Teoria Algorytmów i Obliczeń

Laboratorium

Projekt „Tetris” – program automatycznie układający klocki maksymalizując gęstość ułożenia

Autorzy: Paweł Duszak, Sławomir Fajer, Bartosz Woźniak

Spis treści

Opis projektu

Celem projektu była implementacja oraz wizualizacja wyników działania algorytmu układającego zadany zbiór klocków na planszy o zadanej szerokości. Algorytm ma na celu maksymalizowanie gęstości ułożenia klocków na planszy (wysokość ułożenia klocków powinna być możliwie minimalna). Zadaniem problemu jest zbiór klocków, który wczytywany jest z pliku w następującym formacie:

Szerokość_Studni
Wysokość_Klocka_1 Szerokość_Klocka_1
[Tablica dwuwymiarowa o podanym powyżej rozmiarze składająca się z 1 i 0. 1 – definiuje kształt klocka], np.
1 0 0 0
1 0 0 0
1 1 1 1
Wysokość_Klocka_2 Szerokość_Klocka_2
… itd.

Zakładamy poprawność wczytywanego pliku, tzn. wczytywany plik musi być dokładnie w formacie zdefiniowanym powyżej. Ponadto klocki muszą być spójne (każda jedynka musi mieć wspólny bok z inną jedynką). Podczas wczytywania pliku wykrywane są duplikaty (również obrócone) i takie zduplikowane klocki są odrzucane, czyli taki sam klocek nie jest wczytywany kilkukrotnie. Po wczytaniu pliku użytkownik ma możliwość przeglądania klocków i wybrania ile sztuk jakich płytek chce ułożyć. Użytkownik ma również możliwość wybrania kroku algorytmu – ile klocków jest układanych w jednym ruchu oraz parametru K – który wpływa na jakość ułożenia klocków (jest to parametr algorytmu, zostanie opisany w kolejnej sekcji). Po zdefiniowaniu niezbędnych parametrów, istnieje możliwość kliknięcia przycisku start i oglądania wizualizacji działania (pokazywane są kolejne zmiany na planszach) lub klikania przycisku Next i oglądania wizualizacji kolejnych kroków algorytmu po każdym kliknięciu. W każdym momencie użytkownik ma również możliwość serializowania stanu gry, który może później przywrócić i wznowić obliczenia.

Częścią projektu było również przygotowanie przykładowego zestawu klocków.

Opis algorytmu

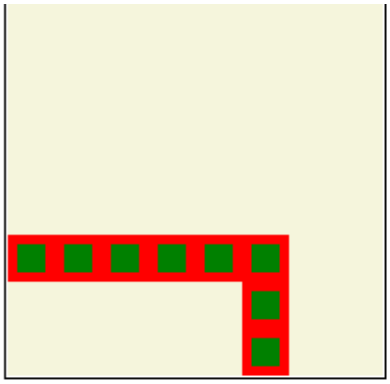
Algorytm układania klocków jest algorytmem zachłannym z nawrotami. Głównym parametrem algorytmu jest K – określa on ile najlepszych rozwiązań w każdym kroku spośród wszystkich możliwych będzie poddanych dalszej analizie.

Zasada działania algorytmu:

- Próba dostawienia każdego dostępnego klocka dla każdej symulacji
- Wybór K najlepszych symulacji przy użyciu funkcji oceniającej

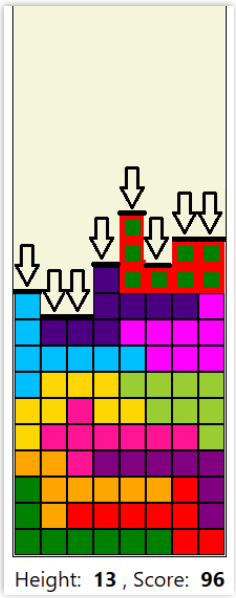
Dwoma najważniejszymi częściami algorytmu są: funkcja znajdująca najlepsze położenie płytki oraz funkcja oceniająca.

Funkcja oceniająca powinna wyznaczać gęstość ułożenia klocków i na tej podstawie powinniśmy wybierać K najlepszych rozwiązań. Niestety takie rozwiązanie nie jest rozwiązaniem optymalnym. Rozważmy następujący przykład:

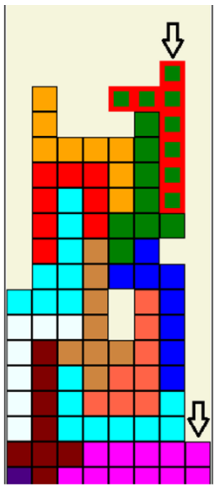


Jeśli obrócimy ten klocek o 180 stopni, otrzymamy układ o takim samym współczynniku gęstości (stosunku zajętych pól do wszystkich pól pod linią wyznaczającą maksymalną wysokość). Obrócony klocek tworzyłby jednak lepszą bazę do układania na nim następnych klocków, dlatego też zdecydowaliśmy się zmienić funkcję oceniającą, aby wybierała właśnie to lepsze rozwiązanie w pierwszej kolejności.

Gęstość liczona zawsze jest pod górną linią (jak na rysunku poniżej), dla każdej kolumny osobno (każda kolumna ma więc inną wysokość).



Takie rozwiązanie okazało się jednak generować pewien problem – powstawały kominy jak na rysunku poniżej. Ważnym jest, aby układane klocki tworzyły na górze możliwie płaską powierzchnię, dlatego też wprowadziliśmy dodatkowy parametr: - Stała * (MaxCol - MinCol), gdzie MaxCol to maksymalna wysokość kolumny, MinCol – minimalna wysokość kolumny, Stała = 1 (na podstawie przeprowadzonych testów stwierdzono, że daje najlepsze rezultaty).



Podsumowując, funkcja oceniająca wyrażona jest następującym wzorem:

Wartość = (100 * ilość pełnych) / (ilość wszystkich) - stała * (MaxCol – MinCol)

Czyli gęstość liczona dla każdej kolumny osobno wyrażona w procentach pomniejszona o różnicę pomiędzy wysokością najwyższej i najniższej kolumny.

Zaobserwowaliśmy, iż dla dużej ilości klocków, wąskich studni, czyli dużej wysokości układanego zbioru klocków wyniki znacznie się pogarszały, ponieważ znaczna część gęstości była liczona na podstawie dużego zbioru już ułożonych klocków, co powodowało wyrównywanie się znacznie różniących się od siebie ułożeń. Dlatego też liczymy gęstość od pewnego poziomu (kilka poziomów poniżej najniższej kolumny), aby mieć pewność, iż zajmujemy się tylko znaczącymi klockami.

Program wyświetla wartość naszej funkcji oceniającej – Score, wartość prawdziwej gęstości – Density oraz wysokość studni – Height.

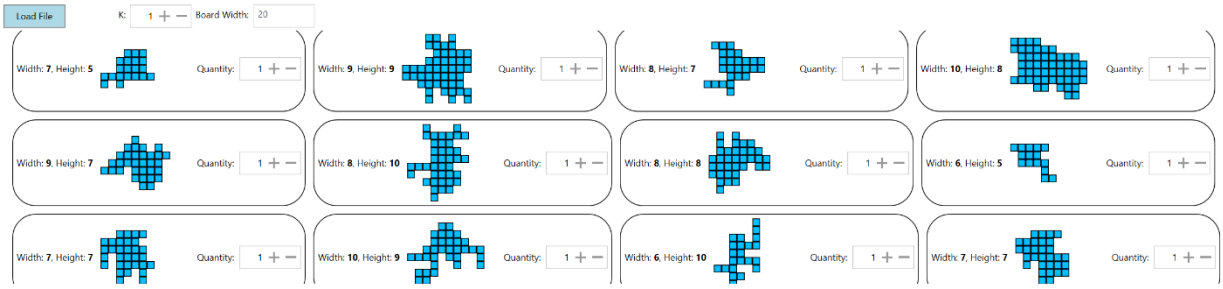
Każdy klocek wraz z jego obrotami próbujemy ułożyć na każdej planszy możliwie najniżej i najbardziej po lewo. Następnie wybieramy najlepszych K ułożeń na podstawie funkcji oceniającej.

Opis danych użytych do testowania

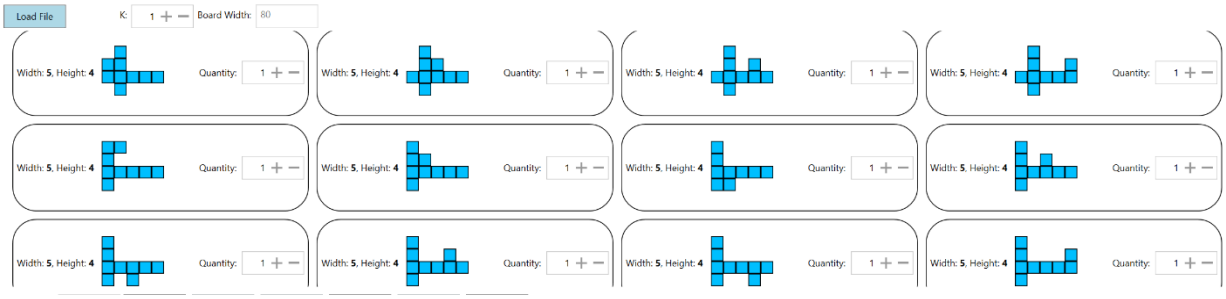
Do testowania użyliśmy sześciu różnych zestawów klocków. Każdy zestaw zawierał co najmniej 100 klocków (użyliśmy kliku klocków każdego rodzaju z zestawu). Każdy zestaw klocków został przetestowany dla stosunkowo małej i dużej szerokości studni oraz dla małej i dużej wartości parametru K.

Opis kolejnych zestawów klocków:

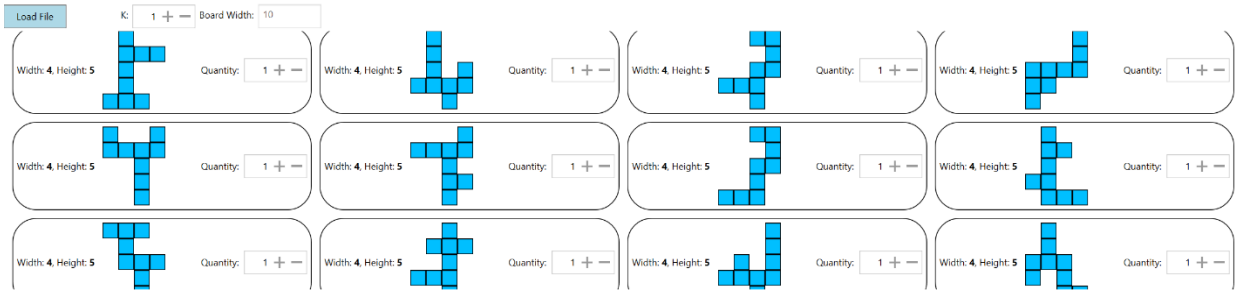
- Zestaw klocków przygotowany przez grupę Marcina Rudnika – zestaw bardzo różnorodnych klocków – zestaw zawiera bardzo duże i małe klocki, co powoduje słabe ułożenie, ponieważ małe klocki są układane w pierwszej kolejności (są niższe i lepiej upakowane). Wynika ta z zachłanności algorytmu.



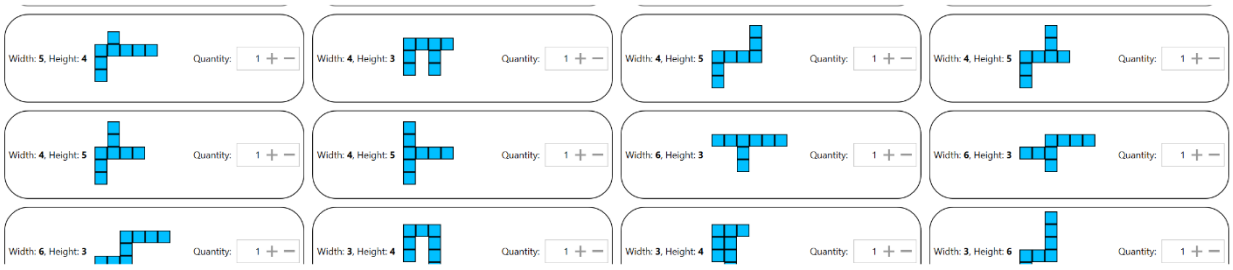
- Zestaw przygotowany przez grupę Anny Bekas – zestaw różnorodnych klocków, podobnej wielkości, przypominających trochę klocek w kształcie litery L z różnymi wypustkami. Klocki mogą być źle układane, ponieważ słabo do siebie pasują, mogą powstawać duże dziury podczas minimalizacji gęstości w danym kroku.



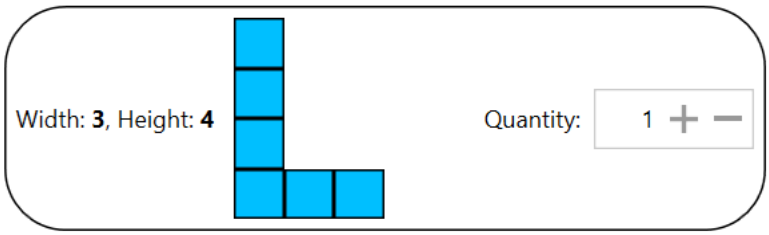
- Zestaw przygotowany przez grupę Mirona Marczuka – zestaw prostokątów – zestaw podobny do zestawu powyżej



- Zestaw przygotowany przez nas – zestaw 60 klocków, których pole jest mniejsze lub równe 8 oraz które są opisane na prostokącie (nie kwadracie) o krótszym boku większym od dwóch



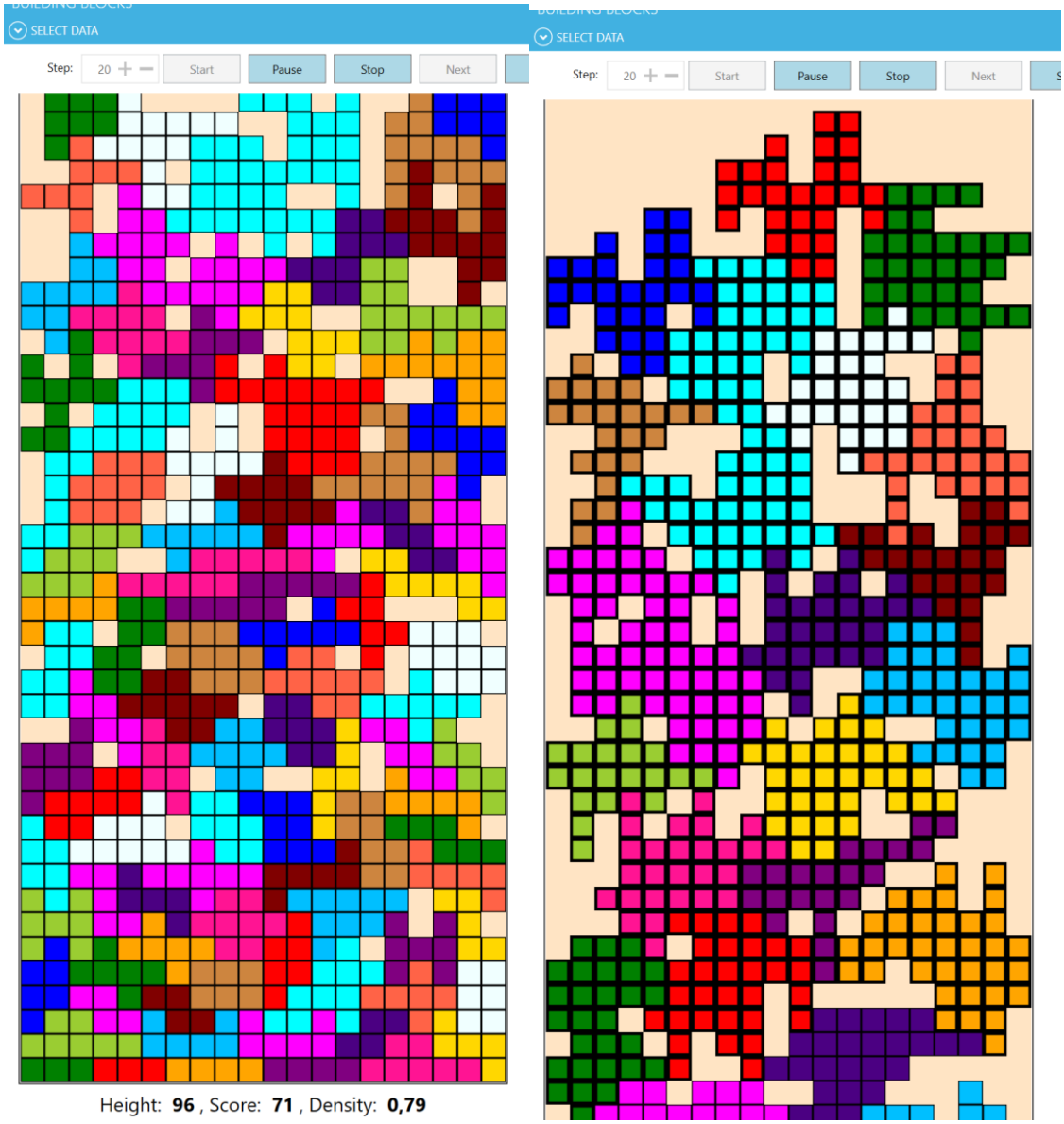
- Zestaw składający się z jednego klocka (100 razy) w kształcie litery L – może dawać bardzo słabe wyniki dla niektórych algorytmów, opisane szczegółowo w początkowej części sekcji algorytm, nasz algorytm sobie radzi



- Zestaw psujący dobre wyniki – składający się z kwadratu 5 na 5 i prostokąta 5 na 1, szerokość studni 6 – bardzo łatwy zestaw do ułożenia dla człowieka, aby uzyskać gęstość 100 procent, jednak ciężko znaleźć algorytm, który ułoży to poprawnie minimalizując na każdym kroku wysokość/gęstość – zła decyzja w pierwszym kroku (wybór linii poziomo) będzie powodować gorsze wyniki

Wyniki testów

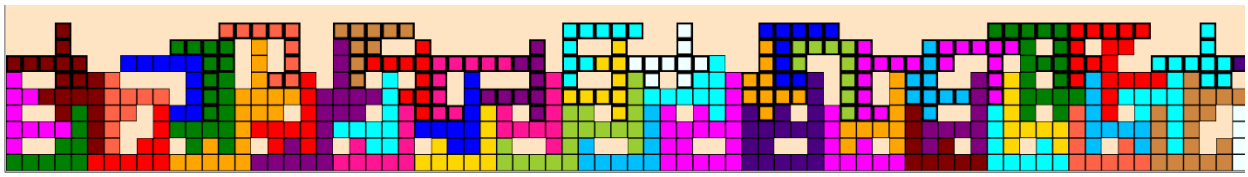
- Zestaw klocków przygotowany przez grupę Marcina Rudnika



Height: **96** , Score: **71** , Density: **0,79**

Algorytm uzyskał gęstość na poziomie 79%, co jest dobrym wynikiem. Niestety małe klocki zostały ułożone w pierwszej kolejności, dlatego gęstość na dole (obrazek po lewej stronie) jest znacznie większa, niż gęstość na górze (obrazek po prawej stronie), gdzie zostały ułożone duże klocki. Optymalnym rozwiązaniem byłoby ułożenie najpierw dużych klocków, a następnie dopasowywanie małych w wolne miejsca. Jednak mimo to, również ze względu na dość wąską studnię, otrzymana gęstość jest zadawalająca.

- Zestaw przygotowany przez grupę Anny Bekas



Height: **9** , Score: **81** , Density: **0,75**

Algorytm uzyskał gęstość na poziomie 75%, co również jest dobrym wynikiem. Algorytm układał klocki rozsądnie, ciężko znaleźć lepsze ułożenie (jednak klocki są słabo do siebie dopasowane). Również duża szerokość studni negatywnie wpływa na gęstość. Złożoność czasowa wykonywanego algorytmu była bardzo duża. Znacznie większa, niż w poprzednim teście.

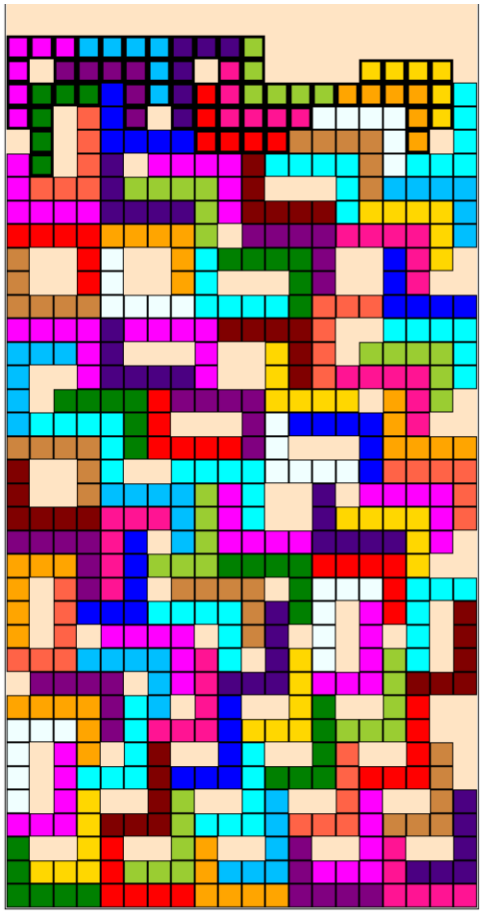
- Zestaw przygotowany przez grupę Mirona Marczuka

Zestaw daje podobne wyniki jak opisane powyżej.

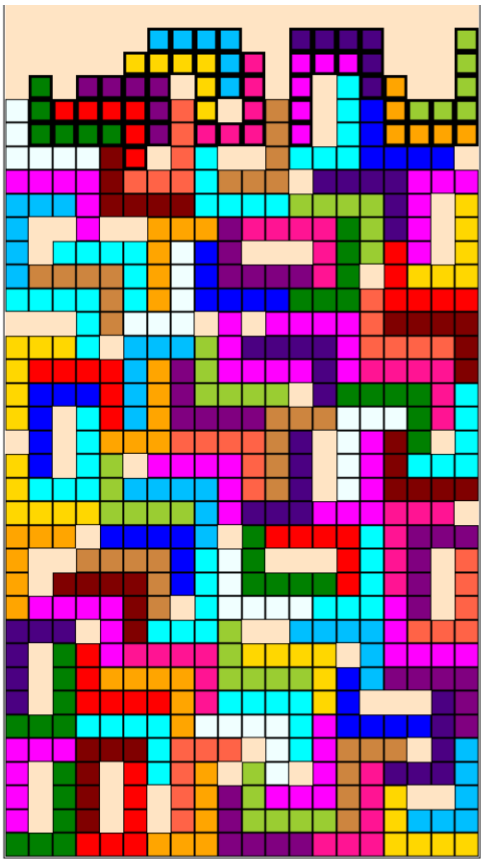
- Zestaw przygotowany przez nas

Zestaw daje podobne wyniki jak opisane powyżej.

- Zestaw składający się z jednego klocka w kształcie litery L



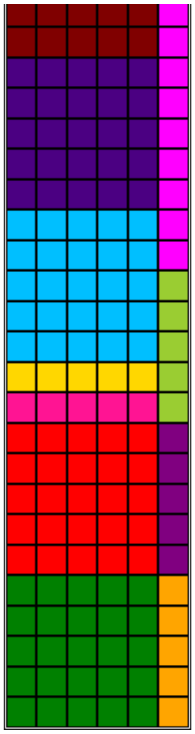
Height: **37** , Score: **84** , Density: **0,81**



Height: **35** , Score: **86** , Density: **0,86**

Algorytm uzyskał gęstość na poziomie 81% dla K = 1, czyli wynik jest zadowalający. Algorytm jest zachłanny, ale stara się wypełniać puste przestrzenie. Dla K = 30 uzyskujemy gęstość na poziomie 86%, czyli poprawa jest znaczna. Wydajność algorytmu jednak znacznie spadła ze wzrostem K.

- Zestaw psujący dobre wyniki



Height: **252** , Score: **88** , Density: **0,99**

Uzyskaliśmy gęstość na poziomie 99%, więc bardzo dobrą, jednak wiemy, iż można łatwo uzyskać gęstość 100% i wysokość 250 (50 kwadratów wysokości 5). Algorytm znajduje identyczne rozwiązanie dla K = 1 i K = 30. Jednak znacząco tracimy na wydajności.

Wnioski

Na podstawie przeprowadzonych testów jesteśmy w stanie stwierdzić, iż działanie algorytmu jest w miarę optymalne. Dla większości różnych zestawów klocków, które zostały „losowo" wybrane (nie są to przypadki graniczne) otrzymujemy gęstość na poziomie 70 – 80 procent. Złożoność czasowa algorytmu jest akceptowalna. Jednocześnie możemy wskazać pewne zależności uzyskanego wyniku od parametru nawrotu K oraz szerokości studni.

Szerokość studni negatywnie wpływa na uzyskany wynik gęstości, czyli wraz ze wzrostem szerokości studni gęstość zwykle maleje. Jest to spowodowane faktem, iż mamy większą powierzchnię na górze studni, powierzchnia ta może nie być płaska. Możemy powiedzieć, iż czasami brakuje klocków na wypełnienie umownego górnego wiersza klocków, dlatego gęstość maleje. Wąskie studnie mogą natomiast zostawiać większe luki, jednak powierzchnia na górze jest znacznie mniejsza, co w testowanych przez nas przypadkach rekompensuje straty z nawiązką. Szerokość studni również negatywnie wpływa na złożoność czasową, ponieważ w takiej studni zwykle jest więcej wolnej przestrzeni, więc każdy klocek próbuje być wstawiony w znacznie więcej miejsc, więc czas wykonywania programu jest znacznie większy.

Przeanalizowaliśmy również wpływ parametru K na złożoność czasową i gęstość ułożenia klocków. Oczywistym jest, że złożoność czasowa znacznie rośnie wraz ze wzrostem K, ponieważ mamy więc plansz do sprawdzenia i każdy klocek próbujemy ułożyć na kliku planszach. Wzrost parametru K również wpływa

na poprawę gęstości. Lepiej widać to na mniejszych zbiorach danych, niż opisanych w niniejszej dokumentacji. Przy tak dużych zestawach klocków znacznie tracimy na złożoności czasowej, a poprawa gęstości jest niewielka.