- `bitcount`: retorna a quantidade de números 1 de um determinado número em sua representação binária;
- `breadth first search`: do português, busca em largura, começa pelo vértice raiz e explora todos os vértices vizinhos. Então, para cada um desses vértices mais próximos, explora-se os seus vértices vizinhos inexplorados e assim por diante, até que ele encontre o alvo da busca;
- `bucketsort`: ordena um *array* dividindo-o em um número finito de recipientes (k = passando como parâmetro como o maior número possível no *array*) cria os k *buckets*, adiciona 1 em cada valor dentro do array exemplo: k=4, valor=3 = [0,0,1,0] enumera o vetor que contém 0 ou 1 [0,0,1,0] = [(0,0),(1,0),(2,1),(3,0)] e adiciona ordenado em um *array* multiplicando um iterador para cada posição, montando assim o vetor ordenado;
- `depth first search`: do português, busca em profundidade, começa num nó raiz (*startnode*) e vai se aprofundando cada vez mais, até que o dado (*goalnode*) de busca seja encontrado ou até que ele se depare com um nó que não possui filhos (*leafnode*). Então a busca retrocede e começa no próximo nó;
- `detect cycle`: retorna verdadeiro se o grafo direcionado contém pelo menos um ciclo e falso caso não;
- `find first in sorted`: retorna a primeira aparição de um determinado valor em um *array*, caso não exista retorna -1, utilizada busca binaria;
- `find in sorted`: encontra por busca binária a posição de um determinado valor;
- `flatten`: retorna um único *array*, com um elemento cada posição. Caso tenha uma lista no *array* enviado, essa lista é particionada e cada elemento dessa lista é adicionado no vetor final a partir da sua posição inicial;
- `gcd`: o máximo divisor comum entre dois ou mais números reais é o maior número real que é fator de tais números;
- `get factors`: retorna um vetor com os valores da fatoração do número;
- `hanoi`: com 3 pilastras, inicialmente *start* em 1 e *end* em 3. É necessário mover todos os discos para o pino da direita Deve mover um disco de cada vez, sendo que um disco maior nunca pode ficar em cima de um disco menor. O programa recebe como parâmetro o número de discos, em qual pilastra começa e em qual devemos colocar, como padrão começa na mais esquerda 1 e deve terminar na mais a direita 3. Ele retorna as movimentações que devemos fazer (x,n) da pilastra x para pilastra n;
- `is valid parenthesization`: verifica se os parênteses estão dispostos de maneira correta, isto é, sempre quando aberto "(" deve, obrigatoriamente, que existir o que fecha ")";

- `kheapsort`: algoritmo de ordenação, com *heap*. Se P é um nó pai de C, então a chave (o valor) de P é maior que ou igual a (em uma *heap* máxima) ou menor que ou igual a (em uma *heap* mínima) chave de C;
- `knapsack`: problema da mochila. Solicita como parâmetro a capacidade que a mochila consegue carregar e a lista dos itens contendo cada item (a,b), a = peso, b = valor;
- `kth`: solicita como parâmetro um vetor, e um número k, onde k deve ser menor que o tamanho do array, e então encontra qual o valor k está o valor ordenado;
- `lcs length`: retorna o comprimento da maior subsequência presente em ambas as *strings* analisadas;
- `levenshtein`: encontra o número mínimo de operações necessárias para transformar uma *string* em outra;
- `lis`: retorna o comprimento da subsequência mais longa de um *array* no qual seus elementos são dispostos em ordem crescente;
- `longest common subsequence`: encontra qual a maior sequência dos elemento em duas strings;
- `max sublist sum`: retorna a maior soma de n números consecutivos do *array*;
- `mergesort`: algoritmo de ordenação que divide o problema em vários subproblemas e resolve esses subproblemas através da recursividade, após todos os subproblemas terem sido resolvidos ocorre a conquista que é a união das resoluções dos subproblemas;
- `minimum spanning tree`: *spanning tree*, é a arvore formada por um grafo não direcionado que abrange todos os vértices. Então a *minimum spanning tree*, é o mínimo "caminho" do grafo que abrange todos os vértices;
- `next palindrome`: um número inteiro positivo é denominado palíndromo se sua representação no sistema decimal for a mesma quando lido da esquerda para a direita e da direita para a esquerda. Dado um número palíndromo, a função retorna o próximo número palíndromo;
- `next permutation`: reorganiza os números na próxima maior permutação de números lexicograficamente. Se tal arranjo não for possível, ele reorganizará na ordem mais baixa possível (ou seja, classificado em ordem crescente);
- `pascal`: o triângulo de Pascal é um triângulo numérico infinito formado por números binomiais (n, k) onde n representa o número da linha e k representa o número da coluna, iniciando a contagem a partir do zero. O programa retorna uma lista com cada linha, começando com a primeira;

- `possible change`: o programa retornar quantas maneiras podemos fazer mudança no *array* informado para termos um determinado valor informado;
- `powerset`: o conjunto de todos os subconjuntos de um conjunto dado A é chamado de conjunto de partes (ou conjunto potência) de A, denotado por P(A) ou 2 A 2**A;
- `quicksort`: algoritmo de ordenação, um elemento é selecionado como o pivô e a lista é dividida de forma que todos os elementos antes do pivô sejam menores do que ele e todos os elementos após o pivô sejam maiores do que ele. No final do processo, o pivô estará em sua posição final, e haverá duas sulistas não ordenadas. Essa operação é chamada de particionamento. Recursivamente é ordenando as sub lista dos elementos menores e maiores;
- `reverse linked list`: dado o nó principal de uma *linked list*, a programa reverte a *linked list*. Exemplo: input = 1-¿2-¿3-¿NULL output: 3-¿2-¿1-¿NULL;
- `rpn eva`: avalia expressões em notação polonesa reversa;
- `shortest path length`: é dado um gráfico ponderado, um vértice destino e um vértice origem, o programa retorna o menor caminho da origem até o destino;
- `shortest path lengths`: é dado um grafo ponderado e o número total de seus vértices, o programa retorna um dicionário com todos os possíveis caminhos no grafo juntamente com o seus respectivos pesos;
- `shortest paths`: dado um grafo orientado e ponderado, e um nó, o programa retorna um dicionario com todos os vértices e os pesos deles referente ao nó enviado;
- `shunting yard`: o algoritmo recebe uma expressão matemática especificada em notação de infixo. Ele retorna uma *string* de notação pós-fixada, também conhecida como notação polonesa reversa (RPN);
- `sieve`: dado um número n, imprime todos os primos menores ou iguais a n. Também é dado que n é um número pequeno. Retorna um *array* com os números primos;
- `sqrt`: dado um *epsilon* o programa retorna a raiz quadrada, ou aproximada se não exata, do número *epsilon* com uma "margem de erro";
- `subsequences`: dado um valor no formato (a, b, k) o programa retornará um *array* com a quantidade máxima de subsequências que é possível realizar com valores de a - b-1 no tamanho de k;

- `to base`: dado um número em decimal A e uma base B, o programa resulta na representação do número decimal A na base B;
- `topological ordering`: a ordenação topológica para Grafo Acíclico Direcionado (DAG) é uma ordenação linear de vértices tal que para cada aresta direcionada (u, v), o vértice u vem antes de v na ordenação. Cada DAG pode ter uma ou mais ordenações topológicas;
- `wrap`: dado parágrafo em texto (uma *string*) retorna uma lista na qual cada elemento representa uma linha que tenha, no máximo, o comprimento de caracteres de largura informado por parâmetro.