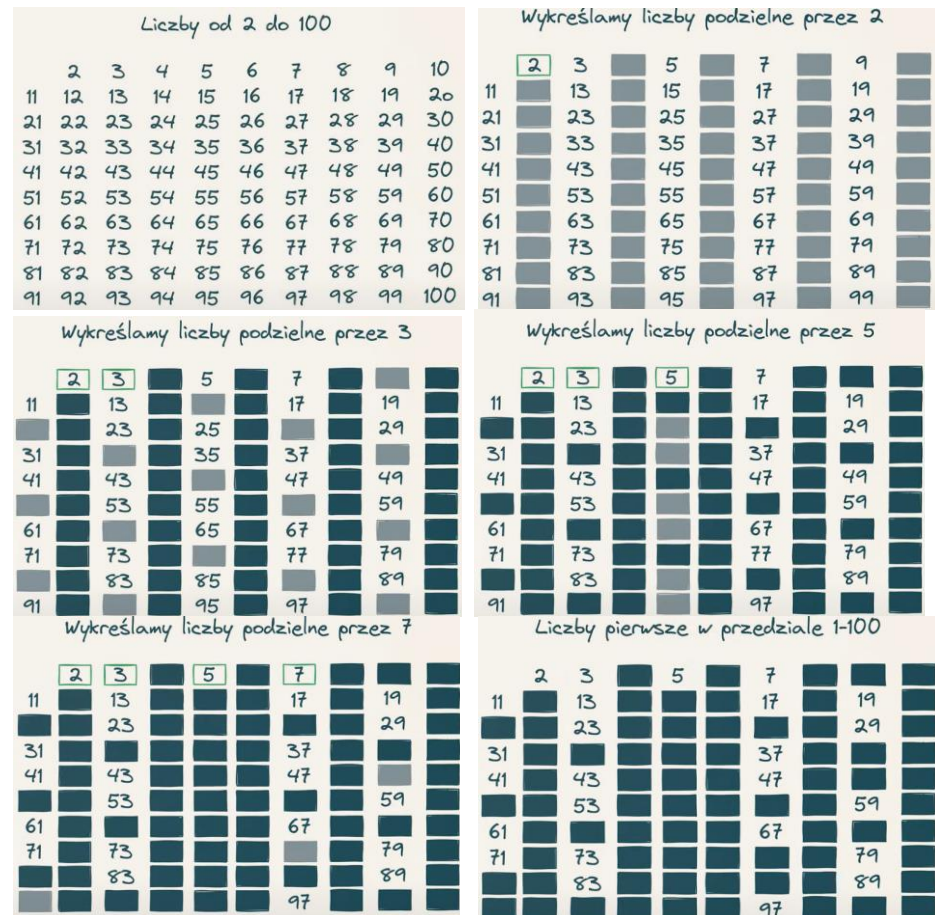
4. Sito Eratostenesa

Sito E. – to algorytm do **znajdowania wszystkich liczb pierwszych w przedziale od 2 do n**.

- Działa on przez **wykreślanie z tego zakresu liczb, które są wielokrotnościami liczb pierwszych**.
- Zaczynamy od najmniejszej liczby pierwszej czyli 2** i wykreślamy wszystkie jej wielokrotności.
- Potem **przechodzimy do następnych liczb**, które nie zostały jeszcze wykreślone i

znów **usuwamy ich wielokrotności**, powtarzając ten proces aż **do pierwiastka z n**.

- Na koniec **wszystkie niewykreślone liczby są liczbami pierwszymi**.
- Jest on efektywniejszy, niż standardowe sprawdzanie pierwszości liczby.



5. **A. Zamiana**
liczby z systemu
dziesiętnego na
dowolny

Aby zamienić **liczbę** z systemu dziesiętnego na dowolny inny system **dzielimy ją przez podstawę nowego systemu i zapisujemy resztę z dzielenia**.

- Proces powtarzamy aż liczba stanie się równa 0. Te reszty zapisane w odwrotnej kolejności tworzą liczbę w nowym systemie

Zamiana 270_{10} na system szesnastkowy

$$270 / 16 = 16 \text{ reszty } 14 \rightarrow E$$

$$16 / 16 = 1 \text{ reszty } 0$$

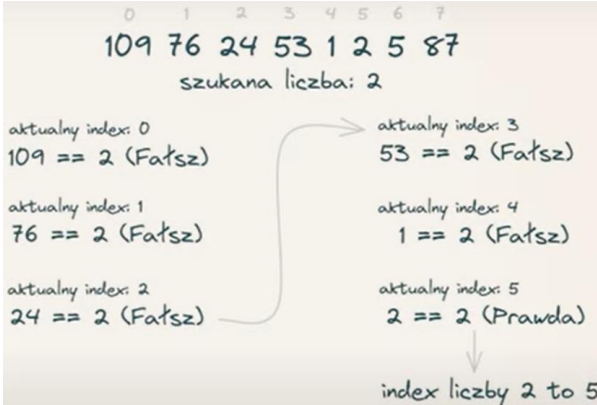
$$1 / 16 = 0 \text{ reszty } 1$$

wynikiem jest zestawienie reszt z dzielenia zapisane od końca $270_{10} = 10E_{16}$



6. **B. Zamiana liczby** z
systemu dowolnego na
dziesiętny (czyli
odwrotnie jak powyżej)

Aby to zrobić **przechodźmy po cyfrach liczby i każdą cyfrę mnożymy przez podstawę systemu podniesioną do potęgi odpowiadającej pozycji cyfry**.

- Pierwsza cyfra od końca ma **potęgę 0**, kolejna 1 itd.
- Następnie **sumujemy wyniki potęgowania i mnożenia**, aby otrzymać zamienioną liczbę na system dziesiętny

		<p>Analiza liczby $1101_2 \leftarrow$ podstawa systemu</p> <p>wagi cyfr \rightarrow 3 2 1 0</p> <p>1 1 0 1</p> <p>$1101_2 = x_{10}$</p> $x = 1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$ $x = 1 * 8 + 1 * 4 + 0 * 2 + 1 * 1$ $x = 8 + 4 + 1$ $x = 13$ $1101_2 = 13_{10}$
7.	<p>A. Przeszukiwanie liniowe</p>	<p>To zagadnienie pojawia się w kontekście złożoności obliczeniowej.</p> <p>Przeszukiwanie liniowe – najprostsze możliwe przeszukiwanie tablicy.</p> <ul style="list-style-type: none"> Przechodzimy po kolei po każdym elemencie tablicy i sprawdzamy czy jest on równy elementowi, którego szukamy. Można zakończyć działanie, kiedy odnajdzie się szukaną wartość. Przeszukiwanie liniowe ma złożoność liniową 
8.	<p>B. Przeszukiwanie binarne</p>	<p>Jest to algorytm – do przeszukiwania tablicy TYLKO POSORTOWANEJ.</p> <ul style="list-style-type: none"> Opiera się on na zasadzie dziel i zwyciężaj. Za każdym razem dzielimy tablicę na pół i wybieramy tę część, w której może znajdować się wyszukiwany element. Proces powtarzamy aż: znajdziemy element lub zmniejszymy przedział do zera. Przeszukiwanie binarne – ma złożoność logarytmiczną, czyli działa efektywniej, niż przeszukiwanie liniowe.

		<div><div><div>01234567891011</div><div>25172528324673747698109</div><div>szukana liczba: 76</div><div>lewy: 0 prawy: 11</div><div>index = $\frac{(0 + 11)}{2} = 5,5 \approx 5$</div><div>32 < 76</div><div>01234567891011</div><div>25172528324673747698109</div><div>lewy: 6 prawy: 11</div><div>index = $\frac{(6 + 11)}{2} = 8,5 \approx 8$</div><div>74 < 76</div></div></div>																																																																																
9.	<div><div><div>Wyszukiwanie wzorca w tekście metodą naiwną</div></div></div>	<div><div><div>Metoda ta polega na – przesuwaniu wzorca wzdłuż tekstu i porównaniu go z odpowiadającym mu fragmentem. Jeśli wzorzec pasuje do fragmentu, oznacza to, że wzorzec został znaleziony.</div><div><div>• Algorytm działa dopóki nie przeanalizuje całego tekstu.</div></div></div><div><div><div>napis: KAPIBARA wzorzec: BAR</div><div><div>I porównanie: <table><tr><td>K</td><td>A</td><td>P</td><td>I</td><td>B</td><td>A</td><td>R</td><td>A</td></tr><tr><td></td><td></td><td></td><td></td><td>B</td><td>A</td><td>R</td><td></td></tr></table> KAP != BAR</div><div><div>II porównanie: <table><tr><td>K</td><td>A</td><td>P</td><td>I</td><td>B</td><td>A</td><td>R</td><td>A</td></tr><tr><td></td><td></td><td></td><td>B</td><td>A</td><td>R</td><td></td><td></td></tr></table> API != BAR</div><div><div>III porównanie: <table><tr><td>K</td><td>A</td><td>P</td><td>I</td><td>B</td><td>A</td><td>R</td><td>A</td></tr><tr><td></td><td></td><td></td><td>B</td><td>A</td><td>R</td><td></td><td></td></tr></table> PIB != BAR</div><div><div>IV porównanie: <table><tr><td>K</td><td>A</td><td>P</td><td>I</td><td>B</td><td>A</td><td>R</td><td>A</td></tr><tr><td></td><td></td><td></td><td>B</td><td>A</td><td>R</td><td></td><td></td></tr></table> IBA != BAR</div><div><div>V porównanie: <table><tr><td>K</td><td>A</td><td>P</td><td>I</td><td>B</td><td>A</td><td>R</td><td>A</td></tr><tr><td></td><td></td><td></td><td>B</td><td>A</td><td>R</td><td></td><td></td></tr></table> BAR == BAR</div></div></div></div></div></div></div></div></div>	K	A	P	I	B	A	R	A					B	A	R		K	A	P	I	B	A	R	A				B	A	R			K	A	P	I	B	A	R	A				B	A	R			K	A	P	I	B	A	R	A				B	A	R			K	A	P	I	B	A	R	A				B	A	R		
K	A	P	I	B	A	R	A																																																																											
				B	A	R																																																																												
K	A	P	I	B	A	R	A																																																																											
			B	A	R																																																																													
K	A	P	I	B	A	R	A																																																																											
			B	A	R																																																																													
K	A	P	I	B	A	R	A																																																																											
			B	A	R																																																																													
K	A	P	I	B	A	R	A																																																																											
			B	A	R																																																																													
10.	<div><div><div>ALGORYTMY ZWIĄZANE Z MATEMATYKĄ:</div><div><div>A. Wyznaczanie miejsca zerowego funkcji</div></div></div></div>	<div><div><div>Wyznaczanie miejsc zerowych w funkcji. Za pomocą metody poławiania, która polega na – stopniowym zawężaniu przedziałów, w którym znajduje się miejsce zerowe.</div><div><div>• Podobnie jak w przeszukiwaniu binarnym dzielimy na pół i sprawdzamy, w której części znajduje się miejsce zerowe.</div><div>• Proces ten powtarzamy, aż różnica między końcami przedziału będzie mniejsza od zadanej dokładności.</div></div></div><div><div><div><div><div>$f(x) = x^3 - 6x - 4$ $a = 2$ $b = 4$ $e = 0,5$</div><div><div><div>$a = 2$ $b = 4$ $\text{środek} = 3$</div><div><div><div><div></div><div></div><div></div></div></div><div><div><div>$a - b > 0,5$ \downarrow $f(a) * f(\text{środek}) < 0 \rightarrow b = \text{środek}$ $(b = 3)$</div></div></div></div></div></div></div></div></div></div></div>																																																																																

		<p> $a = 2$ $b = 3$ $\text{środek} = 2,5$ </p>  <p> $a - b > 0,5$ $f(a) * f(\text{środek}) > 0 \rightarrow a = \text{śr}$ $(a = 2,5)$ </p> <p> $a = 2,5$ $b = 3$ $\text{środek} = 2,75$ </p>  <p> $a - b = 0,5 \rightarrow$ dokładność została osiągnięta, więc przybliżone miejsce zerowe znajduje się w $f(2,75)$ </p>
11.	<p>B. Obliczanie pierwiastka kwadratowego (metodą Newtona Raphsona)</p>	<p>Obliczamy pierwiastek kwadratowy liczby poprzez iteracyjne poprawianie przybliżenia.</p> <p>Zaczynamy od początkowego przybliżenia i używamy wzoru iteracyjnego do obliczania kolejnych przybliżeń. Proces ten powtarzamy aż różnica między kolejnymi przybliżeniami będzie mniejsza niż zadana dokładność.</p> <ul style="list-style-type: none"> Każde nowe przybliżenie jest obliczane na podstawie poprzedniego, co pozwala na szybsze zbliżenie się do rzeczywistej wartości pierwiastka. <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p> $n = 7$ $e = 0,01$ $x_0 = 3,5$ $x_1 = (3,5 + 7 / 3,5) / 2 = 2,75$ $(0,75)$ $2,75 - 3,5 \geq 0,01$ $x_0 = 2,75$ $x_1 = (2,75 + 7 / 2,75) / 2 \approx 2,64773$ $(0,10227)$ $2,64773 - 2,75 \geq 0,01$ $x_0 = 2,64773$ $x_1 = (2,64773 + 7 / 2,64773) / 2 \approx 2,64575$ $(0,00198)$ $2,64575 - 2,64773 < 0,01$ $\sqrt{7} \approx 2,64575$ </p> </div>
12.	<p>C. Obliczanie wartości wielomianu za pomocą schematu Hornera</p>	<p>Algorytm ten służy do szybkiego obliczania wartości wielomianu, zmniejsza on liczbę mnożeń do minimum, czyli przykładowo: z wielomianu mającego pięć mnożeń robimy wielomian w postaci zagnieżdżonej, który ma już tylko trzy mnożenia.</p> <ul style="list-style-type: none"> Aby obliczyć wartość wielomianu dla danej liczby – zaczynamy od największego współczynnika i stopniowo przetwarzamy kolejne współczynniki. Wykonujemy operacje mnożenia i dodawania w jednym kroku. Mnożymy aktualny wynik poprzez wartość zmiennej i dodajemy kolejny współczynnik, kontynuujemy ten proces aż do uwzględnienia wszystkich współczynników, co pozwala szybko uzyskać wartość wielomianu. <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p> $w(x) = 2x^3 + x^2 + 3x + 9$ $w(x) = 2 * x * x * x + x * x + 3 * x + 9$ </p> <p style="text-align: right; font-size: 1.5em; color: #0070c0;">5 mnożeń</p> </div>

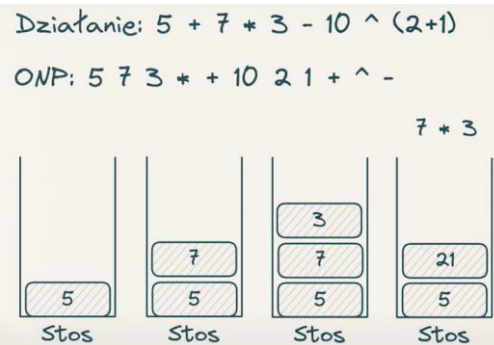
		$w(x) = 2x^3 + x^2 + 3x + 9$ $= x * (2x^2 + x + 3) + 9$ $= x * (x * (2 * x + 1) + 3) + 9$ <p>3 mnożenia</p>
13.	A. Potęgowanie sposobem naiwnym	<p>To najprostsza metoda obliczania potęgi liczby.</p> <p>Działa ona w ten sposób, że:</p> <ul style="list-style-type: none"> • zaczynamy od wartości 1 i wielokrotnie mnożymy ją przez podstawę potęgi tyle razy, ile wynosi wykładnik np. jeśli chcemy mnożyć dwa do czwartej, to po kolei mnożymy jeden przez 2, potem przez 2 i tak dalej, aż osiągniemy liczbę mnożeń równą wykładnikowi, czyli jak w poniższym przykładzie cztery razy • Choć to najprostsza metoda, może być mniej wydajna dla dużych wykładników ponieważ wymaga wielu operacji mnożenia $2^4 = 1 * 2 * 2 * 2 * 2$
14.	B. Potęgowanie szybkie	<p>To efektywniejszy niż poprzedni sposób potęgowania.</p> <p>Algorytm ten polega na:</p> <ul style="list-style-type: none"> • zmniejszaniu operacji liczby mnożeń, dzięki czemu działa znacznie szybciej. • Dzielimy wykładnik na mniejsze części i obliczamy potęgowanie rekurencyjne dla mniejszych części. Jeśli wykładnik jest nieparzysty dodajemy dodatkowe mnożenia poprzez podstawę do rezultatu. • Przykładowo $4^{10} = 16^5 = 256^2 * 16 = 65536^1 * 16 = 1048576$ $4^{10} = (4 * 4) * (4 * 4) * (4 * 4) * (4 * 4) * (4 * 4)$ $= (4 * 4)^5 = 16^5$ $16^5 = (16 * 16) * (16 * 16) * 16 = (16 * 16)^2 * 16$ $= 256^2 * 16$ $256^2 * 16 = 65536^1 * 16$ $4^{10} = 65536^1 * 16 = 65536 * 16 = 1048576$
15.	C. Potęgowanie modulo	<p>Algorytm ten polega na:</p> <ul style="list-style-type: none"> • obliczaniu reszty z dzielenia potęgi danej liczby przez inną liczbę. <p>Potęgowanie naiwne modulo</p> $26^5 \% 33 = (((((((1 * 26) \% 33) * 26) \% 33) * 26) \% 33) * 26) \% 33) * 26) \% 33 =$ $= ((((((26 * 26) \% 33) * 26) \% 33) * 26) \% 33) * 26) \% 33 =$ $= ((((((676 \% 33) * 26) \% 33) * 26) \% 33) * 26) \% 33 =$ $= ((((((16 * 26) \% 33) * 26) \% 33) * 26) \% 33 =$ $= ((((((416 \% 33) * 26) \% 33) * 26) \% 33 =$

		$= ((20 * 26) \% 33) * 26) \% 33 =$ $= ((520 \% 33) * 26) \% 33 =$ $= (25 * 26) \% 33 =$ $= 650 \% 33 =$ $= 23$ $26^5 \% 33 = (((((26 * 26) \% 33) * (26 * 26) \% 33) \% 33) * 26) \% 33 =$ $= (((((26 * 26) \% 33)^2 \% 33) * 26) \% 33 =$ $= (((676 \% 33)^2 \% 33) * 26) \% 33 =$ $= ((16^2 \% 33) * 26) \% 33 =$ $= ((256 \% 33) * 26) \% 33 =$ $= (25 * 26) \% 33 =$ $= 650 \% 33 =$ $= 23$
16.	Wydawanie reszty z użyciem algorytmu zachłannego	<p>Zaczynamy od założeń, że mamy nieskończoną liczbę monet o określonych nominałach chcemy wydać konkretną kwotę używając jak najmniejszej liczby monet.</p> <ul style="list-style-type: none"> Przykładowo chcemy wydać 63 grosze, nominały którymi dysponujemy to: 1gr, 2gr, 5gr, 10gr, 20gr, 50gr Algorytm polega na tym, że – <i>najpierw wybieramy monetę o największym nominalu, która nie przekracza pozostałej kwoty do wydania, postępujemy tak. dopóki nie uzbieramy całej kwoty</i> <p>nominały: (1) (2) (5) (10) (20) (50) reszta: 63 groszy</p> <p>1. wybieramy monetę (50) (50 ≤ 63) 3. wybieramy monetę (2) (2 ≤ 3) 63 - (50) = 13 3 - (2) = 1</p> <p>2. wybieramy monetę (10) (10 ≤ 13) 4. wybieramy monetę (1) (1 ≤ 1) 13 - (10) = 3 1 - (1) = 0</p> <p>wynik: (50) (10) (2) (1)</p>
17.	Odwrotna Notacja Polska (ONP)	<p>Jest ona metodą – zapisu wyrażeń arytmetycznych bez zastosowania nawiasów.</p> <ul style="list-style-type: none"> W tej notacji symbole operacji występują po argumentach <p>W poniższej tablicy widać przykład wyrażeń w ONP</p>

Normalna notacja	ONP
$1 + 3$	1 3 +
$6 + 4 * 3$	6 4 3 * +
$7 / (3 - 5)$	7 3 5 - /
$2 ^ (5 + 1) + (9 - 3)$	2 5 1 + ^ 9 3 - +
$(8 - 5) * (4 + 6)$	8 5 - 4 6 + *

Algorytm obliczania wartości ONP działa, w ten sposób, że:

- odczytujemy po kolei każdy znak wyrażenia; jeśli jest on **liczbą** to **zapisujemy go na stosie**, jeśli jest **operatorem** to **pobieramy dwie ostatnie liczby ze stosu**. Wykonujemy na nich działanie i wynik umieszczamy z powrotem na stosie. Powtarzamy to działanie tak długo dopóki nie przejdziemy przez całe wyrażenie ONP. **Pozostała liczba na stosie będzie naszym wynikiem.**



18.

A. Sortowanie bąbelkowe

To **najprostszy** algorytm sortujący.

Polega on na:

- **porównaniu ze sobą sąsiednich elementów tablicy**, dopóki nie zostanie ona posortowana.
- Dodatkowo można wykonać **operację optymalizacji**, aby zmniejszyć liczbę porównań w każdej iteracji.

Tu dodaj od siebie sposób działania algorytmu:

.....

.....

.....

.....

19.	B. Sortowanie przez wybór	<p>Polega ono na – znajdowaniu najmniejszego elementu po prawej stronie od aktualnego elementu, czyli w nieposortowanej części.</p> <ul style="list-style-type: none"> Po znalezieniu aktualny element i minimalny element zamieniają się miejscami tę czynność wykonujemy do momentu przejścia po całej tablicy.
20.	C. Sortowanie przez wstawianie	<p>Polega ono na – pobieraniu pierwszego elementu z prawej strony tablicy, czyli z części nieposortowanej i wstawianie go w odpowiednie miejsce w lewej części, czyli części posortowanej.</p> <ul style="list-style-type: none"> Algorytm rozpoczyna się od indeksu pierwszego, a indeksuje aż do momentu dojścia do końca tablicy

21.	D. Sortowanie kubetkowe	<p>Ma ono wiele różnych wersji.</p> <p>Poniższa wersja polega na – utworzeniu liczników dla każdej wartości w tablicy i ustawieniu ich na początku na 0. Te liczniki to nasze kubeczki.</p> <ul style="list-style-type: none"> • Kubeczki są tablicą o długości największej wartości w naszej tablicy plus 1. • Przechodząc po tablicy każde wystąpienie danej liczby odnotowujemy zwiększając zawartość danego kubeczka o 1. Po zliczeniu wszystkich elementów wypisujemy numer danego kubeczka tyle razy ile wynosi jego zawartość. W ten sposób otrzymujemy posortowaną tablicę
22.	E. Sortowanie szybkie (Sortowania rekurencyjne)	<p>Polega ono na wybieraniu elementu podziału, czyli naszego pivota.</p> <ul style="list-style-type: none"> • Po lewej stronie od pivota umieszczamy elementy od niego mniejsze a po prawej analogicznie większe. Wtedy następuje podział tablicy na mniejsze i większe elementy od pivota. W tych tablicach również wybieramy pivota dokonując znowu tego samego podziału i powielamy tę czynność aż do momentu kiedy przedział jest JEDNOELEMENTOWY.



23.	F. Sortowanie przez scalanie (to sortowanie rekurencyjne)	<p>Polega ono na – dzieleniu tablicy na dwie części, dopóki nie pozostanie jeden element.</p> <ul style="list-style-type: none"> Powstałe posortowane tablice łączy się w jeden posortowany ciąg.
24.	A. Szyfrowanie Cezara	<p>To najprostsza i jedna z najstarszych metod szyfrowania tekstu.</p> <ul style="list-style-type: none"> Każda litera tekstu jawnego jest zastępowana inną literą, która jest oddalona od niej o klucz, czyli o stałą liczbę pozycji w alfabecie

- Mamy **tekst jawny**, który chcemy zaszyfrować i **klucz** będący wartością liczbową, o którą będziemy przesuwać litery
- np. jeśli klucz wynosi 3, to litera a stanie się literą d, litera b stanie się literą e itd.

tekst jawny: ZUPA klucz: 3

1. Szyfrowanie litery Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Z → C

2. Szyfrowanie litery U

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

U → X

3. Szyfrowanie litery P

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

P → S

4. Szyfrowanie litery A

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A → D

szyfrogram: CXSD

25.

B. Szyfrowanie płotkowe

Polega ono na – **zapisaniu tekstu jawnego w sposób przypominający płotek**

- **Wielkość płotka** jest definiowana poprzez **klucz szyfrujący**, który też jest wartością liczbową
- Następnie dzięki odpowiedniemu **odczytaniu tekstu z płotka** powstaje **szyfrogram**

tekst jawny: WSZYSTKO klucz: 3

1. Stworzenie płotka

	0	1	2	3	4	5	6	7
0	W				S			
1		S		Y		T		O
2			Z				K	

2. Odczytanie liter wierszami

	0	1	2	3	4	5	6	7
0	W				S			
1		S		Y		T		O
2			Z				K	

W S

	0	1	2	3	4	5	6	7
0	W				S			
1		S		Y		T		O
2			Z				K	

W S S Y T O

	0	1	2	3	4	5	6	7
0	W				S			
1		S		Y		T		O
2			Z				K	

W S S Y T O Z K

3. Odczytanie szyfrogramu

szyfrogram: WSSYTOZK

26.

C. Szyfrowanie Vigenere'a

Polega ono na **wykorzystaniu wielu różnych podstawień Cezarowych w zależności od litery klucza**

- Zaczynamy od **wypełnienia macierzy literami alfabetu**, następnie **każdą literę tekstu jawnego zamieniamy według klucza przy pomocy macierzy**, a kluczem w tym szyfrze jest **inne słowo**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

tekst jawny: **INFORMATYKA**

- Szyfrowanie litery I: $I \xrightarrow{M} U$
- Szyfrowanie litery N: $N \xrightarrow{A} N$
- Szyfrowanie litery F: $F \xrightarrow{T} Y$
- Szyfrowanie litery O: $O \xrightarrow{U} I$
- Szyfrowanie litery R: $R \xrightarrow{R} I$
- Szyfrowanie litery M: $M \xrightarrow{A} M$
- Szyfrowanie litery A: $A \xrightarrow{M} M$
- Szyfrowanie litery T: $T \xrightarrow{A} T$
- Szyfrowanie litery Y: $Y \xrightarrow{T} R$
- Szyfrowanie litery K: $K \xrightarrow{U} E$
- Szyfrowanie litery A: $A \xrightarrow{R} R$

szyfrogram: **UNYIIMMTRER**

klucz: **MATURA**

27.

D. Szyfrowanie Playfair

To **szyfr blokowy, który szyfruje litery parami**

- Klucz jest napisem** używanym do wygenerowania macierzy 5 na 5 złożonej z liter alfabetu
 - Litery są wpisane do macierzy zgodnie z kolejnością** z jaką mamy w kluczu, a następnie w porządku alfabetycznym
 - W macierzy może zmieścić się tylko 25 liter**, a alfabet angielski ma ich 26, zatem najczęściej stosuje się rozwiązanie, w którym **i** oraz **j** jest traktowane jako **jedna ta sama litera**
 - Poniżej jest **przykładowa macierz wygenerowana na podstawie klucza szyfr**
- A. Jeśli litery tekstu jawnego są **w tej samej kolumnie** to zamieniamy każdą z liter na literę znajdującą się **poniżej**

- B. Jeśli litery są **w tym samym wierszu** to zamieniamy każdą z liter na literę znajdującą się **po prawej stronie**
- C. Jeżeli litery **tworzą prostokąt** to zamieniamy każdą z liter na literę znajdującą się w tym samym wierszu, ale w **kolumnie drugiej litery z pary**

Tabela

S	Z	Y	F	R
A	B	C	D	E
G	H	I	K	L
M	N	O	P	Q
T	U	V	W	X

Szyfrowanie pary "MA"

S	Z	Y	F	R
A	B	C	D	E
G	H	I	K	L
M	N	O	P	Q
T	U	V	W	X

M → T
A → G

Szyfrowanie pary "TU"

S	Z	Y	F	R
A	B	C	D	E
G	H	I	K	L
M	N	O	P	Q
T	U	V	W	X

T → U
U → V

Szyfrowanie pary "RA"

S	Z	Y	F	R
A	B	C	D	E
G	H	I	K	L
M	N	O	P	Q
T	U	V	W	X

R → S
A → E

28.

INNE ALGORYTMY SZYFROWANIA I SORTOWANIA

Dopisz samemu

1.
2.

29.

Szukanie najdłuższego wspólnego podciągu