Programowanie dla fizyków 2

Raport

Rekurencja i backtracking

Katarzyna Gajewska
Maciej Kucharski
Tomasz Biel

11 czerwiec 2021

1. Wstęp

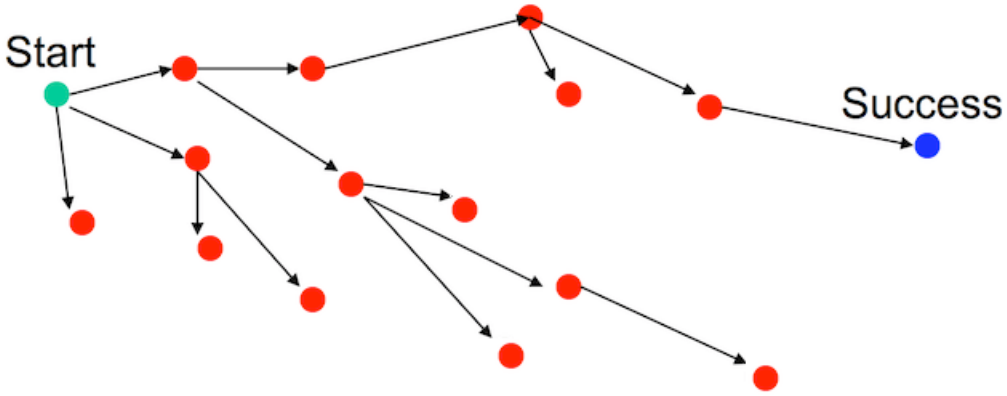
Jako przedmiot naszego projektu wybraliśmy zagadnienie backtrackingu oraz rekurencji. Opiszemy na czym polegają oba zagadnienia i jak działa nasz program wykorzystujący je. Przeanalizujemy kod wykonanego algorytmu krok po kroku.

2. Rekurencja

[1]Krótko mówiąc rekurencja jest to odwoływanie się np. funkcji lub definicjido samej siebie. Definicja rekurencyjna składa się z dwóch części. W pierwszej,zwanej podstawową lub warunkiem początkowym, są wyliczo-ne elementy podstawowe, stanowiące części składowe wszystkichpozostałych elementów zbioru. W drugiej części, zwanej krokiemindukcyjnym, są podane reguły umożliwiające konstruowanie nowych obiektów z elementów podstawowych lub obiektów zbudo-wanych wcześniej. Reguły te można stosować wielokrotnie, two-rząc nowe obiekty.

3. Backtracking

[2]Backtracking (algorytm z nawrotami) jest to algorytm wy-szukiwania polegający na znajdowaniu wyniku metodą "prób ibłędów" z oznaczaniem niepowodzeń, dzięki czemu te same błę-dy nie są popełniane dwukrotnie.



Rysunek 1: Graficzne zobrazowanie backtrackingu

4. Zastosowanie algorytmów z nawrotami

Backtracking - inaczej algorytm z nawrotami, jest narzędziem wykorzystującym rekurencję, służącym do znalezienia porządanego rozwiązania metodą prób i błędów. Algorytm ten sprawdza każdą z opcji raz za razem, aż znajdzie prawidłowe rozwiązanie. Jest to dużo szybsze i wydajniejsze niż sprawdzanie ręczne wszystkich kombinacji z danego zagadnienia. Są różne przykłady zastosowania tego algorytmu. Kilka z nich omówiliśmy w prezentacji. Teraz skupimy się raczej na głębszej analizie naszego kodu.

5. Analiza kodu - Sudoku

Nasz program zawiera trzy pliki. Jeden z ciałem klasy, drugi z jej nagłówkami oraz trzeci zawierający wszystkie instrukcje związane ze sprawdzaniem i znajdywaniem rozwiązania. Nie ma sensu wchodzić w tak trywialne rzeczy jak opisywanie pliku z ciałem klasy czy jej nagłówkami. Przedstawione będą jedynie te elementy kodu, które odpowiadają za rozwiązywanie sudoku i sprawdzanie wszystkich możliwości. Przed szukaniem rozwiązania, program musi mieć określony plan działania - narzucone jest w jakiej kolejności ma sprawdzać odpowiednie klatki sudoku, pilnuje czy wpisano odpowiednią ilość cyfr, na bieżąco sprawdza w jakiej klatce się znajdujemy, itd. Można wymieniać wszystkie najmniejsze szczegóły, ale lepiej przejść do najbardziej interesującego nas fragmentu, czyli do formuły związanej z backtrackingiem i rekurencją:

```
bool Sudoku::Rozwiaz(int wiersz, int kolumna)
{
    if (wiersz == (9 - 1) && kolumna == 9)
        return true;

    if (kolumna == 9)
    {
        wiersz++;
        kolumna = 0;
    }

    if (liczby[wiersz][kolumna] > 0)
        return Rozwiaz(wiersz, kolumna + 1);

    for (int num = 1; num <= 9; num++)
    {
        if (SprawdzLiczbe(wiersz, kolumna, num))
        {
            liczby[wiersz][kolumna] = num;

            if (Rozwiaz(wiersz, kolumna + 1))
                return true;
        }

        liczby[wiersz][kolumna] = 0;
    }
    return false;
}
```

Rysunek 2: Element kodu z rekurencją

Program w tym miejscu kolejno sprawdza czy jesteśmy w ostatniej klatce sudoku. Jeśli tak to znaczy że sudoku zostało rozwiązane całe. Kolejna część odpowiada za przechodzenie do następnego wiersza jeśli jesteśmy w ostatniej klatce wiersza poprzedniego. Następnie, jeśli liczba w danej klatce nie jest zerem, czyli klatka jest wypełnione stałą cyfrą, to program przechodzi do następnej. W następnym kroku program wstawia liczbę do pustej klatki po sprawdzedniu czy ona pasuje. Później zabieg ten wykonywany jest od początku dla kolejnych klatek i cyfr. W ten sposób sprawdzane są wszystkie wartości. Jeśli dana wartość w pewnym momencie nie będzie pasować, to program wykasuje tyle cyfr ile potrzeba i zacznie sprawdzać od pewnego momentu dla kolejnej możliwej do wpisania cyfry. I tak w kółko aż nie znajdziemy rozwiązania.
Warto jeszcze pokazać sposób sprawdzania każdej cyfry czy możliwe jest jej wpisanie:

```
bool Sudoku::SprawdzLiczbe(int wiersz, int kolumna, int liczba)
{
    // Sprawdzamy wiersze
    for (int x = 0; x <= 8; x++)
        if (liczby[wiersz][x] == liczba)
            return false;

    // Sprawdzamy kolumny
    for (int x = 0; x <= 8; x++)
        if (liczby[x][kolumna] == liczba)
            return false;

    // Sprawdzamy kwadrat
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (liczby[i + (wiersz - wiersz % 3)][j + (kolumna - kolumna % 3)] == liczba)
                return false;

    return true;
}
```

Rysunek 3: Element kodu ze sprawdzaniem cyfr

W pierwszej części sprawdzany jest caly wiersz danej klatki, czy liczba którą chcemy wpisać nie powtarza się w żandej klatce tego wiersza. W drugiej części dzieje się to samo, ale dla kolumny. Trzecia część sprawdza czy nie występuje taka sama liczba jaką chcemy wpisać, w kwadracie 3x3, w którym w danym momencie się znajdujemy.
Tak oto przedstawione zostały główne elementy kodu, a przede wszystkim to co odpowiada za backtracking.

6. Podsumowanie

Realizując projekt zapoznaliśmy się z pojęciem backtrackingu i rekurencji. Nauczyliśmy się stosować te zagadnienia w języku programowania. Nie da się wyciągnąc obszernych wniosków na ten temat. Algorytm ten nie jest jakimś wykwintnym narzędziem stosującym skomplikowane wzory. Sprawdza po prostu metodą prób i błędów wszystkie możliwości aż znajdzie rozwiązanie. Jest t "siłowe" rozwiązanie lecz skuteczne.

7. Bibliografia

Literatura

[1] https://pl.wikipedia.org/wiki/Rekurencja

[2] https://encyklopedia.interia.pl/informatyka/news-algorytm-z-nawrotami,nId,2087177