

- BRUTY

- maski bitowe: przejrzanie wszystkich podciągów [video](#)
- generowanie wszystkich ciągów n-elementowych
 - * generowanie wszystkich permutacji
- wyznaczanie ścieżki między dwoma wierzchołkami (graf/drzewo)
- skomplikowane backtracki

- DEBUG [artykuł](#)

- pisanie generatorek
 - * sprawdzarka w bashu [plik zip](#)
 - * sprawdzarka w C++
 - * generowanie losowych liczb, permutacji, drzew, grafów skierowanych / nieskierowanych / spójnych / bez multikrawędzi
- wykrywanie błędów
 - * flagi kompilacji [plik zip](#)
 - * czas wykonania programu oraz zużyta pamięć [plik zip](#)
 - * używanie narzędzia gdb do wykrywania runtime error'ów (↑ [plik zip](#))
 - * nagłówki (↑ [pierwszy zip](#))

- GRAFY

- Drzewa [artykuł](#)
 - * jump-pointery
 - najniższy wspólny przodek (LCA)
 - operacje na ścieżkach (np. otrzymanie maximum)
 - * właściwości numeracji wierzchołków w drzewie
 - preorder, postorder, podobne [artykuł](#)
 - * dynamiki na drzewie
 - * znane algorytmy: szukanie centroidów, średnicy drzewa
- Graf dwudzielny
 - * określanie dwudzielności grafu
 - * skojarzenia w grafach dwudzielnych [video](#) [artykuł](#)
 - * twierdzenie Halla (↑ [artykuł](#))
 - * twierdzenie Königa [video](#) [artykuł](#)
- Skierowany Graf Acykliczny (DAG)
 - * sortowanie topologiczne (toposort) [artykuł](#)
 - * dynamiki na DAGach (↑ [artykuł](#))
 - * podział na warstwy
- Grafy [artykuł](#)
 - * DFS, BFS [video](#) [artykuł](#)
 - * drzewo DFS [video](#)
 - funkcja low & mosty, punkty artykulacji, dwuspójne [video](#) [video2](#) [artykuł](#)
 - * Dijkstra [artykuł](#)
 - * rozbitcie wierzchołkowe
 - * potęgowanie macierzy sąsiedztwa [video](#) [artykuł](#)
 - * silne spójne składowe (SCC) [artykuł](#)
 - * cykl Eulera [video](#) [artykuł](#)
 - * minimalne drzewo rozpinające (MST)
 - algorytm Kruskala (mniej ważny: algorytm Prima) [video](#) [artykuł](#)
 - Prufer Code do brutowania wszystkich możliwych drzew [article](#)
 - * rozbitcie wierzchołkowe
 - * 2-SAT [video](#)

- Meduzy [artykuł](#)
- Grafy planarne [artykuł](#)
 - * wzór Eulera ([↑ artykuł](#))
 - * zapytania o istnienie ścieżki w grafie planarnym ([↑ artykuł](#))
- STRUKTURY DANYCH
 - Trie [artykuł](#)
 - Drzewa przedziałowe [video artykuł](#)
 - * drzewa przedział-przedział [video](#) ([↑ video](#)) ([↑ artykuł](#))
 - * drzewa trwałe, wskaźnikowe
 - Sqrt [artykuł](#)
 - * sqrt decomposition
 - * split & rebuild
 - * algorytm mo ([↑ artykuł](#))
 - kolejka monotoniczna [artykuł](#)
 - mniejszy do większego
 - find & union [artykuł video](#)
 - sumy prefixowe [artykuł](#)
 - `std::vector` jako stos, `std::deque` jako kolejka, `std::set`, `std::map` [artykuł](#)
 - sort
 - * `std::sort`, komparator i błędy z komparatorem
 - * sortowanie kubelkowe [artykuł](#)
 - * skalowanie [artykuł](#)
- DYNAMIKI [artykuł](#)
 - problem plecakowy (różne odmiany, sqrt optymalizacja) [artykuł](#)
 - dynamiki kombinatoryczne [artykuł](#)
 - dynamiki przedziałowe [artykuł](#)
 - dynamiki optymalizacyjne ([↑ artykuł](#))
 - dynamiki wykładnicze [artykuł](#)
 - potęgowanie macierzy [video artykuł](#)
- MATMA
 - arytmetyka modulo
 - * najmniejszy wspólny dzielnik (NWD) [video artykuł](#)
 - rozszerzony algorytm euklidesa, odwrotność modulo [video artykuł](#)
 - * funkcja Phi (totient Eulera) [video](#)
 - Małe Twierdzenie Fermat (MTF) ([↑ video](#))
 - * Chińskie Twierdzenie o Resztach (CTR) ([↑ video](#))
 - * szybkie potęgowanie [artykuł](#)
 - znajdowanie dzielników (sqrt), dzielników pierwszych (sito Eratostenesa/sqrt) [artykuł](#)
 - teoria gier [artykuł](#)
 - * twierdzenie Sprague-Grundy'ego [video video2 artykuł](#)
 - potęgowanie macierzy [video artykuł](#)
 - podstawy kombinatoryki
 - * dwumian Newton'a
 - * zasada włączeń i wyłączeń
 - rozwiązania probabilistyczne, paradoks urodzeń
 - suma ciągu harmonicznego, złożoność sita

- GEOMETRIA [artykuł](#) [artykuł2](#)
 - iloczyn wektorowy ([↑ artykuł](#))
 - pole wielokąta [artykuł](#)
 - wielokąty wypukłe ([↑ artykuł](#))
 - * otoczka wypukła [artykuł](#)
 - zmiatanie [video](#) [artykuł](#)
 - sortowanie kątowe [artykuł](#)
 - geometria obliczeniowa [video](#) [video2](#)
 - najbliższa para punktów [video](#)
 - najdalsza para punktów
- ALGORYTMY TEKSTOWE [artykuł](#)
 - liniowe algorytmy tekstowe [video](#) [artykuł](#)
 - * tablica pi, algorytm KMP [video](#) ([↑ video](#))
 - * algorytm Manachera [video](#)
 - * tablica prefixo-prefixowa ([↑ video](#))
 - * szablony ([↑ video](#))
 - * okresy ([↑ video](#))
 - hashowanie (posłów, podciągów, przypisanie literkom / liczbom losowe wagi i sprawdzanie sumy)([↑ video](#)) [artykuł](#)
 - drzewo Trie [artykuł](#)
 - tablica suffixowa [video](#) [artykuł](#)
- OPTYMALIZACJE
 - dziel i zwyciężaj [artykuł](#)
 - * bardziej skomplikowane przykłady dziel i zwyciężaj
 - * meet in the middle, czyli wzorcówka dla $n \leq 40$ [artykuł](#)
 - wyszukiwanie binarne [artykuł](#)
 - * wyszukiwanie binarne po wyniku
 - * wyszukiwanie binarne wielu rzeczy na raz / równoległe wyszukiwanie binarne
 - koszt zamortyzowany
 - * gąsienica [artykuł](#)
 - * zadania na stos
 - skalowanie [artykuł](#)
 - lider ciągu
 - odpowiednie struktury danych
 - bitsety

- PODEJŚCIA

- dynamik
- algorytm zachłanny **artykuł**
- sposób "liczba dobrych obiektów = liczba wszystkich obiektów - liczba złych obiektów"
- czy warunek konieczny = warunek wystarczający?
- odpowiednie przekształcenie równania; uniezależnienie funkcji od jakiejś zmiennej; zauważenie wypukłości
- zastanowić się nad łatwiejszym problemem, bez jakiegoś elementu z treści
- sprowadzić problem do innego, łatwiejszego/mniejszego problemu
- sprowadzić problem 2D do problemu 1D (zamiatanie; niezależność wyniku dla współrzędnych X od współrzędnych Y)
- konstrukcja grafu
- określenie struktury grafu
- napisanie bruta przed wzorcówką (przynajmniej gwarantowane punkty są, a potem sprawdzaczka)
- optymalizacja bruta do wzorcówki
- czy można poprawić (może zachłannie) rozwiązanie nieoptymalne?
- czy są ciekawe fakty w rozwiązaniach optymalnych (może się do tego przydać brute)
- sklepać brute który sprawdza obserwacje, zawsze jeśli potrzebujemy zoptymalizować dp, wypisać wartości na małym przykładzie
- sprawdzić czy w zadaniu czegoś jest "mało" (np. czy wynik jest mały, albo jakaś zmienna, może się do tego przydać brute)
- odpowiednio "wzbogacić" jakiś algorytm (np. Dijkstrę)
- spróbować obliczyć wkład do wyniku jakiegoś elementu
- jeżeli miałoby się teraz zrobić to zadanie i to nie jest hardkor, jakie obserwacje by pomogły?
- cokolwiek poniżej 10^9 operacji ma szansę wejść na 100 pkt
- coś ciekawego w limitach w zadaniu i subtaskach?
- obserwacje, obserwacje, obserwacje!
- co można wykonać offline? czy jest coś, czego kolejność nie ma znaczenia?
- co można posortować? czy jest zawsze jakaś pewna optymalna kolejność / czy można ustalić jak ta kolejność wygląda?
- narysować dużo swoich własnych przykładów i coś z nich wywnioskować
- skupić się na jakimś specjalnym elemencie, najczęściej najmniejszego/największego (np. spojrzeć na jedynekę w permutacji)
- szacowanie wyniku - czy wynik jest mały? czy można skonstruować algorytm, który zawsze znajdzie górne ograniczenie na wynik?
- nie gardzić punktami gdy leżą na ulicy (podzadania)

- UWAGI, CZĘSTE BŁĘDY

- pamiętaj o pamięci... 10^6 intów = 4 MB
- long longi...
- na OIu flaga -fsanitize=address lub -fsanitize=undefined może nie działać, trzeba wtedy skorzystać z kompilatora g++-4.9 zamiast clang++
- na OIu jakiegokolwiek komparatory do sort'ów wymagają, by argumenty były const
- funkcja resize() w vectorach nie czyści (nie resetuje wartości) pozostałych elementów - kłopotliwe jak się ma tablice globalne i wiele testów w jednym pliku wejściowym
- pamiętaj o trzech magicznych liniijkach podczas korzystania z cin, cout
- nie można szybko sprawdzać odległości między iteratorami w secie/mapie (w szczególności, set nie potrafi szybko odpowiadać na pytania "którym elementem jest x")
- jak w treści jest jakiś dziwny warunek i nie wiadomo co on w treści robi (lub jakaś zmienna jest podejrzanie mała), jest to klucz do rozwiązania

- MATERIAŁY

- solve.edu.pl/~sparingi/resources
- was.zaa.mimuw.edu.pl (lepsza jakość nagrań na [youtube](https://www.youtube.com))
- komendium.meetit.pl
- cp-algorithms.com (artykuły po angielsku)
- książka "Wprowadzenie do algorytmów" by Cormen, podobno w bibliotece szkolnej
- książka "Zaprzyjaźnij się z algorytmami" - podstawy
- książki Olimpiady Informatycznej
 - * Przygody Bajtazara
 - * W Poszukiwaniu Wyzwań
 - * niebieskie książeczki - omówienia zadań z OI