O algorytmach, wyszukiwaniu binarnym i sortowaniu bąbelkowym

* Uwaga, to nie jest wierny skrypt przedstawionej prezentacji. Plik zawiera zmienione treści.

Definicja algorytmu

Algorytm jest skończonym, uporządkowanym ciągiem jasno zdefiniowanych czynności koniecznych do wykonania zadania. Innymi słowy, algorytm to sposób postępowania prowadzący do rozwiązania problemu, który przeprowadza system z pewnego stanu początkowego do pożądanego stanu końcowego. Co ważne, algorytmy zwykle formułowane są w sposób ścisły, często w oparciu o język matematyki. I taki algorytm na przykład może zostać zaimplementowany w postaci programu komputerowego. Trzeba jednak pamiętać, że istnieją także problemy, które nie sposób scharakteryzować za pomocą języka formalnego. To na przykład przepisy kulinarne opisywane słownie, tzn z użyciem języka mówionego. Jeszcze innym przykładem może być twórczość artystyczna, gdzie ważna jest wyobraźnia i intuicja, a nie zasady zero jedynkowe. Chociaż w tym przypadku może dobrze byłoby sięgnąć po biografię Johna Nasha, "biografię genialnego matematyka".

Schemat blokowy

Niektóre problemy można przedstawić na schemacie blokowym, w którym każdy blok ma ściśle określone zastosowanie. Dostępnych jest wiele ciekawych narzędzi, które umożliwiają tworzenie takich schematów. VS Studio firmy MS również posiada takie rozwiązanie.

Historia algorytmu

Początkowo słowem algorism nazywano czynności konieczne do wykonywania obliczeń z użyciem dziesiętnego systemu liczbowego. Obecne znaczenie słowa algorytm, jako zestawu ścisłych reguł, powstało wraz z rozwojem matematyki i techniki. Zbudowanie maszyn, które same mogły realizować pewne proste algorytmy, stało się przełomem. Początkowo miały one postać układów mechanicznych mogących realizować proste obliczenia. Teraz mamy komputery.

Cechy dobrego algorytmu

- poprawny (integralny): dla każdego poprawnego zestawu danych, po wykonaniu skończonej liczby czynności, prowadzi to poprawnych wyników
- jednoznaczny: dla identycznego zestawu danych początkowych, algorytm zawsze zwróci identyczny wynik
- kompletny (skończony): algorytm wykonuje się w skończonej liczbie kroków
- sprawny i optymalny: rozwiązuje dany problem możliwie szybko, jest pozbawiony redundancji (dzięki temu oszczędza pamięć)

Prawa autorskie

W niektórych krajach (między innymi w Stanach Zjednoczonych) algorytmy mogą zostać opatentowane, jeżeli zostaną zaimplementowane w jakimś praktycznym celu. Przeciwnicy tego podejścia twierdzą, że patentowanie algorytmów spowalnia rozwój informatyki, bo jeden producent może uzyskać monopol na pisanie oprogramowania tworzącego pewne typy plików (jak było to w przypadku GIF). Wiele koncernów komputerowych prowadzi między sobą spory prawne dotyczące praw własności do niektórych patentów. Kontrargumentem zwolenników patentów na oprogramowanie jest prawo własności intelektualnej, zakładające, że program jest intelektualną własnością twórcy. Podobnie ma się to w przypadku utworów muzycznych.

Przykłady algorytmów

- algorytm Euklidesa (dotyczy wyznaczenia największego wspólnego dzielnika dwóch liczb)
- algorytm Dijkstry (dotyczy obliczania najkrótszej drogi między dwoma punktami)
- algorytm Fermata (dotyczy sprawdzenia poprawności wpisanej liczby)
- algorytmy sortowania
- algorytmy wyszukiwania

Dziedzina badająca algorytmy to algorytmika.

Algorytmy wyszukiwania: (zasada działania, przykład, zastosowanie)

Wyszukiwania dokonujemy w pewnym zbiorze elementów i w określonych celach. Istnieje kilka algorytmów wyszukiwania. Jeśli jednak nie dysponujemy żadną wiedzą na temat zbioru, który chcemy przeszukiwać, wtedy musimy sprawdzić wszystkie jego elementy, krok po kroku. Takie wyszukiwanie nazywamy wyszukiwaniem liniowym. Jeśli jednak wiemy, że przeszukiwany zbiór zawiera elementy, które są w pewien sposób posegregowane, ułożone, jak na przykład słownik i zawarte w nim słowa ułożone są w kolejności alfabetycznej, możemy użyć wyszukiwania binarnego, które wydaje się być o wiele bardziej wydajnym sposobem. Takie wyszukiwanie polega na wielokrotnym dzieleniu na pół danego podzbioru, zaczynając od zbioru głównego. Popularnym przykładem jest sytuacja, w której komputer przechowuje w pamięci liczbę z zakresu 1-100, a zadaniem użytkownika jest znalezienie tej liczby.
W tym celu dzielimy taki przedział na pół. Wybieramy liczbę 50. Okazuje się, że liczba 50 nie jest liczbą szukaną. Wiemy już jednak, że musi być to liczba większa niż 50. Tym samym przedział, w którym szukana liczba się znajduje, znacząco się ograniczył. W przykładzie to obszar objęty pomarańczową pętlą. Nie będziemy już szukać liczby z zakresu od 1-100, a 51-100. Większe prawdopodobieństwo trafienia uzyskamy ponownie dzieląc nasz przedział na pół, czyli wybierając liczbę 75. Jak widać liczba 62 również nie jest szukaną odpowiedzią, ale przedział, w którym znajduje się szukana, ponownie się zmniejszył. I tak wygląda schemat działania.

Algorytm sortowania bąbelkowego: (zasada działania, przykład, zastosowanie)

Sortowanie bąbelkowe polega na wyznaczeniu posortowanego rosnąco lub malejąco ciągu liczb. Program taki działa na zasadzie wagi. Porównuje jedną liczbę z drugą i zależnie od rodzaju sortowania, zamienia je miejscami. Jeżeli na przykład mamy do posortowania przedstawiony zbiór liczb w kolejności od najmniejszej do największej to program sprawdzi najpierw parę liczb, które stoją na końcu: 4 i 0.
W tym przypadku 0 jest większe, dlatego zostanie przesunięte w lewo. Innymi słowy liczba 4 zastąpi 0. Następnie to liczba 2 będzie poddana sprawdzeniu w parze z zerem.
Schemat powtarza się do momentu uzyskania oczekiwanego wyniku.